Lecture Notes in Artificial Intelligence 5785

Edited by R. Goebel, J. Siekmann, and W. Wahlster

Subseries of Lecture Notes in Computer Science

Lluís Godo Andrea Pugliese (Eds.)

Scalable Uncertainty Management

Third International Conference, SUM 2009
Washington, DC, USA, September 28-30, 2009
Proceedings

Springer

Volume Editors

Lluís Godo
Articial Intelligence Reseach Institute (IIIA)
Spanish National Research Council (CSIC)
Campus de la Universitat, 08193, Bellaterra, Spain
E-mail: godo@iiia.csic.es

Andrea Pugliese
University of Calabria, Department of Electronics
Computer Science and Systems
Via P. Bucci, 41C, 87036-Rende (CS), Italy
E-mail: apugliese@deis.unical.it

Library of Congress Control Number: 2009934039

CR Subject Classification (1998): I.2, I.2.3, F.3, F.4, H.2.1, I.2.4

LNCS Sublibrary: SL 7 – Artificial Intelligence

ISSN 0302-9743
ISBN-10 3-642-04387-9 Springer Berlin Heidelberg New York
ISBN-13 978-3-642-04387-1 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12755936 06/3180 5 4 3 2 1 0

Preface

A large amount of available data nowadays includes uncertain, imprecise and inconsistent information. The possibility of integrating and exploiting these data calls for sound and efficient techniques for managing uncertainty and handling inconsistency. These issues, which have been traditionally addressed within the artificial intelligence community, already play a key role in fields like databases or the semantic Web.

The annual International Conference on Scalable Uncertainty Management (SUM), grown out of this large interest in uncertainty and inconsistency, aims at bringing together all those interested in the management of uncertainty and inconsistency at large, fostering the collaboration and cross-fertilization between the reasoning under uncertainty community and the database and semantic Web communities.

This volume contains the papers presented at the Third International Conference on Scalable Uncertainty Management (SUM 2009), which was held in Washington DC, USA, September 28-30, 2009, following the successful previous editions of SUM 2007 in Washington DC, USA, and SUM 2008 in Naples, Italy. It contains 21 technical papers, which were selected out of 30 submitted papers in a rigorous reviewing process, where each paper was reviewed by three Program Committee members. The volume also contains abstracts of the two invited talks. We wish to thank all authors who submitted papers and all conference participants for fruitful discussions. We are grateful to Amol Deshpande and Thomas Lukasiewicz for their invited talks at the conference. We would like to thank all the Program Committee members and external referees for their timely expertise in carefully reviewing the submissions. Special thanks are due to Henri Prade and V.S. Subrahmanian, General Chairs of SUM 2009, for their invaluable help and support in organizing the conference. Many thanks also to the developers of the EasyChair conference system, which we used for the reviewing process and the preparation of this volume.

September 2009

Lluis Godo
Andrea Pugliese

Organization

General Chairs

Henri Prade	IRIT, France
V.S. Subrahmanian	University of Maryland, USA

Program Chairs

Lluis Godo	IIIA - CSIC, Spain
Andrea Pugliese	University of Calabria, Italy

Program Committee

Chitta Baral	Arizona State University, USA
Leopoldo Bertossi	Carleton University, Canada
Salem Benferhat	University of Artois, France
Bir Bhanu	University of California-Riverside, USA
Fabio Gagliardi Cozman	University of Sao Paulo, Brazil
Michael I. Dekhtyar	Tver State University, Russia
Juergen Dix	TU Clausthal, Germany
Francesco Donini	University of Tuscia, Italy
Didier Dubois	IRIT, France
Thomas Eiter	TU Vienna, Austria
Nicola Fanizzi	University of Bari, Italy
Filippo Furfaro	University of Calabria, Italy
John Grant	Towson University, USA
Sergio Greco	University of Calabria, Italy
Pascal Hitzler	University of Karlsruhe, Germany
Eyke Hüllermeier	University of Marburg, Germany
Edward Hung	The Hong Kong Polytechnic University, Hong Kong, SAR China
Anthony Hunter	University College London, UK
T.S. Jayram	IBM Almaden Research Center, USA
Churn-Jung Liau	Academia Sinica, Taiwan
Weiru Liu	Queen's University Belfast, UK
Thomas Lukasiewicz	Oxford University, UK; TU Vienna, Austria
Serafín Moral	University of Granada, Spain
Dan Olteanu	Oxford University, UK
Gabriella Pasi	University of Milan-Bicocca, Italy
Olivier Pivert	INRIA-ENSSAT, France
Michael Pittarelli	State University of New York, USA
David Poole	University of British Columbia, Canada

Henri Prade	IRIT, France
Emad Saad	Gulf University for Science and Technology, Kuwait
Domenico Saccà	University of Calabria, Italy
Maria Luisa Sapino	University of Turin, Italy
Prakash Shenoy	University of Kansas, USA
Umberto Straccia	ISTI-CNR, Italy
Heiner Stuckenschmidt	University of Mannheim, Germany
V.S. Subrahmanian	University of Maryland, USA
Maurice van Keulen	University of Twente, The Netherlands
Peter Vojtáš	Charles University, Czech Republic
Nic Wilson	University College Cork, Ireland

External Referees

Stefano Aguzzoli
Silvia Calegari
Martine De Cock
Christian Meilicke
Marco Mernberger
Stefan Woltran

Sponsoring Institutions

This conference was partially supported by the Department of Electronics, Informatics, and Systems (DEIS) of the University of Calabria, by the Institute of High-Performance Computing and Networking (ICAR) of the Italian National Research Council (CNR), by the University of Maryland Institute for Advanced Computer Studies (UMIACS), and by the Spanish Consolider project *Agreement Technologies* (CSD2007-022).

Table of Contents

PrDB: Managing Large-Scale Correlated Probabilistic Databases (Abstract)

Amol Deshpande

University of Maryland, College Park, MD, USA
amol@cs.umd.edu

Increasing numbers of real-world application domains are generating data that is inherently noisy, incomplete, and probabilistic in nature. Statistical inference and probabilistic modeling often introduce another layer of uncertainty on top of that. Examples of such data include measurement data collected by sensor networks, observation data in the context of social networks, scientific and biomedical data, and data collected by various online cyber-sources. Over the last few years, numerous approaches have been proposed, and several systems built, to integrate uncertainty into databases. However, these approaches typically make simplistic and restrictive assumptions concerning the types of uncertainties that can be represented. Most importantly, they often make highly restrictive independence assumptions, and cannot easily model rich correlations among the tuples or attribute values. Furthermore, they typically lack support for specifying uncertainties at different levels of abstractions, needed to handle large-scale uncertain datasets.

In this talk, I will begin by presenting our work on building a probabilistic data management system, called PrDB, aimed at supporting rich correlation structures often present in real-world uncertain datasets. I will present the PrDB representation model, which is based on *probabilistic graphical models*, and its key abstractions, and show how these enable PrDB to support uncertainties specified at various abstraction levels, from *schema-level* uncertainties that apply to entire relations to *tuple-specific* uncertainties that apply only to a specific tuple or a specific set of tuples. Query evaluation in PrDB can be seen as equivalent to *inference* in graphical models, and I will present some of the key novel techniques that we have developed to efficiently evaluate various types of queries over large-scale probabilistic databases. I will then briefly discuss our ongoing work and some of the open research challenges in this area.

More details about the PrDB system and its key technologies can be found in [1,2,3].

References

1. Deshpande, A., Getoor, L., Sen, P.: Graphical Models for Uncertain Data. In: Aggarwal, C. (ed.) Managing and Mining Uncertain Data. Springer, Heidelberg (2008)
2. Kanagal, B., Deshpande, A.: Indexing Correlated Probabilistic Databases. In: SIGMOD (2009)
3. Sen, P., Deshpande, A., Getoor, L.: PrDB: Managing and Exploiting Rich Correlations in Probabilistic Databases. The International Journal on Very Large Data Bases (2009)

L. Godo and A. Pugliese (Eds.): SUM 2009, LNAI 5785, p. 1, 2009.

Uncertainty in the Semantic Web

Thomas Lukasiewicz*

Computing Laboratory, University of Oxford
Wolfson Building, Parks Road, Oxford OX1 3QD, UK
thomas.lukasiewicz@comlab.ox.ac.uk

Abstract. During the recent decade, significant research activities have been directed towards the Semantic Web as future substitute of the Web. Many experts predict that the next huge step forward in Web information technology will be achieved by adding semantics to Web data. An important role in research towards the Semantic Web is played by formalisms and technologies for handling uncertainty and/or vagueness. In this paper, I give an overview of some own recent formalisms for handling uncertainty and/or vagueness in the Semantic Web.

1 Introduction

The *Semantic Web* [1,2,3,4] has recently attracted much attention, both from academia and industry, and is widely regarded as the next step in the evolution of the World Wide Web. It aims at an extension of the current Web by standards and technologies that help machines to understand the information on the Web so that they can support richer discovery, data integration, navigation, and automation of tasks. The main ideas behind it are to add a machine-understandable "meaning" to Web pages, to use ontologies for a precise definition of shared terms in Web resources, to use KR technology for automated reasoning from Web resources, and to apply cooperative agent technology for processing the information of the Web.

The Semantic Web consists of several hierarchical layers, where the *Ontology layer*, in form of the *OWL Web Ontology Language* [5], is currently the highest layer of sufficient maturity. OWL consists of three increasingly expressive sublanguages, namely, *OWL Lite*, *OWL DL*, and *OWL Full*. OWL Lite and OWL DL are essentially very expressive description logics with an RDF syntax. As shown in [6], ontology entailment in OWL Lite (resp., OWL DL) reduces to knowledge base (un)satisfiability in the description logic $\mathcal{SHIF}(\mathbf{D})$ (resp., $\mathcal{SHOIN}(\mathbf{D})$). On top of the Ontology layer, sophisticated representation and reasoning capabilities for the *Rules*, *Logic*, and *Proof layers* of the Semantic Web are currently being developed next.

In particular, a key requirement of the layered architecture of the Semantic Web is to integrate the Rules and the Ontology layer. Here, it is crucial to allow for building rules on top of ontologies, that is, for rule-based systems that use vocabulary from ontology knowledge bases. Another type of combination is to build ontologies on top of rules, where ontological definitions are supplemented by rules or imported from rules.

* Alternative address: Institut für Informationssysteme, Technische Universität Wien, Favoritenstraße 9-11, 1040 Wien, Austria, lukasiewicz@kr.tuwien.ac.at

L. Godo and A. Pugliese (Eds.): SUM 2009, LNAI 5785, pp. 2–11, 2009.

Both types of integration have been realized in recent hybrid integrations of rules and ontologies, called *description logic programs* (or *dl-programs*), which are of the form $KB = (L, P)$, where L is a description logic knowledge base, and P is a finite set of rules involving either queries to L in a loose integration (see, e.g., [7,8]) or concepts and roles from L as unary resp. binary predicates in a tight integration (see, e.g., [9]).

However, classical ontology languages and description logics as well as formalisms integrating rules and ontologies are less suitable in all those domains where the information to be represented comes along with (*quantitative*) *uncertainty* and/or *vagueness* (or *imprecision*). For this reason, during the recent years, handling uncertainty and vagueness has started to play an important role in research towards the Semantic Web. A recent forum for approaches to uncertainty reasoning in the Semantic Web is the annual *International Workshop on Uncertainty Reasoning for the Semantic Web (URSW)* at the *International Semantic Web Conference (ISWC)*. There has also been a W3C Incubator Group on *Uncertainty Reasoning for the World Wide Web*. The research focuses especially on probabilistic and fuzzy extensions of description logics, ontology languages, and formalisms integrating rules and ontologies. Note that probabilistic formalisms allow to encode ambiguous information, such as "John is a student with the probability 0.7 and a teacher with the probability 0.3" (roughly, John is either a teacher or a student, but more likely a student), while fuzzy approaches allow to encode vague or imprecise information, such as "John is tall with the degree of truth 0.7" (roughly, John is quite tall). Formalisms for dealing with uncertainty and vagueness are especially applied in ontology mapping, data integration, information retrieval, and database querying. For example, some of the most prominent technologies for dealing with uncertainty are probably the ranking algorithms standing behind Web search engines. Vagueness and imprecision also abound in multimedia information processing and retrieval, and are an important aspect of natural language interfaces to the Web.

In this paper, I give an overview of some own recent extensions of description logics and description logic programs by probabilistic uncertainty and fuzzy vagueness. The rest of this paper is organized as follows. In Section 2, I describe an approach to probabilistic description logics for the Semantic Web. Sections 3 and 4 focus on approaches to probabilistic and fuzzy description logic programs for the Semantic Web, respectively, while Section 5 describes an approach to description logic programs for handling both uncertainty and vagueness in a uniform framework for the Semantic Web. For a more detailed overview of extensions of description logics for handling uncertainty and vagueness in the Semantic Web, I also refer the reader to the recent survey [10].

2 Probabilistic Description Logics

In this section, we briefly describe the probabilistic description logic P-$\mathcal{SHOIN}(\mathbf{D})$, which is a probabilistic generalization of the description logic $\mathcal{SHOIN}(\mathbf{D})$ behind OWL DL towards sophisticated formalisms for reasoning under probabilistic uncertainty in the Semantic Web [11]. Closely related probabilistic generalizations of the *DL-Lite* family of tractable description logics (which lies between the Semantic Web languages RDFS and OWL Lite) and the description logics $\mathcal{SHIF}(\mathbf{D})$ and $\mathcal{SHOQ}(\mathbf{D})$ (which stand behind OWL Lite and DAML+OIL, respectively) have been introduced in [11,12]. A companion paper [13] combines *DL-Lite* with Bayesian networks.

The syntax of P-$\mathcal{SHOIN}(\mathbf{D})$ uses the notion of a conditional constraint from [14] to express probabilistic knowledge in addition to the axioms of $\mathcal{SHOIN}(\mathbf{D})$. Its semantics is based on the notion of lexicographic entailment in probabilistic default reasoning [15,16], which is a probabilistic generalization of the sophisticated notion of lexicographic entailment by Lehmann [17] in default reasoning from conditional knowledge bases. Due to this semantics, P-$\mathcal{SHOIN}(\mathbf{D})$ allows for expressing both terminological probabilistic knowledge about concepts and roles, and also assertional probabilistic knowledge about instances of concepts and roles. It naturally interprets terminological and assertional probabilistic knowledge as statistical knowledge about concepts and roles, and as degrees of belief about instances of concepts and roles, respectively, and allows for deriving both statistical knowledge and degrees of belief. As an important additional feature, it also allows for expressing default knowledge about concepts (as a special case of terminological probabilistic knowledge), which is semantically interpreted as in Lehmann's lexicographic default entailment [17].

Example 2.1. Suppose a classical description logic knowledge base T is used to encode knowledge about cars and their properties (e.g., that sports cars and roadsters are cars). A probabilistic knowledge base $KB = (T, P, (P_o)_{o \in \mathbf{I}_P})$ in P-$\mathcal{SHOIN}(\mathbf{D})$ then extends T by terminological default and terminological probabilistic knowledge in P as well as by assertional probabilistic knowledge in P_o for certain objects $o \in \mathbf{I}_P$. For example, the terminological default knowledge (1) "generally, cars do not have a red color" and (2) "generally, sports cars have a red color", and the terminological probabilistic knowledge (3) "cars have four wheels with a probability of at least 0.9", can be expressed by the following conditional constraints in P:

$$\begin{array}{ll} (1) & (\neg\exists\, \mathit{HasColor}.\{\mathit{red}\} \mid \mathit{Car})[1,1], \\ (2) & (\exists\, \mathit{HasColor}.\{\mathit{red}\} \mid \mathit{SportsCar})[1,1], \\ (3) & (\mathit{HasFourWheels} \mid \mathit{Car})[0.9,1]\,. \end{array}$$

Suppose we want to encode some probabilistic information about John's car (which we have not seen so far). Then, the set of probabilistic individuals $\mathbf{I}_P$ contains the individual *John's car*, and the assertional probabilistic knowledge (4) "John's car is a sports car with a probability of at least 0.8" (we know that John likes sports cars) can be expressed by the following conditional constraint in $P_{\mathit{John's\ car}}$:

$$(4)\ \ (\mathit{SportsCar} \mid \top)[0.8,1]\,.$$

Then, the following are some (terminological default and terminological probabilistic) tight lexicographic consequences of $PT = (T, P)$:

$$\begin{array}{l} (\neg\exists\, \mathit{HasColor}.\{\mathit{red}\} \mid \mathit{Car})[1,1], \\ (\exists\, \mathit{HasColor}.\{\mathit{red}\} \mid \mathit{SportsCar})[1,1], \\ (\mathit{HasFourWheels} \mid \mathit{Car})[0.9,1], \\ (\neg\exists\, \mathit{HasColor}.\{\mathit{red}\} \mid \mathit{Roadster})[1,1], \\ (\mathit{HasFourWheels} \mid \mathit{SportsCar})[0.9,1], \\ (\mathit{HasFourWheels} \mid \mathit{Roadster})[0.9,1]\,. \end{array}$$

Hence, in addition to the sentences (1) to (3) directly encoded in P, we also conclude "generally, roadsters do not have a red color", "sports cars have four wheels with a

probability of at least 0.9", and "roadsters have four wheels with a probability of at least 0.9". Observe here that the default property of not having a red color and the probabilistic property of having four wheels with a probability of at least 0.9 are inherited from cars down to roadsters. Roughly, the tight lexicographic consequences of $PT = (T, P)$ are given by all those conditional constraints that (a) are either in P, or (b) can be constructed by inheritance along subconcept relationships from the ones in P and are not overridden by more specific pieces of knowledge in P.

The following conditional constraints for the probabilistic individual *John's car* are some (assertional probabilistic) tight lexicographic consequences of KB, which informally say that John's car is a sports car, has a red color, and has four wheels with probabilities of at least 0.8, 0.8, and 0.72, respectively:

$$(\mathit{SportsCar} \mid \top)[0.8, 1],$$
$$(\exists\, \mathit{HasColor}.\{\mathit{red}\} \mid \top)[0.8, 1],$$
$$(\mathit{HasFourWheels} \mid \top)[0.72, 1]\,.$$

3 Probabilistic Description Logic Programs

We now summarize the main ideas behind loosely and tightly coupled probabilistic dl-programs, introduced in [18,19,20,21] and [22,23,24,25,26], respectively. For further details on the syntax and semantics of these programs, their background, and their semantic and computational properties, we refer to the above works.

Loosely coupled probabilistic dl-programs [18,19,20] are a combination of loosely coupled dl-programs under the answer set and the well-founded semantics with probabilistic uncertainty as in Bayesian networks. Roughly, they consist of a loosely coupled dl-program (L, P) under different "total choices" B (they are the full joint instantiations of a set of random variables, and they serve as pairwise exclusive and exhaustive possible worlds), and a probability distribution μ over the set of total choices B. One then obtains a probability distribution over Herbrand models, since every total choice B along with the loosely coupled dl-program produces a set of Herbrand models of which the probabilities sum up to $\mu(B)$. As in the classical case, the answer set semantics of loosely coupled probabilistic dl-programs is a refinement of the well-founded semantics of loosely coupled probabilistic dl-programs. Consistency checking and tight query processing (i.e., computing the entailed tight interval for the probability of a conditional or unconditional event) in such probabilistic dl-programs under the answer set semantics can be reduced to consistency checking and query processing in loosely coupled dl-programs under the answer set semantics, while tight query processing under the well-founded semantics can be done in an anytime fashion by reduction to loosely coupled dl-programs under the well-founded semantics. For suitably restricted description logic components, the latter can be done in polynomial time in the data complexity. Query processing for stratified loosely coupled probabilistic dl-programs can be reduced to computing the canonical model of stratified loosely coupled dl-programs. Loosely coupled probabilistic dl-programs can especially be used for (database-oriented) probabilistic data integration in the Semantic Web, where probabilistic uncertainty is used to handle inconsistencies between different data sources [21].

Example 3.1. A university database may use a loosely coupled dl-program (L, P) to encode ontological and rule-based knowledge about students and exams. A probabilistic dl-program $KB = (L, P', C, \mu)$ then additionally allows for encoding probabilistic knowledge. For example, the following two probabilistic rules in P' along with a probability distribution on a set of random variables may express that if two master (resp., bachelor) students have given the same exam, then there is a probability of 0.9 (resp., 0.7) that they are friends:

$$\begin{aligned} \mathit{friends}(X,Y) \leftarrow\ & \mathit{given\_same\_exam}(X,Y), DL[\mathit{master\_student}(X)], \\ & DL[\mathit{master\_student}(Y)],\ \mathit{choice}_m\,; \\ \mathit{friends}(X,Y) \leftarrow\ & \mathit{given\_same\_exam}(X,Y), DL[\mathit{bachelor\_student}(X)], \\ & DL[\mathit{bachelor\_student}(Y)],\ \mathit{choice}_b\,. \end{aligned}$$

Here, we assume the set $C = \{V_m, V_b\}$ of value sets $V_m = \{\mathit{choice}_m, \mathit{not\_choice}_m\}$ and $V_b = \{\mathit{choice}_b, \mathit{not\_choice}_b\}$ of two random variables X_m resp. X_b and the probability distribution μ on all their joint instantiations, given by $\mu\colon\ \mathit{choice}_m, \mathit{not\_choice}_m, \mathit{choice}_b, \mathit{not\_choice}_b \mapsto 0.9, 0.1, 0.7, 0.3$ under probabilistic independence. For example, the joint instantiation $\mathit{choice}_m, \mathit{choice}_b$ is associated with the probability $0.9 \times 0.7 = 0.63$. Asking about the entailed tight interval for the probability that john and bill are friends can then be expressed by a probabilistic query $\exists(\mathit{friends}(\mathit{john}, \mathit{bill}))[R, S]$, whose answer depends on the available concrete knowledge about john and bill (namely, whether they have given the same exams, and are both master or bachelor students).

Tightly coupled probabilistic dl-programs [22,23] are a tight combination of disjunctive logic programs under the answer set semantics with description logics and Bayesian probabilities. They are a logic-based representation formalism that naturally fits into the landscape of Semantic Web languages. Tightly coupled probabilistic dl-programs can especially be used for representing mappings between ontologies [24,25], which are a common way of approaching the semantic heterogeneity problem on the Semantic Web. Here, they allow in particular for resolving inconsistencies and for merging mappings from different matchers based on the level of confidence assigned to different rules (see below). Furthermore, tightly coupled probabilistic description logic programs also provide a natural integration of ontologies, action languages, and Bayesian probabilities towards Web Services. Consistency checking and query processing in tightly coupled probabilistic dl-programs can be reduced to consistency checking and cautious/brave reasoning, respectively, in tightly coupled disjunctive dl-programs. Under certain restrictions, these problems have a polynomial data complexity.

Example 3.2. The two correspondences between two ontologies O_1 and O_2 that (i) an element of $\mathit{Collection}$ in O_1 is an element of Book in O_2 with the probability 0.62, and (ii) an element of $\mathit{Proceedings}$ in O_1 is an element of $\mathit{Proceedings}$ in O_2 with the probability 0.73 (found by the matching system hmatch) can be expressed by the following two probabilistic rules:

$$\begin{aligned} & O_2 : \mathit{Book}(X) \leftarrow O_1 : \mathit{Collection}(X) \wedge \mathit{hmatch}_1; \\ & O_2 : \mathit{Proceedings}(X) \leftarrow O_1 : \mathit{Proceedings}(X) \wedge \mathit{hmatch}_2. \end{aligned}$$

Here, we assume the set $\mathcal{C} = \{\{hmatch_i, not\_hmatch_i\} \mid i \in \{1,2\}\}$ of values of two random variables and the probability distribution μ on all joint instantiations of these variables, given by $\mu\colon hmatch_1, not\_hmatch_1, hmatch_2, not\_hmatch_2 \mapsto 0.62, 0.38, 0.73, 0.27$ under probabilistic independence.

Similarly, two other correspondences between O_1 and O_2 (found by the matching system falcon) are expressed by the following two probabilistic rules:

$$O_2 : InCollection(X) \leftarrow O_1 : Collection(X) \wedge falcon_1;$$
$$O_2 : Proceedings(X) \leftarrow O_1 : Proceedings(X) \wedge falcon_2,$$

where we assume the set $\mathcal{C}' = \{\{falcon_i, not\_falcon_i\} \mid i \in \{1,2\}\}$ of values of two random variables and the probability distribution μ' on all joint instantiations of these variables, given by $\mu'\colon falcon_1, not\_falcon_1, falcon_2, not\_falcon_2 \mapsto 0.94, 0.06, 0.96, 0.04$ under probabilistic independence.

Using the trust probabilities 0.55 and 0.45 for hmatch and falcon, respectively, for resolving inconsistencies between rules, we can now define a merged mapping set that consists of the following probabilistic rules:

$$O_2 : Book(X) \leftarrow O_1 : Collection(X) \wedge hmatch_1 \wedge sel\_hmatch_1;$$
$$O_2 : InCollection(X) \leftarrow O_1 : Collection(X) \wedge falcon_1 \wedge sel\_falcon_1;$$
$$O_2 : Proceedings(X) \leftarrow O_1 : Proceedings(X) \wedge hmatch_2;$$
$$O_2 : Proceedings(X) \leftarrow O_1 : Proceedings(X) \wedge falcon_2.$$

Here, we assume the set $\mathcal{C}''$ of values of random variables and the probability distribution μ'' on all joint instantiations of these variables, which are obtained from $\mathcal{C} \cup \mathcal{C}'$ and $\mu \cdot \mu'$ (defined as $(\mu \cdot \mu')(B\,B') = \mu(B) \cdot \mu'(B')$, for all joint instantiations B of $\mathcal{C}$ and B' of $\mathcal{C}'$), respectively, by adding the values $\{sel\_hmatch_1, sel\_falcon_1\}$ of a new random variable, with the probabilities $sel\_hmatch_1, sel\_falcon_1 \mapsto 0.55, 0.45$ under probabilistic independence, for resolving the inconsistency between the first two rules.

A companion approach to probabilistic description logic programs [26] combines probabilistic logic programs, probabilistic default theories, and the description logics behind OWL Lite and OWL DL. It is based on new notions of entailment for reasoning with conditional constraints, which realize the principle of inheritance with overriding for both classical and purely probabilistic knowledge. They are obtained by generalizing previous formalisms for probabilistic default reasoning with conditional constraints (similarly as for P-$\mathcal{SHOIN}(\mathbf{D})$ in Section 2). In addition to dealing with probabilistic knowledge, these notions of entailment thus also allow for handling default knowledge.

4 Fuzzy Description Logic Programs

We next briefly describe loosely and tightly coupled fuzzy dl-programs, which have been introduced in [27,28] and [29,30], respectively, and extended by a top-k retrieval technique in [33]. All these fuzzy dl-programs have natural special cases where query processing can be done in polynomial time in the data complexity. For further details on their syntax and semantics, background, and properties, we refer to the above works.

Towards dealing with vagueness and imprecision in the reasoning layers of the Semantic Web, loosely coupled (normal) fuzzy dl-programs under the answer set semantics [27,28] generalize normal dl-programs under the answer set semantics by fuzzy vagueness and imprecision in both the description logic and the logic program component. This is the first approach to fuzzy dl-programs that may contain default negations in rule bodies. Query processing in such fuzzy dl-programs can be done by reduction to normal dl-programs under the answer set semantics. In the special cases of positive and stratified loosely coupled fuzzy dl-programs, the answer set semantics coincides with a canonical least model and an iterative least model semantics, respectively, and has a characterization in terms of a fixpoint and an iterative fixpoint semantics, respectively.

Example 4.1. Consider the fuzzy description logic knowledge base L of a car shopping Web site, which defines especially (i) the fuzzy concepts of sports cars ($SportsCar$), "at most 22 000 €" ($LeqAbout22000$), and "around 150 horse power" ($Around150HP$), (ii) the attributes of the price and of the horse power of a car ($hasInvoice$ resp. $hasHP$), and (iii) the properties of some concrete cars (such as a $MazdaMX5Miata$ and a $MitsubishiES$). Then, a loosely coupled fuzzy dl-program $KB = (L, P)$ is given by the set of fuzzy dl-rules P, which contains only the following fuzzy dl-rule encoding the request of a buyer (asking for a sports car costing at most 22 000 € and having around 150 horse power), where $\otimes$ may be the conjunction strategy of, e.g., Gödel Logic (that is, $x \otimes y = \min(x, y)$, for all $x, y \in [0, 1]$, is used to evaluate $\wedge$ and $\leftarrow$ on truth values):

$$query(x) \leftarrow_{\otimes} DL[SportsCar](x) \wedge_{\otimes} DL[\exists hasInvoice.LeqAbout22000](x) \wedge_{\otimes} DL[\exists hasHP.Around150HP](x) \geqslant 1\,.$$

The above fuzzy dl-program $KB = (L, P)$ is positive, and has a minimal model M_{KB}, which defines the degree to which some concrete cars in the description logic knowledge base L match the buyer's request, for example,

$$M_{KB}(query(MazdaMX5Miata)) = 0.36\,, \quad M_{KB}(query(MitsubishiES)) = 0.32\,.$$

That is, the car $MazdaMX5Miata$ is ranked top with the degree 0.36, while the car $MitsubishiES$ is ranked second with the degree 0.32.

Tightly coupled fuzzy dl-programs under the answer set semantics [29,30] are a tight integration of fuzzy disjunctive logic programs under the answer set semantics with fuzzy description logics. They are also a generalization of tightly coupled disjunctive dl-programs by fuzzy vagueness in both the description logic and the logic program component. This is the first approach to fuzzy dl-programs that may contain disjunctions in rule heads. Query processing in such programs can essentially be done by a reduction to tightly coupled disjunctive dl-programs. A closely related work [33] explores the evaluation of ranked top-k queries. It shows in particular how to compute the top-k answers in data-complexity tractable tightly coupled fuzzy dl-programs.

Example 4.2. A tightly coupled fuzzy dl-program $KB = (L, P)$ is given by a suitable fuzzy description logic knowledge base L and the set of fuzzy rules P, which contains only the following fuzzy rule (where $x \otimes y = \min(x, y)$):

$$query(x) \leftarrow_{\otimes} SportyCar(x) \wedge_{\otimes} hasInvoice(x, y_1) \wedge_{\otimes} hasHorsePower(x, y_2) \wedge_{\otimes} LeqAbout22000(y_1) \wedge_{\otimes} Around150(y_2) \geqslant 1\,.$$

Informally, *query* collects all sports cars, and ranks them according to whether they cost at most around 22 000 € and have around 150 HP. Another fuzzy rule involving also a negation in its body and a disjunction in its head is given as follows (where $\ominus x = 1 - x$ and $x \oplus y = \max(x, y)$):

$$Small(x) \vee_{\oplus} Old(x) \leftarrow_{\otimes} Car(x) \wedge_{\otimes} hasInvoice(x, y) \wedge_{\otimes} not_{\ominus} GeqAbout15000(y) \geqslant 0.7\,.$$

This rule says that a car costing at most around 15 000 € is either small or old. Notice here that $Small$ and Old may be two concepts in the fuzzy description logic knowledge base L. That is, the tightly coupled approach to fuzzy dl-programs under the answer set semantics also allows for using the rules in P to express relationships between the concepts and roles in L. This is not possible in the loosely coupled approach to fuzzy dl-programs under the answer set semantics in [27,28], since the dl-queries there can only occur in rule bodies, but not in rule heads.

5 Probabilistic Fuzzy Description Logic Programs

We finally describe (loosely coupled) probabilistic fuzzy dl-programs [31,32], which combine fuzzy description logics, fuzzy logic programs (with stratified default-negation), and probabilistic uncertainty in a uniform framework for the Semantic Web. Intuitively, they allow for defining several rankings on ground atoms using fuzzy vagueness, and then for merging these rankings using probabilistic uncertainty (by associating with each ranking a probabilistic weight and building the weighted sum of all rankings). Such programs also give rise to important concepts dealing with both probabilistic uncertainty and fuzzy vagueness, such as the expected truth value of a crisp sentence and the probability of a vague sentence.

Example 5.1. A (loosely coupled) probabilistic fuzzy dl-program is given by a suitable fuzzy description logic knowledge base L and the following set of fuzzy dl-rules P, modeling some query reformulation / retrieval steps using ontology mapping rules:

$$query(x) \leftarrow_{\otimes} SportyCar(x) \wedge_{\otimes} hasPrice(x, y_1) \wedge_{\otimes} hasPower(x, y_2) \wedge_{\otimes} DL[LeqAbout22000](y_1) \wedge_{\otimes} DL[Around150HP](y_2) \geqslant 1\,, \quad (1)$$

$$SportyCar(x) \leftarrow_{\otimes} DL[SportsCar](x) \wedge_{\otimes} sc_{pos} \geqslant 0.9\,, \quad (2)$$

$$hasPrice(x, y) \leftarrow_{\otimes} DL[hasInvoice](x, y) \wedge_{\otimes} hi_{pos} \geqslant 0.8\,, \quad (3)$$

$$hasPower(x, y) \leftarrow_{\otimes} DL[hasHP](x, y) \wedge_{\otimes} hhp_{pos} \geqslant 0.8\,, \quad (4)$$

where we assume the set $C = \{\{sc_{pos}, sc_{neg}\}, \{hi_{pos}, hi_{neg}\}, \{hhp_{pos}, hhp_{neg}\}\}$ of values of random variables and the probability distribution μ on all joint instantiations of these variables, given by $\mu\colon sc_{pos}, sc_{neg}, hi_{pos}, hi_{neg}, hhp_{pos}, hhp_{neg} \mapsto 0.91, 0.09, 0.78, 0.22, 0.83, 0.17$ under probabilistic independence. Here, rule (1) is the buyer's request, but in a "different" terminology than the one of the car selling site. Rules (2)–(4) are so-called ontology alignment mapping rules. For example, rule (2) states that the predicate "SportyCar" of the buyer's terminology refers to the concept "SportsCar" of the selected site with probability 0.91.

The following may be some tight consequences of the above probabilistic fuzzy dl-program (where for ground atoms q, we use $(\mathbf{E}[q])[L,U]$ to denote that the expected truth value of q lies in the interval $[L,U]$):

$$(\mathbf{E}[query(MazdaMX5Miata)])[0.21,0.21],\ (\mathbf{E}[query(MitsubishiES)])[0.19,0.19]\ .$$

That is, the car $MazdaMX5Miata$ is ranked first with the degree 0.21, while the car $MitsubishiES$ is ranked second with the degree 0.19.

Acknowledgments. This work has been supported by the German Research Foundation (DFG) under the Heisenberg Programme.

References

1. Berners-Lee, T.: Weaving the Web. Harper, San Francisco (1999)
2. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Sci. Amer. 284(5), 34–43 (2001)
3. Fensel, D., Wahlster, W., Lieberman, H., Hendler, J. (eds.): Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential. MIT Press, Cambridge (2002)
4. Horrocks, I., Patel-Schneider, P.F., van Harmelen, F.: From $\mathcal{SHIQ}$ and RDF to OWL: The making of a web ontology language. J. Web Sem. 1(1), 7–26 (2003)
5. W3C: OWL web ontology language overview (2004) W3C Recommendation (February 10, 2004), `http://www.w3.org/TR/2004/REC-owl-features-20040210/`
6. Horrocks, I., Patel-Schneider, P.F.: Reducing OWL entailment to description logic satisfiability. In: Fensel, D., Sycara, K., Mylopoulos, J. (eds.) ISWC 2003. LNCS, vol. 2870, pp. 17–29. Springer, Heidelberg (2003)
7. Eiter, T., Ianni, G., Lukasiewicz, T., Schindlauer, R., Tompits, H.: Combining answer set programming with description logics for the Semantic Web. Artif. Intell. 172(12/13), 1495–1539 (2008)
8. Eiter, T., Lukasiewicz, T., Schindlauer, R., Tompits, H.: Well-founded semantics for description logic programs in the Semantic Web. In: Antoniou, G., Boley, H. (eds.) RuleML 2004. LNCS, vol. 3323, pp. 81–97. Springer, Heidelberg (2004)
9. Lukasiewicz, T.: A novel combination of answer set programming with description logics for the Semantic Web. In: Franconi, E., Kifer, M., May, W. (eds.) ESWC 2007. LNCS, vol. 4519, pp. 384–398. Springer, Heidelberg (2007)
10. Lukasiewicz, T., Straccia, U.: Managing uncertainty and vagueness in description logics for the Semantic Web. J. Web Sem. 6(4), 291–308 (2008)
11. Lukasiewicz, T.: Expressive probabilistic description logics. Artif. Intell. 172(6/7), 852–883 (2008)
12. Giugno, R., Lukasiewicz, T.: P-$\mathcal{SHOQ}(\mathbf{D})$: A probabilistic extension of $\mathcal{SHOQ}(\mathbf{D})$ for probabilistic ontologies in the Semantic web. In: Flesca, S., Greco, S., Leone, N., Ianni, G. (eds.) JELIA 2002. LNCS (LNAI), vol. 2424, pp. 86–97. Springer, Heidelberg (2002)
13. d'Amato, C., Fanizzi, N., Lukasiewicz, T.: Tractable reasoning with Bayesian description logics. In: Greco, S., Lukasiewicz, T. (eds.) SUM 2008. LNCS (LNAI), vol. 5291, pp. 146–159. Springer, Heidelberg (2008)
14. Lukasiewicz, T.: Probabilistic deduction with conditional constraints over basic events. J. Artif. Intell. Res. 10, 199–241 (1999)
15. Lukasiewicz, T.: Probabilistic logic programming under inheritance with overriding. In: Proceedings UAI-2001, pp. 329–336. Morgan Kaufmann, San Francisco (2001)

16. Lukasiewicz, T.: Probabilistic default reasoning with conditional constraints. Ann. Math. Artif. Intell. 34(1–3), 35–88 (2002)
17. Lehmann, D.: Another perspective on default reasoning. Ann. Math. Artif. Intell. 15(1), 61–82 (1995)
18. Lukasiewicz, T.: Probabilistic description logic programs. In: Godo, L. (ed.) ECSQARU 2005. LNCS (LNAI), vol. 3571, pp. 737–749. Springer, Heidelberg (2005)
19. Lukasiewicz, T.: Probabilistic description logic programs. Int. J. Approx. Reasoning 45(2), 288–307 (2007)
20. Lukasiewicz, T.: Tractable probabilistic description logic programs. In: Prade, H., Subrahmanian, V.S. (eds.) SUM 2007. LNCS (LNAI), vol. 4772, pp. 143–156. Springer, Heidelberg (2007)
21. Calì, A., Lukasiewicz, T.: An approach to probabilistic data integration for the Semantic web. In: da Costa, P.C.G., d'Amato, C., Fanizzi, N., Laskey, K.B., Laskey, K.J., Lukasiewicz, T., Nickles, M., Pool, M. (eds.) URSW 2005 - 2007. LNCS (LNAI), vol. 5327, pp. 52–65. Springer, Heidelberg (2008)
22. Calì, A., Lukasiewicz, T.: Tightly integrated probabilistic description logic programs for the Semantic web. In: Dahl, V., Niemelä, I. (eds.) ICLP 2007. LNCS, vol. 4670, pp. 428–429. Springer, Heidelberg (2007)
23. Calì, A., Lukasiewicz, T., Predoiu, L., Stuckenschmidt, H.: Tightly coupled probabilistic description logic programs for the Semantic Web. J. Data Sem. 12, 95–130 (2009)
24. Calì, A., Lukasiewicz, T., Predoiu, L., Stuckenschmidt, H.: Rule-based approaches for representing probabilistic ontology mappings. In: da Costa, P.C.G., d'Amato, C., Fanizzi, N., Laskey, K.B., Laskey, K.J., Lukasiewicz, T., Nickles, M., Pool, M. (eds.) URSW 2005 - 2007. LNCS (LNAI), vol. 5327, pp. 66–87. Springer, Heidelberg (2008)
25. Calì, A., Lukasiewicz, T., Predoiu, L., Stuckenschmidt, H.: Tightly integrated probabilistic description logic programs for representing ontology mappings. In: Hartmann, S., Kern-Isberner, G. (eds.) FoIKS 2008. LNCS, vol. 4932, pp. 178–198. Springer, Heidelberg (2008)
26. Lukasiewicz, T.: Probabilistic description logic programs under inheritance with overriding for the Semantic Web. Int. J. Approx. Reasoning 49(1), 18–34 (2008)
27. Lukasiewicz, T.: Fuzzy description logic programs under the answer set semantics for the Semantic Web. In: Proceedings RuleML-2006, pp. 89–96. IEEE Computer Society, Los Alamitos (2006)
28. Lukasiewicz, T.: Fuzzy description logic programs under the answer set semantics for the Semantic Web. Fundam. Inform. 82(3), 289–310 (2008)
29. Lukasiewicz, T., Straccia, U.: Tightly integrated fuzzy description logic programs under the answer set semantics for the Semantic web. In: Marchiori, M., Pan, J.Z., Marie, C.d.S. (eds.) RR 2007. LNCS, vol. 4524, pp. 289–298. Springer, Heidelberg (2007)
30. Lukasiewicz, T., Straccia, U.: Tightly coupled fuzzy description logic programs under the answer set semantics for the Semantic Web. Int. J. Semantic Web Inf. Syst. 4(3), 68–89 (2008)
31. Lukasiewicz, T., Straccia, U.: Description logic programs under probabilistic uncertainty and fuzzy vagueness. In: Mellouli, K. (ed.) ECSQARU 2007. LNCS (LNAI), vol. 4724, pp. 187–198. Springer, Heidelberg (2007)
32. Lukasiewicz, T., Straccia, U.: Description logic programs under probabilistic uncertainty and fuzzy vagueness. Int. J. Approx. Reasoning 50(6), 837–853 (2009)
33. Lukasiewicz, T., Straccia, U.: Top-k retrieval in description logic programs under vagueness for the Semantic Web. In: Prade, H., Subrahmanian, V.S. (eds.) SUM 2007. LNCS (LNAI), vol. 4772, pp. 16–30. Springer, Heidelberg (2007)

Bridging the Gap between Abstract Argumentation Systems and Logic

Leila Amgoud and Philippe Besnard

IRIT-CNRS, 118, route de Narbonne,
31062 Toulouse Cedex 4 France
{amgoud,besnard}@irit.fr

Abstract. Dung's argumentation system takes as input a set of *arguments* and a *binary relation* encoding attacks among these arguments, and returns different *extensions* of arguments. However, no indication is given on how to instantiate this setting, i.e. how to build arguments from a *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 formulas forming an extension is inconsistent). This is due to the gap between the abstract setting and the knowledge base from which it is defined.

The purpose of this paper is twofold: First it proposes to fill in this gap by extending Dung's system. The idea is to consider all the ingredients involved in an argumentation problem. We start with an abstract monotonic logic which consists of a set of formulas and a consequence operator. We show how to build arguments from a knowledge base using the consequence operator of the logic. Second, we show that the choice of an attack relation is crucial for ensuring consistent results, and should not be arbitrary. In particular, we argue that an attack relation should be at least grounded on the *minimal conflicts* contained in the knowledge base. Moreover, due to the binary character of this relation, some attack relations may lead to unintended results. Namely, symmetric relations are not suitable when ternary (or more) minimal conflicts are in the knowledge base. We propose then the characteristics of attack relations that ensure sound results.

1 Introduction

Argumentation is a reasoning model based on the construction and evaluation of arguments in order to increase or decrease the acceptability of a given standpoint. It is used, for instance, for handling inconsistency in knowledge bases (e.g. [2,9]) and for decision making (e.g. [1,3]).

One of the most abstract argumentation systems in existing literature is Dung's one. It consists of a set of arguments and a binary relation encoding attacks among these arguments. Since its original formulation, this system has become very popular and different instantiations of it have been defined. Unfortunately, some of them such as the one presented in [8] can lead to very unintuitive results. In [4], it has been shown that this instantiation violates key postulates like the consistency of extensions. An extension satisfies consistency if the set of formulas used in arguments of that extension is consistent. What is worth noticing is that this postulate refers to the internal structure

L. Godo and A. Pugliese (Eds.): SUM 2009, LNAI 5785, pp. 12–27, 2009.

of an argument (i.e. its formulas) while the link between these formulas and the acceptability semantics is not clear. The link between the inconsistency of a base and the attack relation is also not clear. In summary, there is a gap between the abstract setting and the knowledge base from which it is built. Thus, basic choices like the definition of an argument and of an attack relation are made in an ad hoc way.

The purpose of this paper is twofold: First it proposes to fill in this gap by extending Dung's system. The idea is to consider all the ingredients involved in an argumentation problem ranging from the logical language to the output of the system. We start with an *abstract monotonic logic* as defined in [10]. Tarski defines an abstract monotonic logic as a set of formulas and a consequence operator that satisfies some axiom. We show how to build arguments from any subset of formulas using the consequence operator. Second, we show that the choice of an attack relation is crucial for ensuring consistent results and should not be arbitrary. In particular, an attack relation should be at least grounded on the minimal conflicts contained in the knowledge base. Moreover, due to the binary character of the attack relation in Dung's system, some attack relations may lead to undesirable results. Namely, symmetric relations are not suitable when ternary (or more) minimal conflicts are included in the base. We propose then the characteristics of an attack relation that should be used for ensuring sound results.

Section 2 recalls Tarski's axiomatization of a monotonic logic. Section 3 details our extension of Dung's system. Section 4 presents examples that show the importance of choosing correctly an attack relation. Section 5 studies the properties of such a relation while Section 6 gives recommendations on how to choose one.

2 Tarski's Abstract Consequence Operations

Alfred Tarski [10] defines an *abstract logic* as a pair $(\mathscr{L}, \mathsf{CN})$ where members of $\mathscr{L}$ are called *well-formed formulas*, and CN is a *consequence operator*. CN is any function from $2^{\mathscr{L}}$ to $2^{\mathscr{L}}$ that satisfies the following axioms:

1. $X \subseteq \mathsf{CN}(X)$ **(Expansion)**
2. $\mathsf{CN}(\mathsf{CN}(X)) = \mathsf{CN}(X)$ **(Idempotence)**
3. $\mathsf{CN}(X) = \bigcup_{Y \subseteq_f X} \mathsf{CN}(Y)$ **(Finiteness)**
4. $\mathsf{CN}(\{x\}) = \mathscr{L}$ for some $x \in \mathscr{L}$ **(Absurdity)**
5. $\mathsf{CN}(\emptyset) \neq \mathscr{L}$ **(Coherence)**

Notation: $Y \subseteq_f X$ means that Y is a finite subset of X.

Intuitively, $\mathsf{CN}(X)$ returns the set of formulas that are logical consequences of X according to the logic in question. It can easily be shown from the above axioms that CN is a closure operator, that is, CN enjoys properties such as:

Property 1. Let $X, X', X'' \subseteq \mathscr{L}$.

1. $X \subseteq X' \Rightarrow \mathsf{CN}(X) \subseteq \mathsf{CN}(X')$. **(Monotonicity)**
2. $\mathsf{CN}(X) \cup \mathsf{CN}(X') \subseteq \mathsf{CN}(X \cup X')$.
3. $\mathsf{CN}(X) = \mathsf{CN}(X') \Rightarrow \mathsf{CN}(X \cup X'') = \mathsf{CN}(X' \cup X'')$.

Almost all well-known monotonic logics (classical logic, intuitionistic logic, modal logic, etc.) can be viewed as special cases of Tarski's notion of an abstract logic.

Once $(\mathscr{L}, \mathsf{CN})$ is fixed, we can define a notion of *consistency* as follows:

Definition 1 (Consistency). *Let $X \subseteq \mathscr{L}$. X is* consistent *wrt $(\mathscr{L}, \mathsf{CN})$ iff $\mathsf{CN}(X) \neq \mathscr{L}$. It is* inconsistent *otherwise.*

This says that X is consistent iff its set of consequences is not the set of all formulas. The coherence requirement forces $\emptyset$ to always be consistent - this makes sense for any reasonable logic as saying emptiness should intuitively be consistent.

Property 2. Let $X \subseteq \mathscr{L}$.

1. If X is consistent, then $\mathsf{CN}(X)$ is consistent as well.
2. $\forall X' \subseteq X$, if X is consistent, then X' is consistent.
3. $\forall X' \subseteq X$, if X' is inconsistent, then X is inconsistent.

In what follows we introduce a concept that is useful for the rest of the paper.

Definition 2 (Adjunctive). *$(\mathscr{L}, \mathsf{CN})$ is* adjunctive *iff for all x and y in $\mathscr{L}$, if $\mathsf{CN}(\{x, y\}) \neq \mathsf{CN}(\{x\})$ and $\mathsf{CN}(\{x, y\}) \neq \mathsf{CN}(\{y\})$ then there exists $z \in \mathscr{L}$ such that $\mathsf{CN}(\{z\}) = \mathsf{CN}(\{x, y\})$.*

Intuitively, an adjunctive logic infers, from the union of two formulas $\{x, y\}$, some formula(s) that can be inferred neither from x alone nor from y alone (except, of course, when y ensues from x or vice-versa). In fact, all well-known logics are adjunctive.[1] A logic which is not adjunctive could for instance fail to deny $x \vee y$ from the premises $\{\neg x, \neg y\}$.

Throughout the paper, the following assumption is made:

Assumption 1. *The logic $(\mathscr{L}, \mathsf{CN})$ is adjunctive.*

From now on, we will consider a *knowledge base* Σ which is a subset of the logical language $\mathscr{L}$ ($\Sigma \subseteq \mathscr{L}$). With no loss of generality and for the sake of simplicity, the knowledge base Σ is assumed to be free of tautologies:

Assumption 2. *For all $x \in \Sigma$, $x \notin \mathsf{CN}(\emptyset)$.*

If Σ is inconsistent, then it contains *minimal conflicts*.

Definition 3 (Minimal conflict). *Let Σ be a knowledge base, and $C \subseteq \Sigma$. The set C is a* minimal conflict *iff:*

- *C is inconsistent*
- *$\forall x \in C$, $C \backslash \{x\}$ is consistent*

Let $\mathscr{C}_\Sigma$ denote the set of all minimal conflicts of Σ.

[1] A few *very restricted* fragments of well-known logics fail to be adjunctive, e.g., the pure implicational fragment of classical logic as it is negationless, disjunctionless, and, of course, conjunctionless.

3 An Extension of Dung's Abstract System

Argumentation is a reasoning paradigm that follows three main steps: i) constructing *arguments* and counterarguments ii) defining the *status* of each argument, and iii) concluding or specifying the *justified conclusions*. In what follows, we refine Dung's system by defining all the above items involved in argumentation without losing generality. We start with an abstract logic $(\mathscr{L}, \mathsf{CN})$ from which the notions of argument and attacks between arguments are defined. More precisely, arguments are built from a knowledge base, say Σ, containing formulas of the language $\mathscr{L}$. An *argument* gives a reason for believing a statement or choosing an action, etc. Formally:

Definition 4 (Argument). *Let Σ be a knowledge base. An* argument *is a pair (X, x) such that:*

1. $X \subseteq \Sigma$
2. X *is consistent*
3. $x \in \mathsf{CN}(X)$
4. $\nexists X' \subset X$ *such that* $x \in \mathsf{CN}(X')$

Notations: $\mathtt{S}upp$ and $\mathtt{C}onc$ denote respectively the *support* X and the *conclusion* x of an argument (X, x). For $\mathcal{S} \subseteq \Sigma$, let $\mathtt{A}rg(\mathcal{S})$ denote the set of all arguments that can be built from $\mathcal{S}$ by means of Definition 4.

Due to Assumption 2 ($x \notin \mathsf{CN}(\emptyset)$ for all $x \in \Sigma$), it can also be shown that each consistent formula in Σ gives birth to an argument:

Property 3. For all $x \in \Sigma$ s.t. the set $\{x\}$ is consistent, there exists $a \in \mathtt{A}rg(\Sigma)$ where $\mathtt{S}upp(a) = \{x\}$.

Since CN is monotonic, constructing arguments is a monotonic process: Additional knowledge never makes the set of arguments to shrink but only gives rise to extra arguments that may interact with the existing arguments.

Property 4. $\mathtt{A}rg(\Sigma) \subseteq \mathtt{A}rg(\Sigma')$ whenever $\Sigma \subseteq \Sigma' \subseteq \mathscr{L}$.

We show next that any proper subset of a minimal conflict is the support of at least one argument. It is a result of utmost importance as regards encoding the attack relation between arguments.

Proposition 1. *Let $(\mathscr{L}, \mathsf{CN})$ be adjunctive. For all non-empty proper subset X of some minimal conflict $C \in \mathscr{C}_\Sigma$, there exists $a \in \mathtt{A}rg(\Sigma)$ s.t. $\mathtt{S}upp(a) = X$.*

Proposition 1 is indeed fundamental because it says that if statements from Σ contradict others then it is always possible to define an argument exhibiting the conflict.

We refine Dung's abstract framework as follows.

Definition 5 (Argumentation system). *Given a knowledge base Σ, an* argumentation system *(AS) over Σ is a pair $(\mathtt{A}rg(\Sigma), \mathcal{R})$ such that $\mathtt{A}rg(\Sigma)$ is a set of arguments defined from Σ and $\mathcal{R} \subseteq \mathtt{A}rg(\Sigma) \times \mathtt{A}rg(\Sigma)$ is an* attack *relation.*

The attack relation captures the different disagreements that may exist between arguments. [6] is silent on how to proceed in order to obtain a reasonable $\mathcal{R}$ in practice. It happens, as pointed out by [4], that it is in fact an error-prone step. We will see in the next section how can things go wrong in this respect, and, in the subsequent section, to what extent our definitions help to circumvent the problem. For the time being, let us focus on what role $\mathcal{R}$ plays in Dung's approach.

Among all the arguments, it is important to know which arguments to rely on for inferring conclusions from a base Σ. In [6], different acceptability semantics have been proposed. The basic idea behind these semantics is the following: for a rational agent, an argument is acceptable if he can defend this argument against all attacks on it. All the arguments acceptable for a rational agent will be gathered in a so-called *extension*. An extension must satisfy a consistency requirement and must defend all its elements.

Definition 6 (Conflict-free, Defence). *Let* $(\mathtt{A}rg(\Sigma), \mathcal{R})$ *be an AS, and* $\mathcal{B} \subseteq \mathtt{A}rg(\Sigma)$.

- $\mathcal{B}$ *is* conflict-free *iff* $\nexists a, b \in \mathcal{B}$ *such that* $(a, b) \in \mathcal{R}$.
- $\mathcal{B}$ defends *an argument* a *iff* $\forall b \in \mathtt{A}rg(\Sigma)$, *if* $(b, a) \in \mathcal{R}$, *then* $\exists c \in \mathcal{B}$ *such that* $(c, b) \in \mathcal{R}$.

The fundamental semantics in [6] is the one that features admissible extensions. The other semantics (i.e., preferred, stable, complete and grounded) are based on it. We only include the definition for the admissible semantics.

Definition 7 (Admissible semantics). *Let* $\mathcal{B}$ *be a conflict-free set of arguments.* $\mathcal{B}$ *is an* admissible *extension iff* $\mathcal{B}$ *defends all its elements.*

Since the notion of the acceptability is defined, we can now characterize the possible conclusions that can be drawn from Σ according to an argumentation system over Σ. The idea is to conclude x if x is the conclusion of at least an argument which belongs to every extension of the system.

Definition 8 (Output of the system). *Let* $(\mathtt{A}rg(\Sigma), \mathcal{R})$ *be an AS over a knowledge base* Σ. *Let* $\mathcal{E}_1, \ldots, \mathcal{E}_n$ *be the extensions of* $(\mathtt{A}rg(\Sigma), \mathcal{R})$ *under a given semantics.*[2] *For* $x \in \mathscr{L}$, x *is a* conclusion *of* Σ *iff* $\exists a \in \mathtt{A}rg(\Sigma)$ *such that* $\mathtt{Conc}(a) = x$ *and* $a \in \mathcal{E}_1 \cap \cdots \cap \mathcal{E}_n$. *We write* $\mathtt{Output}(\Sigma)$ *to denote the set of all conclusions of* Σ.

In [4], it has been argued that the extensions of an argumentation system should ensure consistent results. This means that the base built from the supports of arguments of each extension should be consistent.

Definition 9 (Consistency of extensions). *Let* $(\mathtt{A}rg(\Sigma), \mathcal{R})$ *be an AS. An extension* $\mathcal{E}$ *(under a given semantics)* satisfies consistency *iff* $\bigcup_{a \in \mathcal{E}} \mathtt{Supp}(a)$ *is consistent.*

It can be shown that if all the extensions of an argumentation system enjoy consistency, then the output of the system is consistent as well.

Proposition 2. *Let* $(\mathtt{A}rg(\Sigma), \mathcal{R})$ *be an AS over a knowledge base* Σ. *Let* $\mathcal{E}_1, \ldots, \mathcal{E}_n$ *be the extensions of* $(\mathtt{A}rg(\Sigma), \mathcal{R})$ *under a given semantics. If* $\forall \mathcal{E}_{i=1,\ldots,n}$, $\mathcal{E}_i$ *satisfies consistency, then* $\mathtt{Output}(\Sigma)$ *is consistent.*

[2] One can use any semantics: complete, stable, preferred, or grounded. However, admissible semantics is not recommended since the empty set is admissible. Thus, no argument can belong to all the extensions.

4 Some Problematic Cases

In the previous section we have provided a clear definition of an argument and how it is built from a knowledge base Σ. However, there still is no indication on how $\mathcal{R}$ is defined and how it is related to Σ. Moreover, in [4] it has been shown that there are some instantiations of Dung's system that violate extension consistency. Does this mean that the notion of being conflict-free is not sufficient to ensure consistency? It is sufficient provided that the attack relation is defined in an appropriate way. In this section, we present some problematic examples that shed light on the minimal requirements for defining an attack relation. Throughout the section, we consider propositional logic.

Let us start with an attack relation that is not related to inconsistency in Σ.

Example 1 (Independence from minimal conflict). *Let $\Sigma = \{x, \neg x\}$. So, $\mathscr{C}_\Sigma = \{\Sigma\}$. Take $\mathtt{Arg}(\Sigma) = \{a_1, a_2\}$ where $a_1 = (\{x\}, x)$ and $a_2 = (\{\neg x\}, \neg x)$. Assume that $\mathcal{R} = \emptyset$. This means that $\mathcal{R}$ does not depend at all on minimal conflicts in Σ. In this case, $(\mathtt{Arg}(\Sigma), \mathcal{R})$ has an admissible extension that violates consistency, it is $\{a_1, a_2\}$.*

Requirement 1. *An attack relation should "capture" at least the minimal conflicts of the knowledge base at hand.*

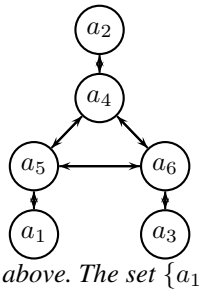
Let us now consider an attack relation that captures all the minimal conflicts of Σ. We start with a *symmetric* relation, due to [7], called *rebut*. An argument a *rebuts* an argument b iff $\mathtt{Conc}(a) \equiv \neg\mathtt{Conc}(b)$ (or vice-versa).

Example 2 (Binary minimal conflict). *The two arguments $a_1 = (\{x\}, x)$ and $a_2 = (\{\neg x\}, \neg x)$ rebut each other. So, the system $(\mathtt{Arg}(\Sigma), Rebut)$ has two admissible extensions $\{a_1\}$ and $\{a_2\}$. They each satisfy consistency.*

Unfortunately, the fact that an attack relation captures the minimal conflicts of a knowledge base does not always ensure consistency of the extensions. Let us consider a knowledge base displaying a ternary minimal conflict.

Example 3 (Ternary conflict). *Let $\Sigma = \{x, y, x \rightarrow \neg y\}$. Let $\mathtt{Arg}(\Sigma)$ consist of the following arguments:*

- $a_1 = (\{x\}, x)$
- $a_2 = (\{y\}, y)$
- $a_3 = (\{x \rightarrow \neg y\}, x \rightarrow \neg y)$
- $a_4 = (\{x, x \rightarrow \neg y\}, \neg y)$
- $a_5 = (\{y, x \rightarrow \neg y\}, \neg x)$
- $a_6 = (\{x, y\}, x \wedge y)$



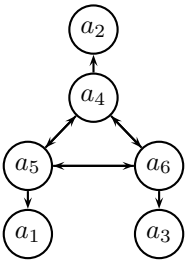
Let $\mathcal{R}$ be such as depicted in figure above. The set $\{a_1, a_2, a_3\}$ is an admissible extension of $(\mathtt{Arg}(\Sigma), \mathcal{R})$. However, $\mathtt{Supp}(a_1) \cup \mathtt{Supp}(a_2) \cup \mathtt{Supp}(a_3)$ is inconsistent.

This example shows that a conflict-free set of arguments may fail consistency. This is due to the fact that $\mathcal{R}$ is binary, in compliance with Dung's definitions imposing the attack relation to be binary. Thus, the ternary conflict between a_1, a_2 and a_3 is not captured.

Requirement 2. *If a knowledge base has ternary (or more) minimal conflicts, the attack relation should be asymmetric.*

Indeed, [5] shows that an argumentation system which uses the asymmetric, *assumption attack* defined in [7] yields consistent extensions. An argument a *undercuts* b iff $\exists h' \in \texttt{Supp}(b)$ such that $\texttt{Conc}(a) \equiv \neg h'$ (or $h' \equiv \neg\texttt{Conc}(a)$).

Example 4 (Ternary conflict cont.). *The attack relation $\mathcal{R}$, in the sense of undercut, between arguments of $\texttt{A}rg(\Sigma)$ from Example 3 is now such as depicted in the figure below:*



There are three maximal (wrt set inclusion) admissible extensions: $\{a_1, a_2, a_6\}$, $\{a_2, a_3, a_5\}$, and $\{a_1, a_3, a_4\}$. All three extensions satisfy consistency.

5 Properties of Attack Relations

In this section, we discuss primitive properties that may be expected from an attack relation. The first of these is about the origin of $\mathcal{R}$. As argued in the previous section, an attack relation should capture at least the minimal conflicts arising from the knowledge base Σ under consideration.

Definition 10 (Capturing conflicts). *Let $C \in \mathscr{C}_\Sigma$. A pair (a, b) in $\mathcal{R}$* captures *C if $C \subseteq \texttt{Supp}(a) \cup \texttt{Supp}(b)$.*

Alas, that an attack relation captures all the minimal conflicts is not sufficient to ensure consistency of extensions.

Example 5. *Let $\texttt{A}rg(\Sigma) = \{a, b, c\}$, $\mathscr{C}_\Sigma = \{C\}$ and $\mathcal{R} = \{(a, b)\}$. If $C \subseteq \texttt{Supp}(a) \cup \texttt{Supp}(b)$ and $C \subseteq \texttt{Supp}(a) \cup \texttt{Supp}(c)$, then $\mathcal{R}$ captures (not "faithfully", though) the minimal conflict C but the admissible extension $\{a, c\}$ violates consistency.*

Definition 11 (Conflict-sensitive). *An attack relation $\mathcal{R}$ is* conflict-sensitive *iff for all a and b in $\texttt{A}rg(\Sigma)$ such that there exists a minimal conflict $C \subseteq \texttt{Supp}(a) \cup \texttt{Supp}(b)$ then either $(a, b) \in \mathcal{R}$ or $(b, a) \in \mathcal{R}$.*

Being conflict-sensitive means this: If Σ provides evidence (according to CN) that the supports of a and b conflict with each other, then this conflict shows in $\mathcal{R}$ (i.e., either $(a, b) \in \mathcal{R}$ holds or $(b, a) \in \mathcal{R}$ holds). In other words, being conflict-sensitive ensures that, when passing from Σ to $(\texttt{A}rg(\Sigma), \mathcal{R})$, no conflict is "forgotten" in $\mathcal{R}$.

Example 6. *The attack relation $\mathcal{R}$ of Example 5 is not conflict-sensitive since neither (a, c) nor (c, a) is in $\mathcal{R}$.*

In case the knowledge base does not contain an inconsistent formula, if $\mathcal{R}$ is conflict-sensitive then $\mathcal{R}$ captures all minimal conflicts.

Proposition 3. *Let $\mathscr{C}_\Sigma$ s.t. $\forall C \in \mathscr{C}_\Sigma$, $|C| > 1$. If $\mathcal{R}$ is sensitive, then $\mathcal{R}$ captures all minimal conflicts of $\mathscr{C}_\Sigma$.*

That an attack relation is conflict-sensitive and captures all the minimal conflicts need not mean that it is strictly based on minimal conflicts. Next is an illustration:

Example 7. *Consider $\mathtt{Arg}(\Sigma) = \{a, b, c\}$, $\mathscr{C}_\Sigma = \{C\}$ and $\mathcal{R} = \{(a, b), (a, c)\}$. Assume $C \subseteq \mathtt{Supp}(a) \cup \mathtt{Supp}(b)$. Then, $\mathcal{R}$ is conflict-sensitive and captures the minimal conflicts in $\mathscr{C}_\Sigma$. But $\mathcal{R}$ contains an attack, (a, c), which is unrelated to $\mathscr{C}_\Sigma$.*

Definition 12 (Conflict-dependent). *An attack relation $\mathcal{R}$ is* conflict-dependent *iff for all a and b in $\mathtt{Arg}(\Sigma)$, $(a, b) \in \mathcal{R}$ implies that there exists a minimal conflict $C \subseteq \mathtt{Supp}(a) \cup \mathtt{Supp}(b)$.*

Being conflict-dependent means this: $\mathcal{R}$ shows no attack from a to b unless Σ provides evidence (according to CN) that the supports of a and b conflict with each other. That is, being conflict-dependent ensures that, when passing from Σ to $(\mathtt{Arg}(\Sigma), \mathcal{R})$, no conflict is "invented" in $\mathcal{R}$.

Example 8. *$\mathcal{R}$ in Example 7 is not conflict-dependent since the attack (a, c) does not depend on the minimal conflict C.*

Clearly, an attack relation that is conflict-sensitive need not be conflict-dependent. Conversely, in Example 5, $\mathcal{R}$ illustrates the fact that an attack relation that is conflict-dependent is not necessarily conflict-sensitive. Note that an attack relation which is conflict-dependent exhibits no self-attack:

Proposition 4. *Let $(\mathtt{Arg}(\Sigma), \mathcal{R})$ be s. t. $\mathcal{R}$ is conflict-dependent. For all $a \in \mathtt{Arg}(\Sigma)$, $(a, a) \notin \mathcal{R}$.*

When the attack relation is conflict-dependent, if a set of arguments is such that its corresponding base (set-theoretic union of supports) is consistent then it is a conflict-free set:

Proposition 5. *Let $(\mathtt{Arg}(\Sigma), \mathcal{R})$ be s. t. $\mathcal{R}$ is conflict-dependent. $\forall \mathcal{B} \subseteq \mathtt{Arg}(\Sigma)$, if $\bigcup_{a \in \mathcal{B}} \mathtt{Supp}(a)$ is consistent, $\mathcal{B}$ is conflict-free.*

When $\mathcal{R}$ is both conflict-sensitive and conflict-dependent, $\mathcal{R}$ then essentially makes no difference between arguments that have equivalent supports:

Proposition 6. *Let $(\mathtt{Arg}(\Sigma), \mathcal{R})$ such that $\mathcal{R}$ is conflict-sensitive and conflict-dependent. For all a and b in $\mathtt{Arg}(\Sigma)$, if $\mathsf{CN}(\mathtt{Supp}(a)) = \mathsf{CN}(\mathtt{Supp}(b))$, then for all $c \in \mathtt{Arg}(\Sigma)$ $(a, c) \in \mathcal{R}$ or $(c, a) \in \mathcal{R}$ iff $(b, c) \in \mathcal{R}$ or $(c, b) \in \mathcal{R}$.*

Definitions 10-12 characterize the "origin" of the attack relation $\mathcal{R}$ by relating it to the minimal conflicts of the knowledge base Σ. It is then natural that the attack relation conforms with the minimal conflicts. In what follows, we present rules about $\mathcal{R}$ that relate to the minimal conflicts.

Definition 13 (Homogeneous relation). *Let a and b in $\mathtt{Arg}(\Sigma)$ such that $\mathtt{Supp}(a) \subseteq \mathtt{Supp}(b)$. For all $c \in \mathtt{Arg}(\Sigma)$,*

R1: $(a, c) \in \mathcal{R} \Rightarrow (b, c) \in \mathcal{R}$.
R2: $(c, a) \in \mathcal{R} \Rightarrow (c, b) \in \mathcal{R}$.

$\mathcal{R}$ is homogeneous *if it satisfies both $R1$ and $R2$.*

The above two rules capture exactly the idea of Property 2 which says that if a set of formulas is inconsistent, then all its supersets are inconsistent as well. This property is captured by two rules since an attack relation is not necessarily symmetric whereas inconsistency is not oriented. We show that, when the attack relation is symmetric, if it satisfies one of the above two rules, then it also satisfies the other.

Property 5. Let $\mathcal{R}$ be symmetric. If $\mathcal{R}$ satisfies rule $R1$ (resp. $R2$) then it satisfies $R2$ (resp. $R1$).

Finally, according to the definition of a minimal conflict C, any partition of C into two subsets X_1 and X_2, it holds that X_1 and X_2 are consistent. Since Proposition 1 ensures that X_1 and X_2 are the supports of arguments, then it is natural to consider that those arguments are conflicting.

Definition 14 (Conflict-exhaustive). *Let $(\mathtt{Arg}(\Sigma), \mathcal{R})$ be an AS over a knowledge base Σ.*

- *$\mathcal{R}$ is* strongly conflict-exhaustive *iff for all $C \in \mathscr{C}_\Sigma$ s.t. $|C| > 1$, for all non-empty proper subset X of C, there exist a and b in $\mathtt{Arg}(\Sigma)$ s.t. $\mathtt{Supp}(a) = X$, $\mathtt{Supp}(b) = C \setminus X$, $(a, b) \in \mathcal{R}$ and $(b, a) \in \mathcal{R}$.*
- *$\mathcal{R}$ is* conflict-exhaustive *iff for all $C \in \mathscr{C}_\Sigma$ s.t. $|C| > 1$, for all non-empty proper subset X of C, there exist a and b in $\mathtt{Arg}(\Sigma)$ s.t. $\mathtt{Supp}(a) = X$, $\mathtt{Supp}(b) = C \setminus X$ and either $(a, b) \in \mathcal{R}$ or $(b, a) \in \mathcal{R}$.*
- *$\mathcal{R}$ is* weakly conflict-exhaustive *iff $\forall C \in \mathscr{C}_\Sigma$ s.t. $|C| > 1$, for all $x \in C$, there exist a and b in $\mathtt{Arg}(\Sigma)$ s.t. $\mathtt{Supp}(a) = \{x\}$, $\mathtt{Supp}(b) = C \setminus \{x\}$ and $(b, a) \in \mathcal{R}$.*

The following property highlights the links between the different notions presented in this section.

Property 6. Let $\mathcal{R} \subseteq \mathtt{Arg}(\Sigma) \times \mathtt{Arg}(\Sigma)$.

1. $\mathcal{R}$ is strongly conflict-exhaustive, then $\mathcal{R}$ is weakly conflict-exhaustive and $\mathcal{R}$ is conflict-exhaustive.
2. If $\mathcal{R}$ is homogeneous and conflict-exhaustive, then $\mathcal{R}$ is conflict-sensitive.
3. If $\mathcal{R}$ is conflict-sensitive, then it is conflict-exhaustive.

Finally, one can characterize symmetric relations using the above properties.

Proposition 7. *If $\mathcal{R}$ is conflict-dependent, homogeneous, and strongly conflict-exhaustive, then $\mathcal{R}$ is symmetric.*

6 Choosing an Attack Relation

This section studies the appropriate attack relation of an argumentation system. By appropriate relation, we mean a relation that ensures at least extension consistency. We will study two cases: the case where all the minimal conflicts that are contained in a base are binary, and the case of a base containing ternary or more minimal conflicts.

6.1 Case of Binary Minimal Conflicts

Throughout this section, we will consider a knowledge base Σ whose minimal conflicts are all *binary*. This means that any minimal conflict contains exactly two formulas. Thus, there is no inconsistent formula in the base.

The following result is of great importance since it provides a class of attack relations that ensure extension consistency.

Proposition 8. *Let $(\mathtt{A}rg(\Sigma), \mathcal{R})$ be an AS over Σ s.t. all minimal conflicts of Σ are binary. If $\mathcal{R}$ is conflict-sensitive, then for all $\mathcal{B} \subseteq \mathtt{A}rg(\Sigma)$, that $\mathcal{B}$ is conflict-free implies that $\bigcup_{a \in \mathcal{B}} \mathtt{Supp}(a)$ is consistent.*

The above result is not surprising. As shown in the previous section, in order for a conflict-free set of arguments to ensure consistency, the attack relation should capture the minimal conflicts. However, it is not necessary to be conflict-dependent. This latter is however important to show that any set of arguments that satisfies consistency is conflict-free (see Proposition 5).

Corollary 1. *Let $(\mathtt{A}rg(\Sigma), \mathcal{R})$ be an AS over Σ s.t. all minimal conflicts of Σ are binary. If $\mathcal{R}$ is conflict-sensitive, then $(\mathtt{A}rg(\Sigma), \mathcal{R})$ satisfies extension consistency.*

From this corollary and Property 6, the following result holds.

Corollary 2. *Let $(\mathtt{A}rg(\Sigma), \mathcal{R})$ be an AS over Σ s.t. all minimal conflicts of Σ are binary. If $\mathcal{R}$ is homogeneous and conflict-exhaustive then $(\mathtt{A}rg(\Sigma), \mathcal{R})$ satisfies consistency extension.*

Another direct consequence of Proposition 5, and Proposition 8 is the following:

Corollary 3. *Let $(\mathtt{A}rg(\Sigma), \mathcal{R})$ be an AS over Σ s.t. all minimal conflicts of Σ are binary. Let $\mathcal{R}$ be conflict-sensitive and conflict-dependent. For all $\mathcal{B} \subseteq \mathtt{A}rg(\Sigma)$, $\mathcal{B}$ is conflict-free iff $\bigcup_{a \in \mathcal{B}} \mathtt{Supp}(a)$ is consistent.*

It is worth noticing that symmetric relations can be used in the binary case. This means that it is possible to have an attack relation that is both sensitive and symmetric. Indeed, according to Proposition 7, when $\mathcal{R}$ is conflict-dependent, homogeneous, and strongly conflict-exhaustive, then $\mathcal{R}$ is symmetric. From Property 6 it is clear that when $\mathcal{R}$ is strongly exhaustive, then it is also conflict-exhaustive. Thus, $\mathcal{R}$ is sensitive.

6.2 Case of General Minimal Conflicts

In the previous section, we have shown that when all the minimal conflicts of a knowledge base are binary, then a symmetric relation can be used and ensures extension consistency. Unfortunately, this is not the case when ternary or more minimal conflicts are present in a base. The following result shows that symmetric relations lead to the violation of consistency.

Proposition 9. *Let* $\Sigma = \{x_1, \ldots, x_n\}$ *where* $n > 2$ *and* $\mathscr{C}_\Sigma = \{\Sigma\}$. *Let* $a_1, \ldots, a_n \in \mathtt{Arg}(\Sigma)$ *s.t.* $\mathtt{Supp}(a_i) = \{x_i\}$. *If* $\mathcal{R}$ *is conflict-dependent and symmetric, then* $\mathcal{E} = \{a_1, \ldots, a_n\}$ *is an admissible extension of* $(\mathtt{Arg}(\Sigma), \mathcal{R})$.

A direct consequence of the above result is the following:

Corollary 4. *Let* $\Sigma = \{x_1, \ldots, x_n\}$ *where* $n > 2$ *and* $\mathscr{C}_\Sigma = \{\Sigma\}$. *If* $\mathcal{R}$ *is conflict-dependent and symmetric, then the AS* $(\mathtt{Arg}(\Sigma), \mathcal{R})$ *over* Σ *violates extension consistency.*

The previous result can be generalized as follows:

Corollary 5. *Let* $\mathscr{C}_\Sigma$ *s.t.* $\exists C \in \mathscr{C}_\Sigma$ *and* $|C| > 2$. *If* $\mathcal{R}$ *is conflict-dependent and symmetric, then the AS* $(\mathtt{Arg}(\Sigma), \mathcal{R})$ *over* Σ *violates extension consistency.*

Let us now present a class of attack relations that ensure consistency. The following result states that when the relation $\mathcal{R}$ satisfies both $R2$ and $R4$ and is weakly exhaustive, then the corresponding argumentation system satisfies extension consistency.

Proposition 10. *If* $\mathcal{R}$ *satisfies R2, R4 and is weakly exhaustive, then* $(\mathtt{Arg}(\Sigma), \mathcal{R})$ *satisfies extension consistency.*

Note that there may exist other classes of asymmetric attack relations that ensure extensions consistency.

7 Conclusion

The paper extended Dung's argumentation framework by taking into account the logic from which arguments are built. The new framework is general since it is grounded on an abstract monotonic logic. Thus, a wide variety of logics can be used even those that are not yet considered in argumentation like temporal logic, modal logic. The extension has two main advantages: First, it enforces the framework to make safe conclusions. Second, it relates the different notions of Dung's system, like the attack relation and conflict-free, to the knowledge base that is under study. The paper also presented a formal methodology for defining arguments from a knowledge base, and for choosing an appropriate attack relation.

There are a number of ways to extend this work. One future direction is to complete the study of attack relations that ensure consistency. Another direction that we want to pursue is to consider the case of non-adjunctive logics. Recall that all the results presented in this paper hold in the case of an adjunctive logic.

References

1. Amgoud, L., Prade, H.: Using arguments for making and explaining decisions. Artificial Intelligence J. 173, 413–436 (2009)
2. Besnard, P., Hunter, A.: Elements of Argumentation. MIT Press, Cambridge (2008)
3. Bonet, B., Geffner, H.: Arguing for decisions: A qualitative model of decision making. In: Proc. of UAI 1996, pp. 98–105 (1996)
4. Caminada, M., Amgoud, L.: On the evaluation of argumentation formalisms. Artificial Intelligence J. 171(5-6), 286–310 (2007)
5. Cayrol, C.: On the relation between argumentation and non-monotonic coherence-based entailment. In: Proc. of IJCAI 1995, pp. 1443–1448 (1995)
6. Dung, P.M.: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artificial Intelligence J. 77, 321–357 (1995)
7. Elvang-Gøransson, M., Fox, J., Krause, P.: Acceptability of arguments as logical uncertainty. In: Moral, S., Kruse, R., Clarke, E. (eds.) ECSQARU 1993. LNCS, vol. 747, pp. 85–90. Springer, Heidelberg (1993)
8. Prakken, H., Sartor, G.: Argument-based extended logic programming with defeasible priorities. J. of Applied Non-Classical Logics 7, 25–75 (1997)
9. Simari, G., Loui, R.: A mathematical treatment of defeasible reasoning and its implementation. Artificial Intelligence J. 53, 125–157 (1992)
10. Tarski, A.: On Some Fundamental Concepts of Metamathematics. Oxford Uni. Press, Oxford (1956)

Appendix

Proof. of Property 1 Let $X, X', X'' \subseteq \mathscr{L}$.

1. Assume that $X \subseteq X'$. According to Finiteness axiom, it holds that $\mathsf{CN}(X) = \bigcup_{Y \subseteq_f X} \mathsf{CN}(Y)$. $\bigcup_{Y \subseteq_f X} \mathsf{CN}(Y) \subseteq \bigcup_{Y \subseteq_f X'} \mathsf{CN}(Y) = \mathsf{CN}(X')$ since $Y \subseteq_f X \subset X'$.
2. $X \subseteq X \cup X'$, thus by monotonicity, $\mathsf{CN}(X) \subseteq \mathsf{CN}(X \cup X')$ (a). Similarly, $X' \subseteq X \cup X'$ thus $\mathsf{CN}(X') \subseteq \mathsf{CN}(X \cup X')$ (b). From (a) and (b), $\mathsf{CN}(X) \cup \mathsf{CN}(X') \subseteq \mathsf{CN}(X \cup X')$.
3. Assume that $\mathsf{CN}(X) = \mathsf{CN}(X')$. According to Expansion axiom, the following inclusions hold: $X' \subseteq \mathsf{CN}(X')$ and $X'' \subseteq \mathsf{CN}(X'')$. Thus, $X' \cup X'' \subseteq \mathsf{CN}(X') \cup \mathsf{CN}(X'') = \mathsf{CN}(X) \cup \mathsf{CN}(X'')$ (a). Moreover, $X \subseteq X \cup X''$ thus $\mathsf{CN}(X) \subseteq \mathsf{CN}(X \cup X'')$ (a'). Similarly, $X'' \subseteq X \cup X''$ thus $\mathsf{CN}(X'') \subseteq \mathsf{CN}(X \cup X'')$ (b'). From (a') and (b'), $\mathsf{CN}(X) \cup \mathsf{CN}(X'') \subseteq \mathsf{CN}(X \cup X'')$. From (a), it follows that $X' \cup X'' \subseteq \mathsf{CN}(X \cup X'')$. By monotonicity, $\mathsf{CN}(X' \cup X'') \subseteq \mathsf{CN}(\mathsf{CN}(X \cup X''))$. Finally, by Idempotence axiom, the inclusion $\mathsf{CN}(X' \cup X'') \subseteq \mathsf{CN}(X \cup X'')$ holds. To show that $\mathsf{CN}(X \cup X'') \subseteq \mathsf{CN}(X' \cup X'')$, the same reasoning is applied by starting with X instead of X'.

Proof. of Property 2 Let $X \subseteq \mathscr{L}$. Assume that X is consistent, then $\mathsf{CN}(X) \neq \mathscr{L}$ (1)

1. Let us now assume that $\mathsf{CN}(X)$ is inconsistent. This means that $\mathsf{CN}(\mathsf{CN}(X)) = \mathscr{L}$. However, according to Idempotence axiom, $\mathsf{CN}(\mathsf{CN}(X)) = \mathsf{CN}(X)$. Thus, $\mathsf{CN}(X) = \mathscr{L}$, this contradicts (1).

2. Assume that $\exists X' \subseteq X$ s.t. X' is inconsistent. Thus $\mathsf{CN}(X') = \mathscr{L}$. However, since $X' \subseteq X$ then $\mathsf{CN}(X') \subseteq \mathsf{CN}(X)$ (according to the monotonicity axiom). Thus, $\mathscr{L} \subseteq \mathsf{CN}(X)$. Since $\mathsf{CN}(X) \subseteq \mathscr{L}$, then $\mathsf{CN}(X) = \mathscr{L}$. Thus, X is inconsistent. Contradiction.
3. Let $X' \subseteq X$. Assume that X' is inconsistent. Since $X' \subseteq X$ thus $\mathsf{CN}(X') \subseteq \mathsf{CN}(X)$ (by monotonicity axiom). Since X' is inconsistent, $\mathsf{CN}(X') = \mathscr{L}$. Consequently, $\mathsf{CN}(X) = \mathscr{L}$ which means that X is inconsistent.

Proof. of Property 4 Let $\Sigma \subseteq \Sigma' \subseteq \mathscr{L}$. Assume that $(X, x) \in \mathtt{A}rg(\Sigma)$ and $(X, x) \notin \mathtt{A}rg(\Sigma')$. Since $(X, x) \notin \mathtt{A}rg(\Sigma')$, then there are four possible cases:

1. X is not a subset of Σ'. This is impossible since $(X, x) \in \mathtt{A}rg(\Sigma)$, thus $X \subseteq \Sigma \subseteq \Sigma'$.
2. X is inconsistent, this is impossible since $(X, x) \in \mathtt{A}rg(\Sigma)$.
3. $x \notin \mathsf{CN}(X)$. This is impossible since $X \subseteq \Sigma \subseteq \Sigma'$ and by monotonicity, $\mathsf{CN}(X) \subseteq \mathsf{CN}(\Sigma) \subseteq \mathsf{CN}(\Sigma')$.
4. X is not minimal. Thus, $\exists X' \subset X$ that satisfies conditions 1-3 of Def. 4. This is impossible since $(X, x) \in \mathtt{A}rg(\Sigma)$.

Lemma 1. *If X is a non-empty proper subset of a minimal conflict of Σ, then there exists $x \in \mathscr{L}$ s.t. $\mathsf{CN}(\{x\}) = \mathsf{CN}(X)$.*

Proof. Assume X is a non-empty proper subset of a minimal conflict $C \in \mathscr{C}_\Sigma$. We prove the lemma by induction, after we first take care to show that X is finite. By Tarski's requirements, there exists $x_0 \in \mathscr{L}$ s.t. $\mathsf{CN}(\{x_0\}) = \mathscr{L}$. Since C is a conflict, $\mathsf{CN}(C) = \mathsf{CN}(\{x_0\})$. As a consequence, $x_0 \in \mathsf{CN}(C)$. However, $\mathsf{CN}(C) = \bigcup_{C' \subseteq_f C} \mathsf{CN}(C')$ by Tarski's requirements. Thus, $x_0 \in \mathsf{CN}(C)$ means that there exists $C' \subseteq_f C$ s.t. $x_0 \in \mathsf{CN}(C')$. This says that C' is a conflict. Since C is a minimal conflict, $C = C'$ and it follows that C is finite. Of course, so is X: Let us write $X = \{x_1, \ldots, x_n\}$. *Base step:* $n = 1$. Taking x to be x_1 is enough. *Induction step:* Assume the lemma is true up to rank $n - 1$. As CN is a closure operator, $\mathsf{CN}(\{x_1, \ldots, x_n\}) = \mathsf{CN}(\mathsf{CN}(\{x_1, \ldots, x_{n-1}\}) \cup \{x_n\})$. The induction hypothesis entails $\exists x \in \mathscr{L}$ s.t. $\mathsf{CN}(\mathsf{CN}(\{x_1, \ldots, x_{n-1}\}) \cup \{x_n\}) = \mathsf{CN}(\mathsf{CN}(\{x\}) \cup \{x_n\})$. Then, $\mathsf{CN}(\{x_1, \ldots, x_n\}) = \mathsf{CN}(\{x, x_n\})$. As $\mathsf{CN}(\{x, x_n\}) \neq \mathsf{CN}(\{x_n\})$ and $\mathsf{CN}(\{x, x_n\}) \neq \mathsf{CN}(\{x\})$ (otherwise C cannot be minimal), there exists $y \in \mathscr{L}$ s.t. $\mathsf{CN}(\{x, x_n\}) = \mathsf{CN}(\{y\})$ because $(\mathscr{L}, \mathsf{CN})$ is adjunctive. Since $\mathsf{CN}(\{x_1, \ldots, x_n\}) = \mathsf{CN}(\{x, x_n\})$ was just proved, it follows that $\mathsf{CN}(\{y\}) = \mathsf{CN}(\{x_1, \ldots, x_n\})$.

Proof. of Proposition 1 Let $C \in \mathscr{C}_\Sigma$ and $X \subseteq C$ such that X is non-empty. Assume $\nexists a \in \mathtt{A}rg(\Sigma)$ such that $\mathtt{S}upp(a) = X$. I.e., there exists no x such that X is a minimal consistent set satisfying $x \in \mathsf{CN}(X)$. So, for all $x \in \mathscr{L}$, if $x \in \mathsf{CN}(X)$ then $\exists Y \subset X$ such that $x \in \mathsf{CN}(Y)$. In short, for all $x \in \mathsf{CN}(X)$, there exists $Y \subset X$ such that $x \in \mathsf{CN}(Y)$. Lemma 1 says that there exists $z \in \mathscr{L}$ such that $\mathsf{CN}(\{z\}) = \mathsf{CN}(X)$. However, $z \notin \mathsf{CN}(Y)$ for all $Y \subset X$ otherwise C would fail to be minimal (because $Y \cup (C \setminus X) \subset C$ while $z \in \mathsf{CN}(Y)$ implies $\mathsf{CN}(C) = \mathsf{CN}(X \cup (C \setminus X)) = \mathsf{CN}(\{z\} \cup (C \setminus X)) \subseteq \mathsf{CN}(Y \cup (C \setminus X))$). A contradiction arises.

Proof. of Proposition 2 Let Σ be a knowledge base, $(\mathtt{A}rg(\Sigma), \mathcal{R})$ its associated argumentation system, and $\mathcal{E}_1, \ldots, \mathcal{E}_n$ the different extensions under a given semantics. Let us assume that all the extensions satisfy consistency. Assume also that $\mathtt{Output}(\Sigma)$ is inconsistent. This means that $\exists X \subseteq \mathtt{Output}(\Sigma)$ such that X is a minimal (for set inclusion) inconsistent set. Let $X = \{x_1, \ldots, x_m\}$. Since each element of X is in $\mathtt{Output}(\Sigma)$, then $\exists a_i \in \mathtt{A}rg(\Sigma)$ for each x_i such that $\mathtt{Conc}(a_i) = x_i$ and $a_i \in \mathcal{E}_1 \cap \ldots \cap \mathcal{E}_n$. Let $a_1, \ldots, a_m$ be those arguments.

Let us now consider a given extension $\mathcal{E}_k$. It is clear that $a_1, \ldots, a_m \in \mathcal{E}_k$. Thus, $\mathtt{Supp}(a_1) \cup \ldots \cup \mathtt{Supp}(a_m) \subseteq \bigcup \mathtt{Supp}(a)$ where $a \in \mathcal{E}_k$. According to the Monotonicity axiom, it holds that $\mathsf{CN}(\mathtt{Supp}(a_1) \cup \ldots \cup \mathtt{Supp}(a_m)) \subseteq \mathsf{CN}(\bigcup \mathtt{Supp}(a))$ where $a \in \mathcal{E}_k$. (a)

Besides, $\forall x_i \in X, x_i \in \mathsf{CN}(\mathtt{Supp}(a_i))$ (according to the definition of an argument). Thus, $\{x_i\} \subseteq \mathsf{CN}(\mathtt{Supp}(a_i))$. It follows that $\{x_1, \ldots, x_m\} \subseteq \mathsf{CN}(\mathtt{Supp}(a_1)) \cup \ldots \cup \mathsf{CN}(\mathtt{Supp}(a_m))$. (b)

From (a) and (b), it follows that $X \subseteq \mathsf{CN}(\bigcup \mathtt{Supp}(a))$ where $a \in \mathcal{E}_k$. Thus, $\mathsf{CN}(X) \subseteq \mathsf{CN}(\mathsf{CN}(\bigcup \mathtt{Supp}(a)))$. According to Idempotence axiom, $\mathsf{CN}(\mathsf{CN}(\bigcup \mathtt{Supp}(a))) = \mathsf{CN}(\bigcup \mathtt{Supp}(a))$. Since X is inconsistent, then $\mathsf{CN}(X) = \mathscr{L}$, thus $\mathsf{CN}(\bigcup \mathtt{Supp}(a)) = \mathscr{L}$. Thus, $\bigcup \mathtt{Supp}(a)$ is inconsistent. This contradicts the fact that $\mathcal{E}_k$ satisfies consistency.

Proof. of Proposition 3 Let $\mathscr{C}_\Sigma$ s.t. $\forall C \in \mathscr{C}_\Sigma$, $|C| > 1$. Assume that $\exists C \in \mathscr{C}_\Sigma$ s.t. C is not captured. Thus, $\nexists a, b \in \mathtt{A}rg(\Sigma)$ s.t. $C \subseteq \mathtt{Supp}(a) \cup \mathtt{Supp}(b)$ and $(a, b) \in \mathcal{R}$ or $(b, a) \in \mathcal{R}$. There are two cases:

case 1: $\nexists a, b \in \mathtt{A}rg(\Sigma)$ s.t. $C \subseteq \mathtt{Supp}(a) \cup \mathtt{Supp}(b)$. Since $|C| > 1$, then C can be partitioned into 2 consistent non-empty subsets X_1 and X_2. From Proposition 1, since $(\mathscr{L}, \mathsf{CN})$ is adjunctive, then $\exists a_1, a_2 \in \mathtt{A}rg(\Sigma)$ s.t. $\mathtt{Supp}(a_1) = X_1$ and $\mathtt{Supp}(a_2) = X_2$.

case 2: $(a, b) \notin \mathcal{R}$ and $(b, a) \notin \mathcal{R}$. From case 1, it is clear that $\exists a, b \in \mathtt{A}rg(\Sigma)$ s.t. $C \subseteq \mathtt{Supp}(a) \cup \mathtt{Supp}(b)$. Since $\mathcal{R}$ is sensitive, then $\mathcal{R}$ contains either (a, b) or (b, a).

Proof. of Proposition 4 Assume that $\mathcal{R}$ is conflict-dependent and $a \in \mathtt{A}rg(\Sigma)$ s.t. $(a, a) \in \mathcal{R}$. Since $\mathcal{R}$ is conflict-dependent, then $\exists C \in \mathscr{C}_\Sigma$ s.t. $C \subseteq \mathtt{Supp}(a)$. This means that $\mathtt{Supp}(a)$ is inconsistent. This contradicts the fact that a is an argument.

Proof. of Proposition 5 Let $\mathcal{B} \subseteq \mathtt{A}rg(\Sigma)$. Since $\bigcup_{a \in \mathcal{B}} \mathtt{Supp}(a)$ is consistent, then so is $\mathtt{Supp}(a) \cup \mathtt{Supp}(b)$ for all a and b in $\mathcal{B}$ (according to Property 2). Hence, there exist no minimal conflict $C \subseteq \mathtt{Supp}(a) \cup \mathtt{Supp}(b)$. By the definition of $\mathcal{R}$ being conflict-dependent, $(a, b) \notin \mathcal{R}$ ensues.

Proof. of Proposition 6 Due to symmetry, it is enough to show that if $a\mathcal{R}c$ then $b\mathcal{R}c$ or $c\mathcal{R}b$. Assume $a\mathcal{R}c$. Since $\mathcal{R}$ is conflict-dependent, $C \subseteq \mathtt{Supp}(a) \cup \mathtt{Supp}(c)$ for some $C \in \mathscr{C}_\Sigma$. By Tarski's requirements for CN, it follows, due to $\mathsf{CN}(\mathtt{Supp}(a)) = \mathsf{CN}(\mathtt{Supp}(b))$, that $\mathtt{Supp}(b) \cup \mathtt{Supp}(c)$ contains a minimal conflict. Since $\mathcal{R}$ is conflict-sensitive, $b\mathcal{R}c$ or $c\mathcal{R}b$.

Proof. of Property 5 Let $\mathcal{R}$ be a symmetric relation that satisfies rule $R1$. Let a, b, c in $\mathtt{Arg}(\Sigma)$ such that $\mathtt{Supp}(a) \subseteq \mathtt{Supp}(b)$. According to $R1$, if $(a, c) \in \mathcal{R}$ then $(b, c) \in \mathcal{R}$. However, since $\mathcal{R}$ is symmetric, $(a, c) \in \mathcal{R}$ means that $(c, a) \in \mathcal{R}$ and $(b, c) \in \mathcal{R}$ means that $(c, b) \in \mathcal{R}$. Same reasoning holds for $R2$.

Proof. of Property 6 1) The implications follow from Definition 14.
2) Let $a, b \in \mathtt{Arg}(\Sigma)$ and $C \in \mathscr{C}_\Sigma$ such that $C \subseteq \mathtt{Supp}(a) \cup \mathtt{Supp}(b)$. Clearly, $\mathtt{Supp}(a) \cap C$ is non-empty (otherwise, $\mathtt{Supp}(b)$ would be inconsistent). Of course, $\mathtt{Supp}(a) \cap C$ is a subset of C. Let X be $\mathtt{Supp}(a) \cap C$. According to Proposition 1, since $(\mathscr{L}, \mathsf{CN})$ is adjunctive then there exist $x, y \in \mathtt{Arg}(\Sigma)$ with $\mathtt{Supp}(x) = X$ and $\mathtt{Supp}(y) = C \setminus X$. Since $\mathcal{R}$ is conflict-exhaustive, then either $(x, y) \in \mathcal{R}$ or $(y, x) \in \mathcal{R}$. Both cases are symmetric, so we only consider $(x, y) \in \mathcal{R}$. That $\mathtt{Supp}(a) \cap C$ is a subset of $\mathtt{Supp}(a)$ means $\mathtt{Supp}(x) \subseteq \mathtt{Supp}(a)$. Therefore, $(x, y) \in \mathcal{R}$ yields $(a, y) \in \mathcal{R}$ because $\mathcal{R}$ is homogeneous. On the other hand, $C \subseteq \mathtt{Supp}(a) \cup \mathtt{Supp}(b)$ implies that $C \setminus (\mathtt{Supp}(a) \cap C)$ is a subset of $\mathtt{Supp}(b)$. That is, $\mathtt{Supp}(y)$ is a subset of $\mathtt{Supp}(b)$. Therefore, $(a, y) \in \mathcal{R}$ yields $(a, b) \in \mathcal{R}$ because $\mathcal{R}$ is homogeneous. Overall, we obtain that either $(a, b) \in \mathcal{R}$ or $(b, a) \in \mathcal{R}$ holds.
3) Assume that $\mathcal{R}$ is conflict-sensitive but not conflict-exhaustive. Since $\mathcal{R}$ is not conflict-exhaustive, this means that $\exists C \in \mathscr{C}_\Sigma$ such that for $X \subset C$, $\exists a, b \in \mathtt{Arg}(\Sigma)$ such that $\mathtt{Supp}(a) = X$ and $\mathtt{Supp}(b) = C \setminus X$ (according to Proposition 1) and $(a, b) \notin \mathcal{R}$ and $(b, a) \notin \mathcal{R}$. However, $\mathtt{Supp}(a) \cup \mathtt{Supp}(b) = C$. Thus, since $\mathcal{R}$ is conflict-sensitive, either $(a, b) \in \mathcal{R}$ or $(b, a) \in \mathcal{R}$ holds. Contradiction.

Proof. of Proposition 7 Assume that $\mathcal{R}$ is conflict-dependent, satisfies rules R1, R2, and is strongly conflict-exhaustive. Assume also that $\mathcal{R}$ is not symmetric. Thus, $\exists (a, b) \in \mathcal{R}$ but $(b, a) \notin \mathcal{R}$. Since $\mathcal{R}$ is conflict-dependent, then $\exists C \subseteq \mathtt{Supp}(a) \cup \mathtt{Supp}(b)$ with $C \in \mathscr{C}_\Sigma$. Let $X_1 = C \cap \mathtt{Supp}(a)$ and $X_2 = C \setminus X_1$. According to Proposition 1, $\exists a', b' \in \mathtt{Arg}(\Sigma)$ s.t. $\mathtt{Supp}(a') = X_1$ and $\mathtt{Supp}(b') = X_2$. It is clear that a' is a sub-argument of a and b' is a sub-argument of b. Since $\mathcal{R}$ is strongly conflict-exhaustive, then $(b', a') \in \mathcal{R}$. Since $\mathcal{R}$ satisfies R2, it follows that $(b', a) \in \mathcal{R}$. Finally, since $\mathcal{R}$ satisfies R1, $(b, a) \in \mathcal{R}$ holds.

Proof. of Proposition 8 Let $\mathcal{B} \subseteq \mathtt{Arg}(\Sigma)$ s.t. $\mathcal{B}$ is conflict-free. Thus, $\forall a, b \in \mathcal{B}$, neither $(a, b) \in \mathcal{R}$ nor $(b, a) \in \mathcal{R}$ hold. By the definition of being conflict-sensitive, this means that there exist no minimal conflict $C \subseteq \mathtt{Supp}(a) \cup \mathtt{Supp}(b)$. A consequence is that $\bigcup_{a \in \mathcal{E}} \mathtt{Supp}(a)$ is consistent because all minimal conflicts in Σ are binary (cf the assumption). That is, $\bigcup_{a \in \mathcal{B}} \mathtt{Supp}(a)$ is consistent.

Proof. of Proposition 9 Let $\mathscr{C}_\Sigma = \{\Sigma\}$ s.t. $\Sigma = \{x_1, \ldots, x_n\}$ and $n > 2$. Let $a_1, \ldots, a_n \in \mathtt{Arg}(\Sigma)$ such that $\mathtt{Supp}(a_i) = \{x_i\}$. Assume that the set $\mathcal{E} = \{a_1, \ldots, a_n\}$ is not an admissible extension.

Case 1: $\mathcal{E}$ is not conflict-free. Thus, $\exists a_i, a_j \in \mathcal{E}$ such that $(a_i, a_j) \in \mathcal{R}$. Since $\mathcal{R}$ is conflict-dependent, $\exists C' \subseteq \mathtt{Supp}(a_i) \cup \mathtt{Supp}(a_j)$ such that C' is a minimal conflict. Thus, the set $\{x_i, x_j\}$ is inconsistent. However, $|C| > 2$ then $\{x_i, x_j\} \subset C$. Consequently, $\{x_i, x_j\}$ is consistent.
Case 2: $\mathcal{E}$ does not defend its elements. This means that $\exists a \in \mathtt{Arg}(\Sigma)$, $a_i \in \mathcal{E}$ such that $(a, a_i) \in \mathcal{R}$ and $\mathcal{E}$ does not defend a_i. Since $\mathcal{R}$ is symmetric, it holds that $(a_i, a) \in \mathcal{R}$, thus $\mathcal{E}$ defends a_i.

Proof. of Corollary 4 Let $\mathscr{C}_\Sigma = \{\Sigma\}$ with $\Sigma = \{x_1, \ldots, x_n\}$ and $n > 2$. Let $a_1, \ldots, a_n \in \mathtt{A}rg(\Sigma)$ such that $\mathtt{Supp}(a_i) = \{x_i\}$. Assume that $\mathcal{R}$ is conflict-dependent and symmetric. According to Proposition 9, the set $\mathcal{E} = \{a_1, \ldots, a_n\}$ is an admissible extension. Thus, $\mathtt{Supp}(a_1) \cup \ldots \cup \mathtt{Supp}(a_n) = C$, thus inconsistent.

Proof. Corollary 5 Let $C = \{x_1, \ldots, x_n\}$ where $n > 2$. Assume that $C \in \mathscr{C}_\Sigma$ and that $\mathcal{R}$ is both conflict-dependent and symmetric. From Proposition 9, the set $\mathcal{E} = \{a_1, \ldots, a_n\}$, with $\mathtt{Supp}(a_i) = \{x_i\}$, is an admissible extension of $(\mathtt{A}rg(\Sigma), \mathcal{R})$. Moreover, $\mathcal{E}$ violates extension consistency. Thus, $(\mathtt{A}rg(\Sigma), \mathcal{R})$ violates extension consistency.

Proof. of Proposition 10 Let $(\mathtt{A}rg(\Sigma), \mathcal{R})$ be an argumentation system such that $\mathcal{R}$ satisfies R2, R4 and is weakly exhaustive. Assume that $\mathcal{E}$ is an admissible extension. Let $\mathtt{Formulas}(\mathcal{E}) = \bigcup_{a \in \mathcal{E}} \mathtt{Supp}(a)$. Assume that $\mathcal{E}$ violates consistency. This means that $\exists C \in \mathscr{C}_\Sigma$ such that $C \subseteq \mathtt{Formulas}(\mathcal{E})$. Let $C = \{x_1, \ldots, x_n\}$.
Consider $x_1 \in C$. According to Proposition 1, $\exists b \in \mathtt{A}rg(\Sigma)$ such that $\mathtt{Supp}(b) = \{x_1\}$. Moreover, $\exists a \in \mathtt{A}rg(\Sigma)$ such that $\mathtt{Supp}(a) = C \setminus \{x_1\}$. Since $x_1 \in \mathtt{Formulas}(\mathcal{E})$, then $\exists a' \in \mathcal{E}$ with $x_1 \in \mathtt{Supp}(a')$. Thus, b is a sub-argument of a'.
Since $\mathcal{R}$ is weakly exhaustive, it holds that $(a, b) \in \mathcal{R}$. Since $\mathcal{R}$ satisfies rule $R2$ and b is a sub-argument of a', then $(a, a') \in \mathcal{R}$.
However, $\mathcal{E}$ is an admissible extension, this means that $a \notin \mathcal{E}$. Since $(a, a') \in \mathcal{R}$ and $a' \in \mathcal{E}$, if follows that $\mathcal{E}$ defends a' against a. This means that $\exists a'' \in \mathcal{E}$ such that $(a'', a) \in \mathcal{R}$. Let $x_2 \in C$, then $\{x_2\} \subseteq \mathtt{Supp}(a)$, by Proposition 1 $\exists b' \in \mathtt{A}rg(\Sigma)$ such that $\mathtt{Supp}(b') = \{x_2\}$. According to rule R4, $(a'', b') \in \mathcal{R}$. However, $x_2 \in C$, thus, $x_2 \in \mathtt{Formulas}(\mathcal{E})$ thus, $\exists d \in \mathcal{E}$ such that $x_2 \in \mathtt{Supp}(d)$. Thus, b' is a sub-argument of d. Since R satisfies rule R2, it holds that $(a'', d) \in \mathcal{R}$. This contradicts the fact that $\mathcal{E}$ is conflict-free.

Modeling Unreliable Observations in Bayesian Networks by Credal Networks

Alessandro Antonucci and Alberto Piatti

Dalle Molle Institute for Artificial Intelligence (IDSIA)
Galleria 2, Via Cantonale
CH6927 - Manno - Lugano, Switzerland
{alessandro,alberto.piatti}@idsia.ch

Abstract. Bayesian networks are probabilistic graphical models widely employed in AI for the implementation of knowledge-based systems. Standard inference algorithms can update the beliefs about a variable of interest in the network after the observation of some other variables. This is usually achieved under the assumption that the observations could reveal the actual states of the variables in a fully reliable way. We propose a procedure for a more general modeling of the observations, which allows for updating beliefs in different situations, including various cases of unreliable, incomplete, uncertain and also missing observations. This is achieved by augmenting the original Bayesian network with a number of auxiliary variables corresponding to the observations. For a flexible modeling of the observational process, the quantification of the relations between these auxiliary variables and those of the original Bayesian network is done by *credal sets*, i.e., convex sets of probability mass functions. Without any lack of generality, we show how this can be done by simply estimating the bounds of likelihoods of the observations for the different values of the observed variables. Overall, the Bayesian network is transformed into a *credal network*, for which a standard updating problem has to be solved. Finally, a number of transformations that might simplify the updating of the resulting credal network is provided.

1 Introduction

Bayesian networks ([14], Section 2.1) are graphical tools allowing for the specification of a probabilistic model through combination of local models, each involving only some of the variables in the domain. This feature makes those probabilistic graphical models particularly effective for capturing expert knowledge about specific problems. Once the network has been specified, different kinds of probabilistic information can be extracted from it. Updating beliefs about a variable of interest after the observation of some other variables is a typical task, that might be solved through standard inference algorithms. The underlying assumption for this task is that the observations are able to reveal the actual state of the variables in a fully reliable way. Yet, in many situations this might not be true: unreliable sensors, imperfect diagnostic tests, qualitative observations, etc. are not always supposed to be correct. This problem is addressed

L. Godo and A. Pugliese (Eds.): SUM 2009, LNAI 5785, pp. 28–39, 2009.

in Section 4. Following a standard epistemological paradigm, we regard the outcomes of the observations as the (actual) values of *manifest* variables, which are in general distinguished from the *latent* (and hence directly unobservable) variables, whose states we try to detect through the observations.

Following these ideas, we might regard at a Bayesian network implementing a knowledge-based systems as a model of the relations between a set of latent variables, which should be augmented by a manifest variable for each observation we perform. This step is achieved in Section 5 by a simple graphical transformation. We indeed model the particular *observational process* relating the manifest to the latent variables by an appropriate quantification of the corresponding conditional probabilities. In order to provide a flexible model of these relations, we adopt *credal sets*, i.e., convex sets of probability mass functions ([11], Section 2.2). Without any lack of generality, this quantification can be achieved in a very compact way: we simply need to estimate the bounds for the likelihoods of the observations corresponding to the different values of the observed variables (this is proved on the basis of the technical results reported in Section 3). Our approach is quite general, and we can easily generalize to set-valued probabilistic assessments many kinds of observations (e.g., data missing-at-random [12] and Pearl's virtual evidence method [14, Sect. 2.2.2]).

Overall, the method we present transforms the original Bayesian network into a *credal network* ([5], Section 2.3). In the new updating problem all the observed variables are manifest, and the outcomes of the observation should therefore correspond to the actual state of the relative variables. Thus, we can employ standard inference algorithms for credal networks to update our beliefs. Moreover, in Section 6, we describe a simple variable elimination procedure for some of the variables in the credal network: the manifest variables associated to latent variables that are root or leaf nodes in the original Bayesian network can be eliminated by simple local computations. Conclusions and outlooks are in Section 7, while the proofs of the theorems are in Appendix A.

2 Bayesian and Credal Networks

In this section we review the basics of Bayesian networks (BNs) and their extension to convex sets of probabilities, i.e., credal networks (CNs). Both the models are based on a collection of random variables $\mathbf{X}$, which take values in finite sets, and a directed acyclic graph $\mathcal{G}$, whose nodes are associated to the variables of $\mathbf{X}$. For both models, we assume the *Markov condition* to make $\mathcal{G}$ represent probabilistic independence relations between the variables in $\mathbf{X}$: every variable is independent of its non-descendant non-parents conditional on its parents. What makes BNs and CNs different is a different notion of independence and a different characterization of the conditional mass functions for each variable given the possible values of the parents, which will be detailed next.

Regarding notation, for each $X_i \in \mathbf{X}$, Ω_{X_i} is the possibility space of X_i, x_i a generic element of Ω_{X_i}. For binary variables we simply set $\Omega_{X_i} = \{x_i, \neg x_i\}$. Let $P(X_i)$ denote a mass function for X_i and $P(x_i)$ the probability that $X_i = x_i$. A

similar notation with uppercase subscripts (e.g., X_E) denotes arrays (and sets) of variables in $\mathbf{X}$. The parents of X_i, according to $\mathcal{G}$, are denoted by Π_i and for each $\pi_i \in \Omega_{\Pi_i}$, $P(X_i|\pi_i)$ is the conditional mass function for X_i given the joint value π_i of the parents of X_i. Moreover, the nodes of $\mathcal{G}$ without children and without parents are called respectively *leaf* and *root* nodes.

2.1 Bayesian Networks

In the case of BNs, the modeling phase involves specifying a conditional mass function $P(X_i|\pi_i)$ for each $X_i \in \mathbf{X}$ and $\pi_i \in \Omega_{\Pi_i}$; and the standard notion of probabilistic independence is assumed in the Markov condition. A BN can therefore be regarded as a joint probability mass function over $\mathbf{X} \equiv (X_1, \ldots, X_n)$, that factorizes as follows: $P(\mathbf{x}) = \prod_{i=1}^{n} P(x_i|\pi_i)$, for each $\mathbf{x} \in \Omega_{\mathbf{X}}$, because of the Markov condition. The updated belief about a queried variable X_q, given some evidence $X_E = x_E$, is:

$$P(x_q|x_E) = \frac{\sum_{x_M} \prod_{i=1}^{n} P(x_i|\pi_i)}{\sum_{x_M, x_q} \prod_{i=1}^{n} P(x_i|\pi_i)}, \tag{1}$$

where $X_M \equiv \mathbf{X} \setminus (\{X_q\} \cup X_E)$, the domains of the arguments of the sums are left implicit and the values of x_i and π_i are consistent with (x_q, x_M, x_E). Despite its hardness in general, Eq. (1) can be efficiently solved for polytree-shaped BNs with standard propagation schemes based on local computations and message propagation [14]. Similar techniques apply also for general topologies with increased computational time. Fig. 1 reports a simple knowledge-based system based on a small BN to be used in the rest of the paper to explain how our ideas apply in practice.

2.2 Credal Sets

CNs relax BNs by allowing for *imprecise probability* statements: in our assumptions, the conditional mass functions of a CN are just required to belong to a finitely generated *credal set*, i.e., the convex hull of a finite number of mass functions over a variable. Geometrically, a credal set is a *polytope*. A credal set contains an infinite number of mass functions, but only a finite number of *extreme mass functions*: those corresponding to the *vertices* of the polytope, which are, in general, a subset of the generating mass functions. It is possible to show that updating based on a credal set is equivalent to that based only on its vertices. A credal set over X will be denoted as $K(X)$. It is easy to verify that, if X is binary, any credal set has at most two vertices and can be specified by a single linear constraint $l \leq P(x) \leq u$.

2.3 Credal Networks

In order to specify a CN over the variables in $\mathbf{X}$ based on $\mathcal{G}$, a collection of conditional credal sets $K(X_i|\pi_i)$, one for each $\pi_i \in \Omega_{\Pi_i}$, should be provided

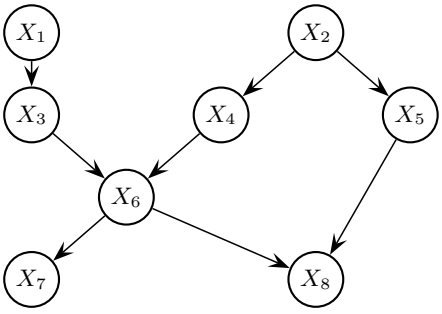


Fig. 1. The Asia network [10]. This is a small BN that calculates the probability of a patient having tuberculosis, lung cancer or bronchitis respectively based on different factors. The semantic of the variables, which are all binary and referred to a particular patient is: *he visited Asia* (X_1), *he smokes* (X_2), *he has tuberculosis* (X_3), *he has lung cancer* (X_4), *he has bronchitis* (X_5), *he has tuberculosis or cancer* (X_6), *his X-rays show something* (X_7), *he has dyspnea* (X_8).

separately for each $X_i \in \mathbf{X}$; while, regarding Markov condition, we assume *strong independence* [6]. A CN associated to these local specifications is said to be with *separately specified* credal sets. The specification becomes global considering the *strong extension* of the CN, i.e.,

$$K(\mathbf{X}) \equiv \mathrm{CH}\left\{ \prod_{i=1}^{n} P(X_i|\Pi_i) : P(X_i|\pi_i) \in K(X_i|\pi_i) \quad \begin{array}{l} \forall \pi_i \in \Omega_{\Pi_i}, \\ \forall i = 1, \ldots, n \end{array} \right\}, \tag{2}$$

where CH denotes the convex hull of a set of functions. Let $\{P_k(\mathbf{X})\}_{k=1}^{n_v}$ denote the set of vertices of $K(\mathbf{X})$. It is an obvious remark that, for each $k = 1, \ldots, n_v$, $P_k(\mathbf{X})$ is the joint mass function of a BN over $\mathcal{G}$. For this reason a CN can be regarded as a finite set of BNs. In the case of CNs, updating is intended as the computation of tight bounds of the probabilities of a queried variable, given some evidences, i.e., Eq. (1) generalizes as:

$$\underline{P}(x_q|x_E) = \min_{k=1,\ldots,n_v} \frac{\sum_{x_M} \prod_{i=1}^{n} P_k(x_i|\pi_i)}{\sum_{x_M, x_q} \prod_{i=1}^{n} P_k(x_i|\pi_i)}, \tag{3}$$

and similarly with a maximum replacing the minimum for upper probabilities $\overline{P}(x_q|x_E)$. Exact updating in CNs displays higher complexity than in BNs: updating in polytree-shaped CNs is NP-complete, and $\mathrm{NP}^{\mathrm{PP}}$-complete in general CNs [7]. Also non-separate specifications of CNs can be provided [2]. Here, we sometimes consider *extensive* specifications where a list of possible values for the conditional probability tables $P(X_i|\Pi_i)$ is provided instead of the (separate) specification of the conditional credal sets $K(X_i|\pi_i)$ for each $\pi_i \in \Omega_{\Pi_i}$.

3 Equivalence Relations for Credal Networks Updating

In this section, we prove some simple equivalences between CNs with respect to the updating problem. The results are simple and the proofs are reported in App. A for sake of completeness. Let us consider the updating of a CN as in Eq. (3). We obtain a new CN through each one of the four following transformations.

Transformation 1. *For each $X_i \in X_E$, iterate the following operations for each children X_j of X_i (i.e., for each X_j such that $X_i \in \Pi_j$): (i) remove from $\mathcal{G}$ the arcs $X_i \to X_j$; (ii) redefine the conditional credal sets of X_j as $K(X_j|\pi'_j) := K(X_j|\pi'_j, x_i)$ for each $\pi'_j \in \Omega_{\Pi'_j}$, with $\Pi'_j = \Pi_j \setminus \{X_i\}$ and the value of x_i consistent with x_E.*

Transformation 2. *Assume all the nodes in X_E to be leaf. For each $X_i \in X_E$, iterate the following operations: (i) make X_i a binary variable by shrinking its possibility space to $\Omega'_{X_i} := \{x_i, \neg x_i\}$; (ii) redefine the conditional credal sets $K(X'_i|\pi_i)$ by the constraint*

$$\underline{P}(X_i = x_i|\pi_i) \leq P(X'_i = x_i|\pi_i) \leq \overline{P}(X_i = x_i|\pi_i), \tag{4}$$

for each $\pi_i \in \Omega_{\Pi_i}$, with the values of x_i consistent with x_E.

Transformation 3. *Assume all the nodes in X_E to be leaf, binary, and with the same (and unique) parent X_j. Transform the (joint) variable X_E into a (single) binary variable with $\Omega_{X_E} := \{x_E, \neg x_E\}$. Define its conditional credal sets $K(X_E|x_j)$ by the constraints*

$$\prod_{i \in E} \underline{P}(x_i|x_j) \leq P(x_E|x_j) \leq \prod_{i \in E} \overline{P}(x_i|x_j), \tag{5}$$

for each $x_j \in \Omega_j$, with the values of x_i consistent with x_E.

Transformation 4. *Assume all the nodes in X_E leaf and binary. For each $X_i \in X_E$, consider an extensive quantification $\{P_k(X_i|\Pi_i)\}_{k=1}^{m_i}$ of its conditional probability table.*[1] *For each $k = 1, \ldots, m_i$, define the table $P'(X_i|\Pi_i)$ such that $P'_k(x_i|\pi_i) = \alpha_k \cdot P_k(x_i|\pi_i)$ for each $\pi_i \in \Omega_{\Pi_i}$, where α_k is an arbitrary nonnegative constant such that all the new probabilities remains less than or equal to one. Then, let $\{P'_k(X_i|\Pi_i)\}_{k=1}^{m_i}$ be the new extensive specification for the node X_i.*

Theorem 1. *The updating problem in Eq. (3) can be equivalently discussed in the CNs returned by the Transformation 1, 2, 3 and 4.*

According to this result, we have that, with respect to belief updating: (i) observed nodes can be always regarded as leaf and binary; (ii) only the (lower and

[1] If the credal set are separately specified, we can obtain an extensive specification by taking all the possible combinations of the vertices of the credal sets.

upper) probabilities for the observed state matter; (iii) if two observed nodes have a single and common parent, they can be regarded as a single node; (iv) if the quantification of the observed node is extensive (or is formulated in an extensive form) what really matters are the ratios between the probabilities of the observed state for the different value of the relative latent variable.

4 The Observational Process

Both BNs and CNs are widely used in AI for the implementation of knowledge-based systems. In fact, the decomposition properties of these graphical tools allow for a compact and simple modeling of the knowledge of an expert. Inference algorithms solving the updating tasks in Eq. (1) and Eq. (3) can be indeed used to extract probabilistic information about the posterior beliefs for the variable of interest X_q after the observation of x_E for the joint variables X_E. Yet, the idea that the outcomes in x_E would always correspond to the actual states of the variables in X_E is not always realistic. In general situations, the outcome of an observation should be better regarded as a different variable from the one we try to measure.

This idea follows an epistemological paradigm in the conceptualization of theoretical systems: the *holistic construal* proposed in [3] for the representation of a theory. This is based on a distinction between theoretical constructs, that are unobservable (latent) real entities, and empirical constructs (or measures), that are observable (manifest) empirical entities representing the observation, perception or measurement of theoretical constructs. Accordingly, any theory can be divided into two parts: "one that specifies relationships between theoretical constructs and another that describes relationships between (theoretical) constructs and measures".

In the case of probabilistic theories, this seems to corresponds to the approach to the modeling of missing data in [17], where a *latent level* and a *manifest level* of the information are distinguished. In the first, theoretical constructs are represented as latent variables[2] and the relationships between them are represented through conditional (in)-dependence relations and conditional probabilities. In the manifest level, the observations (empirical constructs) are represented as manifest variables and the correspondence rules, i.e., the relationships between the underlying theoretical constructs and their observations, are represented through conditional (in)-dependence relations and conditional probabilities. The sum of the observations and the correspondence rules is called *observational process*. The result of this approach is a single model, embedding both the formal theory and the observational process.

Note that the interpretation associated to the latent variables is *realistic* [4]: they are considered as (unobservable) real entities that exist independent of their

[2] Following [15], a *latent variable* is a random variable whose realizations are unobservable (hidden), while a *manifest variable* is a random variable whose realizations can be directly observed.

observation. In the realistic view, latent variables (theoretical entities) are considered to be causes of the observed phenomena and are usually modeled through a *reflective model* [4,9], i.e., a model where manifest variables (observations) are specified conditional on the underlying latent variables.

5 Modeling the Observations

5.1 A Simple Example

Let us explain by means of the example in Fig. 1 how the ideas of the previous section may apply to BNs, The fact that a patient have or not cancer (or tuberculosis, dyspnea, etc.) is clearly a real fact (*realistic interpretation*), but in general we can verify that only through tests that are not fully reliable. In order to decide whether or not the patient is a smoker (or has been to Asia, etc.) we can just ask him, but no guarantees about the truth of its answer are given. Similarly, the reliability of tests measuring dyspnea are not always accurate [13] (or an x-ray might blurred, etc.). Similar considerations can be done for any variable in the BN, which should be therefore regarded as the latent level of our theory (and all its variables as latent ones). In fact, the answers of the patient about having been in Asia or being a smoker, as well as the outcomes of the test of dyspnea or of the observation of the x-ray should be therefore regarded as the (actual) values of new manifest variables.

Generally speaking, only the relative latent variable is expected to affect the state of a manifest variable (*reflective model*). Accordingly, the fact that the patient would tell us whether he smokes or not might be influenced by the fact that he really smokes or not (e.g., we could assume that a non-smoker would never lie, while most of the smokers would do that). On the contrary, it seems hard to assume that the fact that he visited or not Asia might affect his answer about smoking. These assumptions makes it possible to embed the new manifest variables into the original network as depicted in Fig. 2.

5.2 The General Transformation

Now let us consider a generic BN over the variables $\mathbf{X} := (X_1, \ldots, X_n)$ implementing a knowledge-based system. Let $X_E \subset \mathbf{X}$ denote the variables we observe. Assume that these observations might be not fully reliable (and hence it might have sense to consider many observations for a single variable). For each $X_i \in X_E$, let o_i^j denote the outcome of the j-th observation of X_i, with $j = 1, \ldots, n_i$. The outcomes o_i^j are regarded as the actual values of a corresponding set of manifest variables O_i^j, while the variables in X_E as well as the other variables in $\mathbf{X}$ are regarded as latent variables. In general the outcome of an observation might also be missing, and we therefore set $o_i^j \in \Omega_{X_i} \cup \{*\} =: \Omega_{O_i^j}$ for each $i = 1, \ldots, n$ and $j = 1, \ldots, n_i$, where $*$ denotes a missing outcome. Let also $\mathbf{O} := \{O_i^j\}_{i=1,\ldots,n}^{j=1,\ldots,n_i}$.

At this point, we should extend the original BN defined over $\mathbf{X}$ to a probabilistic model over the whole set of variables $(\mathbf{X}, \mathbf{O})$. To this aim, we assume

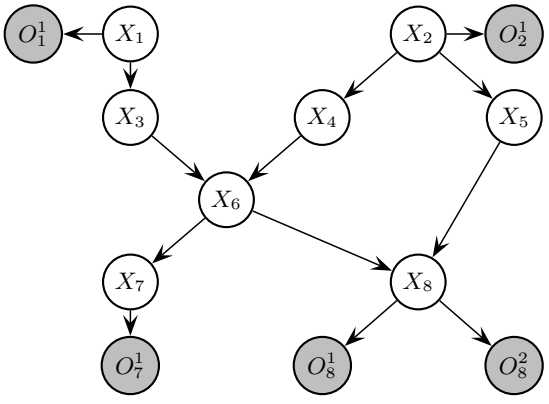


Fig. 2. The Asia network in Fig. 1 with manifest variables (gray nodes) modeling the observations of four latent variables. Two different tests about dyspnea are considered.

each manifest variables to be affected only by its corresponding latent variable.[3] In the probabilistic framework this corresponds to the conditional independence between a manifest variables and all the other variables given its relative latent variable. According to the Markov condition this means that the manifest variable is a child of its latent variable. Thus, we augment the directed acyclic graph over $(\mathbf{O}, \mathbf{X})$ by simply adding to $\mathcal{G}$ an arc $X_i \to O_i^j$, for each $j = 1, \ldots, n_i$ and $i = 1, \ldots, n$.

Finally, to complete the specification of the model, we quantify $P(O_i^j|x_i)$ for each $i = 1, \ldots, n$, $j = 1, \ldots, n_i$, $x_i \in \mathbf{X}_i$. In the BNs framework, those probabilistic values should be precisely assessed. This might be problematic because statistical data about these relations are rarely available, while the quantification is often based on expert qualitative judgements. For this reason, it seems much more realistic, at least in general, to allow for an *imprecise-probability* specifications and give therefore the expert the freedom to model his uncertainty about the observational process through credal sets $K(O_i^j|x_i)$ instead of precise probability mass functions $P(O_i^j|x_i)$.

Overall, this procedure transforms a BN without an explicit model of the observations into a CN which embeds in its structure the model of the observational process.

Finally, let us detail how the results in Th. 1 allow for a simplification of the CN to be updated. First, the manifest variables in $\mathbf{O}$ can be always assumed leaf. This basically means that we can model situation where the outcome of some observation is affected by the outcome of some other observation without connecting the corresponding nodes (provided that in the quantification of the corresponding credal sets we take care of this relations). As an example, the

[3] In principle also more general situations can be considered without big problems. Here we focus on this assumption as in real application it is very often satisfied.

topology in Fig. 2 could model a situation where the outcome of the first of dyspnea (O_8^1) has an effect on the second test (O_8^2) without tracing any arc between these two nodes. These manifest variables can be also assumed to be binary, and we set $\Omega_i^j := \{o_i^j, \neg o_i^j\}$ for each $i = 1, \ldots, n$ and $j = 1, \ldots, n_i$. This basically means that, in the quantification of $K(O_i^j|x_i)$, we might simply assess $\underline{P}(o_i^j|x_i)$ and $\overline{P}(o_i^j|x_i)$ for each $x_i \in \Omega_i$. Note that these numbers are the lower and upper *likelihood* for the outcome of the observation corresponding to the different values of the relative latent variable. If the specification is extensive, we can also unequivocally describe the quantification in terms of a collection of *likelihood ratios*. Finally, we can cluster together the manifest variable corresponding to the observations of a latent variable.

5.3 Possible Models of the Observations

Overall, the problem of updating beliefs in a BN with unreliable observations has been mapped into a "standard", in the sense that now only manifest variables are observed, updating problem for a CN. The above procedure is quite general and proper specifications of the conditional credal sets can be used to describe many different observational process, including various cases of unreliable, incomplete, uncertain, and also missing observations. Let us describe how this can be done in some important cases.

- Fully reliable observation: the outcome cannot be missing, and we have $P(o_i|x_i) = 1$ if $o_i = x_i$ and zero otherwise. This means that the manifest and the latent variable coincides and, in practice, an explicit modeling of the observation is unnecessary.
- Fully unreliable observation: this condition of complete ignorance is modeled by a *vacuous* quantification of the credal sets $K(O_i|x_i)$. This corresponds to the *conservative updating rule* (CUR) for modeling missing data [8]. Notably, the specification employed in [1] to describe the same situation can be regarded as an equivalent (but extensive) specification of the same model.
- Missing-at-random (MAR, [12]): this is a model of an unselective missingness process, i.e., $P(O_i = *|x_i)$ constant for each $x_i \in \Omega_i$. Note also that the situation where we assume to *do not observe* a variable is a particular case of MAR (with the constant equal to one). A situation of this kind corresponds to assume (unconditional) independence between O_i and X_i. Accordingly we can remove the arc $X_i \to O_i$, and that proves that also in this case the explicit modeling of the observational process in unnecessary. Yet, in our framework we can consider relaxed version of the MAR assumption, where for instance, all the conditional probabilities are not all equal, but are supposed to vary in the same intervals. This seems particularly suited to model *epistemic irrelevance* and should be investigated in more detail in the future.
- Pearl's virtual evidence [14, Sect. 2.2.2]): the method proposed by Pearl for modeling ambiguous observations can be clearly modeled in our framework. Moreover, we can easily consider situations where sets of likelihood ratios

(corresponding to an extensive specification) or a single interval-valued likelihood ratio (corresponding to a separate specification) could be reported.

6 Variable Elimination for Root and Leaf Nodes

In this section we call *observational* CN, a CN obtained from a BN after the transformation in Section 5.2. In order to update our beliefs about the variable of interest in the original BN after the unreliable observations, an updating problem as in Equation 3 should be therefore solved for the observational CN. With respect to the original BN, the size of the CN is larger because we have augmented the model with a new variable for each observation. Yet, in the previous section we have already shown that the observations of a same latent variable can be equivalently clustered into a single variable. Here, we want to show how is possible to eliminate the observation nodes corresponding to the latent variables that are root or leaf nodes in the original BN.[4] The results are based on two simple transformations.

Transformation 5. *Let X_i be a latent variable corresponding to a root node in the original BN (which is clearly a root node also in the observational CN), and O_i the relative manifest variable. Consider the unconditional probability mass function $\tilde{P}(X_i)$ such that*

$$\tilde{p}(x_i) := \left[1 + \frac{\sum_{x'_i \neq x_i \in \Omega_i} p(x'_i) \cdot p(o_i|x'_i)}{p(x_i) \cdot p(o_i|x_i)}\right]^{-1}, \tag{6}$$

for each $x_i \in \Omega_i$, with $P(o_i|x_i) \in \{\underline{P}(o_i|x_i), \overline{P}(o_i|x_i)\}$, and where $P(X_i)$ is the (unconditional) mass function associated to X_i. Let $K(X_i)$ be the convex hull of the $2^{|\Omega_i|}$ possible specifications of $\tilde{P}(X_i)$. Finally, replace $P(X_i)$ with $K(X_i)$ in the specification of the CN, and remove O_i from it.

Theorem 2. *An updating problem for an observational CN can be equivalently discussed in the CN obtained by applying Transformation 5 to each node that was a root node in the original BN.*

Something similar can be done also for the latent variables that are leaf nodes in the original BN.

Transformation 6. *Let X_i be a latent variable corresponding to a leaf node in the original BN, and O_i the relative manifest variable. For each $\pi_i \in \Omega_{\Pi_i}$, consider the conditional probability mass function $\tilde{P}(O_i)$ such that:*

$$\tilde{p}(o_i|\pi_i) := \sum_{x_i \in \Omega_i} P(o_i|x_i) \cdot P(x_i|\pi_i), \tag{7}$$

[4] This assumption is not particularly constraining as, for most of the BNs implementing knowledge-based systems, the variables to be observed are of this kind.

with $P(o_i|x_i) \in \{\underline{P}(o_i|x_i), \overline{P}(o_i|x_i)\}$*, and where* $P(X_i|\pi_i)$ *is a conditional mass function associated to* X_i*. Let* $K(O_i|\pi_i)$ *denote the credal set whose two vertices produce the minimum and the maximum for* $\tilde{P}(O_i|\pi_i)$ *among all the* $2^{|\Omega_i|}$ *possible specifications of* $\tilde{P}(O_i|\pi_i)$*. Remove* X_i *from the network, and let the parents* Π_i *of* X_i *to become parents of* O_i*. Finally, set* $K(O_i|\pi_i)$ *as the conditional credal sets associated to* O_i *for each* $\pi_i \in \Omega_{\Pi_i}$*.*

Theorem 3. *An updating problem for an observational CN can be equivalently discussed in the CN obtained by applying Transformation 6 to each node that was a leaf node in the original BN.*

As an example of the application of Th. 2, we can remove O_1^1 and O_2^1 from the observational CN in Fig. 2 by means of Transformation 5, and hence by an appropriate redefinition of the unconditional credal sets for the nodes X_1 and X_2. Similarly, in order to apply Th. 3, we can remove X_7 and X_8 (which is replaced by the cluster of O_8^1 and O_8^2 by means of Transformation 6.

7 Conclusions and Outlooks

We have defined a general protocol for the modeling of unreliable observation in Bayesian networks. The procedure transform the model into a credal network, for which some procedures for variable elimination by local computations are proposed. As a future direction for this research we want to develop specific algorithm for the particular class of credal networks returned by the transformation. Moreover, we want to further investigate the possible models of uncertain and missing observation that can be describe by our framework.

References

1. Antonucci, A., Zaffalon, M.: Equivalence between bayesian and credal nets on an updating problem. In: Lawry, J., Miranda, E., Bugarin, A., Li, S., Gil, M.A., Grzegorzewski, P., Hryniewicz, O. (eds.) Soft Methods for Integrated Uncertainty Modelling, Springer (Proceedings of the third international conference on Soft Methods in Probability and Statistics: SMPS 2006), pp. 223–230. Springer, Heidelberg (2006)
2. Antonucci, A., Zaffalon, M.: Decision-theoretic specification of credal networks: A unified language for uncertain modeling with sets of bayesian networks. Int. J. Approx. Reasoning 49(2), 345–361 (2008)
3. Bagozzi, R.P., Phillips, L.W.: Representing and testing organizational theories: A holistic construal. Administrative Science Quarterly 27(3), 459–489 (1982)
4. Borsboom, D., Mellenbergh, G.J., Heerden, J.v.: The theoretical status of latent variables. Psychological Review 110(2), 203–219 (2002)
5. Cozman, F.G.: Credal networks. Artificial Intelligence 120, 199–233 (2000)
6. Cozman, F.G.: Graphical models for imprecise probabilities. Int. J. Approx. Reasoning 39(2-3), 167–184 (2005)
7. de Campos, C.P., Cozman, F.G.: The inferential complexity of Bayesian and credal networks. In: Proceedings of the International Joint Conference on Artificial Intelligence, Edinburgh, pp. 1313–1318 (2005)

8. de Cooman, G., Zaffalon, M.: Updating beliefs with incomplete observations. Artificial Intelligence 159, 75–125 (2004)
9. Edwards, J.R., Bagozzi, R.P.: On the nature and direction of relationships between constructs and measures. Psychological Methods 5(2), 155–174 (2000)
10. Lauritzen, S.L., Spiegelhalter, D.J.: Local computations with probabilities on graphical structures and their application to expert systems. Journal of the Royal Statistical Society (B) 50, 157–224 (1988)
11. Levi, I.: The Enterprise of Knowledge. MIT Press, London (1980)
12. Little, R.J.A., Rubin, D.B.: Statistical Analysis with Missing Data. Wiley, New York (1987)
13. Mahler, D.A.: Mechanisms and measurement of dyspnea in chronic obstructive pulmonary disease. The Proceedings of the American Thoracic Society 3, 234–238 (2006)
14. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann, San Mateo (1988)
15. Skrondal, A., Rabe-Hesketh, S.: Generalized latent variable modeling: multilevel, longitudinal, and structural equation models. Chapman and Hall/CRC, Boca Raton (2004)
16. Zaffalon, M.: The naive credal classifier. Journal of Statistical Planning and Inference 105(1), 5–21 (2002)
17. Zaffalon, M., Miranda, E.: Conservative inference rule for uncertain reasoning under incompleteness. Journal of Artificial Intelligence Research 34, 757–821 (2009)

A Proofs

Proof (Theorem 1). Let us first prove the thesis for Tr. 1 and Tr. 2 in the special case where the CN is a BN and X_E is made of a single node. It is sufficient to observe that in both the numerator and denominator of the second side of Eq. (1) there is no sum over x_E. Thus, we simply have the proof, which can be easily extended to the case where X_E has more than a single variable, and to the case of CNs by simply considering Eq. (3). Regarding the result for Tr. 3, it is sufficient to consider the expression in [16, Eq. 3.8] by assuming the prior precise. Finally, for Tr. 4, we can consider the result in [14, Sect. 2.2.2] for the case of BNs, and then generalize it to CNs by simply regarding a CN as a collection of BNs.

Proof (Theorems 2 and 3). By Tr. 5 we compute the conditional credal set $K(X_i|o_i)$, while by Tr. 6 we marginalize out X_i from the conditional credal set $K(O_i, X_i|\pi_i)$. The results therefore follows from the d-separation properties of CNs.

Interventions in Possibilistic Logic

Salem Benferhat[1], Didier Dubois[2], and Henri Prade[2]

[1] CRIL - Université d'Artois
Rue Jean Souvraz, SP 18, 62307 Lens Cedex, France
[2] IRIT, CNRS and University of Toulouse, 31062 Toulouse Cedex 9, France
benferhat@cril.univ-artois.fr, dubois@irit.fr, prade@irit.fr

Abstract. An intervention is a tool that enables us to distinguish between causality and simple correlation. The use of interventions has been only implemented in Bayesian net structures (or in their possibilistic counterpart) until now. The paper proposes an approach to the representation and the handling of intervention-like pieces of knowledge, in the setting of possibilistic logic. It is compatible with a modeling of the way agents perceive causal relations in reported sequences of events, on the basis of their own beliefs about how the world normally evolves. These beliefs can also be represented in a possibilistic logic framework.

1 Introduction

Modelling causation is known to be a difficult task. For instance, probabilistic approaches to this issue have often been controversial, because the notion of dependence in probability theory is symmetric. The use of Bayesian networks that rely on directed acyclic graphs may have seemed to address the issue, interpreting the directed arcs as a form of causation. However, the direction of arcs is merely a matter of convention, as any ranking of the variables would lead to an equally valid network representation. In fact, a Bayesian network is often viewed as a compact representation of a joint probability distribution, instead of the representation of some causal relations between variables. The notion of intervention, which has been informally discussed by philosophers (e.g., [15]), has been formally introduced by Pearl, initially in [7] in the setting of Spohn's ordinal conditional functions [14], but then essentially developed in the setting of probabilistic networks [11], [12], as a tool for distinguishing genuine causality relations from mere correlations. It amounts to describing actions and their results. In order to test if the value of a variable causally affects the value of another, the idea is to imagine an "external" action enforcing the value of the first variable and check if it is enough to alter the value of the other variable. In practice, Pearl's idea, implemented in the directed setting of probabilistic Bayesian networks, amounts to setting the value of some variable and to cut the links that constrain the value of this variable in the original graph, the rest of the network being unchanged.

Recently, Benferhat and Smaoui [3] have introduced the notion of intervention in the setting of possibilistic networks, which are counterparts of probabilistic

L. Godo and A. Pugliese (Eds.): SUM 2009, LNAI 5785, pp. 40–54, 2009.

Bayesian networks in a more qualitative, but still directed, setting. They have shown that the computational handling of interventions is then easier. However, the question arises whether or not the notion of intervention requires the use of graphical representations. Namely, how can intervention be modelled in non-directed representations of knowledge?

In this paper, we discuss the representation of interventions in the non directed framework of possibilistic logic both at the semantic and at the syntactic level. We then show its use for getting rid of spurious causal relations, after a reminder on an approach to causal ascription where generic beliefs are represented by nonmonotonic consequence relations.

2 Possibilistic Logic: A Short Background

Propositional possibilistic logic [5] has been essentially developed as a formalism for handling qualitative uncertainty or preferences in a framework where syntax and proof machinery remain close to the one of classical logic. A possibilistic logic atomic formula is a pair made of i) a well-formed classical propositional logic formula, and ii) a weight expressing its certainty or priority. Thus, a standard atomic possibilistic logic expression is denoted as a pair (ϕ, α), where ϕ is a propositional logic formula and $\alpha \in (0, 1]$ is interpreted as a lower bound of a necessity measure N, i.e., (ϕ, α) is semantically interpreted as $N(\phi) \geq \alpha$, where N is a necessity measure. Necessity measures N are characterized by the decomposability property $N(\phi \wedge \psi) = \min(N(\phi), N(\psi))$, and are dual of possibility measures Π (namely $N(\phi) = 1 - \Pi(\neg\phi)$). A possibilistic logic base is a set of possibilistic logic formulas. It is viewed as a conjunction of atomic possibilistic formulas. A possibilistic logic base can be easily put in clausal form thanks to the decomposability property of necessity measures.

A possibilistic logic base is semantically equivalent to a possibility distribution that restricts the set of interpretations (w. r. t. the considered language) that are more or less compatible with the base. More precisely, a propositional possibilistic logic base $\mathcal{B} = \{(p_i, \alpha_i) | i = 1, n\}$ is semantically associated with the possibility distribution

$$\pi_{\mathcal{B}}(\omega) = \min_{i=1,n} \pi_{(p_i,\alpha_i)}(\omega)$$

with $\pi_{(p_i,\alpha_i)}(\omega) = 1$ if $\omega \vDash p_i$, and $\pi_{(p_i,\alpha_i)}(\omega) = 1 - \alpha_i$ if $\omega \vDash \neg p_i$.

Thus, $\pi_{\mathcal{B}}$ is obtained as the min-based conjunctive combination of the representations of each formula in $\mathcal{B}$. Moreover, an interpretation ω is all the more possible as it does not violate any formula p_i with a high certainty level α_i. So, the possibility degree $\pi_{\mathcal{B}}(\omega)$ is all the less as the weight of the most certain possibilistic logic formula violated by ω is greater. The normalization of the possibility distribution $\pi_{\mathcal{B}}$, i.e., $\exists\ \omega,\ \pi_{\mathcal{B}}(\omega) = 1$ is equivalent to the consistency of the classical logic base $\mathcal{B}^* = \{p_i | i = 1, n\}$.

The basic inference rule in possibilistic logic put in clausal form is the following resolution rule, here written in the propositional case:

$$(\neg p \vee q, \alpha); (p \vee r, \beta) \vdash (q \vee r, \min(\alpha, \beta)).$$

Using this rule repeatedly, a refutation-based proof procedure that is sound and complete w. r. t. the semantics exists for propositional possibilistic logic [5]. When $\forall i,\ \alpha_i = 1$, $\mathcal{B} = \{(p_i, \alpha_i) | i = 1, n\}$ is isomorphic to the classical propositional logic base $\mathcal{B}^* = \{p_i | i = 1, n\}$, and has the same deductive closure.

3 Introducing Interventions

Roughly speaking, interventions on a knowledge base are actions performed on this base, such as setting the values of propositional variables and possibly deleting (or deactivating) formulas, in order to observe the behavior of the base under such changes. It is expected that an intervention has no irreversible effect on the knowledge base.

Following the notation introduced by Pearl [11], we use the new variable $do(s)$ to denote the fact of setting the value of literal s at 'true', this literal being of the form p or $\neg p$. Interventions will be performed only on consistent possibilistic logic bases. Moreover, it is assumed in the next subsection that $\exists \omega, \omega \vDash s$ and $\pi(\omega) = 1$. Namely, interventions should be fully possible apriori (in subsection 3.1). Otherwise, performing an intervention would create an inconsistency.

In the following, we first consider interventions that only amount to enforcing the value of some variables, before discussing interventions where formulas involving enforced variables are actually deleted from the the knowledge base (or made inactive), thus cutting the links with other variables.

3.1 Interventions by Simple Enforcement

If we want to set to true a propositional variable, say p, it seems natural to add it to the considered possibilistic logic base $\mathcal{B}$. But, once p is added, the original possibility distribution $\pi_{\mathcal{B}}$ cannot be recovered from the knowledge of $\pi_{\mathcal{B} \cup \{(p,1)\}} = \min(\pi_{\mathcal{B}}, \pi_{(p,1)})$ where $\pi_{(p,1)}(\omega) = 1$ if $\omega \vDash p$ and $\pi_{(p,1)}(\omega) = 0$ otherwise.

However, a trick can be used for both preserving the information $\pi_{\mathcal{B}}$, and the impact of the input information, viewed as an intervention, assumed for the moment to be elementary, i.e. pertaining to one variable. Let us introduce the new variable $\mathrm{do}(p)$ where p is a literal. The idea is to augment $\mathcal{B}$ by a formula $\neg \mathrm{do}(p) \vee p$, stating that if we intervene on p, then p becomes true. The set of interpretations Ω is then expanded, by adding the new variable $\mathrm{do}(p)$ to the language, into $(\Omega)^{\mathrm{do}(p)} = \{\omega \neg \mathrm{do}(p) \text{ for } \omega \in \Omega\} \cup \{\omega \mathrm{do}(p) \text{ for } \omega \in \Omega\}$. This is a natural technique that allows us to distinguish between the action of enforcing the truth of p, from the mere observation that p is true. Let us take an example.

Example 1. Let us consider a very simple propositional logic base $\mathcal{B}_1 = \{\neg p \vee q\}$, where the only impossible worlds are those where p is true and q is false. (when the weight is not explicitly specified, as in $\mathcal{B}_1$, it simply means that the formula is fully certain, namely it is equal to 1). Indeed, let $\pi_{\mathcal{B}_1}$ be the possibility distribution associated with $\mathcal{B}_1$. As can be seen, $\pi_{\mathcal{B}_1}(\omega) = 1$ if ω is pq, $\neg pq$, or $\neg p \neg q$, and $\pi_{\mathcal{B}_1}(\omega) = 0$ if ω is $p \neg q$.

Assume we study the intervention on q. So there are two cases, either we force $\neg q$ by introducing $\mathrm{do}(\neg q)$, or we force q by introducing $\mathrm{do}(q)$. Let us consider first the case with $do(\neg q)$. Then $\mathcal{B}_1$ becomes $\mathcal{B}_1^{\mathrm{do}(\neg q)} = \{\neg p \vee q, \neg \mathrm{do}(\neg q) \vee \neg q\}$. The last clause expresses the effectiveness of the manipulation.

Table 1. Interventions on $\mathcal{B}_1 = \{\neg p \vee q\}$ by doing $\neg q$ and doing q

	p	q	$\mathrm{do}(\neg q)$	doing $\neg q$ (or not)	$\mathrm{do}(q)$	doing q (or not)
1.	1	1	1	0	1	$1 : p \wedge q \wedge \mathrm{do}(q)$
2.	1	1	0	$1 : p \wedge q \wedge \neg \mathrm{do}(\neg q)$	0	$1 : p \wedge q \wedge \neg \mathrm{do}(q)$
3.	1	0	1	0	1	0
4.	1	0	0	0	0	0
5.	0	1	1	0	1	$1 : \neg p \wedge q \wedge \mathrm{do}(q)$
6.	0	1	0	$1 : \neg p \wedge q \wedge \neg \mathrm{do}(\neg q)$	0	$1 : \neg p \wedge q \wedge \neg \mathrm{do}(q)$
7.	0	0	1	$1 : \neg p \wedge \neg q \wedge \mathrm{do}(\neg q)$	1	0
8.	0	0	0	$1 : \neg p \wedge \neg q \wedge \neg \mathrm{do}(\neg q)$	0	$1 : \neg p \wedge \neg q \wedge \neg \mathrm{do}(q)$

We can observe that $\pi_{\mathcal{B}_1^{\mathrm{do}(\neg q)}}(\omega) = 1$ only if ω is $\neg p \neg q x$ for $x = \mathrm{do}(\neg q)$. See Table 1. Comparing $\pi_{\mathcal{B}_1^{\mathrm{do}(\neg q)}}$ with $\pi_{\mathcal{B}_1}$ reveals what is made impossible by the intervention $do(\neg q)$, namely $p \wedge q$ and $\neg p \wedge q$. Moreover, $\mathcal{B}_1^{\mathrm{do}(\neg q)} \cup \{\mathrm{do}(\neg q)\}$ is logically equivalent to $\{\neg p \wedge \neg q\}$.

Now consider the case where $\mathcal{B}_1$ becomes $\mathcal{B}_1^{\mathrm{do}(q)} = \{\neg p \vee q, \neg \mathrm{do}(q) \vee q\}$. What is made impossible by the intervention $\mathrm{do}(q)$ is then $\neg p \wedge \neg q$. $\mathcal{B}_1^{\mathrm{do}(q)} \cup \{\mathrm{do}(q)\}$ is logically equivalent to $\{q\}$. See Table 1.

Observe also that $\pi_{\mathcal{B}_1} = \max(\pi_{\mathcal{B}_1^{\mathrm{do}(\neg q)}}, \pi_{\mathcal{B}_1^{\mathrm{do}(q)}})$. ◇

It is important to note that our way to handle interventions, by adding extra variables, is fully consistent with the way interventions are handled in Bayesian causal networks. Indeed, one natural way to deal with interventions in graphical models, is to add for each variable A, a new variable "DO_A". The obtained graph is called "augmented graph". Interventions in initial causal graph are handled as observations in the augmented graph. Clearly, this paper follows the same spirit. The following results can be established in the propositional case.

Proposition 1. *Let* $\mathcal{B}^{do(p)} = \mathcal{B} \cup \{\neg do(p) \vee p\}$. *Then*
$\pi_{\mathcal{B}^{do(p)}}(\omega x) = \pi_B(\omega)$ *if* $x = \neg do(p)$;
$\pi_{\mathcal{B}^{do(p)}}(\omega x) = \pi_{\mathcal{B} \cup \{(p,1)\}}(\omega)$ *if* $x = do(p)$.
$\pi_{\mathcal{B}} = \max(\pi_{\mathcal{B}^{do(p)}}, \pi_{\mathcal{B}^{do(\neg p)}})$.

Clearly, what is impossible before an intervention remains impossible after in the above view. This will not be always the case with the type of interventions we consider next, since then some clause(s) of the initial base could be put aside.

3.2 Interventions with Cut

In the above view of the idea of intervention, only the effect of setting the value of a variable has been described. In the above example, when applying $\mathrm{do}(\neg q)$, we have kept $\neg p \vee q$ in $\mathcal{B}_1^{\mathrm{do}(\neg q)}$, thus continuing to enforce the constraint "if q is false then p is false".

However, there is a stronger view of the idea of intervention, used by Pearl [12], where when "doing", we do not only set the value of a variable, but we should also cut the links with this variable. The intuition behind this cutting of links, is that interventions are results of external actions, and hence beliefs on direct causes of the intervened variable should not change. This is now illustrated on an example.

Example 2. We consider the example discussed by Sloman and Lagnado [13]. We have three balls A, B, C, put in this order that may roll in a gauge. The base $\mathcal{B}_2 = \{\neg a \vee b, \neg b \vee c\}$, where a (resp. b, c) means A (resp. B, C) moves, expresses that if A moves, B moves, and if B moves, C moves. Let us consider the intervention $\mathrm{do}(\neg b)$, i.e., B is maintained and does not move.

Let $\mathcal{B}_2^{\mathrm{do}(\neg b)} = \{\neg a \vee b, \neg b \vee c, \neg \mathrm{do}(\neg b) \vee \neg b\}$. As can be easily seen (see Table 2), once $\mathrm{do}(\neg b)$ is performed, the only remaining possible worlds are such that $\neg a \wedge \neg b$ holds.

This result may be found too drastic in the sense that once $\mathrm{do}(\neg b)$ is performed, it may be no longer the case that the laws relating the object of the intervention to the rest of the system still apply. In our example, the fact that, after intervention, ball B can no longer move makes the application of rule "if B does not move, A does not move" debatable, since in some sense the intervention may be viewed as breaking the constraint linking A and B. ◇

The above discussion leads us to the following definition.

Definition 1. *Let $\mathcal{B}$ be a propositional logic base put in clausal form. The intervention aiming at doing p with cut results in a new propositional logic base* $\mathcal{B}^{do(p),cut} = (\mathcal{B} \backslash \{\neg p \vee q \text{ s.t. } \neg p \vee q \in \mathcal{B} \text{ and } q \text{ is a disjunct}\}) \cup_q \{\neg do(p) \vee p, do(p) \vee \neg p \vee q\}$.

Table 2. Sloman and Lagnado' example

	a	b	c	B_2 *no intervention*	$B_2^{do(\neg b)}$ *for* $do(\neg b) = 1$	*intervention with cut*	*intervention on* B_2 *modified*
1.abc	1	1	1	1	0	0	0
2.$ab\neg c$	1	1	0	0	0	0	0
3.$a\neg bc$	1	0	1	0	**0**	**1**	**0**
4.$a\neg b\neg c$	1	0	0	0	**0**	**1**	**1**
5.$\neg abc$	0	1	1	1	0	0	0
6.$\neg ab\neg c$	0	1	0	0	0	0	0
7.$\neg a\neg bc$	0	0	1	1	1	1	1
8.$\neg a\neg b\neg c$	0	0	0	1	1	1	1

Thus, $\mathcal{B}^{\mathrm{do}(p),cut}$ is obtained from $\mathcal{B}$ by deleting all the clauses that can be resolved with p, and adding the success clause $\neg\mathrm{do}(p) \vee p$, as well as weakened versions of the deleted clauses that are now "conditioned" by $\neg\mathrm{do}(p)$ (in order to recover them if the intervention does not take place, while they are inhibited if it takes place).

Note that the deletion of *all* the clauses that can be resolved with the result p of doing the action $\mathrm{do}(p)$ is a matter of cautiousness. Indeed in the above balls example, if it seems natural to consider that "if B does not move, A does not move" is no longer valid whence B is stopped by doing the action (since B no longer interacts with the rest of the system), a rule such as "if B does not move, B is easy to grasp" would not need to be deleted if it was part of the knowledge. However, since there is no straightforward criterion to determine what has to be deleted, the above definition of the intervention with cut, deletes all the clauses that can be resolved with the result of the action of doing.

At the semantic level, the result of an intervention with cut can be described in the following way.

Proposition 2. *Let $\mathcal{B}$ be a consistent propositional logic base put in clausal form. Let $do(p)$ be an intervention to be performed with cut. Let $\mathcal{B}^{\neg p} = \{\neg p \vee q$ s.t. $(\neg p \vee q) \in \mathcal{B}$ and q is a disjunct$\}$. Let $\mathcal{B}^{do(p),cut}$ denote the resulting base. The models and counter-models of $\mathcal{B}^{do(p),cut}$ can be obtained from those of $\mathcal{B}$ in the following way.*

- *if $x = \neg do(p)$, $\pi_{\mathcal{B}^{do(p),cut}}(\omega x) = \pi_B(\omega)$;*
- *if $x = do(p)$*
  - *$\pi_{\mathcal{B}^{do(p),cut}}(\omega x) = 1$, if $\pi_{\mathcal{B}^{\neg p}}(\omega) = 0$ and $\pi_{\mathcal{B}\setminus\mathcal{B}^{\neg p}}(\omega) = 1$;*
  - *$\pi_{\mathcal{B}^{do(p),cut}}(\omega x) = \pi_{\mathcal{B}\cup\{(p,1)\}}(\omega)$ otherwise.*

Proof: If $x = \neg\mathrm{do}(p)$, then $\neg\mathrm{do}(p) \vee p$ is set to true, and $\mathrm{do}(p) \vee \neg p \vee q$ reduces to $\neg p \vee q$, which restores the deleted formulae. If $x = \mathrm{do}(p)$, then p is also enforced to true but there is no constraint on the truth value of q since $\mathrm{do}(p) \vee \neg p \vee q$ is set to true. So, $\pi_{\mathcal{B}^{\mathrm{do}(p),cut}}(\omega x) = 1$ even if some formulas in $\mathcal{B}^{\neg p}$ are violated, as long as formulas in the rest of $\mathcal{B}$ are satisfied; i.e. it sanctions the deletion of $\neg p \vee q$. QED

This is illustrated on the example of Table 2. In this example, after doing $\neg b$ with cut, $\mathcal{B}_2^{\mathrm{do}(\neg b),cut} = \{\mathrm{do}(\neg b) \vee \neg a \vee b, \neg\mathrm{do}(\neg b) \vee \neg b, \neg b \vee c\}$. When doing $\neg b$, its models reduce to those of $\neg b$. The column "intervention with cut" is obtained by applying the two following rules to the initial column "no intervention" representing $\mathcal{B}_2$:

- the interpretations $abc, \neg abc$, that are possible when b is true, become impossible;
- the interpretations $a\neg bc, a\neg b\neg c$ that are impossible when b is false become possible again (since the formula $\neg a \vee b$ that affects b is dropped when doing $\neg b$), unless they are made impossible due to formulas in $\mathcal{B} \setminus \mathcal{B}^{\neg p}$.

To illustrate this last point, consider the knowledge base $\mathcal{B}$ modified.

Example 3. Consider the set

$$\mathcal{B}_2^{mod} = \{\neg a \vee b, \neg b \vee c\ , \neg a \vee \neg c\}.$$

Then, any interpretation where both a and c are true remains impossible, even if b is made false, since it is made impossible by a formula $\neg a \vee \neg c$ unrelated to b. See especially the values in bold in Table 2. $\diamond$

The approach can handle the case of "combined interventions". Indeed one may fix several variables simultaneously. For instance, in agreement with Definition 2, setting both p and q to true with respect to a base $\mathcal{B}$, amounts to stating

$$\mathcal{B}^{\mathrm{do}(p),\mathrm{do}(q),cut} = \mathcal{B} \setminus \{\neg p \vee r, \neg q \vee s \text{ s.t. } (\neg p \vee r) \in \mathcal{B}, (\neg q \vee s) \in \mathcal{B} \text{ and } r, s \text{ are disjuncts}\} \cup_{r,s} \{\neg \mathrm{do}(p) \vee p, \neg \mathrm{do}(q) \vee q, \mathrm{do}(p) \vee \neg p \vee r, \mathrm{do}(q) \vee \neg q \vee s\}.$$

We have discussed the ideas of interventions without, and with cut in the setting of propositional logic, independently from uncertainty. However, the above approach can be extended to situations with graded certainty levels. It basically amounts to replacing the possibility degrees that are 0, by possibility degrees strictly smaller than 1 corresponding to the violation of possibilistic formulas (p_i, α_i) with $0 < \alpha_i < 1$ (in agreement with the semantics recalled in section 2). The extension is easy. It is not explicitly made here due to space limitation.

4 Ascription of Causal Relations Based on Default Rules

People try to make sense of what they observe or of what is reported to them. Ascribing causality relations is one of the main ways to do it. A formal definition of this process has been recently proposed [6], [4]. Namely, given a context C (C represents the available contextual information that does not evolve), a sequence of events $\neg B_t$, A_t, $B_{t'}$ is reported, where t' denotes a time instant strictly after t (B'_t means that B is reported as true at time t'). Besides, the agent that receives the sequenced information has some knowledge about what is the normal (thus expected) course of things in context C, as well as in context $C \wedge A$, in particular regarding the truth status of B.

The idea that B is normally true in the context C is encoded at the syntactic level by means of a conditional assertion $C \mid\!\sim B$, where $\mid\!\sim$ is a non-classical consequence relation symbol. This language can capture the fact that the agent may either believes that $C \mid\!\sim B$ (the agent expects B to be true in context C), or that $C \mid\!\sim \neg B$ (the agent expects that B to be false in context C), or that $C \not\mid\!\sim B$ and $C \not\mid\!\sim \neg B$ (the truth or the falsity of B is contingent in context C according to the agent), where $\mid\!\sim$ is a nonmonotonic consequence relation stating what is normal, while $\not\mid\!\sim$ denotes its negation. It is assumed that $\mid\!\sim$ obeys the postulates of system P [9]. Similarly, in the context $C \wedge A$, the agent may have the same type of beliefs (it is supposed that $C \wedge A$ is consistent). Note that the truth of A may refer to an action that was performed (e.g. "the driver drank before taking his car"), or to some state of facts that is still true (e.g., "the driver is inebriated"). A may be also an event that does not result from an action (for instance "a storm

has broken out"). The following definition of (perceived) causality is borrowed from [6]:

Definition 2. *(Causality) Suppose that an agent becomes aware of the sequence of events* $\neg B_t$, A_t, $B_{t'}$. *Let* C *be context, i.e. the conjunction of all the other facts known by the agent at instant* $t' > t$. *Let* $\vdash\!\!\sim$ *be a nonmonotonic consequence relation. If the agent believes that* $C \vdash\!\!\sim \neg B$,

- *and then* $C \wedge A \vdash\!\!\sim B$, *the agent perceives* A *in context* C *as being the* cause *of* B *becoming true, which is denoted* $C : A \Rightarrow_{ca} B$).
- *and only thinks that* $C \wedge A \not\vdash\!\!\sim \neg B$, *then the agent considers that in context* C, A has facilitated *the fact that* B *became true.*

Among the noticeable properties of causality defined in this way, let us mention that [4]:

- If $C : A \Rightarrow_{ca} B$ then $C \vdash\!\!\sim \neg A$. It means that a potential cause can be only an abnormal fact with respect to the current context C.
- Causality satisfies a restricted form of transitivity: if $C : A \Rightarrow_{ca} B$, if $C : B \Rightarrow_{ca} D$ and if moreover $B \wedge C \vdash\!\!\sim A$, then $C : A \Rightarrow_{ca} D$.

These two properties are valid for $\Rightarrow_{ca}$ provided that $\vdash\!\!\sim$ is a "preferential" non monotonic consequence relation in the sense of [9].

Note that in this approach, the knowledge the agent is aware of about the normal course of the world, whether correct or not, is qualitative, and separates what is normal in the context from what is exceptional in this context. This distinguishes this approach from the one of Halpern and Pearl [8], which is based on the use of structural equations and demands a richer knowledge about the behavior of the considered system.

Consider the following illustrative example.

Example 4. Mary is told that Peter has a lung cancer, and she knows that he is an heavy smoker, and that until recently he was not ill. Mary believes that in general, one has not a lung cancer, but that if one smokes a lot, it is normal to get it sooner or later, i.e., $C \vdash\!\!\sim \neg$cancer and $C \wedge$ smoking $\vdash\!\!\sim$ cancer. Then Mary considers that "smoking" caused Peter's cancer (i.e., C: smoking $\Rightarrow_{ca}$ cancer). If her beliefs obey the rationality postulates of "preferential" inference, Mary also thinks that generally one does not smoke: $C \vdash\!\!\sim \neg$smoking.

However the predictions provided by this approach may be not satisfactory, as shown now. Indeed suppose that Mary moreover believes that

$C \vdash\!\!\sim \neg$yellow teeth; $C \wedge$ smoking $\vdash\!\!\sim$ yellow teeth;
$C \wedge$ yellow teeth $\not\vdash\!\!\sim \neg$cancer.

i.e., according to Mary, in general teeth are not yellow, smoking makes teeth yellow, and that lung cancer is not exceptional for somebody who has yellow teeth. Thus for Mary, if Peter's teeth are yellow, this leads to think that i) "the fact that Peter is an heavy smoker is the cause of his yellow teeth" (which is satisfactory), but also that ii) "the fact that Peter has yellow teeth facilitated (according to the above definition) the fact that he has a lung cancer" ! If Mary

had the stronger belief that $C \wedge$ yellow teeth $\not\sim$ cancer (thus expressing that yellow teeth are a strong indication of lung cancer), then Mary should conclude that having yellow teeth causes lung cancer!

The notion of intervention provides a way for distinguishing causality from mere correlation. The idea is to simulate the effect of actions (which may be fictitious). Let us denote such an action by $do(A)$ (for distinguishing the action that makes A true from the fact that A is true). In the example, Mary may then refer to a supplementary piece of belief such that: $C \wedge do(\text{yellow teeth}) \not\sim \neg\text{cancer}$, i.e., if the teeth of somebody are made yellow, cancer should remain the exception (Mary already believes that $C \not\sim \neg\text{cancer}$). $\diamond$

Such intervention-based beliefs can be used for blocking undesirable facilitation or causality judgements as the ones mentioned above (note however that the conditionals $C \wedge A \not\sim B$, $C \wedge do(A) \not\sim B$, and $C \wedge do(A) \not\sim A$ constitute a consistent set of beliefs). Before discussing the use of intervention-based beliefs in the next section, we need first to recall how conditionals can be represented in possibilistic logic [2].

A default rule "generally, if p then q" can be represented in the framework of possibility theory by a constraint on a possibility ranking over interpretations, of the form $\Pi(p \wedge q) > \Pi(p \wedge \neg q)$, which expresses that in the context where p is true, having q true is strictly more possible than having q false [2]. In this setting, it has been shown that a collection of default rules represented by a set of such constraints, induces a unique qualitative possibility ranking $>_\pi$ under the form of an ordered partition $(E_1, \ldots, E_m)$ of the set of interpretations (such as if $i > j, \omega \in E_i$ and $\omega' \in E_j$ then $\omega >_\pi \omega'$). This ranking is the least restrictive solution of the considered set of constraints, the one that leads to a minimal number of classes of equally plausible situations.

Example 5. The default rules $C \not\sim \neg$yellow teeth, and $C \wedge$smoking $\not\sim$ yellow teeth correspond to the constraints (where s stands for "smoking" and y for "yellow"):

$$\Pi(C \wedge \neg y) > \Pi(C \wedge y)$$
$$\Pi(C \wedge s \wedge y) > \Pi(C \wedge s \wedge \neg y)$$

Using the max-decomposability of possibility measures w. r. t. disjunction, with the following notations :

$$\Omega = \{\omega_1 = Csy, \omega_2 = C\neg sy, \omega_3 = Cs\neg y, \omega_4 = C\neg s\neg y, \omega_5 = \neg Csy,$$
$$\omega_6 = \neg C\neg sy, \omega_7 = \neg Cs\neg y, \omega_8 = \neg C\neg s\neg y\},$$

the constraints can be rewritten

$$\max(\pi(\omega_3), \pi(\omega_4)) > \max(\pi(\omega_1), \pi(\omega_2)),$$

and

$$\pi(\omega_1) > \pi(\omega_3),$$

which leads to a partition into three classes:

$$\omega_4 =_\pi \omega_5 =_\pi \omega_6 =_\pi \omega_7 =_\pi \omega_8 >_\pi \omega_1 =_\pi \omega_2 >_\pi \omega_3$$

where $\omega_i =_\pi \omega_j$ (resp. $\omega_i >_\pi \omega_j$) translates $\pi(\omega_i) = \pi(\omega_j)$ (resp. $\pi(\omega_i) > \pi(\omega_j)$) in terms of complete preorder. At the syntactic level, this is equivalent to the possibilistic logic base $\{(\neg C \vee \neg y, \alpha), (\neg s \vee \neg C \vee y, \beta)\}$, with $\alpha < \beta$. This means that priority is given to the most specific rule.

Similarly, the set of default rules $\{C \mid\sim \neg\text{yellow teeth}, C \mid\sim \neg\text{cancer}, C \wedge \text{smoking} \mid\sim \text{yellow teeth}, C \wedge \text{smoking} \mid\sim \text{cancer}\}$ can be represented by the possibilistic base:

$$\mathcal{K} = \{(\neg C \vee \neg y, \alpha), (\neg C \vee \neg c, \alpha), (\neg C \vee \neg s \vee y, \beta), (\neg C \vee \neg s \vee c, \beta)\},$$

with $\alpha < \beta$ (general rules are less prioritary than specific and exceptional rules). At the semantic level, without introducing the context C (that does not play any particular role here) in the notations, letting

$$\omega_1 = csy, \omega_2 = c\neg sy, \omega_3 = cs\neg y, \omega_4 = c\neg s\neg y,$$
$$\omega_5 = \neg csy, \omega_6 = \neg c\neg sy, \omega_7 = \neg cs\neg y, \omega_8 = \neg c\neg s\neg y,$$

leads to the following qualitative possibility ranking:

$$\omega_8 >_\pi \omega_1 =_\pi \omega_2 =_\pi \omega_4 =_\pi \omega_6 >_\pi \omega_3 =_\pi \omega_5 =_\pi \omega_7.$$

However, the approach cannot take into account the information

$$C \wedge \text{yellow teeth} \not\mid\sim \neg\text{cancer},$$

which might be thought of as expressible by $\Pi(C \wedge y \wedge c) \geq \Pi(C \wedge y \wedge \neg c)$, and corresponds to the constraint $\max(\pi(\omega 1), \pi(\omega 2)) \geq \max(\pi(\omega 5), \pi(\omega 6))$. But this constraint is already satisfied by the above ranking, and has by the way no syntactic counterpart in possibilistic logic!

Besides, the stronger piece of information "yellow teeth are indications that the person shall develop a lung cancer: $C \wedge$yellow teeth $\mid\sim$ cancer, which corresponds to

$$\Pi(C \wedge y \wedge c) > \Pi(C \wedge y \wedge \neg c),$$

leads to the ranking

$$\omega_8 >_\pi \omega_1 =_\pi \omega_2 =_\pi \omega_4 >_\pi \omega_6 =_\pi \omega_3 =_\pi \omega_5 =_\pi \omega_7$$

and to the base

$$\mathcal{K}' = \{(\neg y, \alpha), (\neg c, \alpha), (\neg s \vee y, \beta), (\neg s \vee c, \beta), (\neg y \vee c, \beta)\}$$

with $\alpha < \beta$. But, we are going to see now that in such a base, causality cannot be set apart from correlation without intervention-based knowledge.

5 Adding Intervention-Based Information

Let us again consider the example that illustrates the problem raised by the ascription of causal relationships. This basic example is made of three statements

- heavy smoking (s) leads to cancer (c);
- heavy smoking (s) makes teeth yellow (y);
- having yellow teeth (y) is an indication for (lung) cancer (c).

Let us make some remarks before considering a logical formalization. Intuitively the two first statements express causal relations and sound "deductive", while the last one is "inductive", according to the opposition (stressed by Pearl [10]) between, e.g., "fire is the cause for smoke", and "seeing smoke suggests the presence of fire". Clearly, the three statements (i)-(ii)-(iii) are generic, but liable to many exceptions, whose rate may be expressed by means of conditional probabilities, or certainty levels.

Let us represent the three statements in the setting of propositional logic, assuming for the moment that they have no exception : $\mathcal{K} = \{\neg s \vee c, \neg s \vee y, \neg y \vee c\}$. Note that in fact it corresponds to the β level cut of the base $\mathcal{K}'$ computed at the end of Section 4 (i.e., the formulas associated to a level at least equal to β), which corresponds to the default rules that are the most specific. At the semantic level, it can be easily checked that $\mathcal{K}$ has only four models. These models are scy, $\neg scy$, $\neg sc\neg y$, $\neg s\neg c\neg y$, which correspond to four possible worlds and can be summarized in at least three different ways, according to the variable we start with:

1. either one smokes and one has a cancer and yellow teeth, or one does not smoke and it is not possible to have yellow teeth without having cancer;
2. either one has yellow teeth and one has a cancer, or one has no yellow teeth and one does not smoke;
3. either one has not a cancer and one has no yellow teeth and one does not smoke, or one has a cancer and it is impossible to smoke without having yellow teeth.

In this poor representation setting, y and s play the same role with respect to c, i.e. it expresses that yellow teeth cause cancer as sure as smoking does. This can be expressed in terms of the possibility distribution:

$\pi_{\mathcal{K}}(\omega) = 1$ if $\omega \in \{scy, \neg scy, \neg sc\neg y, \neg s\neg c\neg y\}$, $\pi_{\mathcal{K}}(\omega) = 0$ otherwise.

As can be seen,

- $\mathcal{K}$ delimits the non impossible worlds;

- in $\mathcal{K}$ one has lost the directed nature of causal links and of the evocation link (which cannot be simply guessed from the presence of the positive and negative literals), and in $\mathcal{K}$ no distinction is made between the two types of links.

Following Pearl [12] (see also Sloman and Lagnado [13] for a psychological validation), the idea is to perform a thought experiment and to express its believed results, i.e., in the example, that "if one makes the teeth of non-smokers yellow, they will not get a cancer for that reason", which suggests the conditional: $\neg$smoking $\wedge\, do(yellowteeth) \mid\!\sim \neg$cancer.

Let $do(y)$ be a new variable expressing that teeth are painted in yellow, then it might seem natural to complete $\mathcal{K}$ in the following way (without performing any cut):

Table 3. Yellow teeth example 1

s	c	y	$\neg s \vee c$	$\neg s \vee y$	$\neg y \vee c$	$\mathcal{K}$
1	1	1	1	1	1	1
1	1	0	1	0	1	0
1	0	1	0	1	0	0
1	0	0	0	0	1	0
0	1	1	1	1	1	1
0	1	0	1	1	1	1
0	0	1	1	1	0	0
0	0	0	1	1	1	1

Table 4. Yellow teeth example 2

s	c	y	$do(y)$	$\neg s \vee c$	$\neg s \vee y$	$\neg do(y) \vee y$	$\neg do(y) \vee s \vee \neg c$	$\mathcal{K}^*_{do(y)}$	$\neg y \vee c$	$\mathcal{K}_{do(y)}$
1	1	1	1	1	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1	1	1
1	1	0	1	1	0	0	1	0	1	0
1	1	0	0	1	0	0	1	0	1	0
1	0	1	1	0	1	1	1	0	0	0
1	0	1	0	0	1	1	1	0	0	0
1	0	0	1	0	0	0	1	0	1	0
1	0	0	0	0	0	1	1	0	1	0
0	1	1	1	1	1	1	0	0	1	0
0	1	1	0	1	1	1	1	1	1	1
0	1	0	1	1	1	0	0	0	1	0
0	1	0	0	1	1	1	1	1	1	1
0	0	1	1	1	1	1	1	1	0	0
0	0	1	0	1	1	1	1	1	0	0
0	0	0	1	1	1	0	1	0	1	0
0	0	0	0	1	1	1	1	1	1	1

$$\begin{aligned}\mathcal{K}_{do(y)} &= \mathcal{K} \cup \{\neg do(y) \vee y, \neg do(y) \vee s \vee \neg c\} \\ &= \{\neg s \vee c, \neg s \vee y, \neg y \vee c, \neg do(y) \vee y, \neg do(y) \vee s \vee \neg c\},\end{aligned}$$

thus adding to $\mathcal{K}$ the fact that painting teeth in yellow makes them yellow, but is not a source of cancer for non-smokers. The only models of $\mathcal{K}_{do(y)}$ are

$$\{scydo(y), scy\neg do(y), negscy\neg do(y), \neg sc\neg y\neg do(y), \neg s\neg c\neg y\neg do(y)\}.$$

We have thus lost information, in particular the inconsistency of $\mathcal{K}$, which states the impossibility of having yellow teeth without cancer (see 1., above), together with $\{\neg do(y) \vee y, \neg do(y) \vee s \vee \neg c\}$, which has $do(y)y\neg s\neg c$ and $do(y)ys\neg c$ among its models, thus stating the non impossibility of having yellow teeth without cancer.

As stated in the directed setting of probabilistic Bayesian nets by Pearl [12], intervention does not only amount to setting the value of a variable, but it also cuts the links that previously constrained its value, the remaining being unchanged. In the example, we thus withdraw formula $\neg y \vee c$, thus withdrawing all the formulas that involve the negative literal $\neg y$, in order to cut the possibility of any logical resolution with $\neg do(y) \vee y$. This gives, once adding the fact that making teeth of a non-smoker yellow does not lead to cancer:

$$\begin{aligned}\mathcal{K}^*_{do(y)} &= (\mathcal{K} - \{\neg y \vee c\}) \cup \{\neg do(y) \vee y, \neg do(y) \vee s \vee \neg c\} \\ &= \{\neg s \vee c, \neg s \vee y, \neg do(y) \vee y, \neg do(y) \vee s \vee \neg c\}.\end{aligned}$$

The models of $\mathcal{K}^*_{do(y)}$ are given in Table 4. Thus $\mathcal{K}^*_{do(y)}$ has 7 models:

$$\begin{gathered}scydo(y),\ scy\neg do(y),\ \neg scy\neg do(y),\ \neg sc\neg y\neg do(y), \\ \neg s\neg cydo(y),\ \neg s\neg cy\neg do(y),\ \neg s\neg c\neg y\neg do(y),\end{gathered}$$

i.e., either one smokes and one gets a cancer and yellow teeth, or one does not smoke and it is impossible of not having yellow teeth once they have been made yellow, but one may have yellow teeth without being threatened by a cancer: $\neg s\neg cydo(y)$ is a model of $\mathcal{K}^*_{do(y)}$. Thus having yellow teeth no longer causes cancer! Performing a 'do' operation in a logical framework amounts to a "revision" operation, which changes the base $\mathcal{K}$ into the base $\mathcal{K}^*_{do(y)}$ defined above.

In the general case, intervention consists not only of introducing a new variable $do(p)$ controling a literal p and inhibiting the existing causal links bearing on it, but also introducing specific knowledge pertaining to the intervention on p (here, $\neg do(y) \vee s \vee \neg c$), distinguishing between the known effect of intervention on a literal p of interest and the spurious causation between p and t. Then definition 1 becomes

Definition 3. *Let $\mathcal{K}$ be a set of logical formulas in clausal form such that $\mathcal{K} \not\vdash t$ and $\mathcal{K} \not\vdash \neg t$, where t is a literal that appears in $\mathcal{K}$. Let p be another literal appearing in $\mathcal{K}$. The intervention on p in connection with some effect t consists in*

1. *Adding the clause $\neg do(p) \vee p$ to $\mathcal{K}$.*
2. *Adding all known results of intervention on p, of the form $\neg do(p) \vee q \vee \neg t$, that deny the causal influence of doing p on the truth of t.*
3. *Deleting all clauses that unify with p of the form $\neg p \vee r$ and adding substitutes of the form $do(p) \vee \neg p \vee r$.*

Let $\mathcal{K}^{I(p,t)}$ be the result of this transformation. Note that the last point means that one deletes the problematic links such as $\neg p \vee t$ that refers to a spurious causation between p and t.

Proposition 3. *If $\mathcal{K} \not\vdash t$ and $\mathcal{K} \not\vdash \neg t$ then $\mathcal{K}^{I(p,t)}$ is consistent with $p \wedge \neg t$.*

Proof : We must show that $\mathcal{K}^{I(p,t)} \cup \{p\} \not\vdash t$. Since all clauses that unify with p have been modified by adding literal $do(p)$, $\mathcal{K}^{I(p,t)} \cup \{p\}$ can only entail clauses of the form $do(p) \vee r$ by unification with p. Such clauses can also unify with

$\neg do(p) \vee p$ and yield $p \vee r$. As $\mathcal{K} \not\vdash t$ and $\mathcal{K} \not\vdash \neg t$, and there is no additional clause produced by the transformation of $\mathcal{K}$ that can produce t, $\mathcal{K}^{I(p,t)}$ is consistent with $p \wedge \neg t$. In particular, if $\neg p \vee t \in \mathcal{K}$, then only $do(p) \vee t$ and $p \vee t$ can be produced.

This result shows that there is a counter-example to an hypothetical causal link from p to t. For instance on the yellow teeth example, one can check that

$$\mathcal{K}^{I(p,t)} = \{\neg s \vee c, \neg s \vee y, \neg do(y) \vee y, do(y) \vee \neg y \vee c, \neg do(y) \vee s \vee \neg c\},$$

namely step 2 adds the known fact $\neg do(y) \vee s \vee \neg c$ and step 3 removes $\neg y \vee c$ and adds $do(y) \vee \neg y \vee c$. It ensures that making teeth yellow (adding $do(y)$ to the above knowledge base) cannot imply c. However yellow teeth still suggest cancer if this colour is not the effect of an action (this is expressed by adding $\neg do(y)$), but is the result of smoking (when s is a known fact).

This approach provides a tool for deleting from a knowledge base the links that are known to be non-causal due to intervention. Thus, intervention can refine the perception of causal links provided by the default rule-based approach presented in Section 4.

This technique can be generalized to possibilistic logic bases, which thus enables us to work with the representations of default rules. In the example, at the semantic level, the initial Table 3 would become Table 5 below where the interpretations that are completely possible are unchanged. The knowledge pertaining to interventions could then be dealt with directly at the syntactic level in the setting of possibilistic logic, from the modified base $\mathcal{K}^*_{do(y)}$, viewed as a level cut of a possibilistic logic base.

Table 5. Yellow teeth example 3

s	c	j	$\neg s \vee c$	$\neg s \vee y$	$\neg y \vee c$	$\mathcal{K}$
1	1	1	1	1	1	1
1	1	0	1	$1-\beta$	1	$1-\beta$
1	0	1	$1-\alpha$	1	$1-\beta$	$\min(1-\alpha, 1-\beta)$
1	0	0	$1-\alpha$	$1-\beta$	1	$\min(1-\alpha, 1-\beta)$
0	1	1	1	1	1	1
0	1	0	1	1	1	1
0	0	1	1	1	$1-\beta$	$1-\beta$
0	0	0	1	1	1	1

6 Concluding Remarks

The paper has shown how interventions (with or without cut) can be handled in a propositional logic setting. It has outlined how interventions, once expressed in such a setting, can supplement the approach that models the ascription of causal relations in a reported sequence of events on the basis of default rules expressing generic beliefs, in order to distinguish causality from correlation. The proposed approach is still to be compared with the treatment of interventions in the setting of possibilistic networks [3], since a possibilistic network can be

translated into a possibilistic logic base and vice versa [1]. However, the paper has shown that the notion of intervention makes sense in non directed settings.

Acknowledgements

This work is financially supported by the ANR project MICRAC (NT05-3-44479).

References

1. Benferhat, S., Dubois, D., Garcia, L., Prade, H.: On the transformation between possibilistic logic bases and possibilistic causal networks. Inter. J. of Approximate Reasoning 29, 135–173 (2002)
2. Benferhat, S., Dubois, D., Prade, H.: Nonmonotonic reasoning, conditional objects and possibility theory. Artificial Intelligence 92, 259–276 (1997)
3. Benferhat, S., Smaoui, S.: Possibilistic causal networks for handling interventions: A new propagation algorithm. In: Proc. 22nd AAAI Conf. on Artif. Intelligence (AAAI 2007), July 2007, pp. 373–378 (2007)
4. Bonnefon, J.F., Da Silva Neves, R.M., Dubois, D., Prade, H.: Background default knowledge and causality ascriptions. In: Proc. 17th Europ. Conf. on Artif. Intel (ECAI 2006), vol. 29(8-1(9)), pp. 11–15. Riva del Garda (2006)
5. Dubois, D., Lang, J., Prade, H.: Possibilistic logic. In: Gabbay, D., Hogger, C., Robin-son, J., Nute, D. (eds.) Handbook de Logic in Artificial Intelligence and Logic Programming, vol. 3, pp. 439–513. Oxford Univ. Pr., Oxford (1994)
6. Dubois, D., Prade, H.: Modeling the role of (ab)normality in the ascription of causality judgements by agents. In: Morgenstern, L., Pagnucco, M. (eds.) Proc. of IJCAI-05 Workshop on Nonmonotonic Reasoning, Action, and Change (NRAC 2005), Edinburg, August 1, pp. 22–27 (2005)
7. Goldszmidt, M., Pearl, J.: Rank-based Systems: A Simple Approach to Belief Revision, belief update, and reasoning about evidence and actions. In: Proc. of 3rd Inter. Conf. on Principles of Knowledge Representation and Reasoning (KR 1992), Cambridge, MA, October 25-29, pp. 661–672. Morgan Kaufmann, San Francisco (1992)
8. Halpern, J., Pearl, J.: Causes and explanations: a structural-model approach. Part I: Causes. Part II: Explanations. British Journal for the Philosophy of Science 56, 843–887, 889–911 (2005)
9. Kraus, S., Lehmann, D., Magidor, M.: Nonmonotonic reasoning, preferential models and cumulative logics. Arti cial Intelligence 44, 167–207 (1990)
10. Pearl, J.: Embracing causality in default reasoning. Artificial Intelligence 35, 259–271 (1988)
11. Pearl, J.: A probabilistic calculus of actions. In: Lopez de Mantaras, R., Poole, D. (eds.) Proc. 10th Conf. Uncertainty in A I (UAI-1994), Seattle, July 29-31, pp. 454–462. Morgan Kaufmann, San Francisco (1994)
12. Pearl, J.: Causality. Cambridge Univ. Pr., Cambridge (2000)
13. Sloman, S.A., Lagnado, D.A.: Do we "do"? Cognitive Science 29, 5–39 (2005)
14. Spohn, W.: A general non-probabilistic theory of inductive reasoning. In: Shachter, R., Levitt, T., Kanal, L., Lemmer, J. (eds.) Uncertainty in Artificial Intelligence, North-Holland, vol. 4, pp. 149–158 (1990)
15. Woodward, J.: Causation and manipulability. In: Zalta, E.N. (ed.) Stanford Encyclopedia of Philosophy (2001)
`http://plato.stanford.edu/entries/causation-mani/`

An Analysis of Sum-Based Incommensurable Belief Base Merging

Salem Benferhat, Sylvain Lagrue, and Julien Rossit

CRIL-CNRS UMR 8081,
Université d'Artois, Faculté des Sciences Jean Perrin
rue Jean Souvraz, SP18. 62307 Lens France
{benferhat,lagrue,rossit}@cril.univ-artois.fr

Abstract. Different methods have been proposed for merging multiple and potentially conflicting informations. Sum-based operators offer a natural method for merging commensurable prioritized belief bases. Their popularity is due to the fact that they satisfy the majority property and they adopt a non cautious attitude in deriving plausible conclusions.

This paper analyses the sum-based merging operator when sources to merge are incommensurable, namely they do not share the same meaning of uncertainty scales. We first show that the obtained merging operator can be equivalently characterized either in terms of an infinite set of compatible scales, or by a well-known Pareto ordering on a set of models. We then study different families of compatible scales useful for merging process. This paper also provides a postulates-based analysis of our merging operators.

1 Introduction

The problem of merging multiple-source information is crucial for many applications. Indeed, many situations require to take into account several potentially conflicting pieces of information, such as distributed databases frameworks, multi-agent systems, or distributed information in general (e.g. semantic web). This kind of situation leads to perform some combination operations on available pieces of information, which is well known as a *data fusion problem*.

Different merging operators have been proposed in the literature to merge prioritized pieces of information issued from different sources [1, 2, 3]. Most of existing merging methods assume that ranks associated with beliefs are commensurable from one source to another. This *commensurability assumption* may make sense in some situations, when it is possible to obtain a reference scale between sources. However, it can appear to be too strong for other applications. Only few works have addressed the issue of merging incommensurable ranked belief bases [4, 5].

In this paper, we first provide, in Section 3, a natural extension of the sum-based fusion mode to deal with incommensurable belief bases. This extension uses the concept of compatible scales to define the result of merging. We show, in the same section, that the fusion can also be characterized in terms of a

L. Godo and A. Pugliese (Eds.): SUM 2009, LNAI 5785, pp. 55–67, 2009.

Pareto-ordering on possible worlds. We then analyze, in Section 4, the logical behavior of the sum-based merging operator in commensurable and incommensurable cases. As a surprising result, the majority postulate is no longer valid when dealing with incommensurable belief bases. Even worse, the sum-based merging operator become majority-independent. Section 5 presents several inference relations based on some selection functions of compatible scales (such as the ones obtained from linear transformations or bounded compatible scales). In particular, we analyze the impact of these selection functions on the satisfaction of rationality postulates, and on the prudence of merging operators. Section 6 provides a brief comparison with some other merging approaches.

Before developing in details these results, we first provide some backgrounds on ranked belief bases and sum-based merging of commensurable belief bases.

2 Merging Ranked Belief Bases

Let $\mathcal{L}$ be a *finite* propositional language. We denote by Ω the set of interpretations of $\mathcal{L}$ and by ω an element of Ω. Greek letters φ and ψ denote propositional formulas. $\equiv$ denotes a logical equivalence, $Mod(\varphi)$ represents the set of models of φ, $\top$ and $\bot$ represent respectively a tautology and a contradiction.

2.1 Ranked Belief Bases

A ranked belief base is a multi-set of ranked formulas. It contains beliefs provided by a given source. The term *belief* is used when pieces of information provided by sources are uncertain. A *ranked belief* is represented by a propositional formula associated with a rank. This rank represents the amount of uncertainty associated with the formula, and can simply expresses the reliability of the source which provides this belief. A ranked belief base is a convenient framework to represent uncertain (or prioritized) pieces of information:

Definition 1 (Ranked belief base). *A ranked belief base B_i is a multi-set of ranked propositional formulas $B_i = \{(\varphi_{ij}, R_{B_i}(\varphi_{ij})),\ j \in \{1, ..., m_i\}\}$, where $\varphi_{ij} \in \mathcal{L}$, and $R_{B_i}(\varphi_{ij}) \in \mathbb{N}^*$.*

$(\varphi_{ij}, R_{B_i}(\varphi_{ij}))$ means that φ_{ij} has a priority rank of at least $R_{B_i}(\varphi_{ij})$. Intuitively, formulas associated with highest ranks are those which are preferred for a given source (or agent). Only strictly positive ranks are represented. Ranked belief bases are used in different frameworks, such as possibility theory [6] or ordinal conditional functions (OCF) [7,8,9]. We denote by B_i^* the set of propositional formulas obtained from B_i by ignoring ranks associated with formulas.

Given a ranked belief base, a total pre-order on interpretations of Ω can be derived as follows: ω is preferred to ω' if and only if the strongest belief falsified by ω is less important than the strongest belief falsified by ω'. More precisely:

Definition 2 (κ-functions). *A ranking function κ_{B_i} associated with a ranked belief base B_i is a function that maps each interpretation $\omega \in \Omega$ to an integer $\kappa_{B_i}(\omega)$ such that:*

$$\kappa_B(\omega) = \begin{cases} 0 & \text{if } \forall(\varphi_{ij},\ R_{B_i}(\varphi_{ij})) \in B_i,\ \omega \models \varphi_{ij} \\ max\{R_{B_i}(\varphi_{ij}) : \omega \not\models \varphi_{ij},\ (\varphi_{ij}, R_{B_i}(\varphi_{ij})) \in B_i\} & \text{otherwise.} \end{cases}$$

Interpretations associated with lower ranks represent agent's current beliefs. This ordering is the basis of possibilistic logic [6] and adjustment revision [8].

Example 1. Let us consider the ranked belief base $B = \{(\neg a \vee b, 8), (a \vee b, 5), (a, 2)\}$. Table 1 gives the κ-function κ_B associated with B. ω_3 is the preferred interpretation and will represent agent's current beliefs.

Table 1. An example of κ-function

$\omega_i \in \Omega$	a b	$\kappa_B(\omega_i)$
ω_0	0 0	5
ω_1	0 1	2
ω_2	1 0	8
ω_3	1 1	**0**

It is important to note that beliefs are inserted as they are and as they come from their sources, and we do not add derived beliefs. This is the spirit of what is called "belief bases" by Nebel [10]. Hence, the same belief can be present several times in B_i and this explains why we consider it as a multi-set. Equivalent beliefs have different identification (the identification can be an arbitrary numbering of the beliefs in B_i). We do not make these identifications explicit since it renders the notation heavy. Formulas of belief bases are distinguished from plausible conclusions which are derived from beliefs.

2.2 Sum-Based Fusion of Commensurable Bases

This section recalls a *sum-based* merging method. Let $E = \{B_1, ..., B_n\}$ be a multi-set of n ranked belief bases issued from n sources, and let μ be a propositional formula representing integrity constraints to satisfy. The aim of merging is, given E and μ, to rank-order different interpretations of Ω with respect to pieces of information provided by sources. This ordering is often obtained using a merging operator denoted here by $\triangle^\mu$. Given E, $\triangle^\mu$, we denote by $\lhd^E$ the ordering on Ω induced by $\triangle^\mu$ and E. We denote by $\triangle^\mu(E)$ the so-called belief set which represents the set of actual beliefs obtained after merging E and μ by $\triangle^\mu$. $\triangle^\mu(E)$ is defined as usually, namely it is such that its models are those which are minimal with respect to $\lhd^E$. In the literature, different methods for merging E have been proposed (e.g. [3,2,11]). This paper focuses on a sum-based fusion, denoted by $\triangle^\mu_\Sigma$.

To compute the result of merging, each interpretation ω is associated with a *profile*, denoted by $\nu_E(\omega)$, and defined by:

$$\nu_E(\omega) = \langle \kappa_{B_1}(\omega), ..., \kappa_{B_n}(\omega) \rangle$$

It represents the consistency degree of an interpretation ω with respect to ranked bases to merge. The computation of $\triangle^\mu_\Sigma$ is achieved in two steps: first combine

the consistency degrees $\kappa_{B_i}(\omega)$'s with the Sum operator Σ, and then rank-order interpretations with respect to their obtained ranks. More formally:

Definition 3 ($\lhd^E_\Sigma$). *Let ω and ω' be two interpretations of Ω, and $\nu_E(\omega)$, $\nu_E(\omega')$ be their respective profiles. Then : $\omega \lhd^E_\Sigma \omega'$ iff $\Sigma(\nu_E(\omega)) < \Sigma(\nu_E(\omega'))$, where $\Sigma(\nu_E(\omega)) = \sum_{i=1,..,n} \kappa_{B_i}(\omega)$.*

Models of $\triangle^\mu_\Sigma(E)$ are models of μ which are minimal with respect to $\lhd^E_\Sigma$:

$$Mod(\triangle^\mu_\Sigma(E)) = Min(Mod(\mu), \lhd^E_\Sigma)$$

The sum-based merging operator is *majority dependent* : the repetition of a same piece of information may affect the result of merging. This kind of operator is particularly well adapted if sources (or agents) are assumed to be independent.

Example 2. Let us consider $E = \{B_1,\ B_2\ B_3\}$ where $B_1 = \{(a, 6), (b, 3)\}$, $B_2 = \{(a \vee b, 3), (\neg b, 1)\}$ and $B_3 = \{(\neg a, 5\}$. Assume that $\mu \equiv \neg a \vee \neg b$. Profiles associated with interpretations are given by Table 2. We have $Mod(\triangle^\mu_\Sigma(E)) = \{\omega_1\}$.

Table 2. Profiles associated with interpretations

$\omega \in \Omega$	a b	$\nu_E(\omega)$	$\Sigma(\nu_E(\omega))$
ω_0	0 0	$\langle$6,3,0$\rangle$	9
ω_1	**0 1**	$\langle$**6,1,0**$\rangle$	**7**
ω_2	1 0	$\langle$3,0,5$\rangle$	8
ω_3	1 1	$\langle$0,1,5$\rangle$	6

3 Extension of the Sum-Based Operator for Merging Incommensurable Belief Bases

3.1 Compatible Scales Merging Approach

The sum-based merging operator defined above assumes that ranks, associated with formulas, have to be commensurable in order to sum them. Such assumption can be too strong for some applications, for instance when information is obtained from sources with unknown quality (e.g. web). A natural way to merge incommensurable belief bases consists in using possible common scales, called *compatible scales* [5]. A compatible scale affects new ranks to beliefs such that initial relative orders between beliefs of each agent are preserved.

Definition 4 (Compatible scale). *A compatible scale $\mathcal{S}$ is a function that maps $E = \{B_1, ..., B_n\}$ to $E^\mathcal{S} = \{B_1^\mathcal{S}, ..., B_n^\mathcal{S}\}$such that for all $B_i \in E$:*

(i) $B_i^\mathcal{S} = \{(\varphi_{ij}, \mathcal{S}(\varphi_{ij})) :\ (\varphi_{ij}, R_{B_i}(\varphi_{ij})) \in B_i\}$
(ii) $\forall B_i \in E$, $\forall (\varphi_{ij}, R_{B_i}(\varphi_{ij})) \in B_i$, $\forall (\varphi_{ij'}, R_{B_i}(\varphi_{ij'})) \in B_i$,
$R_{B_i}(\varphi_{ij}) \leq R_{B_i}(\varphi_{ij'})$ *iff* $\mathcal{S}(\varphi_{ij}) \leq \mathcal{S}(\varphi_{ij'})$.

The following example shows that compatible scales are not unique.

Example 3. Let us consider $E = \{B_1, B_2, B_3\}$ be belief bases provided by Example 2. Table 3 gives three possible scales: $\mathcal{S}_1$, $\mathcal{S}_2$ and $\mathcal{S}_3$. Scales $\mathcal{S}_1$ and $\mathcal{S}_2$ are compatible, because they preserve initial orders induced by each base. However, $\mathcal{S}_3$ is not a compatible one: it inverses priority between beliefs of B_1.

Table 3. Examples of compatible scales

	φ_{ij}	$R_{B_i}(\varphi_{ij})$	$\mathcal{S}_1(\varphi_{ij})$	$\mathcal{S}_2(\varphi_{ij})$	$\mathcal{S}_3(\varphi_{ij})$
B_1	a	6	2	6	1
	b	3	1	4	5
B_2	$a \vee b$	3	2	3	3
	$\neg b$	1	1	2	1
B_3	$\neg a$	5	1	4	5

As we already pointed out, beliefs are considered as self justified. A compatible scale does not directly handle beliefs but rather *normalize* ranks associated with beliefs. If, for instance, one have two bases: $B_1 = \{(a, 1), (b, 2)\}$, and $B_2 = \{(b, 3), (a, 5)\}$; a possible common scale is $\mathcal{S}$ such that $B_1^{\mathcal{S}} = \{(a, 2), (b, 3)\}$, and $B_2^{\mathcal{S}} = \{(b, 2), (a, 3)\}$. This scale is compatible since it simply preserves the initial ordering between beliefs of B_1 and between beliefs of B_2.

The set of scales compatible with E is denoted by $\mathbb{S}_E$. Note that $\mathbb{S}_E$ is never empty (it is enough to consider a scale that simply uses initial ranks, which is trivially compatible). Given a compatible scale $\mathcal{S} \in \mathbb{S}_E$, we denote by $B_i^{\mathcal{S}}$ the belief base obtained from B_i by using a compatible scale $\mathcal{S}$. More formally, $B_i^{\mathcal{S}}$ is obtained by replacing each pair $(\varphi_{ij}, R_{B_i}(\varphi_{ij}))$ by $(\varphi_{ij}, \mathcal{S}(\varphi_{ij}))$. Moreover, we denote by $E^{\mathcal{S}}$ the multi-set obtained by application of $\mathcal{S}$ on each B_i from E.

Definition 5 (Definition of $\blacktriangleleft_\Sigma^E$). *Let ω, ω' be two interpretations of Ω. Then:*

$$\omega \blacktriangleleft_\Sigma^E \omega' \quad \textit{iff} \quad \forall \mathcal{S} \in \mathbb{S}_E,\ \omega \lhd_\Sigma^{E^{\mathcal{S}}} \omega'$$

where $\lhd_\Sigma^{E^{\mathcal{S}}}$ is the result of applying the Definition 3 on $E^{\mathcal{S}}$.

Models of $\blacktriangle_\Sigma^\mu(E)$ are again: $Mod(\blacktriangle_\Sigma^\mu(E)) = Min(Mod(\mu), \blacktriangleleft_\Sigma^E)$.

Example 4. Let consider again $B_1 = \{(a, 6), (b, 3)\}$, $B_2 = \{(a \vee b, 3), (\neg b, 1)\}$ and $B_3 = \{(\neg a, 5\}$. Table 4 provides profiles associated with interpretations for each of two compatible scales, $\mathcal{S}_1$ and $\mathcal{S}_2$. Bold elements represent models of $\triangle_\Sigma^\mu(E^{\mathcal{S}_i})$. For instance, ω_2, and ω_3 are models of $\triangle_\Sigma^\mu(E^{\mathcal{S}_1})$ according to $\mathcal{S}_1$.

3.2 Characterization of the Result of Merging

This subsection shows that it is possible to characterize the result of fusion without comparing all compatible scales. The following proposition shows that

Table 4. Profiles associated with interpretations

$\omega \in \Omega$	a	b	$\nu_E s_1(\omega)$	$\Sigma_{\mathcal{S}_1}$	$\nu_E s_2(\omega)$	$\Sigma_{\mathcal{S}_2}$
ω_0	0	0	$\langle 2,2,0\rangle$	4	$\langle 6,3,0\rangle$	9
ω_1	0	1	$\langle 2,1,0\rangle$	3	$\langle 6,2,0\rangle$	8
ω_2	1	0	$\langle \mathbf{1},\mathbf{0},\mathbf{1}\rangle$	**2**	$\langle 4,0,4\rangle$	8
ω_3	1	1	$\langle \mathbf{0},\mathbf{1},\mathbf{1}\rangle$	**2**	$\langle \mathbf{0},\mathbf{2},\mathbf{4}\rangle$	**6**

an interpretation ω is a model of $\blacktriangle^{\mu}_{\Sigma}(E)$ if and only if there exists a compatible scale $\mathcal{S}$ where ω is a model of $\triangle^{\mu}_{\Sigma}(E^{\mathcal{S}})$. More formally:

Proposition 1. *Let E be a multi-set of ranked belief bases. Then $\omega \in Mod(\blacktriangle^{\mu}_{\Sigma}(E))$, if and only if there exists a compatible scaling $\mathcal{S}$ such that $\omega \in Mod(\triangle^{\mu}_{\Sigma}(E^{\mathcal{S}}))$.*

We now generalize this proposition by characterizing the whole ordering $\blacktriangleleft^{E}_{\Sigma}$ and not only its minimal elements. It turns out that $\blacktriangleleft^{E}_{\Sigma}$ corresponds to the well-known *Pareto Criterion*. Namely:

Proposition 2. *Let ω and ω' be two interpretations of Ω. Then $\omega \blacktriangleleft^{E}_{\Sigma} \omega'$ iff: (i) $\forall j \in \{1,..,n\},\ \kappa_{B_j}(\omega) \leq \kappa_{B_j}(\omega')$ and (ii) $\exists i \in \{1,...,n\},\ \kappa_{B_i}(\omega) < \kappa_{B_i}(\omega')$.*

The first condition means that ω is at least as preferred as ω' with respect to each belief base, while the second condition means that at least one base expresses a strict preference for ω.

4 Logical Behavior and Rational Postulates

Many postulates have been proposed in the literature to characterize merging operators under constraints (see [3] for details). These postulates are defined when belief bases are represented by propositional formulas. In our framework, these postulates have been adapted (see [5] for details) as follows:

(IC0) $\triangle^{\mu}(E) \models \mu$;
(IC1) If μ is consistent, then $\triangle^{\mu}(E)$ is consistent;
(IC2*) If $\bigwedge_{B\in E} B^*$ is consistent with μ, then
$\triangle^{\mu}(E) \equiv \bigwedge_{B\in E} B^* \wedge \mu$;
(IC3*) If $E_1 \equiv_R E_2$ and $\mu_1 \equiv \mu_2$, then
$\triangle^{\mu_1}(E_1) \equiv \triangle^{\mu_2}(E_2)$;
(IC4*) If $B_1^* \models \mu$ and $B_2^* \models \mu$, then $\triangle^{\mu}(\{B_1, B_2\}) \wedge B_1^*$ is consistent iff $\triangle^{\mu}(\{B_1, B_2\}) \wedge B_2^*$ is consistent;
(IC5) $\triangle^{\mu}(E_1) \wedge \triangle^{\mu}(E_2) \models \triangle^{\mu}(E_1 \bigsqcup E_2)$;
(IC6) If $\triangle^{\mu}(E_1) \wedge \triangle^{\mu}(E_2)$ is consistent, then
$\triangle^{\mu}(E_1 \bigsqcup E_2) \models \triangle^{\mu}(E_1) \wedge \triangle^{\mu}(E_2)$;
(IC7) $\triangle^{\mu_1}(E) \wedge \mu_2 \models \triangle^{\mu_1 \wedge \mu_2}(E)$;
(IC8) If $\triangle^{\mu_1}(E) \wedge \mu_2$ is consistent, then
$\triangle^{\mu_1 \wedge \mu_2}(E) \models \triangle^{\mu_1}(E) \wedge \mu_2$.

Additional postulates have been proposed in [3]:

(IC6') If $\triangle^{\mu}(E_1) \wedge \triangle^{\mu}(E_2)$ is consistent, then
$\triangle^{\mu}(E_1 \bigsqcup E_2) \models \triangle^{\mu}(E_1) \vee \triangle^{\mu}(E_2)$;
(MAJ) $\exists m \in \mathbb{N} :\ \triangle^{\mu}(E \sqcup B_i^m) \models B_i^*$, (with $B_i^m = \{B_i\} \sqcup ... \sqcup \{B_i\}$ m times and $\sqcup$ the multi-set union) ;
(MI) $\forall m,\ \triangle^{\mu}(E_1 \sqcup E_2^m) \equiv \triangle^{\mu}(E_1 \sqcup E_2)$.

The *majority postulate* **(MAJ)** characterizes majoritarian merging operators: if a given set of beliefs is repeated often enough, this set of beliefs should be accepted in the result of merging. At the opposite, the *majority independence postulate* states that the result of merging is independent of the repetition of beliefs. **(IC6')** is a weakened version of **(IC6)**. We also introduce in this paper a stronger version of **(IC4*)**, called the *consensus* postulate:

(CSS) $\forall B_i \in E,\ if\ B_i \models \mu,\ then\ B_i^* \wedge \triangle^{\mu}(E)$ is consistent.

4.1 The Commensurable Case

In the commensurable case, the following proposition shows that $\triangle_{\Sigma}^{\mu}$ satisfies most of the original postulates:

Proposition 3. $\triangle_{\Sigma}^{\mu}$ *satisfies* **(IC0)**, **(IC1)**, **(IC2*)**, **(IC3*)**, **(IC5)**, **(IC6)**, **(IC6')**, **(IC7)**, **(IC8)** *and* **(MAJ)**.

However, $\triangle_{\Sigma}^{\mu}$ falsifies **(IC4*)**, **(MI)** and **(CSS)**. For the lack of space, we only provide a counter-example for **(IC4*)**:

Example 5. Let us consider $B_1 = \{(\neg a, 2), (b, 1)\}$, $B_2 = \{(a, 3), (b, 2)\}$, and $\mu = a \vee b$. We have $B_1^* \models \mu$ and $B_2^* \models \mu$, and from Table 5, $Mod(\triangle_{\Sigma}^{\mu}(E)) = \{\omega_3\}$, and then $\triangle_{\Sigma}^{\mu}(E) \equiv a \wedge b$. Hence, on this example, $\triangle^{\mu}(\{B_1, B_2\}) \wedge B_2^*$ is consistent, but $\triangle^{\mu}(\{B_1, B_2\}) \wedge B_1^*$ is not.

Table 5. Profiles associated with interpretations

	a b	$\kappa_{B_1}(\omega)$	$\kappa_{B_2}(\omega)$	$\nu_E(\omega)$	$\Sigma(\nu_E(\omega))$
ω_0	0 0	1	3	⟨1,3⟩	4
ω_1	0 1	0	3	⟨0,3⟩	3
ω_2	1 0	2	2	⟨2,2⟩	4
ω_3	1 1	2	0	⟨2,0⟩	2

4.2 The Incommensurable Case

When belief bases are incommensurable, we obtain:

Proposition 4. $\blacktriangle_{\Sigma}^{\mu_1}(E)$ *satisfies* **(IC0)**, **(IC1)**, **(IC2*)**, **(IC3*)**, **(IC4*)**, **(IC5)**, **(IC7)** *and* **(CSS)**.

However, $\blacktriangle_{\Sigma}^{\mu}$ falsifies **(IC6)**, **(IC6')**, **(IC8)**, and **(MAJ)**. For the lack of space, we only provide a counter-example for **(IC6)**.

Example 6. Let us consider $\mu = \top$, $E_1 = \{B_1 = \{(a, 1)\}, B_2 = \{(\neg a, 1)\}\}$ and $E_2 = \{B_3 = \{(a, 1)\}\}$. From Tables 6, we have $\blacktriangle^{\mu}_{\Sigma}(E_1) \equiv \top$ and $\blacktriangle^{\mu}_{\Sigma}(E_2) \equiv a$. Furthermore, we have $\blacktriangle^{\mu}_{\Sigma}(E_1 \bigsqcup E_2) \equiv \top$, but $\blacktriangle^{\mu}_{\Sigma}(E_1) \wedge \blacktriangle^{\mu}_{\Sigma}(E_2) \equiv a$. Hence, **(IC6)** is not satisfied since $\blacktriangle^{\mu}_{\Sigma}(E_1) \wedge \blacktriangle^{\mu}_{\Sigma}(E_2)$ is consistent, but $\blacktriangle^{\mu}_{\Sigma}(E_1 \bigsqcup E_2) \not\models \blacktriangle^{\mu}_{\Sigma}(E_1) \wedge \blacktriangle^{\mu}_{\Sigma}(E_2)$.

Table 6. Profiles associated with interpretations

	a	b	$\nu_{E_1}(\omega)$	$\nu_{E_2}(\omega)$	$\nu_{E_1 \bigsqcup E_2}(\omega)$
ω_0	0	0	$\langle$**1,0**$\rangle$	$\langle$1$\rangle$	$\langle$**1,0,1**$\rangle$
ω_1	0	1	$\langle$**1,0**$\rangle$	$\langle$1$\rangle$	$\langle$**1,0,1**$\rangle$
ω_2	1	0	$\langle$**0,1**$\rangle$	$\langle$**0**$\rangle$	$\langle$**0,1,0**$\rangle$
ω_3	1	1	$\langle$**0,1**$\rangle$	$\langle$**0**$\rangle$	$\langle$**0,1,0**$\rangle$

Note that the non-satisfaction of **(IC8)** is due to the fact that when dealing with incommensurable belief bases, $\blacktriangleleft^{E}_{\Sigma}$ is only a partial order. For instance, in [12], the fusion mode based on partial order does not satisfy **(IC8)**.

Regarding the non satisfaction of the majority postulate, the situation is even worst. It can be shown that the *Sum*-based incommensurable belief base merging operator satisfies the *majority independence* postulate.

Proposition 5. $\blacktriangle^{\mu}_{\Sigma}(E)$ *satisfies* **(MI)**.

5 Selection Functions of Compatible Scales

This section restricts our merging operator to particular subsets of compatible scales, in order to derive more plausible conclusions. We discuss the following particular classes of compatible scales:

- *bounded class* $S^{(p)}$: compatible scales such that the highest new rank assigned to a formula cannot exceed a fixed threshold p (a positive integer).
- *linear class* S^{l}: this class only proceeds to a linear transformation of initial ranks;
- *weighted class* S^{w}: this class allows a proportional change of initial weights;
- *shift class* S^{s}: this class allows to hold distance between ranks associated with two distinct formulas from a given source.

Table 7 gives formal definitions of these selection functions:

Table 7. Particular classes of compatible scales

Class	Notation	$\{\mathcal{S}$ *s.t.* $\forall B_i \in E,\ \forall \varphi \in B_i :$ ... $\}$
Bounded	$S^{(p)}$	$\mathcal{S}(\varphi) \leq p$
Linear	S^{l}	$\mathcal{S}(\varphi) = a_i.R_{B_i}(\varphi) + b_i$ with $a_i > 0,\ b_i \geq 0$
Weighted	S^{w}	$\mathcal{S}(\varphi) = a_i.R_{B_i}(\varphi)$ with $a_i > 0$
Shift	S^{s}	$\mathcal{S}(\varphi) = R_{B_i}(\varphi) + b_i$ with $b_i \geq 0$

Bounded compatible scales offer a natural way to select a set of compatible scales, since in practice common scales are bounded. One can remark that the smallest possible p is $p_{min} = max\{|B_i| : B_i \in E\}$, where $|B_i|$ represents the number of different rank (or ranks) in B_i. If $p < p_{min}$, then the set of compatible scales is empty.

The shift compatible scale class allows to hold distance relations between ranks of formulas. Indeed, when applying a such compatible scale $\mathcal{S}$, then: $\forall(\varphi_{ij}, R_{B_i}(\varphi_{ij})), (\varphi_{ij'}, R_{B_i}(\varphi_{ij'})) \in B_i, \mathcal{S}(\varphi_{ij}) - \mathcal{S}(\varphi_{ij'}) = R_{B_i}(\varphi_{ij}) - R_{B_i}(\varphi_{ij'})$.

Weighted compatible scales are obtained by multiplying associated ranks $R_{B_i}(\varphi_{ij})$ by a weight a_i. Intuitively, these weights may represent the reliability of sources (each B_i has a reliability weight a_i), and the merging operator become a weighted sum. Linear compatible scales class generalizes the two above classes (weighted and shift).

We denote by $\blacktriangleleft^{\mu}_{\Sigma,S^{(p)}}$ (resp. $\blacktriangleleft^{\mu}_{\Sigma,S^{l}}$, $\blacktriangleleft^{\mu}_{\Sigma,S^{w}}$, and $\blacktriangleleft^{\mu}_{\Sigma,S^{s}}$) the order obtained from Definition 5 by replacing $\mathbb{S}$ by $S^{(p)}$ (resp. S^l, S^w, and S^s). Following subsections analyze the impact of restricting to particular classes on the cautiousness of our merging operator and on the satisfaction of rational postulates.

5.1 Impact on Cautiousness

As a first surprising result, restricting to classes of affine or linear compatible scales does not affect the result of merging:

Proposition 6. $\forall\omega,\ \omega' \in \Omega, \omega \blacktriangleleft^{E}_{\Sigma,S^{l}_{E}} \omega'$ *iff* $\omega \blacktriangleleft^{E}_{\Sigma,S^{w}_{E}} \omega'$ *iff* $\omega \blacktriangleleft^{E}_{\Sigma} \omega'$.

However, inference based on bounded scales is in general more productive than $\blacktriangle^{\mu}_{\Sigma}$. In fact, inference from bounded scales depends on the value of p, and for a very particular value of p the standard sum-based merging operator is recovered. Indeed, if all bases in E have the same number of different ranks, equal to p_0, and that the maximal rank associated with formulas in each B_i is p_0, then $\forall\omega,\ \omega' \in \Omega$: $\omega \blacktriangleleft^{E}_{\Sigma,S^{(p_0)}_{E}} \omega'$ iff $\omega \lhd^{E^{\mathcal{S}_{p_0}}}_{\Sigma} \omega'$.

Regarding inference based on shift compatible scales, it is also in general more productive than $\blacktriangle^{\mu}_{\Sigma}$. In fact, we can even provide a criterion which allows to characterize the order on possible worlds induced by $\blacktriangleleft^{E}_{\Sigma,S^{s}_{E}}$.

Proposition 7. *Let ω, ω' be two interpretations of Ω. Then $\omega \blacktriangleleft^{E}_{\Sigma,S^{s}_{E}} \omega'$ if and only if: i)$\Sigma(\nu_E(\omega)) < \Sigma(\nu_E(\omega'))$ and ii)$\forall B_i \in \{B_j \in E, \omega' \models B_j\},\ \omega \models B_i$.*

A full picture of the relationships between these different merging operators regarding prudence relations will be provided before the concluding discussion.

5.2 Impact on Rational Postulates

Table 8 summarizes the impact of selection functions on the satisfaction of postulates. In addition to Table 8, $\blacktriangle^{\mu}_{\Sigma,S^{(p)}}$ and $\blacktriangle^{\mu}_{\Sigma,S^{s}}$ both satisfy **(IC0)**, **(IC1)**, **(IC2*)**, **(IC3*)**, **(IC5)**, and **(IC7)**.

Table 8. Rational postulates satisfied in commensurable and incommensurable cases

	(IC4*)	**(IC6)**	**(IC8)**	**(MAJ)**	**(MI)**	**(CSS)**
$\triangle^{\mu}_{\Sigma}$	-	$\surd$	$\surd$	$\surd$	-	-
$\blacktriangle^{\mu}_{\Sigma}$	$\surd$	-	-	-	$\surd$	$\surd$
$\blacktriangle^{\mu}_{\Sigma,S^s}$	$\surd$	-	-	-	-	$\surd$
$\blacktriangle^{\mu}_{\Sigma,S^{(p)}}$	$\surd$	-	-	$\surd$	-	-

The main reason of the non-satisfaction of the majority postulate by $\blacktriangle^{\mu}_{\Sigma}(E)$ is that new ranks which are assigned to belief bases by compatible scales are not bounded. For instance, assume that B_1 contains φ and B_2 contains $\neg\varphi$. Since compatible scales are not bounded, then even if B_1 is repeated m times, it is always possible to find a compatible scale that assigns a high rank to formulas of B_2 (hence to $\neg\varphi$) which blocks the inference of φ. This explains why $\blacktriangle^{\mu}_{\Sigma,S^{(p)}}$ satisfies **(MAJ)** while other compatible based operators not.

$\triangle^{\mu}_{\Sigma}$ satisfies most of postulates except the fairness postulate **(IC4*)** and **(CSS)**. A natural question is whether there exists a single compatible scale that satisfies the fairness postulate and the consensus postulate. The following proposition provides a very particular case where **(IC4*)** and **(CSS)** hold.

Proposition 8. *Let* $E = \{B_1, = \{(\varphi, R_{B_1}(\varphi))\}, B_2 = \{(\varphi', R_{B_2}(\varphi'))\}\}$. *Let* $\mathcal{S}$ *be a compatible scale. Then:* $\triangle^{\mu}_{\Sigma,\mathcal{S}}$ *satisfies* (**IC4***)*and* (**CSS**) *iff* $\mathcal{S}(\varphi) = \mathcal{S}(\varphi')$.

However, in general, there is no hope to recover the satisfaction of the fairness and consensus postulates if one only uses a single compatible scale.

Proposition 9 (of impossibility). *There is no single compatible scale such that* $\triangle^{\mu}_{\Sigma,\mathcal{S}}$ *satisfies the fairness and the consensus postulate for multi-set of sources* E, *namely:* $\nexists\mathcal{S}$ *s.t.* $\forall E:$ $\triangle^{\mu}_{\Sigma,\mathcal{S}}$ *satisfies* (**IC4***) *or* (**CSS**).

For the counter example, it is enough to consider $E = \{B_1, B_2, B_3\}$ with $B_1 = \{(a \wedge c, 1)\}$, $B_2 = \{(\neg a, 1)\}$, and $B_3 = \{(\neg c, 1), (a, 2)\}$.

6 A Comparative Study

This section provides a comparative study of our merging operators with respect to max-based merging and coherence-based merging. Let $\blacktriangle^{\mu}_{Max}(E)$ (resp. $\triangle^{\mu}_{Max}(E)$) be defined exactly as $\blacktriangle^{\mu}_{\Sigma}(E)$ (resp. $\triangle^{\mu}_{\Sigma}(E)$) given by Definition 5 (resp. Definition 3), except that the sum operator Σ is replaced by the maximum operator Max (see [5] for more details).

Note that in the commensurable case, the sum-based and the Max-based merging operators are incomparable. In the incommensurable case, we have a strict inclusion between these two inference relations, namely:

$$\blacktriangle^{\mu}_{\Sigma}(E) \models \blacktriangle^{\mu}_{Max}(E).$$

Another way to deal with merging incommensurable bases is to view the set of bases to merge E as a partially ordered belief bases $(K^E, <_{K^E})$ where K^E is a multi-set containing all formulas in each bases of E, and $<_{K^E}$ is defined by: $\varphi_{ij} <_{K^E} \varphi_{ik}$ iff $\exists B_i \in E$ such as $\varphi_{ij} \in B_i$, $\varphi_{ik} \in B_i$ *and* $R_{B_i}(\varphi_{ij}) < R_{B_i}(\varphi_{ik})$. Computing the result of merging comes down to select a set of preferred interpretation, according to $(K^E, <_{K^E})$. One way to define such preferred interpretations is to use the well-known set inclusion-based criterion defined by [13]:

Definition 6. *An interpretation ω is said to be Incl-preferred to another interpretation ω', denoted by $\omega \lhd_E^{Incl} \omega'$, iff: $\forall \varphi \in K^E$ s.t. $\omega \not\models \varphi$ and $\omega' \models \varphi$, $\exists \psi \in K^E$ s.t. $\omega \models \psi$ and $\omega' \not\models \psi$ and: $\psi <_{K^E} \varphi$.*

This leads to define a merging operator $\triangle^{\mu}_{Incl}$ based on $\lhd_E^{Incl}$. Preferred beliefs for this merging operator are defined as follows:
$Mod(\triangle^{\mu}_{Incl}(E)) = Min(Mod(\mu), \lhd_E^{Incl})$.

The following proposition expresses that when each belief base contains exactly one formula, namely K^E is a set of propositional formulas, then $\triangle^{\mu}_{Incl}$ and $\blacktriangle^{\mu}_{\Sigma}$ provide the same result:

Proposition 10. *Assume that each B_i exactly contains one propositional formula. Then $\forall \varphi \in \mathcal{L}$, $\triangle^{\mu}_{Incl}(E) \models \varphi$ iff $\blacktriangle^{\mu}_{\Sigma}(E) \models \varphi$.*

Again, Proposition 10 shows that $\blacktriangle^{\mu}_{\Sigma}$ has a different behavior in the incommensurable case, since if B_i's are commensurable and contains a single formula, then $\triangle^{\mu}_{\Sigma}$ is more productive than inclusion-based approach. Now, if B_i's contain more than one formula, then $\triangle^{\mu}_{Incl}(E)$ and $\triangle^{\mu}_{\Sigma}(E)$ are incomparable.

Example 7. Let $E_1 = \{B_1, B_2, B_3\}$ be such that $B_1 = \{(a \wedge b, 1)\}$, $B_2 = \{(\neg a \wedge b, 1)\}$ and $B_3 = \{(a, 1)\}$. The sum-based operator will conclude {a} whereas the inclusion-based merging operator will not. Now consider $E_2 = \{B_1, B_2\}$ where $B_1 = \{(b, 2), (a, 1)\}$ and $B_2 = \{(\neg a, 2)\}$. Here, the inclusion-based merging operator will deduce {b} whereas the sum-based merging operator will not.

Figure 1 summarizes the links between all operators in terms of cautiousness: $\triangle_1 \rightarrow \triangle_2$ means that $\triangle_1$ can be inferred by $\triangle_2$. Flat case is also represented.

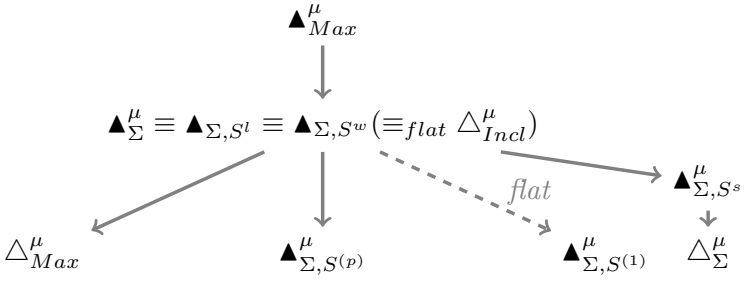


Fig. 1. Cautiousness of the different merging operators

In case of flat bases (namely each B_i contains exactly one formula), then $\triangle^{\mu}_{Incl}$ is more cautious than if one uses bounded scales with $p = 1$. In fact, when each belief base in E contains a single formula, then we can check that the well-known cardinality-based inference can be recovered from bounded scales with $p = 1$.

7 Conclusion

This paper investigated the sum-based merging operator for incommensurable bases. We proposed a characterization of the merging result in terms of compatible scales and in terms of a Pareto-ordering. This paper showed that the behavior of the sum-based merging in incommensurable case departs from the commensurable case, regarding postulates satisfaction and cautiousness properties. In particular, the sum-based merging operator is no longer a majoritarian operator. We analyzed different classes of compatible scales. Bounded compatible scales allow to recover the majority operator, and some coherence based approaches [14] when bases contain a single formula. We also analyzed the fairness postulate **(IC4*)** and the new postulate proposed in this paper, called consensus postulate **(CSS)**. We showed that there is no way to recover these postulates if a single compatible scale is selected. Lastly, this paper provided a comparative study between different merging operators discussed in this paper.

References

1. Everaere, P., Konieczny, S., Marquis, P.: Conflict-based merging operators. In: Proceedings of KR 2008, pp. 348–357 (2008)
2. Lin, J.: Integration of weighted knowledge bases. Artificial Intelligence 83(2), 363–378 (1996)
3. Konieczny, S., Pino Pérez, R.: Merging information under constraints: a logical framework. Journal of Logic and Computation 12(5), 773–808 (2002)
4. Guilin, Q., Liu, W., Bell, D.A.: Merging stratified knowledge bases under constraints. In: Proceedings of AAAI 2006, July 2006, pp. 348–356 (2006)
5. Benferhat, S., Lagrue, S., Rossit, J.: An egalitarist fusion of incommensurable ranked belief bases under constraints. In: Proceedings of AAAI 2007, pp. 367–372 (2007)
6. Dubois, D., Lang, J., Prade, H.: Possibilistic logic. In: Handbook of Logic in Artificial Intelligence and Logic Programming, vol. 3, pp. 439–513. Oxford University Press, Oxford (1994)
7. Spohn, W.: Ordinal conditional functions: a dynamic theory of epistemic state. Causation in Decision, Belief Change and Statistics, 105–134 (1988)
8. Williams, M.A.: Iterated theory base change: A computational model. In: Proceedings of IJCAI 1995, pp. 1541–1547 (1995)
9. Meyer, T.: On the semantics of combination operations. Journal of Applied Non-Classical Logics 11(1-2), 59–84 (2001)
10. Nebel, B.: Belief revision and default reasoning: Syntax-based approaches. In: Proceedings of KR 1991, July 1991, pp. 417–428 (1991)
11. Liberatore, P., Schaerf, M.: Arbitration: A commutative operator for belief revision. In: Proceedings of the 2nd World Conference on the Fundamentals of Artificial Intelligence, pp. 217–228 (1995)

12. Everaere, P., Konieczny, S., Marquis, P.: A diff-based merging operator. In: Proceedings of NMR 2008, pp. 19–25 (2008)
13. Brewka, G.: Preferred subtheories: an extended logical framework for default reasoning. In: Proceedings of IJCAI 1989, pp. 1043–1048 (1989)
14. Benferhat, S., Dubois, D., Prade, H.: Some Syntactic Approaches to the Handling of Inconsistent Knowledge Bases: a Comparative Study Part 2: the Prioritized Case. In: Orłowska, E. (ed.) Logic at Work: Essays Dedicated to the Memory of H. Rasiowa, pp. 437–511. Physica-Verlag (1999)

An Argument-Based Approach to Using Multiple Ontologies

Elizabeth Black[1], Anthony Hunter[2], and Jeff Z. Pan[3]

[1] Department of Engineering Science,
University of Oxford, Parks Road, Oxford, OX1 3PJ, UK
[2] Department of Computer Science,
University College London, Gower Street, London WC1E 6BT, UK
[3] Department of Computing Science,
University of Aberdeen, Aberdeen AB24 3UE, UK

Abstract. Logic-based argumentation offers an approach to querying and revising multiple ontologies that are inconsistent or incoherent. A common assumption for logic-based argumentation is that an argument is a pair $\langle \Phi, \alpha \rangle$ where Φ is a minimal subset of the knowledgebase such that Φ is consistent and Φ entails the claim α. Using dialogue games, agents (each with its own ontology) can exchange arguments and counterarguments concerning formulae of interest. In this paper, we present a novel framework for logic-based argumentation with ontological knowledge. As far as we know, this is the first proposal for argumentation with multiple ontologies via dialogues. It allows two agents to discuss the answer to queries concerning their knowledge (even if it is inconsistent) without one agent having to copy all of their ontology to the other, and without the other agent having to expend time and effort merging that ontology with theirs. Furthermore, it offers the potential for the agents to incrementally improve their knowledge based on the dialogue by checking how it differs from the other agent's.

1 Introduction

Inconsistency and incoherence are recognized as significant problems in managing ontological knowledge (e.g. [11]). These problems are particularly an issue when using multiple ontologies. Current solutions include the "maxcon" approach (which involves merging ontologies by selecting a maximal consistent subset of the union of the multiple ontologies) and the "oracle" approach (which involves constructing a merged consistent ontology by getting extra information to help resolve the conflicts). Unfortunately, the maxcon approach results in a loss of useful information, as it may not be certain which subset to choose, and therefore an arbitrary choice is made, and the oracle approach involves a lot of work that may not be necessary if for example a query can be answered from a small part of the agents' knowledge, and furthermore that this knowledge may not even be in conflict.

To address these problems, here we explore an alternative approach which involves only focusing on the subset of the union of the ontologies that is required for answering queries. Our approach is to keep the ontologies separate, and associate each ontology with an agent. Then, the agents enter into a dialogue in which arguments are exchanged

L. Godo and A. Pugliese (Eds.): SUM 2009, LNAI 5785, pp. 68–79, 2009.

concerning some subject. To simplify our presentation, we restrict consideration to two agents 1 and 2. Each agent i has a personal knowledgebase (or perbase) Δ_i that is a description logic ontology (e.g. OWL). We assume each perbase is finite, consistent and coherent, but that normally $\Delta_1 \cup \Delta_2$ is inconsistent or incoherent. We also assume they use the same description logic, though not necessarily the same vocabulary (i.e. they can use different names for the same concept and/or the same names for different concepts). Each perbase is private (i.e. the contents of the perbase are only available to the agent). The strategy used by the agent dictates what the agent will make public from its perbase. We also assume a finite knowledgebase of lexical matching knowledge called a lexbase Π that contains just formulae of the form $A \sqsubseteq B$ where A and B are named concepts. Each formula in Π is obtained by a lexical matching algorithm, and there is some reasonable probability that it is correct. Π is public knowledge (i.e. the contents are available to both agents) and Π may be inconsistent with either of the perbases. We assume some external agent has decided what formulae are in Π, and that each agent can choose to draw on formulae from the lexbase as required.

The aim of each agent is to co-operate with the other to answer questions about the ontological information they have, even though there may be conflicts between their perbases. This will allow agents to efficiently and effectively use their own ontology with the other agent's knowledge even when there is conflict between them. A further benefit (which we consider in the discussion) is that each agent can improve their own perbase through the argumentation process, and so undertake a form of partial merging of the other agent's ontological knowledge with their own ontological knowledge.

In the rest of this paper, we consider arguments based on description logic, and present a general framework for dialogical argumentation. We show that using dialogical argumentation allows for agents to use multiple ontologies without having to distribute each ontology to every other agent, and without having to merge the ontological knowledge.

2 Logical Arguments

The usual paradigm for **logic-based argumentation** is that there is a large repository of information, represented by Δ, from which logical arguments can be constructed for and against arbitrary claims (e.g. [15,4,1,9,7]). There is no *a priori* restriction on the contents, and the pieces of information in the repository can be arbitrarily complex. Therefore, Δ is not expected to be consistent. It need not even be the case that every single formula in Δ is consistent.

Our framework adopts a very common intuitive notion of a logical argument. Essentially, an argument is a set of formulae that can be used to prove some claim, together with that claim. Each claim is represented by a formula. Provability is represented by a consequence relation that may be for a logic such as classical logic, or a description logic.

Here we focus on **description logics** (DLs) [2], which is a family of logic-based knowledge representation formalisms used as the underpinning of the standard OWL Web ontology language. DLs are characterised by the constructors (such as $C \sqcap D$, $\exists R.C$, $\forall R.C$) for building complex concept descriptions, such as $\texttt{Animal} \sqcap \forall \texttt{Eat.Plant}$

(animals that eat only plants), and role descriptions. A DL ontology $\mathcal{O}$ can consist of concept axioms, such as concept inclusion axioms $C \sqsubseteq D$ (e.g., $\texttt{Cow} \sqsubseteq \texttt{Animal} \sqcap \forall \texttt{Eat}.\texttt{Plant}$), role axioms, such as transitive role axioms, and individual axioms, such as concept assertions (e.g. $\texttt{Cow}(\texttt{daisy})$ and $\texttt{Plant}(\texttt{grass1})$), $C(a)$ and role assertions $R(a, b)$ (e.g. $\texttt{Eat}(\texttt{daisy}, \texttt{grass1})$). We use $A \equiv B$ as shorthand for $A \sqsubseteq B$ and $B \sqsubseteq A$. A DL ontology $\mathcal{O}$ is **consistent** if there exists some interpretation $\mathcal{J}$ that satisfies all its axioms. A concept C is **satisfiable** w.r.t. $\mathcal{O}$ if there exists some interpretation $\mathcal{J}$ of $\mathcal{O}$ such that $C^{\mathcal{J}}$ is non-empty. A DL ontology $\mathcal{O}$ is **coherent** if all the named concepts in $\mathcal{O}$ are satisfiable. Negated axioms are closely related to inconsistencies and changes in ontologies. Since well known DLs do not provide enough expressive power to represent negations of all the axioms, we use two relaxed notions of negation proposed in [8], namely consistency-negation and coherence-negation. Intuitively speaking, an axiom α and any of its consistency-negation (coherence-negation) β are inconsistent (incoherent, resp.) with each other; i.e., $\{\alpha, \beta\}$ is inconsistent (incoherent, resp.).

We let $\mathcal{L}$ denote a DL and $\vdash$ denote the consequence relation of the DL. We use $\alpha, \beta, \gamma, \ldots$ to denote DL formulae (axioms), $\Delta, \Phi, \Psi, \ldots$ to denote sets of DL formulae, and $\mathsf{Names}(\alpha)$ to denote the set of names for concepts, roles and individuals from which a formula α is composed.

Definition 1. *An* **argument** *is a pair $\langle \Phi, \alpha \rangle$ s.t.: (1) $\Phi \subseteq \Delta$; (2) Φ is consistent and coherent; (3) $\Phi \vdash \alpha$; and (4) there is no $\Phi' \subset \Phi$ s.t. $\Phi' \vdash \alpha$. We say that $\langle \Phi, \alpha \rangle$ is an argument for α. We call α the* **claim** *and Φ the* **support** *of the argument.*

An undercut is a counterargument that directly opposes the support of an argument.

Definition 2. *Let $\langle \Phi, \alpha \rangle$, $\langle \Psi, \beta \rangle$ be arguments. $\langle \Psi, \beta \rangle$ is an* **undercut** *for $\langle \Phi, \alpha \rangle$ iff $\{\beta\} \cup \Phi$ is inconsistent or incoherent.*

Example 1. Let $\Delta = \{A \sqsubseteq B, B \sqsubseteq C, A \sqsubseteq D, A \sqsubseteq \neg C\}$. Some arguments include $\langle \{A \sqsubseteq B, B \sqsubseteq C\}, A \sqsubseteq C \rangle$ and $\langle \{A \sqsubseteq \neg C\}, A \sqsubseteq \neg C \rangle$ which undercut each other.

We can capture an argument

together with its undercuts, and by recursion, undercuts to undercuts, in a tree structure as follows. We assume that X denotes this set of arguments.

Definition 3. *An* **argument tree** *for ϕ is a tree (N, E, X, f) where N is a set of nodes, E is a set of edges, X is a set of arguments, and $f : N \mapsto X$ assigns an argument to each node s.t. (1) the root is assigned with an argument for ϕ (the* **root argument***); (2) for each node n, if m is a child of n, then $f(m)$ is an undercut of $f(n)$; and (3) for each node n, if m is an ancestor of n in the branch, then the support of $f(n)$ is not a subset of the support of $f(m)$.*

The **dialectical principle** (widely adopted in the literature on argumentatation) evaluates each argument as defeated or undefeated: An argument is **undefeated** if all the undercuts for it are defeated, and an argument is **defeated** if there is a undercut to it that is undefeated. Any argument with no undercuts is undefeated. For example, let A_1 be the root argument, A_2 and A_3 be undercuts to A_1, and A_4 be an undercut to A_3. In this case, only A_2 and A_4 are undefeated.

3 Dialogue Framework

The communicative acts in a dialogue are called **moves**. We assume that there are always exactly two agents taking part in a dialogue, each with its own identifier taken from the set $\mathcal{I} = \{1, 2\}$. Each agent takes it in turn to make a move to the other agent. We also refer to agents using the variables x and $\overline{x}$ such that if x is 1 then $\overline{x}$ is 2 and if x is 2 then $\overline{x}$ is 1. The format for moves is shown in Table 1, and the set of all moves meeting the format is denoted $\mathcal{M}$. For a move m, the function Sender returns the agent that made the move.

We now informally explain the different types of move: A query move $\langle x, \mathsf{query}, \alpha\rangle$ starts a dialogue with the topic α; A posit move $\langle x, \mathsf{posit}, \langle \Phi, \alpha\rangle\rangle$ asserts an argument $\langle \Phi, \alpha\rangle$ by x for the topic, or an undercut for a previous posit; A concede move $\langle x, \mathsf{concede}, \alpha\rangle$ asserts agent x will regard α as valid; and a close move $\langle x, \mathsf{close}, \gamma\rangle$ is used when an agent has no other moves it can make. Note, for a posit move $m = \langle x, \mathsf{posit}, \langle \Phi, \alpha\rangle\rangle$, we say that m is a posit move for α.

A dialogue is simply a sequence of moves, each of which is made from one participant to the other. As a dialogue progresses over time, we denote each timepoint by a natural number. Each move is indexed by a timepoint and exactly one move is made at each timepoint.

Definition 4. *A* **dialogue***, denoted D^t, is a sequence of moves $m_1, \ldots, m_t$ involving two agents in $\mathcal{I} = \{1, 2\}$, s.t. (1) m_1 is of the form $\langle x, \mathsf{query}, \gamma\rangle$; (2) $\mathsf{Sender}(m_s) \in \mathcal{I}$ for $1 \leq s \leq t$; and (3) $\mathsf{Sender}(m_s) \neq \mathsf{Sender}(m_{s+1})$ for $1 \leq s < t$. The* **topic** *of the dialogue D^t is returned by $\mathsf{Topic}(D^t)$ (i.e. $\mathsf{Topic}(D^t) = \gamma$). The set of all dialogues is denoted $\mathcal{D}$.*

The first move of a dialogue D^t must always be a query move (condition 1), every move of the dialogue must be made to a participant of the dialogue (condition 2), and the agents take it in turns to make moves (condition 3). In order to terminate a dialogue, two close moves must appear one immediately after the other in the sequence (called a *matched-close*).

Definition 5. *Let D^t be a dialogue s.t. $\mathsf{Topic}(D^t) = \gamma$. We say that m_s ($1 < s \leq t$), is a* **matched-close for** *D^t iff $m_{s-1} = \langle x, \mathsf{close}, \gamma\rangle$ and $m_s = \langle \overline{x}, \mathsf{close}, \gamma\rangle$. D^t* **terminates at** *t iff m_t is a matched-close for D^t and there does not exist an s s.t. $s < t$ and D^s terminates at s.*

Table 1. The move format, where γ is a formula, $\langle \Phi, \phi\rangle$ is an argument and $x \in \{1, 2\}$ is the agent that makes the move

Move	Format
query	$\langle x, \mathsf{query}, \gamma\rangle$
posit	$\langle x, \mathsf{posit}, \langle \Phi, \phi\rangle\rangle$
concede	$\langle x, \mathsf{concede}, \gamma\rangle$
close	$\langle x, \mathsf{close}, \gamma\rangle$

Example 2. $\Delta_1 = \{\neg C(b)\}$; $\Delta_2 = \{C(b), R(b,a), C \sqsubseteq \forall R.C, D(a)\}$; $\Pi = \{C \equiv D\}$.

$$\langle 1, \mathsf{query}, C(a)\rangle$$
$$\langle 2, \mathsf{posit}, \langle\{C(b), R(b,a), C \sqsubseteq \forall R.C\}, C(a)\rangle\rangle$$
$$\langle 1, \mathsf{posit}, \langle\{\neg C(b)\}, \neg C(b)\rangle\rangle$$
$$\langle 2, \mathsf{concede}, D(a)\rangle$$
$$\langle 1, \mathsf{concede}, C \equiv D\rangle$$
$$\langle 2, \mathsf{posit}, \langle\{C \equiv D, D(a)\}, C(a)\rangle\rangle$$
$$\langle 1, \mathsf{close}, C(a)\rangle$$
$$\langle 2, \mathsf{close}, C(a)\rangle$$

Agent 2 posits an argument for the topic, and agent 1 provides an undercut to it. Then each agent concedes a formula, and agent 2 uses these to posit an argument for the topic.

Since all our examples are terminated dialogues, from now on we will omit the matched-close moves.

We associate a commitment store with a dialogue, and let it grow monotonically over the course of the dialogue: If an agent posits an argument, the support is added to the commitment store; If an agent concedes a formula, it is added to the commitment store. A commitment store is therefore the union of all the supports of all the arguments that have been publicly posited along with all the formulae that have been publicly conceded by the agents so far. For this reason, when constructing an argument, an agent may make use of not only its own perbase but also those from the commitment store.

Definition 6. *A* **commitment store** Σ^t *is* $\emptyset$ *at* $t = 0$*, and for all* $t \geq 1$*, if* $m_t = \langle x, \mathsf{posit}, \langle \Phi, \phi\rangle\rangle$*, then* $\Sigma^t = \Sigma^{t-1} \cup \Phi$*, else if* $m_t = \langle x, \mathsf{concede}, \alpha\rangle$*, then* $\Sigma^t = \Sigma^{t-1} \cup \{\alpha\}$*, otherwise* $\Sigma^t = \Sigma^{t-1}$.

A protocol is a function that returns the set of moves that are legal for an agent to make at a particular point in a particular type of dialogue. Here we give the specific protocol that we require. It takes the dialogue and the identifier of the agent whose turn it is to move, and returns the set of legal moves that the agent may make.

Definition 7. *Let* D^t *be a dialogue s.t.* $\mathsf{Sender}(m_t) = \overline{x}$*, and* $\mathsf{Topic}(D^t) = \gamma$*. The* **protocol** *for agent* x *is a function* $\mathsf{Protocol}_x : \mathcal{D} \mapsto \wp(\mathcal{M})$ *s.t.* $\mathsf{Protocol}_x(D^t)$ *is*

$$P_x^{\mathsf{posit}}(D^t) \cup P_x^{\mathsf{concede}}(D^t) \cup \{\langle x, \mathsf{close}, \gamma\rangle\}$$

where $P_x^{\mathsf{posit}}(D^t)$ *is* $\{\ \langle x, \mathsf{posit}, \langle \Phi, \phi\rangle\rangle \mid \langle \Phi, \phi\rangle$ *is an argument* $\}$ *and* $P_x^{\mathsf{concede}}(D^t)$ *is* $\{\ \langle x, \mathsf{concede}, \phi\rangle \mid \phi \notin \Sigma^t\ \}$.

Note that it is straightforward to check conformance with the protocol as the protocol only refers to public elements of the dialogue (i.e. it does not refer to perbases). For instance, the dialogue in Ex. 2 conforms to the protocol.

In general, a **strategy** for agent x is a function $\mathsf{Strategy}_x : \mathcal{D} \mapsto \wp(\mathcal{M})$ that takes the dialogue D^t and returns a subset of the legal moves. A strategy is personal to an agent and the moves that it returns depends on the agent's private beliefs (i.e. its perbase Δ_x).

A well-formed dialogue is a dialogue that does not continue after terminated and that is generated by the strategy.

Definition 8. *A dialogue D^t is a* **well-formed dialogue** *iff, for all s ($s < t$), (1) D^s does not terminate at s and (2) if $\mathsf{Sender}(m_s) = \overline{x}$, then $m_{s+1} \in \mathsf{Strategy}_x(D^s)$.*

In the next section, we give a specific strategy for using inconsistent ontologies, and then discuss alternatives to it.

4 An Example of a Strategy

We will shortly give a specific strategy function using the following subsidiary notions.

Definition 9. *Let Ψ be a set of formulae. The set of arguments that can be formed from Ψ is $\mathsf{Args}(\Psi) = \{\langle \Phi, \psi\rangle \mid \Phi \subseteq \Psi$ and $\langle \Phi, \psi\rangle$ is an argument $\}$.*

The posit moves that occur after and including the last posit move for the topic are called live moves.

Definition 10. *For D^t, the set of live moves, $\mathsf{Live}(D^t)$, is*

$$\begin{aligned}\{m_k \mid\ & \text{there is an } i \text{ s.t. } i \leq k \leq t \\ & \text{and } m_i = \langle x, \mathsf{posit}, \langle \Phi, \mathsf{Topic}(D^t)\rangle\rangle \\ & \text{and there is not a } j \text{ s.t. } (i < j \leq t \\ & \qquad \text{and } m_j = \langle x, \mathsf{posit}, \langle \Psi, \mathsf{Topic}(D^t)\rangle\rangle\rangle\}\end{aligned}$$

For instance, for Ex. 2, as sequence $m_1, m_2, ..$ is made, the move m_3 is live until the move m_6 is made.

Given a dialogue, we define as follows whether an argument can be a novel undercut to extend the dialogue, as in Ex. 2, where each undercut is novel when posited.

Definition 11. *$\langle \Phi, \phi\rangle$ is a* **novel undercut** *for D^t iff there is a set of posit moves $\{n_1, ..., n_k\} \subseteq \mathsf{Live}(D^t)$ s.t. (1) n_1 posits an argument for the topic, (2) $\langle \Phi, \phi\rangle$ is an undercut for the posit of n_k, (3) for each i, $(1 < i \leq k)$, n_i is a posit for an undercut for the posit of n_{i-1}, (4) for each i, $(1 \leq i \leq k)$, Φ is not a subset of the support of the posit of n_i.*

We break a dialogue into phases. Intuitively, a phase is started by an agent positing an argument for the topic, and ended either by the dialogue ending, or by the next move being another posit move for the topic. So the live moves are in the latest phase in the dialogue.

Definition 12. *Let γ be the topic of D^t. A sequence of moves $m_i, ..., m_k$ is a* **phase** *in D^t iff $m_i, ..., m_k$ is a subsequence of D^t (i.e. D^t is $m_1, ..., m_i, ..., m_k, ..., m_t$ where $1 \leq i \leq k \leq t$), and m_i is a posit for γ and either t is k or m_{k+1} is a posit of γ and for all j s.t. $i < j \leq k$, m_j is not a posit for γ.*

To ensure that each concede move is relevant, the formula being conceded must have an atom in common with a formula already in the commitment store or with the topic of the dialogue in order to ensure that it can potentially be used with other formulae in a posit move.

$$\mathsf{Strategy}_x(D^t) = \begin{cases} S_x^{\mathsf{counter}}(D^t) & \text{iff } S_x^{\mathsf{counter}}(D^t) \neq \emptyset \\ S_x^{\mathsf{arg}}(D^t) & \text{iff } S_x^{\mathsf{arg}}(D^t) \neq \emptyset \text{ and } S_x^{\mathsf{counter}}(D^t) = \emptyset \\ S_x^{\mathsf{concede}}(D^t) & \text{iff } S_x^{\mathsf{concede}}(D^t) \neq \emptyset \\ & \quad \text{and } S_x^{\mathsf{counter}}(D^t) = \emptyset \text{ and } S_x^{\mathsf{arg}}(D^t) = \emptyset \\ \langle x, \mathsf{close}, \mathsf{Topic}(D^t)\rangle & \text{otherwise} \end{cases}$$

Fig. 1. The strategy function selects moves according to the following preference ordering (starting with the most preferred): posit moves, concede moves, and close moves. The conditions on the r.h.s. of each iff statement above imposes this ordering.

Definition 13. *A formula ϕ is* **relevant** *in D^t iff* $\mathsf{Names}(\phi) \cap \mathsf{Names}(\mathsf{Topic}(D^t)) \neq \emptyset$ *or* $\exists \psi \in \Sigma^t$ *s.t.* $\mathsf{Names}(\phi) \cap \mathsf{Names}(\psi) \neq \emptyset$.

We also require the following subsidiary functions. Essentially, S_x^{arg} gives the posits for the topic of the dialogue that have not been posited before, S_x^{counter} gives the undercuts for any argument in the current phase that have not been given so far in the current phase, and S_x^{concede} gives any formulae from its perbase or the lexbase that is relevant to the topic of the dialogue or to any formula already used in the dialogue.

Definition 14. *For the strategy function given in Fig. 1, we require the following sets of moves where $y \in \{1, 2\}$.*

$$\begin{aligned} S_x^{\mathsf{arg}}(D^t) = \{\langle x, \mathsf{posit}, \langle \Phi, \phi\rangle\rangle \in P_x^{\mathsf{posit}}(D^t) \mid \\ \langle \Phi, \phi\rangle \in \mathsf{Args}(\Delta_x \cup \Sigma^t) \\ \textit{and } \phi = \mathsf{Topic}(D^t) \\ \textit{and } \neg\exists m_i \textit{ s.t. } m_i \textit{ appears in } D^t \\ \textit{and } m_i = \langle y, \mathsf{posit}, \langle \Phi, \phi\rangle\rangle\} \end{aligned}$$

$$\begin{aligned} S_x^{\mathsf{counter}}(D^t) = \{\langle x, \mathsf{posit}, \langle \Phi, \phi\rangle\rangle \in P_x^{\mathsf{posit}}(D^t) \mid \\ \langle \Phi, \phi\rangle \in \mathsf{Args}(\Delta_x \cup \Sigma^t) \\ \textit{and } \langle \Phi, \phi\rangle \textit{ is a novel undercut for } D^t\} \end{aligned}$$

$$\begin{aligned} S_x^{\mathsf{concede}}(D^t) = \{\langle x, \mathsf{concede}, \phi\rangle \in P_x^{\mathsf{concede}}(D^t) \mid \\ (\phi \in \Delta_x \textit{ or } \phi \in \Pi) \\ \textit{and } \phi \textit{ is relevant in } D^t\} \end{aligned}$$

For the strategy defined in Fig. 1, a posit for the topic is made if possible. Then, the agents exhaustively present undercuts to this, and by recursion, undercuts to undercuts. When this is exhausted, the first phase has finished. If another posit for the topic can be made, then the second phase starts, and undercuts to this, and by recursion, undercuts to undercuts are exhaustively presented, thereby bringing the second phase to a close. Subsequent phases are constructed accordingly. The dialogue in Ex. 2 is generated by this strategy.

Now, we consider how to evaluate these dialogues. The set of arguments posited in a phase is called a constellation.

Definition 15. *Let $m_i, ..., m_k$ be a phase in D^t. X is the* **constellation** *for $m_i, ..., m_k$ iff $X = \{\langle \Phi, \alpha \rangle \mid m_j = \langle x, \mathsf{posit}, \langle \Phi, \alpha \rangle \rangle$ and $m_j \in \{m_i, ..., m_k\}\}$.*

Intuitively, a dialogue supports the topic iff there is a constellation that can be obtained from a phase of the dialogue and that the constellation can be arranged as an argument tree with an undefeated root argument for the topic of the dialogue. For this property, we require the following: A **complete** argument tree is an argument tree (N, E, X, f) such that if there is a node $n \in N$, and an argument $A \in X$ where A undercuts $f(n)$ and there is not a node $n' \in N$ such that n' is on the branch from the root to n and the support of A is a subset of the support of $f(n')$, then there is a child m of n such that $f(m)$ is A.

Definition 16. *For a dialogue D^t where $\mathsf{Topic}(D^t) = \alpha$, D^t* **supports** *α iff there is a phase $m_i, ..., m_k$ in D^t s.t. X is the constellation for $m_i, ..., m_k$ and there is a complete argument tree for α (N, E, X, f) s.t. its root argument is undefeated.*

The dialogue in Ex. 2 has two phases and supports $C(a)$.

Example 3. $\Delta_1 = \{\neg D(a), D \sqsubseteq A, A \sqsubseteq \neg C\}$; $\Delta_2 = \{D(a), D \sqsubseteq C, D \sqsubseteq E, E \sqsubseteq \neg A\}$; $\Pi = \emptyset$.

$$\langle 1, \mathsf{query}, C(a) \rangle$$
$$\langle 2, \mathsf{posit}, \langle \{D(a), D \sqsubseteq C\}, C(a) \rangle \rangle$$
$$\langle 1, \mathsf{posit}, \langle \{D \sqsubseteq A, A \sqsubseteq \neg C\}, D \sqsubseteq \neg C \rangle \rangle$$
$$\langle 2, \mathsf{posit}, \langle \{D \sqsubseteq E, E \sqsubseteq \neg A\}, D \sqsubseteq \neg A \rangle \rangle$$
$$\langle 1, \mathsf{posit}, \langle \{\neg D(a)\}, \neg D(a) \rangle \rangle$$

Here there is just one phase $(m_2, ..., m_5)$. Agent 2 gives an argument, then Agent 1 gives an undercut to it, and then Agent 2 gives an undercut to that. Finally, Agent 1 comes back with an undercut to the first argument by Agent 2. As a result, $C(a)$ is not supported.

So each dialogue generated with this strategy ensures that all the possible arguments relevant to the query are presented, and these arguments can be assessed to determine whether the query is supported in the dialogue. Because the strategy returns choices of moves, the dialogue is not necessarily unique given a query, but the alternative dialogues that can be obtained given a query are isomorphic, and hence the constellations will be the same.

With the protocol, we can also define alternative strategies that ensure alternative useful behaviour. For instance, we could define a strategy that stops the dialogue when a phase has occurred that alone can be used to demonstrate support for the topic, or we could define a strategy that also gives the arguments for the negation of the topic, or we could define a strategy that only allows a concede move of a formula when that formula is consistent with its perbase.

5 Properties of Dialogical Argumentation

The constraints on the strategy function are such that we can show that all dialogues terminate (as agents' perbases are finite, hence there are only a finite number of different moves that can be generated and agents cannot repeat these moves *ad infinitum*).

Proposition 1. *For any well-formed dialogue D^t, there exists a D^u s.t. $t \leq u$ and D^u terminates at u and D^u extends D^t (i.e. the first t moves of D^u are given by D^t).*

A dialogue is sound if and only if, if an argument is generated by the dialogue, then it can also be constructed from the union of the perbases and the lexbase.

Definition 17. *Let D^t be a well-formed dialogue. We say that D^t is* **sound** *iff, for each s, if $s \leq t$ and $m_s = \langle x, \mathsf{posit}, \langle \Phi, \phi \rangle \rangle$, then $\langle \Phi, \phi \rangle$ is an argument s.t. $\Phi \subseteq (\Delta_x \cup \Delta_{\overline{x}} \cup \Pi)$.*

When an agent posits an argument, it must be able to construct the argument from its perbase and the commitment store. This is clear from the definition of the strategy. From this, and the fact that the commitment stores are only updated when a posit or concede move is made, we get that a commitment store is always a subset of the union of the perbases and the lexbase. From these observations, we get soundness.

Proposition 2. *If D^t is a well-formed dialogue, then D^t is sound.*

Similarly, a dialogue is complete if and only if, if the dialogue terminates at t and it is possible to construct an argument for the topic of the dialogue from the union of the perbases and lexbase, then that argument will eventually be posited by one of the agents.

Definition 18. *Let D^t be a well-formed dialogue and $\mathsf{Topic}(D^t) = \gamma$. We say that D^t is* **complete** *iff, if there is a argument $\langle \Phi, \gamma \rangle$ s.t. $\Phi \subseteq (\Delta_x \cup \Delta_{\overline{x}} \cup \Pi)$, then there is a move $\langle x, \mathsf{posit}, \langle \Phi, \gamma \rangle \rangle$ in D^t.*

In order to show that all dialogues are complete, we need some further lemmas. The first states: If an agent cannot produce, given their perbase and the commitment store, an argument for the topic of the dialogue, then the strategy forces them to concede formulae from their perbase and the lexbase, thus adding to the commitment store.

Lemma 1. *Let D^t be a well-formed dialogue with $\mathsf{Topic}(D^t) = \gamma$. If $S_x^{\mathsf{arg}}(D^t) = \emptyset$ and $S_x^{\mathsf{counter}}(D^t) = \emptyset$ and there is a $\beta \in \Delta_x \cup \Pi$ s.t. β is relevant for D^t and $\beta \notin \Sigma^t$ then $\langle x, \mathsf{concede}, \beta \rangle \in \mathsf{Strategy}_x(D^t)$.*

Following from the above lemma, we obtain the following lemma that says if there is an argument for the topic of the dialogue that can be obtained by pooling the agents' perbases and the lexbase, then, once the dialogue has terminated, there is the support for this argument in the union of the agent's perbase with the commitment store.

Lemma 2. *Let D^t be a well-formed dialogue that terminates at t with $\mathsf{Topic}(D^t) = \gamma$. If there is a $\Phi \subseteq (\Delta_x \cup \Delta_{\overline{x}} \cup \Pi)$ s.t. $\langle \Phi, \gamma \rangle$ is a argument, then $\Phi \subseteq (\Delta_x \cup \Sigma^t)$.*

The next lemma says that if there is an argument for the topic of the dialogue that can be obtained from an agent's perbase and the commitment store, then the strategy will force the posit of that argument at some point in the dialogue.

Lemma 3. *Let D^t be a well-formed dialogue that terminates at t with $\mathsf{Topic}(D^t) = \gamma$. If there is a $\Phi \subseteq (\Delta_x \cup \Sigma^t)$ s.t. $\langle \Phi, \gamma \rangle$ is an argument, then there is an s s.t. $s < t$ and $m_s = \langle x, \mathsf{posit}, \langle \Phi, \gamma \rangle \rangle$.*

Using the above lemmas, it is straightforward to now show that dialogues are complete.

Proposition 3. *If D^t is a well-formed terminated dialogue, then D^t is complete.*

A dialogue is faithful if it supports the topic iff the arguments that can be constructed from the union of the perbases and the lexbase can be arranged as a complete argument tree for the topic where the root argument is undefeated.

Definition 19. *Let D^t be a well-formed dialogue,* $\mathsf{Topic}(D^t) = \gamma$*, and* $X = \mathsf{Args}(\Delta_x \cup \Delta_{\overline{x}} \cup \Pi)$*. We say that D^t is* **faithful** *when the following equivalence holds.*

$$
\begin{array}{l}
D^t \textit{ supports } \gamma \textit{ iff} \\
\quad \textit{there is a complete argument tree } (N, E, X, f) \textit{ for } \gamma \\
\quad \textit{where } f(n) \textit{ is undefeated for root } n
\end{array}
$$

From completeness and soundness, for topic γ, we get that $\langle \Phi, \gamma \rangle$ is an argument from the union of the agents' perbases and the lexbase iff there is a posit move of $\langle \Phi, \gamma \rangle$ in the dialogue. Furthermore, there is exactly one phase for each of these arguments $\langle \Phi, \gamma \rangle$. We can then generalize the completeness and soundness results so that for each phase, if $\langle \Phi, \gamma \rangle$ is the argument that starts the phase, then the posit moves made in the phase contain exactly, the undercuts of $\langle \Phi, \gamma \rangle$, and by recursion, the novel undercuts to each undercut, that could be obtained from the union of the agents' perbases and the lexbase. Therefore, each phase is isomorphic to a complete argument tree for the topic that can be obtained from the union of the agents' perbases and the lexbase, and each complete argument tree for the topic that can be obtained from the union of the agents' perbases and the lexbase is isomorphic to a phase. Hence, we get the following.

Proposition 4. *If D^t is a well-formed terminated dialogue, then D^t is faithful.*

A corollary of this proposition is that all the minimal inconsistent subsets of the union of the agents' ontologies that involve the topic of the dialogue can be recovered from the commitment store of the dialogue. In other words, from each phase of the dialogue, the minimal inconsistent subsets involving the query can be obtained from the posit starting the phase, and the undercuts to this posit.

So in this section, we have shown that the dialogues always terminate, they are sound (any argument posited is an argument that can come from the union of the agents' ontologies plus the lexbase), they are complete (any argument for the topic obtainable from the union of the agents' ontologies plus the lexbase, is posited in the dialogue), they are faithful (any argument for the topic shown to be undefeated given the union of the agents' ontologies plus the lexbase, is shown to be undefeated in a phase in the dialogue, and *vice versa*). These properties mean that the dialogical argumentation is equivalent to argumentation with the union of the agents' ontologies plus the lexbase, but with the advantage that it is not necessary to copy all of each ontology to each agent in order to undertake the argumentation. Rather, just enough knowledge is exchanged in the posit and concede moves for the argument trees to be implicitly constructed in the dialogue.

6 Conclusions

We have presented a dialogical argumentation framework for using multiple ontologies. In comparison with the maxcon approach, we do not lose information, rather we keep it all, and we do not make arbitrary choices. Furthermore, any inference that can be obtained from the maxcon approach can be obtained from our approach, but not vice versa. In comparison with the oracle approach, we may get inferior inferences (i.e. inferences that with the benefit of some oracle are not deemed to be good), but the significant advantage here is that we do not need to copy and merge all of the ontology for each agent to use knowledge from other agents' ontologies.

There are other proposals for argumentation with ontologies. In [10,13], all the ontological knowledge is in a centralized location, and so they do not get the advantages that come from using dialogues, and in [12], dialogues are used for discussing ontology alignments, but not for querying the ontological knowledge.

Another advantage of our approach is that it allows an agent to determine how its perbase differs from another. This can then be used by the agent to decide how to update its perbase. For instance, if it regards the other agent as more reliable, or if it has had the same conflict with a number of agents, it may choose to delete some of its own knowledge.

Our system also allows the definition of alternative strategies that ensure alternative intelligent behaviour. For instance, we can define a more efficient strategy that only builds a pruned version of the argument tree and yet still produces faithful dialogues. Also we can refine how agents chose to concede a move from the lexbase (e.g. by only allowing a formula to be conceded if it is consistent with the agent's perbase, or by more tightly coupling concession to the search for premises for arguments and counterarguments).

Our proposal is influenced by [5], but it does involve some substantial developments over it: (1) That paper was for a simple propositional defeasible logic whereas this paper is for much richer description logics; (2) That paper was only about finding arguments whereas this paper is about the more complex issue of finding warranted arguments; (3) That paper has different moves, protocol and strategy to this paper; (4) This paper has a more general framework based on phases that is valuable for supporting and auditing diverse protocols and strategies for arguing about ontologies.

The dialogues generated by our system allow agents to jointly construct arguments for a topic and to determine if this topic is supported given the union of available knowledge. As far as we are aware, there are only three other dialogue systems that share this same aim and have been shown to have similar properties to ours (i.e. are faithful) [3,16,6]; However, none deal with ontological knowledge and the first two impose restictions on the distribution of formulae between the agents. [14] also consider completeness properties for general classes of protocol that allow agents to jointly construct argument trees, but they do not allow the joint construction of arguments nor do they consider ontological knowledge.

In future work, we will develop a range of more refined strategies including for conceding formulae from the lexbase. We will also extend our system to address wider issues concerning semantic heterogenity arising between ontologies.

References

1. Amgoud, L., Cayrol, C.: A model of reasoning based on the production of acceptable arguments. Annals of Mathematics and Artificial Intelligence 34, 197–216 (2002)
2. Baader, F., McGuiness, D.L., Nardi, D., Patel-Schneider, P. (eds.): Description Logic Handbook: Theory, implementation and applications. Cambridge University Press, Cambridge (2002)
3. Bentahar, J., Alam, R., Maamar, Z.: An argumentation-based protocol for conflict resolution. In: KR 2008-workshop on Knowledge Representation for Agents and Multi-Agent Systems, pp. 19–35 (2008)
4. Besnard, P., Hunter, A.: A logic-based theory of deductive arguments. Artificial Intelligence 128, 203–235 (2001)
5. Black, E., Hunter, A.: A generative inquiry dialogue system. In: 6th Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems, pp. 1010–1017 (2007)
6. Black, E., Hunter, A.: An inquiry dialogue system. Autonomous Agents and Multi-Agent Systems 19(2), 173–209 (2009)
7. Dung, P., Kowalski, R., Toni, F.: Dialectical proof procedures for assumption-based admissible argumentation. Artificial Intelligence 170, 114–159 (2006)
8. Flouris, G., Huang, Z., Pan, J.Z., Plexousakis, D., Wache, H.: Inconsistencies, negations and changes in ontologies. In: 21st AAAI Conf., pp. 1295–1300 (2006)
9. García, A., Simari, G.: Defeasible logic programming: An argumentative approach. Theory and Practice of Logic Prog. 4(1), 95–138 (2004)
10. Gómez, S., Chesñevar, C., Simari, G.: An argumentative approach to reasoning with inconsistent ontologies. In: Knowledge Representation and Ontologies Workshop, pp. 11–20 (2008)
11. Huang, Z., van Harmelen, F., ten Teije, A.: Reasoning with inconsistent ontologies. In: 19th Int. Joint Conf. on Artificial Intelligence, pp. 454–459 (2005)
12. Laera, L., Blacoe, I., Tamma, V., Payne, T., Euzenat, J., Bench-Capon, T.: Argumentation over ontology correspondences in MAS. In: 6th Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems, pp. 1293–1300 (2007)
13. Moguillansky, M., Rotstein, N., Falappa, M.: A theoretical model to handle ontology debugging & change through argumentation. In: 2nd Int. Workshop on Ontology Dynamics (2008)
14. Parsons, S., McBurney, P., Sklar, E., Wooldridge, M.: On the relevance of utterances in formal inter-agent dialogues. In: 6th Int. Joint Conf. on Autonomous Agents and Mutli-Agent Systems, pp. 1002–1009 (2007)
15. Prakken, H., Sartor, G.: Argument-based extended logic programming with defeasible priorities. Journal of Applied Non-classical Logic 7, 25–75 (1997)
16. Thimm, M., Kern-Isberner, G.: A distributed argumentation framework using defeasible logic programming. In: 2nd Int. Conf. on Computational Models of Argument, pp. 381–392 (2008)

A Model Based on Possibilistic Certainty Levels for Incomplete Databases

Patrick Bosc[1], Olivier Pivert[1], and Henri Prade[2]

[1] Irisa – Enssat, University of Rennes 1
Technopole Anticipa 22305 Lannion Cedex France
[2] IRIT, CNRS and University of Toulouse, 31062 Toulouse Cedex 9, France
bosc@enssat.fr, pivert@enssat.fr, prade@irit.fr

Abstract. This paper deals with the modeling and querying of a database containing uncertain attribute values, in the situation where some knowledge is available about the more or less certain value (or disjunction of values) that a given attribute in a given tuple can take. This is represented in the setting of possibility theory. A relational database model suited to this context is introduced, and selection, join and union operators of relational algebra are extended so as to handle such relations. It is shown that i) the model in question is a strong representation system for the algebraic operators considered, and that ii) the data complexity associated with the extended operators in this context is the same as in the classical database case, which makes the approach highly scalable. A possibilistic logic encoding of the model is also outlined.

1 Introduction

Thirty years ago, two papers authored respectively by E.F. Codd [5] and W. Lipski [16] were published the same year on the issue of extending the relational database model so as to represent unknown (null) values. Since then, many authors have made diverse proposals to model and handle databases involving uncertain or incomplete data. In particular, the last two decades have witnessed an explosion of research on this topic (cf. e.g., [2,3,6,7,11,13,15,18] for some recent ones). The notion of an uncertain (or imprecise) database covers two aspects:

- Attribute uncertainty: i.e. when some attribute values are ill-known;
- Existential uncertainty: i.e. when the existence of some tuples is itself uncertain.

In this paper, we deal with the first aspect only. Even though most of the literature about uncertain databases considers probability theory as the underlying uncertainty model, some approaches rather rest on possibility theory [19]. The initial idea consisting in applying possibility theory to the modeling of incomplete databases goes back to the early 80's with the work by Prade and Testemale [17]. More recent advances on this topic can be found in [4]. In the present paper, we introduce a new idea which is to use the notion of certainty (necessity) to qualify the likeliness that an ill-known piece of data takes a given value. With respect to using probability theory, one expects the following advantages:

L. Godo and A. Pugliese (Eds.): SUM 2009, LNAI 5785, pp. 80–94, 2009.

– the qualitative nature of the model makes easier the elicitation of the degrees attached to the various candidate values;
– in probability theory, the fact that the sum of the degrees from a distribution must equal 1 makes it difficult to deal with incompletely known distributions;
– there does not exist any probabilistic logic which is complete and works locally as possibilistic logic does [10]: this can be problematic in the case where the degrees attached to certain pieces of data must be automatically deduced from those attached to some other pieces of data (when data coming from different sources have to be merged into a single database, for instance).

With respect to probabilistic databases *and* possibilistic ones, the main advantage — and this constitutes a major result of this paper — lies in the fact that operations from relational algebra can be extended in a simple way and with a data complexity which is the same as in a precise database context.

The rest of the paper is organized as follows. Section 2 is devoted to a reminder about basic notions concerning the interpretation of an uncertain database in terms of possible worlds, as well as the property characterizing a strong representation system. In Section 3, we present the main features of the certainty-based model and we describe a method that can be used to prove that it is a strong representation system for a given set of operators. Section 4 gives the definition of the compact operators (i.e., operators working directly on tables of the model) for selection, join and union in this framework. In Section 5, a possibilistic logic encoding of the model is briefly discussed, before concluding.

2 Basic Notions

2.1 Interpretation of an Uncertain Database in Terms of Worlds

The possible worlds model is founded on the fact that uncertainty in data makes it impossible to define what precisely the real world is. One can only describe the set of possible worlds which are consistent with the available information. As far as a table T conveys some imprecision/uncertainty, several interpretations (I) can be drawn from T and the set of all the interpretations of T is denoted by $rep(T)$. The notation $rep(D)$ extends naturally to an uncertain database D involving several tables. Note: a regular database is a special case of an uncertain one which has only one interpretation.

In an incomplete relational database D, attribute values in the tables of D may be imprecisely known and then represented as sets of candidates of various kinds, including null values, sets and intervals, and distributions.

From a semantic point of view, such an uncertain database D can be interpreted in terms of a (possibly infinite) set of usual databases, also called worlds W_1, ..., W_p, and $rep(D) = \{W_1, ..., W_p\}$. This view establishes a strong connection between uncertain and regular databases, which is particularly interesting since it offers a canonical definition of the meaning of queries addressed to uncertain databases as it will be seen later. Any world W_i is obtained by choosing a candidate value in each set appearing in every relation T_j pertaining to D. One

of these (regular) databases, let us say W_k, is supposed to correspond to the actual state of the universe modeled. The assumption of independence between the various sets of candidates is usually made and then any world W_i corresponds to a conjunction of independent choices.

Example 1. Let us consider the uncertain database D involving a single relation *im* whose schema is $IM(\#i, airc, date, place)$. Relation *im* is assumed to describe satellite images of aircrafts. Each image, numbered ($\#i$), was taken on a certain location (*place*) a given day (*date*) and it is supposed that it includes a single aircraft (*airc*). With the extension of *im* depicted in Table 1 six worlds can be drawn, W_1, W_2, W_3, W_4, W_5 and W_6 since there are three candidates for *date* in the first tuple and two candidates for *airc* in the second one. Two of the worlds associated with the uncertain relation *im* are represented in Figure 1. $\diamond$

Table 1. An extension of relation *im*

$\#i$	*airc*	*date*	*place*
i_1	a_1	$\{d_1, d_3, d_7\}$	c_1
i_3	$\{a_3, a_4\}$	d_1	c_2

$\#i$	*airc*	*date*	*place*
i_1	a_1	d_1	c_1
i_3	a_3	d_1	c_2

$\#i$	*airc*	*date*	*place*
i_1	a_1	d_7	c_1
i_3	a_4	d_1	c_2

Fig. 1. Two worlds associated with relation *im*

2.2 Strong Representation Systems and Compact Calculus

When dealing with an uncertain database D, a very important issue is that of the efficiency of the querying process. A naive way of doing would be to make explicit all the interpretations of D (at least when they are finite) in order to query each of them. Such an approach is intractable in practice and it is of prime importance to find a more realistic alternative. To this end, the notion of a representation system has been introduced — initially by Imielinski and Lipski [14] — and discussed in [1]. The basic idea is to look for a way for representing both initial tables and those resulting from queries in such a way that the representation of the result of a query q against any database D (made of tables T_1, ..., T_p) denoted by $q(D)$, is equivalent (in terms of interpretations, or worlds) to the set of results obtained by applying q to every interpretation of D, i.e.:

$$rep(q(D)) = q(rep(D)). \tag{P1}$$

If property P1 holds for a representation system ρ and a subset σ of the relational algebra, ρ is called a *strong representation system* for σ.

From a querying point of view, P1 permits to envisage a direct (or compact) calculus of a query Q, in the sense that Q applies to D itself without making the worlds explicit. So doing, provided that relational operations are defined over tables of the system considered, reasonable performances can be expected.

2.3 Reminder about Possibility Theory

Possibility theory [19] offers a qualitative model for uncertainty where a piece of information is represented by means of a possibility distribution encoding a complete preorder over the possible situations. More formally, a possibility distribution is a function π from a domain X to the unit interval [0, 1] and $\pi(a)$ expresses the degree to which a is a possible value for the considered variable. The normalization condition imposes that at least one of the values of the domain (a_0) is completely possible, i.e., $\pi(a_0) = 1$. When the domain is discrete, a possibility distribution can be written $\{\pi_1/a_1, ..., \pi_n/a_n\}$ where a_i is a candidate value and π_i its possibility degree. Any event E defined in the powerset of X is characterized by two measures Π and N. The axioms related to the measure of possibility Π are the following:

- $\Pi(X) = 1$ (which requires the normalization condition mentioned above),
- $\Pi(\emptyset) = 0$,
- $\Pi(E_1 \cup E_2) = \max(\Pi(E_1), \Pi(E_2))$

and the measure of possibility of the event E is derived from the possibility distribution associated with the concerned variable in the following way: $\Pi(E) = \max_{x \in E} \pi(x)$. It can be seen that the third axiom differs from that of probability theory in the sense that the measure is not additive. The possibility of the conjunction of two events (see, e.g., [9]) only obey the inequality $\Pi(E_1 \cap E_2) \leq \min(\Pi(E_1), \Pi(E_2))$, but if the underlying variables are non interactive (i.e., E_1 and E_2 are logically independent) $\Pi(E_1 \cap E_2) = \min(\Pi(E_1), \Pi(E_2))$.

The only relationship between the possibility of $\overline{E}$ (the opposite event of E) and that of E is: $\max(\Pi(E), \Pi(\overline{E})) = 1$. In order to have a better characterization of the event E, the measure of certainty (or necessity) N has been introduced: $N(E) = 1 - \Pi(\overline{E})$. In other words, the less possible $\overline{E}$, the more certain E. Due to the duality between these two measures, the following formulas hold in the general case:

- $N(X) = 1$; $N(\emptyset) = 0$;
- $N(E_1 \cap E_2) = \min(N(E_1), N(E_2))$; $N(E_1 \cup E_2) \geq \max(N(E_1), N(E_2))$,

and in case of non-interactivity: $N(E_1 \cup E_2) = \max(N(E_1), N(E_2))$.

We now illustrate the use of possibility distributions for representing candidate values for some attributes in the context of the database describing satellite images of aircrafts (cf. Example 1).

In the first tuple of im depicted in Table 2, d_1 is completely possible and it is more possible than d_3. Let us mention that Table 2 corresponds to a relation from the model proposed in [17] while the model proposed in [4] involves an additional attribute expressing the certainty for a tuple to have a representative in any possible world (this is mandatory in order to guarantee property P1).

Table 2. An extension of relation im

$\#i$	$airc$	$date$	$place$
i_1	a_1	$\{1/d_1,\ 0.7/d_3,\ 0.4/d_7\}$	c_1
i_3	$\{1/a_3,\ 1/a_4\}$	d_1	c_2

3 The Model

As that described in [4], the model we propose is based on possibility theory [19], but it represents the values that are more or less certain instead of those which are more or less possible. This corresponds to the most important part of information (in this approach, a possibility distribution is "synthetized" by keeping its most plausible elements). The idea is to attach a certainty level to each piece of data (by default, a piece of data has certainty 1). Certainty is modeled as a lower bound of a necessity measure. We assume that there always exists a key non pervaded with uncertainty in base relations. For instance, $\langle 037$, John, $(40, \alpha)\rangle$ denotes the existence of a person named John, whose age is 40 with certainty α. Then the possibility that his age differs from 40 is upper bounded by $1 - \alpha$ without further information on the respective possibility degrees of other possible values. In the following, uncertainty may pervade *attribute values*, for instance when a source cannot fully guarantee their values, e.g., for those attributes whose value may change without notice, or data evolving with time.

3.1 Main Features of the Model

In the proposed model, to each uncertain attribute value is attached a certainty degree. The underlying possibility distribution associated with an uncertain attribute value $(a,\ \alpha)$ is $\{1/a,\ (1-\alpha)/\omega\}$ where ω denotes $domain(A) - \{a\}$, A being the attribute considered (due to the duality necessity (certainty) / possibility: $N(a) \geq \alpha \Leftrightarrow \Pi(a) \leq 1 - \alpha$ [8]). For instance, let us assume that the domain of attribute City is {Brest, Quimper, Rennes}. The uncertain attribute value (Brest,α) is assumed to correspond to the possibility distribution {1/Brest, $(1-\alpha)$/Quimper, $(1-\alpha)$/Rennes}. More generally, the model can also deal with disjunctive uncertain values, and the underlying possibility distributions are of the form $max(\mu_S(x),\ 1 - \alpha)/x$ where S is an α-certain subset of the attribute domain and $\mu_S(x)$ equals 1 if $x \in S$, 0 otherwise.

Moreover, since some operations (e.g., the selection) may create "maybe tuples", each tuple t from an uncertain relation r has to be associated with a degree N expressing the certainty that t exists in r. It will be denoted by N/t.

Example 2. Let us consider the relation r of schema (#id, Name, City) containing tuple $t_1 = \langle 1,\ John,\ (Rennes,\ 0.8)\rangle$, and the query "find the persons who live in Rennes". Let the domain of attribute City be {Brest, Quimper, Rennes}. The answer contains $0.8/t_1$ since it is 0.8 certain that t_1 satisfies the requirement, while the result of the query "find the persons who live

in Rennes, Brest or Quimper" contains $1/t_1$ since it is totally certain that t_1 satisfies the condition. ◇

To sum up, a tuple $\alpha/\langle 037,\ John,\ (Rennes,\ \beta)\rangle$ from relation r means that it is α certain that person 037 exists in the relation, and that it is β certain that 037 lives in Rennes (independently from the fact that it is or not in relation r).

Given a query, we only look for answers that are somewhat certain. Consider the relations r and s given in Table 3 and a query asking for the persons who live in a city where there is a flea market), then *John* will be retrieved with a certainty level equal to $min(\alpha,\ \beta)$ (in agreement with the calculus of necessity measures [8]). Note that it should also be possible to find the tuples that are more or less certainly *not* answers to the query. There are usually many. All the other tuples are only possible, but we do not try to evaluate to what extent. Here, *Mary* with certainty level $min(\delta,\ \gamma)$ is not an answer to the query.

Table 3. Relations r (top) and s (bottom)

#id	*Name*	*City*	*N*
1	John	(Brest, α)	1
2	Mary	(Lannion, δ)	1

City	*Flea Market*	*N*
Brest	(yes, β)	1
Lannion	(no, γ)	1

It is also possible to accommodate cases of disjunctive information in this setting. For instance, $\langle 3,$ Peter, (Albi $\vee$ Toulouse, 0.8)$\rangle$ represents the fact that it is 0.8-certain that the person number 3 named Peter lives in Albi or in Toulouse.

3.2 Strong Representation System

Let us now examine what becomes of property P1 in such a context. Let us denote by D an uncertain database involving certainty levels, $poss(D)$ the corresponding uncertain database involving possibility distributions (cf. the comment above about the underlying possibility distributions associated with somewhat certain values), q an algebraic query, and q_c the compact version of q suited to the certainty-based model. The counterpart of property P1 is:

$$q_c(D) = \psi(q(rep(poss(D)))) \tag{P2}$$

where $\psi(r')$ denotes the certainty-based relation which gathers the tuples somewhat certainly in the intersection of all the (more or less) possible worlds from the set r' (each world from r' represents a possible result of q applied to D). Operation ψ works as follows. First, one attaches an identifier to every tuple from D involving at least one uncertain value (this is necessary when q involves a projection which removes the keys). Note that this identifier is virtual and does not

impact the result of the operations performed on the worlds (intersection and difference, in particular). When duplicates are eliminated in a world, the list of identifiers attached to the tuples merged is attached to the tuple resulting from the merging. For each tuple present in a world from $q(rep(poss(D)))$, denoting by id its identifier, or (id_1, id_2) in case of a binary operation, one has to check:

- whether there exists a completely possible world from $q(rep(poss(D)))$ from which id — or (id_1, id_2) for a join, id_1 and/or id_2 for an intersection, id_1 for a difference — is absent. If it is the case, $\psi(q(rep(poss(D))))$ does not contain any tuple identified by id;
- otherwise, for each attribute, one gathers into a disjunction V the values associated with id in the different completely possible worlds; the certainty degree associated with such a disjunction equals 1 minus the maximal possibility degree attached to a world containing id associated with a value absent from V. Notice that for the intersection, the "empty tuples" that may have been generated must be considered too. The certainty degree associated with the "compact tuple" produced is equal to 1 minus the maximal possibility degree associated with a world from which id is absent.

Due to space limitation, we only deal with three relational operators in this paper: selection, join and union; however it is possible to use property P2 to prove that the model is a strong representation system for the whole relational algebra. It is important to emphasize that this property makes the certainty-based model we propose the only *simple* uncertain database model that can handle the join operation in a compact and *very efficient* way (with a data complexity which is the same as in the classical database case).

4 The Operators

The goal of this section is to define the compact version of the three operators from $\sigma = \{$selection, join, union$\}$ and to show that the certainty-based database model is a strong representation system — i.e., that property P2 then holds.

4.1 Selection

Let us consider a relation r of schema (A, X) where A is an attribute and X is a set of attributes, and a selection condition ϕ on A. Let us denote by $s(t.A)$ the disjunctive set of values — which may be reduced to a singleton — somewhat certain for attribute A in tuple μ/t, and by $c(t.A)$ the associated certainty level.

Let us first deal with the case where ϕ is of the form $attribute\ \theta\ q$ where q denotes a constant.

$$\begin{aligned}\text{select}(r,\ A\,\theta\,q) = \{&\mu'/t \mid \exists \mu/t \in r \text{ s.t. } \forall a_i \in s(t.A),\ a_i\,\theta\,q \wedge \\ &\mu' = min(\mu,\ 1) = \mu \text{ if } \forall a_i \in domain(A),\ a_i\,\theta\,q; \\ &\mu' = min(\mu,\ c(t.A)) \text{ otherwise}\}.\end{aligned}$$

Proof. Let us show that property P2 is satisfied. Since the selection operates on tuples individually, one can assume that r contains only one tuple. Let us denote it by μ/t and assume that $s(t.A) = (a_1 \vee a_2 \vee \ldots \vee a_n)$ and $c(t.A) = \alpha$. Let us attach identifier k to this tuple. Let us recall that the maximal possibility distribution associated with $s(t.A)$ is $\{1/a_1,\ 1/a_2,\ \ldots,\ 1/a_n,\ 1-\alpha/\omega\}$ where $\omega = domain(A) - \{a_1,\ , a_2,\ \ldots,\ a_n\}$. Three cases may appear:

- $\exists a_i \in \{a_1,\ a_2,\ \ldots,\ a_n\}$ such that $\neg(a_i\ \theta\ q)$:
  then, there exists a completely possible world of the result where k is not present. The certainty degree attached to k is zero and $\psi(q(rep(poss(D))))$ is empty, which is consistent with the definition of the compact selection.
- $\forall a_i \in domain(A)$, $a_i \theta q$: then, k is present in every completely possible world and the only possible world where k is not present is the empty world (possibility $1-\mu$). Hence, the certainty degree attached to k in $\psi(q(rep(poss(D))))$ is $1-(1-\mu) = \mu$. The most possible world where k has an A-value which does not belong to $s(t.A)$ in the result has the possibility degree $1-\alpha$. Hence, the certainty degree attached to the A-value $s(t.A)$ in the tuple identified by k in $\psi(q(rep(poss(D))))$ is $1-(1-\alpha) = \alpha$. This is consistent with the compact definition of selection where $s(t.A)$ and $c(t.A)$ are kept unchanged.
- $\forall a_i \in \{a_1,\ a_2,\ \ldots,\ a_n\},\ a_i\ \theta\ q \wedge \exists u_i \in domain(A)$ such that $\neg(u_i\ \theta\ q)$:
  then, k is present in every completely possible world. The most possible world where k is not present is either the empty world (possibility $1-\mu$) or is made of a tuple $\langle k,\ u_i,\ \text{t.X}\rangle$ where $u_i \in \omega$ (possibility $1-\alpha$). Thus, this most possible world has the possibility degree $max(1-\mu,\ 1-\alpha)$. Hence, the certainty degree attached to k in $\psi(q(rep(poss(D))))$ is $1-max(1-\mu, 1-\alpha) = min(\mu,\ \alpha)$. The most possible world where k has an A-value which does not belong to $s(t.A)$ in the result has the possibility degree $1-\alpha$. Hence, the certainty degree attached to the A-value $s(t.A)$ in the tuple identified by k in $\psi(q(rep(poss(D))))$ is $1-(1-\alpha) = \alpha$. This is consistent with the compact definition of selection. ∎

Let us now consider a condition ϕ of the form: $attribute_1\ \theta\ attribute_2$. The definition of the selection in this case is:

$$\begin{aligned}\text{select}(r,\ A_1\ \theta\ A_2) = \\ \{\mu'/t \mid \exists \mu/t \in r\ \text{ s.t. } \forall a_{1,i} \in s(t.A_1),\ \forall a_{2,j} \in s(t.A_2),\ a_{1,i}\ \theta\ a_{2,j} \wedge \\ \mu' = \mu \text{ iff } \forall (u,\ v) \in (domain(A_1) \times domain(A_2),\ u\ \theta\ v; \\ \mu' = min(\mu,\ c(t.A_1),\ c(t.A_2)) \text{ otherwise}\}.\end{aligned}$$

From the previous definitions, it immediately follows that the data complexity of the selection operation is linear (as usual). The example hereafter illustrates the case of a conjunctive selection condition.

Example 3. Let us consider the database D made of the sole relation *emp* of schema (id, name, city, job). Let us suppose that *emp* only contains tuple t = $0.9/\langle 17,$ John, (Paris, 0.8), (Engineer, 0.7)$\rangle$ and let us consider the query:

$$q = \text{select(emp, city = 'Paris' } \textbf{and} \text{ job = 'Engineer')}.$$

Its compact result is $0.7/\langle 17, \text{John}, (\text{Paris}, 0.8), (\text{Engineer}, 0.7)\rangle$. Let us show that property P2 is satisfied. Identifier 17 is present in every completely possible world. The most possible world where 17 is not present is made of the tuple $\langle 17, \text{John}, \text{Paris}, \omega_2)\rangle$ and has the possibility degree $min(1, 1-0.7)) = 0.3$. Hence, the certainty degree attached to 17 in the result is $1 - 0.3 = 0.7$. The most possible world where 17 has a *city* value different from "Paris" in the result has the possibility degree $1 - 0.8 = 0.2$. Hence, the certainty degree attached to the *city* value "Paris" in the tuple identified by 17 in the result is $1-0.2 = 0.8$. The most possible world where 17 has a *job* value different from "Engineer" in the result has the possibility degree $1-0.7 = 0.3$. Hence the certainty degree attached to the *job* value "Engineer" in the tuple identified by 17 in the result is $1 - 0.3 = 0.7$. The compact calculus is thus correct. ◇

4.2 Join

The compact definition of the join is:

$$\text{join}(r_1,\, r_2,\, A\ \theta\ B) = \{min(\alpha,\, \beta,\, \chi,\, \delta)/t_1 \oplus t_2 \mid \exists \alpha/t_1 \in\ r_1,\, \exists \beta/t_2 \in\ r_2 \text{ s.t.}\ card(s(t_1.A)) = 1 \wedge card(s(t_2.B)) = 1 \wedge s(t_1.A)\ \theta\ s(t_2.A) \wedge c(t_1.A) = \chi \wedge c(t_2.B) = \delta\}$$

where $\oplus$ denotes the concatenation and *card* returns the cardinality of a set. Notice that only the tuples whose value for the join attribute is non-disjunctive (i.e., is a singleton) can participate in the result: for the other ones, one cannot be certain at all that they match a tuple from the other relation. Indeed, for a tuple t_1 whose join attribute value $t_1.A$ is disjunctive, it is always completely possible that a given tuple t_2 does not match, whatever the value of $t_2.B$. The proof is omitted for space reasons, but the example hereafter illustrates the way it works.

Example 4. Consider relations *Person* and *Lab* from Table 4 and the query:

$$join(Person, Lab, Pcity = Lcity)$$

which looks for the pairs (p, l) such that person p (somewhat certainly) lives in a city where a research center l is located. Its compact result is represented in Table 5. Let us show that this result is correct using the proof method described in Subsection 3.2. The only pair of identifiers present in every completely possible world is (12, 21). The most possible world where (12, 21) is not present has the possibility degree $min(1, max(1-\beta, 1-\mu)) = max(1-\beta, 1-\mu) = 1-min(\beta, \mu)$. Hence, the certainty degree attached to (12, 21) in the result is $min(\beta,\, \mu)$. The most possible world where (12, 21) has a *Pcity* value different from "Rennes" has the possibility degree $min(1 - \beta,\, 1) = 1 - \beta$. Hence the certainty degree attached to the *Pcity* value "Rennes" in the tuple identified by (12, 21) in the result is β. The most possible world where (12, 21) has a *Lcity* value different from "Rennes" has the possibility degree $min(1,\, 1 - \mu) = 1 - \mu$. Hence the certainty degree attached to the *Lid* value "Rennes" in the tuple identified by

Table 4. Relations *Person* (top) and *Lab* (bottom)

#Pid	*Pname*	*Pcity*	*N*
11	John	(Rennes ∨ Brest, α)	1
12	Mary	(Rennes, β)	1

#Lid	*Lname*	*Lcity*	*N*
21	INRIA	(Rennes, μ)	1
22	IFREMER	(Brest, ρ)	1
23	CNRS	(Rennes ∨ Brest, ξ)	1

Table 5. Result of the compact join query

#Pid	*Pname*	*Pcity*	*N*	*#Lid*	*Lname*	*Lcity*	*N*
12	Mary	(Rennes, β)	1	21	INRIA	(Rennes, μ)	$min(\beta, \mu)$

(12, 21) in the result is μ. Notice that in case of an equi-join — as it is the case here — both columns *Pid* and *Lid* could be merged into a single column in the resulting table. John cannot take part to the result since one cannot be certain of what is the *Lab* value associated with him. ◇

About the semi-join. An interesting remark about the preceding example is that, even though the result of the join does not contain any tuple involving John, this individual belongs to the result of the semi-join which looks for the persons who (somewhat certainly) live in a city where a research center is located: $Person \ltimes Lab$. This means that the usual equivalence between a semi-join and a join followed by a projection: $r_1 \ltimes r_2 \equiv (r_1 \bowtie r_2)[X]$ where X denotes the attributes of r_1, is not valid anymore in the context of the certainty-based model. However, the semi-join can be defined in a sound way in this framework. Informally, for a tuple t_1 from r_1 to be in the result of the semi-join $r_1 \ltimes r_2$, each value a_i from $s(t_1.A)$ has to be present as a singleton B-value in a tuple from r_2. If an a_i from $s(t_1.A)$ joins with several tuples from r_2, the maximum of the corresponding certainty degrees must be used for computing the certainty degree attached to t_1 in the result. The formal definition is as follows.

$$\text{semijoin}(r_1, r_2, A\ \theta\ B) = \{min(\mu, \alpha, min_{i=1..n}\ \lambda_i)/t_1) \mid \exists \mu/t_1 \in r_1 \text{ s.t. } s(t_1.A) = \{a_1, \ldots, a_n\} \wedge c(t_1.A) = \alpha \wedge \forall a_i \in s(t_1.A), (\exists t_{2j} \in r_2 \text{ s.t. } s(t_{2j}.B) = \{a_i\}) \wedge \lambda_i = max_{j \text{ s.t. } \delta_j/t_{2j} \in r_2 \wedge s(t_{2j}.B)=\{a_i\} \wedge c(t_{2j}.B)=\beta_j} min(\delta_j, \beta_j).$$

Using this definition and the relations from Table 4, the result of the semi-join expressed above is given in Table 6. Note that the tuple $\#Lid = 23$ in Table 4 has no influence on the result, since taking into account tuple $\#Pid = 11$, John may live in Rennes while the CNRS lab may be in Brest (or the converse).

Table 6. Result of the compact semi-join query

#Pid	*Pname*	*Pcity*	*N*
11	John	(Rennes $\vee$ Brest, α)	$min(\alpha, \mu, \rho)$
12	Mary	(Rennes, β)	$min(\beta, \mu)$

The key to the fact that join (and semi-join) can be easily handled in this model lies in the property that a tuple involving disjunctive values can produce at most one tuple in the result (due to the semantics of certainty). This is not the case when a probabilistic or a full possibilistic [4] model is used and it is then necessary to deal with dependencies between tuples in the result (which is a complex and expensive issue). Let us illustrate this point with a simple example where the degrees are either probability or possibility degrees. Let us consider the relations $r(A, B)$ and $s(B, C)$ with the extensions:

$$r = \{\langle\{\alpha/a_1, \beta/a_2, \gamma/a_3\}, b\rangle\} \qquad s = \{\langle b, c_1\rangle, \langle b, c_2\rangle\}$$

where incompleteness is only due to the fact that the actual value of A in the tuple of r is either a_1, or a_2, or a_3. The natural join of r and s leads to a relation $t(A, B, C)$ involving two tuples, but it is mandatory to guarantee that only three possible worlds can be drawn from t (and not 3^2), since attribute A should take the same value in each of the two tuples, in order that property P1 holds. Now, let us perform the natural join of the following relations:

$$r = \{\langle a, \{\alpha/b_1, \beta/b_2, \gamma/b_3\}\rangle\} \qquad s = \{\langle b_1, c_1\rangle, \langle b_3, \{\eta/c_2, \delta/c_3\}\rangle\}.$$

Here, the resulting relation is either empty, or made of a single tuple among three possible: $\langle a, b_1, c_1\rangle$, $\langle a, b_3, c_2\rangle$ and $\langle a, b_3, c_3\rangle$. It is then necessary to express that these four situations are exclusive. On the other hand, in the certainty-based model we propose, the processing of a join is very similar to what it is in a regular database context. It follows from the previous definition that its data complexity is in $\theta(|r| \times |s|)$ — as in the usual case when no indexes are available — and no dependencies between tuples have to be dealt with.

4.3 Union

Union is defined as usual (and has the same data complexity), except that:

- one must keep the duplicates when they correspond to tuples that involve uncertain values, so as to be consistent with a world-based processing (cf. property P2);
- the relations involved must be independent: they must not result from two selections on the same relation, for instance since the usual equivalence select(r, ϕ_1 or ϕ_2) $\equiv$ select(r, ϕ_1) $\cup$ select(r, ϕ_2) does not hold due to the duplicate removal policy.

As to tuples that only involve certain values, one can remove the duplicates as in the classical case.

Example 5. Consider the relations from Table 7. The result of their union is shown in Table 8. Note that the second tuple in Table 8 is redundant w.r.t. the fourth one, but is kept to be consistent with a world-based interpretation. ◇

Table 7. Relations r_1 (top) and r_2 (bottom)

Job	*City*	*N*
(Engineer, 0.6)	(Rennes, 0.7)	1
(Engineer ∨ Manager, 0.7)	(Rennes, 0.8)	0.7
Manager	Brest	1

Job	*City*	*N*
(Engineer ∨ Manager, 0.9)	Rennes	0.9
Manager	Brest	1
(Engineer, 0.8)	(Rennes, 0.4)	0.4

Table 8. Result of the union $r_1 \cup r_2$

Job	*City*	*N*
(Engineer, 0.6)	(Rennes, 0.7)	1
(Engineer ∨ Manager, 0.7)	(Rennes, 0.8)	0.7
Manager	Brest	1
(Engineer ∨ Manager, 0.9)	Rennes	0.9
(Engineer, 0.8)	(Rennes, 0.4)	0.4

5 About a Possibilistic Logic Encoding of the Model

Possibilistic logic [10] provides a computational tool (with a complexity similar to classical logic) for inferring from certainty-qualified formulas by means of the cut rule $(p \vee q, \alpha); (\neg q \vee r, \beta) \vdash (p \vee r, min(\alpha, \beta))$. It can be applied to the computation of the answers to a query to an uncertain database. Due to space limitation, we only survey and illustrate the main issues. Tuples are translated into a possibilistic logic base, applying the following principles:

- keys become variables;
- attributes become predicates;
- tuples become instantiated formulas.

A query such as 'find the x's s. t. condition Q is true', i.e., $\exists x\, Q(x)$? is processed by refutation, adding the formulas corresponding to $\neg Q(x) \vee answer(x)$ to the base, using a small trick due to [12] (see [8]). Let us first take a simple example.

Example 6. A tuple such as $t = \langle 17, John, (Paris, 0.8), (Engineer, 0.7)\rangle$ translates into the possibilistic logic base:

$$K1 = \{(name(17,\ John)),\ (city(17,\ Paris),\ 0.8),\ (job(17,\ Engineer),\ 0.7)\}.$$

Query $q = select(emp, city = \ 'Paris' \ \textbf{and} \ \ job = \ 'Engineer')$ translates into:

$$q = \{((\neg city(x,\ Paris)) \vee (\neg job(x,\ Engineer)) \vee answer(x),\ 1)\}.$$

From $K1 \cup q$, applying resolution and unification, one gets $(answer(17),\ 0.7)$. $\diamond$

The previous example does not require the use of formulas expressing that the values of the attributes are necessarily in the attribute domains. Here is an example, with five tuples, where such formulas are useful:

Example 7. Let us consider the tuples:

$$t_R1 = \langle John, (Brest \vee Vannes, \alpha)\rangle$$
$$t_R2 = \langle Mary, (Rennes, \beta)\rangle$$
$$t_S1 = \langle Brest, (Britanny, 1)\rangle$$
$$t_S2 = \langle Vannes, (Britanny, 1)\rangle$$
$$t_S3 = \langle Rennes, (Britanny, 1)\rangle.$$

Assuming that there are only three cities (Brest, Rennes, Vannes) where people in the database may live, it translates into:

$$\begin{aligned} K2 = \{&(city(John, Brest) \vee city(John, Vannes), \alpha),\\ &(city(John, Brest) \vee city(John, Rennes) \vee city(John, Vannes), 1),\\ &(city(Mary, Rennes), \beta),\\ &(city(Mary, Brest) \vee city(Mary, Rennes) \vee city(Mary, Vannes), 1),\\ &(Britanny(Brest), 1),\ (Britanny(Vannes), 1),\ (Britanny(Rennes), 1)\}. \end{aligned}$$

Considering the request $\exists x\ city(x, y) \wedge Brit.(y)?$, we add

$$q = \{(\neg city(x, y) \vee \neg Brit.(y) \vee answer(x), 1)\}.$$

From $K2 \cup q$, one can deduce $(answer(John), 1)$ and $(answer(Mary), 1)$. If the query is slightly modified into $\exists(x, y)\ city(x, y) \wedge Britanny(y)$?, which translates into $q = \{(\neg city(x, y) \vee \neg Britanny(y) \vee answer(x, y), 1)\}$, we then obtain:

$$(answer(John, Brest) \vee answer(John, Vannes) \vee answer(John, Rennes), 1),$$

and the same for Mary. Notice that we also get $(answer(Mary, Rennes),\ \beta)$, $(answer(John, Brest) \vee answer(John, Vannes), \alpha)$, which are not subsumed by the previous formulas. $\diamond$

Example 8 shows that the approach can provide conditional answers as well.

Example 8. Let be the three tuples:

$$t1 = \langle John, veterinary, (Paris \vee Rennes, \alpha)\rangle;$$
$$t2 = \langle Peter, taxidermist, (Paris, \beta)\rangle;$$
$$t3 = \langle Mary, taxidermist, (Paris \vee Rennes, \gamma)\rangle.$$

It translates into:

$$\begin{aligned} K3 = \{&(city(John, Paris) \vee city(John, Rennes), \alpha),\\ &(job(John, veterinary), 1),\\ &(city(Peter, Paris), \beta), (job(Peter, taxidermist), 1),\\ &(city(Mary, Paris) \vee city(Mary, Rennes), \gamma),\\ &(job(Mary, taxidermist), 1)\}. \end{aligned}$$

Consider the query 'Find the persons who are veterinaries and live in a city where at least a taxidermist lives and the corresponding taxidermists':

$$\begin{aligned} q = \{&(\neg city(x, z) \vee \neg city(y, z) \vee \neg job(x, veterinary) \vee \neg job(y, taxidermist)\\ &\vee answer(x, y), 1)\}. \end{aligned}$$

One can, *for example*, deduce from $K3 \cup q$:

$$(city(John, Rennes) \vee answer(John, Peter), min(\alpha, \beta))$$
$$(city(John, Rennes) \vee city(Mary, Rennes) \vee answer(John, Mary), min(\alpha, \gamma))$$

which expresses that (John, Peter) (resp. (John, Mary)) is an answer with certainty $min(\alpha, \beta)$ (resp. $min(\alpha, \gamma)$) *provided* that John does not live in Rennes (resp. both John and Mary do not live in Rennes – hence they live in Paris). $\diamond$

The possibilistic logic modeling provides an alternative way to prove that the compact definitions of the operators are correct. It is also an expressive setting.

6 Conclusion

In this paper, we have presented a model for incomplete databases based on the notion of certainty levels. The idea is to associate every candidate value (or disjunction of such values) representing an ill-known piece of data with a degree expressing the extent to which the candidate value (or disjunction) is certain. We have extended the operations of selection, join and union in this context and shown that the model constitutes a strong representation system for this set of operators. A very interesting result is that the data complexity of these operations in the context of the certainty-based model is the same as in the classical database case, which makes the approach highly scalable. We have also briefly discussed the way the model can be encoded using possibilistic logic, which makes it possible to benefit from powerful automated reasoning tools, thus leading to a question-answering approach potentially more expressive than the relational framework.

Immediate perspectives for future work concern: i) the extension of the model to the entire relational algebra, ii) the extensive definition of a logical counterpart of the relational database model described here.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley, Reading (1995)
2. Antova, L., Jansen, T., Koch, C., Olteanu, D.: Fast and simple processing of uncertain data. In: Proc. of ICDE 2008, pp. 983–992 (2008)
3. Benjelloun, O., Das Sarma, A., Halevy, A., Widom, J.: ULDBs: Databases with uncertainty and lineage. In: Proc. VLDB 2006, pp. 953–964 (2006)
4. Bosc, P., Pivert, O.: About projection-selection-join queries addressed to possibilistic relational databases. IEEE Trans. on Fuzzy Systems 13, 124–139 (2005)
5. Codd, E.F.: Extending the relational database model to capture more meaning. ACM Transactions on Database Systems 4(4), 397–434 (1979)
6. Dalvi, N., Suciu, D.: Management of probabilistic data: Foundations and challenges. In: Proc. of PODS 2007, pp. 1–12 (2007)
7. Das Sarma, A., Benjelloun, O., Halevy, A., Widom, J.: Working models for uncertain data. In: Proc. of 22nd Int. Conf. on Data Engineering, ICDE (2006)
8. Dubois, D., Prade, H.: Necessity measures and the resolution principle. IEEE Trans. Syst., Man and Cyber. 17, 474–478 (1987)
9. Dubois, D., Prade, H.: Possibility Theory. Plenum, New York (1988)
10. Dubois, D., Lang, J., Prade, H.: Automated reasoning using possibilistic logic: Semantics, belief revision, and variable certainty weights. IEEE Transactions on Knowledge and Data Engineering 6(1), 64–71 (1994)
11. Eiter, T., Lukasiewicz, T., Walter, M.: Extension of the relational algebra to probabilistic complex values. In: Schewe, K.-D., Thalheim, B. (eds.) FoIKS 2000. LNCS, vol. 1762, pp. 94–115. Springer, Heidelberg (2000)
12. Green, C.: Theorem-proving by resolution as a basis for question-answering systems. In: Michie, D., Meltzer, B. (eds.) Machine Intellig., vol. 4, pp. 183–205. Edinb. Uni. Pr. (1969)
13. Green, T.J., Tannen, V.: Models for incomplete and probabilistic information. IEEE Data Eng. Bull. 29, 17–24 (2006)
14. Imielinski, T., Lipski, W.: Incomplete information in relational databases. J. of the ACM 31, 761–791 (1984)
15. Lakshmanan, L., Leone, N., Ross, R., Subrahmanian, V.S.: Probview: A flexible probabilistic system. ACM Trans. Database Syst. 22(3), 419–469 (1997)
16. Lipski, W.: Semantic issues connected with incomplete information databases. ACM Transactions on Database Systems 4(3), 262–296 (1979)
17. Prade, H., Testemale, C.: Generalizing database relational algebra for the treatment of incomplete/uncertain information and vague queries. Information Sciences 34, 115–143 (1984)
18. Ré, C., Dalvi, N., Suciu, D.: Query evaluation on probabilistic databases. IEEE Data Eng. Bull. 29, 25–31 (2006)
19. Zadeh, L.A.: Fuzzy sets as a basis for a theory of possibility. Fuzzy Sets and Systems 1, 3–28 (1978)

A Proposal for Making Argumentation Computationally Capable of Handling Large Repositories of Uncertain Data

Marcela Capobianco and Guillermo R. Simari

Artificial Intelligence Research and Development Laboratory
Department of Computer Science and Engineering
Universidad Nacional del Sur – Av. Alem 1253, (8000) Bahía Blanca ARGENTINA
{mc,grs}@cs.uns.edu.ar

Abstract. Data intensive applications with the capability of handling uncertain, imprecise, and inconsistent information are in constant demand. Efficient computational systems that can perform complicated inferences, obtain the appropriate conclusions, and explain the results are increasingly being required to act upon large databases. Argumentation systems could be used in the construction of interactive systems that are able to reason with large databases and/or different data sources. Notwithstanding, there are two important issues that need to be resolved in order to use argumentation in this kind of practical applications: adding the ability to deal with explicit uncertainty, and improving the computational complexity of argumentation, which so far has been an obstacle for its integration into interactive systems acting on large databases. In this paper we propose an argumentation-based system that has been engineered to address these issues.

1 Introduction

Research in argumentation has provided important results while developing tools for common sense reasoning. As a result, argumentation systems have substantially evolved in the past few years, putting forward a number of argument-based applications in diverse areas where knowledge representation issues play a major role. Clustering algorithms [16], intelligent web search [10], recommender systems [9], agent systems [5], knowledge engineering systems [7], argumentation-based negotiation [19], and natural language assessment [10] are the outcome of one of these lines of research.

Data intensive applications are in constant demand and there is need of computing environments with better intelligent capabilities than those present in today's Database Management Systems (DBMS). Recently, there has been progress in developing efficient techniques to store and retrieve data, and many satisfactory solutions have been found for the associated problems. However, the problem of how to *understand* and *interpret* a large amount of information remains

L. Godo and A. Pugliese (Eds.): SUM 2009, LNAI 5785, pp. 95–110, 2009.

open, particularly when this information is uncertain, imprecise, and/or inconsistent. To do this we need formalisms that can perform complicate inferences, obtain the appropriate conclusions, and explain the results.

Argumentation systems appear as an interesting choice that could result in interactive systems able to reason with large databases and/or different data sources. Nevertheless, there are two important issues that need to be addressed to use argumentation in these kind of practical applications. The first concerns the quality of the information expected by argumentation systems: most of them are unable to deal with explicit uncertainty which is a vital capability in modern applications. The second problem is the computational complexity of argumentation, that so far has been an obstacle for its integration into interactive systems.

Here, we propose an argumentation-based system that addresses these problems. First, it incorporates the treatment of possibilistic uncertainty into the framework following the approach introduced in [11,3]. Second, to solve the problem of performance in argumentation, we have equipped this system with a pre-compiled knowledge component that allows significant speed-ups in the inference process. This development was inspired in *Truth Maintenance Systems* (TMS) [12] that use pre-compiled knowledge to improve the performance of problem solvers. We have also provided the system with the ability to seamlessly incorporate uncertain and/or contradictory information into its knowledge base, using a modular upgrading and revision procedure.

This paper is organized as follows. First, we present the formal definition of our argumentation framework showing its fundamental properties. Next, we propose an architectural software pattern useful for applications adopting our reasoning system. Section 4 discusses how to optimize the inference engine with pre-compiled knowledge and presents the corresponding algorithms. Finally, we state the conclusions of our work.

2 The OP-DeLP Programming Language: Fundamentals

Possibilistic Defeasible Logic Programming (P-DeLP) [1,2] is an important extension of DeLP in which the elements of the language have the form (φ, α), where φ is a DeLP clause or fact. Below, we will introduce the elements of the language necessary in this presentation. Observation based P-DeLP (OP-DeLP) is an optimization of P-DeLP that allows the computation of warranted arguments in a more efficient way, by means of a pre-compiled knowledge component. It also permits a seamless incorporation of new perceived facts into the program codifying the knowledge base of the system. Therefore the resulting system can be used to implement practical applications with performance requirements. The idea of extending the applicability of DeLP in a dynamic setting, incorporating perception and pre-compiled knowledge, was originally conceived in [6]. Thus the OP-DeLP system incorporates elements from two different variants of the DeLP system, O-DeLP [6] and P-DeLP [11]. In what follows we present the formal definition of the resulting system.

2.1 Language

To formally define the language of OP-DeLP we start introducing the concept of signature, that summarizes the elements that change in accordance with the program under consideration.

Definition 1. [Signature] *A* signature Σ *is a tuple* $\langle \mathcal{V}, \text{Pred}, \text{Func} \rangle$, *where* $\mathcal{V}$ *is a countable set of variables,* Pred *is a finite set of predicates, and* Func *is a finite set of functions, such that* $\mathcal{V} \cap (\text{Pred} \cup \text{Func}) = \emptyset$.

As in PROLOG standard notation, variables are denoted with identifiers starting with uppercase letters while functions and predicates start with lowercase letters. Every signature has an associated arity function that assigns a natural number to each function and predicate. As usual, constants are functions with 0 arity and propositions are predicates with a null arity. Next, we define the *alphabet* of OP-DeLP programs and use this to define *terms*, *atoms*, and *literals*:

Definition 2. [Alphabet] *The* alphabet *generated from a given signature* Σ *is composed by the members of* Σ, *the symbol "*$\sim$*" denoting strong negation [15] and the symbols "(", ")", "." and ",".*

Definition 3. [Term] *Let* $\Sigma = \langle \mathcal{V}, \text{Pred}, \text{Func} \rangle$ *be a signature. A* term *of* Σ *is inductively defined as follows: (1) every variable* $V \in \mathcal{V}$ *is a term, (2) every constant* $c \in$ Func *is a term, (3) if* $f \in$ Func, $\text{arity}(f) = n$ *and* $t_1, \ldots, t_n$ *are terms then* $f(t_1, \ldots, t_n)$ *is also a term.*

Definition 4. [Atom] *Let* $\Sigma = \langle \mathcal{V}, \text{Pred}, \text{Func} \rangle$ *be a signature,* $t_1, \ldots, t_n$ *terms of* Σ *and* $p \in$ Pred *such that* $\text{arity}(p) = n$ *then* $p(t_1, \ldots, t_n)$ *is an* atom *of* Σ.

Definition 5. [Literal - Weighted Literal] *Let* Σ *be a signature, then every atom* A *of* Σ *is a positive literal, while every negated atom* $\sim A$ *is a negative literal. A* literal *of* Σ *is a positive literal or a negative literal. A* certainty weighted literal, *or simply a weighted literal, is a pair* (L, α) *where* L *is a literal and* $\alpha \in [0, 1]$ *expresses a lower bound for the certainty of* φ *in terms of a necessity measure.*

OP-DeLP programs are composed by a set of *observations* and a set of *defeasible rules*. Observations are weighted literals and thus have an associated certainty degree. In real world applications, observations model perceived facts. Hence the associated certainty degree may be calculated based on different factors depending on the act of perception itself, such as plausibility of the source, trust on the perception mechanism, etc. Defeasible rules provide a way of performing tentative reasoning as in other argumentation formalisms.

Definition 6. *A* defeasible rule *has the form* $(L_0 \prec L_1, L_2, \ldots, L_k, \alpha)$ *where* L_0 *is a literal,* $L_1, L_2, \ldots, L_k$ *is a non-empty finite set of literals, and* $\alpha \in [0, 1]$ *expresses a lower bound for the certainty of the rule in terms of a necessity measure.*

Ψ	Δ
(`virus(b)`, 0.7)	(`move_inbox(X)` $\prec$ ~`filters(X)`, 0.6)
(`local(b)`, 1)	(~`move_inbox(X)` $\prec$ `move_junk(X)`, 0.8)
(`local(d)`, 1)	(~`move_inbox(X)` $\prec$ `filters(X)`, 0.7)
(~`filters(b)`, 0.9)	(`move_junk(X)` $\prec$ `spam(X)`, 1)
(~`filters(c)`, 0.9)	(`move_junk(X)` $\prec$ `virus(X)`, 1)
(~`filters(d)`, 0.9)	(`spam(X)` $\prec$ `black_list(X)`, 0.7)
(`black_list(c)`, 0.75)	(~`spam(X)` $\prec$ `contacts(X)`, 0.6)
(`black_list(d)`, 0.75)	(~`spam(X)` $\prec$ `local(X)`, 0.7)
(`contacts(d)`, 1)	

Fig. 1. An OP-DeLP program for email filtering

Intuitively a defeasible rule $L_0 \prec L_1, L_2, \ldots, L_k$ can be read as *"$L_1, L_2, \ldots, L_k$ provide tentative reasons to believe in L_0"* [20]. In OP-DeLP these rules also have a certainty degree, that quantifies how strong is the connection between the premises and the conclusion. A defeasible rule with a certainty degree 1 models a strong rule.

A set of weighted literals Γ will be deemed as contradictory, denoted as $\Gamma \vdash\!\sim\!\bot$, iff $\Gamma \vdash\!\sim\!(l, \alpha)$ and $\Gamma \vdash\!\sim\!(\neg l, \beta)$ with α and $\beta > 0$.

Definition 7 (OP-DeLP Program). *An* OP-DeLP program $\mathcal{P}$ *is a pair* $\langle \Psi, \Delta \rangle$*, where* Ψ *is a* non contradictory *finite set of observations and* Δ *is a finite set of defeasible rules.*

Example 1. Fig.1 shows a program for basic email filtering. Observations describe different characteristics of email messages. Thus, `virus(X)` stands for "message `X` has a virus"; `local(X)` indicates that "message `X` is from the local host"; `filters(X)` specifies that "message `X` should be filtered" redirecting it to a particular folder; `black_list(X)` indicates that "message `X` is considered dangerous" because of the server it is coming from; and `contacts(X)` indicates that "the sender of message `X` is in the contact list of the user".

The first rule expresses that if the email does not match with any user-defined filter then it usually should be moved to the "inbox" folder. The second rule indicates that unfiltered messages in the "junk" folder usually should not be moved to the inbox. According to the third rule, messages to be filtered should not be moved to the inbox. The following two rules establish that a message should be moved to the "junk" folder if it is marked as spam or it contains viruses. Finally there are three rules for spam classification: a message is usually labeled as spam if it comes from a server that is in the blacklist. Nevertheless, even if an email comes from a server in the blacklist it is not labeled as spam when the sender is in the contact list of the user. Besides, a message from the local host is usually not classified as spam.

The P-DeLP language [11], which presented the novel idea of mixing argumentation and possibilistic logic, is based on Possibilistic Gödel Logic or PGL [3],

which is able to model both uncertainty and fuzziness and allows for a partial matching mechanism between fuzzy propositional variables. In OP-DeLP, for simplicity reasons, we will avoid fuzzy propositions, and hence it will be based on the necessity-valued classical Possibilistic logic [13]. As a consequence, possibilistic models are defined by possibility distributions on the set of classical interpretations, and the proof method for our formulas, written $\vdash$, is defined by derivation based on the following instance of the generalized modus ponens rule (GMP): $(L_0 \prec L_1 \wedge \cdots \wedge L_k, \gamma), (L_1, \beta_1), \ldots, (L_k, \beta_k) \vdash (L_0, \min(\gamma, \beta_1, \ldots, \beta_k))$, which is a particular instance of the well-known possibilistic resolution rule, and which provides the *non-fuzzy* fragment of OP-DeLP with a complete calculus for determining the maximum degree of possibilistic entailment for weighted literals. Literals in the set of observations Ψ are the basis case of the derivation sequence, for every literal Q in Ψ with a certainty degree α it holds that (Q, α) can be derived from $\mathcal{P} = (\Psi, \Delta)$.

2.2 Inference Engine

Basically, an OP-DeLP program is a set of weighted literals and rules. In this set we can distinguish certain from uncertain information. A clause (γ, α) will be deemed as *certain* if $\alpha = 1$, otherwise it will be *uncertain*. Given an OP-DeLP program $\mathcal{P}$, a query posed to $\mathcal{P}$ corresponds to a ground literal Q which must be supported by an *argument* [20,14].

Definition 8. [Argument]–[Subargument] *Let $\mathcal{P} = \langle \Psi, \Delta \rangle$ be a program, $\mathcal{A} \subseteq \Delta$ is an* argument *for a goal Q with necessity degree $\alpha > 0$, denoted as $\langle \mathcal{A}, Q, \alpha \rangle$, iff:* (1) $\Psi \cup \mathcal{A} \vdash (Q, \alpha)$, (2) $\Psi \cup \mathcal{A}$ *is non contradictory, and* (3) *there is no $\mathcal{A}_1 \subset \mathcal{A}$ such that $\Psi \cup \mathcal{A}_1 \vdash (Q, \beta)$, $\beta > 0$. An argument $\langle \mathcal{A}, Q, \alpha \rangle$ is a* subargument *of $\langle \mathcal{B}, R, \beta \rangle$ iff $\mathcal{A} \subseteq B$.*

Note that in addition to provide a proof to support a ground literal an argument must also be non-contradictory and minimal. The non-contradictory requirement is easy to understand, since it avoids self-defeating arguments [8]. To see why arguments should be minimal consider that adding an unnecessary rule into a set that was already enough to obtain the conclusion Q would only weaken it, adding more opportunities for conflicts with other arguments (this will be clear when the notion of conflict between arguments is formally defined.) Regarding the notion of sub-argumentation, an interesting property can be mentioned.

Proposition 1. *Consider an OP-DeLP program $\mathcal{P}$, and an argument $\langle \mathcal{A}, Q, \alpha \rangle$ based on $\mathcal{P}$. Let $\mathsf{SubArgs}(\langle \mathcal{A}, Q, \alpha \rangle)$ be the set of subarguments of $\langle \mathcal{A}, Q, \alpha \rangle$, then for every $\langle \mathcal{B}, R, \beta \rangle$ in $\mathsf{SubArgs}(\langle \mathcal{A}, Q, \alpha \rangle)$ it holds that $\beta \geq \alpha$.*

Proof. Since $\mathcal{B} \subseteq \mathcal{A}$ then two cases may arise: (1) $\mathcal{B} = \mathcal{A}$: in this case $\alpha = \beta$; (2) $\mathcal{B} \subset \mathcal{A}$: this implies that the derivation of R is part of the derivation of Q. That is, R could be considered as a premise used in the derivation of Q. Since arguments are minimal every rule in the derivation is used to obtain R. Since the GMP rule takes the minimum over the certainty degrees of the rule used in

the derivation and the weighted literals in the body of that rule, it is clear that every subsequent literal derived from R should receive a weight that is lower or equal to β. In particular, $\beta \geq \alpha$.

As in most argumentation frameworks, arguments in O-DeLP can attack each other. This situation is captured by the notion of *counterargument*. Defeat among arguments is defined combining the counterargument relation and a preference criterion "$\preceq$".

Definition 9. [Counter-argument][14] *An argument* $\langle \mathcal{A}_1, Q_1, \alpha \rangle$ counter-argues *an argument* $\langle \mathcal{A}_2, Q_2, \beta \rangle$ *at a literal* Q *if and only if there is a sub-argument* $\langle \mathcal{A}, Q, \gamma \rangle$ *of* $\langle \mathcal{A}_2, Q_2, \beta \rangle$*, (called* disagreement subargument*), such that* Q_1 *and* Q *are complementary literals.*

In order to define defeat among arguments we need a *preference criterion* on conflicting arguments. This criterion is defined on the basis of the necessity measures associated with arguments.

Definition 10. [Preference criterion $\succeq$][11] *Let* $\langle \mathcal{A}_1, Q_1, \alpha_1 \rangle$ *be a counterargument for* $\langle \mathcal{A}_2, Q_2, \alpha_2 \rangle$*. We will say that* $\langle \mathcal{A}_1, Q_1, \alpha_1 \rangle$ *is* preferred *over* $\langle \mathcal{A}_2, Q_2, \alpha_2 \rangle$ *(denoted* $\langle \mathcal{A}_1, Q_1, \alpha_1 \rangle \succeq \langle \mathcal{A}_2, Q_2, \alpha_2 \rangle$*) iff* $\alpha_1 \geq \alpha_2$*. If it is the case that* $\alpha_1 > \alpha_2$*, then we will say that* $\langle \mathcal{A}_1, Q_1, \alpha_1 \rangle$ *is* strictly preferred *over* $\langle \mathcal{A}_2, Q_2, \alpha_2 \rangle$*, denoted* $\langle \mathcal{A}_2, Q_2, \alpha_2 \rangle \succ \langle \mathcal{A}_1, Q_1, \alpha_1 \rangle$*. Otherwise, if* $\alpha_1 = \alpha_2$ *we will say that both arguments are* equi-preferred*, denoted* $\langle \mathcal{A}_2, Q_2, \alpha_2 \rangle \approx \langle \mathcal{A}_1, Q_1, \alpha_1 \rangle$*.*

Definition 11. [Defeat][11] *Let* $\langle \mathcal{A}_1, Q_1, \alpha_1 \rangle$ *and* $\langle \mathcal{A}_2, Q_2, \alpha_2 \rangle$ *be two arguments built from a program* $\mathcal{P}$*. Then* $\langle \mathcal{A}_1, Q_1, \alpha_1 \rangle$ defeats $\langle \mathcal{A}_2, Q_2, \alpha_2 \rangle$ *(or equivalently* $\langle \mathcal{A}_1, Q_1, \alpha_1 \rangle$ *is a* defeater *for* $\langle \mathcal{A}_2, Q_2, \alpha_2 \rangle$*) iff (1) Argument* $\langle \mathcal{A}_1, Q_1, \alpha_1 \rangle$ *counterargues argument* $\langle \mathcal{A}_2, Q_2, \alpha_2 \rangle$ *with disagreement subargument* $\langle \mathcal{A}, Q, \alpha \rangle$*; and (2) Either it is true that* $\langle \mathcal{A}_1, Q_1, \alpha_1 \rangle \succ \langle \mathcal{A}, Q, \alpha \rangle$*, in which case* $\langle \mathcal{A}_1, Q_1, \alpha_1 \rangle$ *will be called a* proper defeater *for* $\langle \mathcal{A}_2, Q_2, \alpha_2 \rangle$*, or* $\langle \mathcal{A}_1, Q_1, \alpha_1 \rangle \approx \langle \mathcal{A}, Q, \alpha \rangle$*, in which case* $\langle \mathcal{A}_1, Q_1, \alpha_1 \rangle$ *will be called a* blocking defeater *for* $\langle \mathcal{A}_2, Q_2, \alpha_2 \rangle$*.*

As in most argumentation systems [8,18], OP-DeLP relies on an exhaustive dialectical analysis which allows to determine if a given argument is *ultimately* undefeated (or *warranted*) wrt a program $\mathcal{P}$. An *argumentation line* starting in an argument $\langle \mathcal{A}_0, Q_0, \alpha_0 \rangle$ is a sequence $[\langle \mathcal{A}_0, Q_0, \alpha_0 \rangle, \langle \mathcal{A}_1, Q_1, \alpha_1 \rangle, \ldots, \langle \mathcal{A}_n, Q_n, \alpha_n \rangle, \ldots]$ that can be thought of as an exchange of arguments between two parties, a *proponent* (evenly-indexed arguments) and an *opponent* (oddly-indexed arguments). In order to avoid *fallacious* reasoning, argumentation theory imposes additional constraints on such an argument exchange to be considered rationally acceptable wrt an OP-DeLP program $\mathcal{P}$, namely:

1. **Non-contradiction:** Given an argumentation line λ, the set of arguments of the proponent (resp. opponent) should be *non-contradictory* wrt $\mathcal{P}$. Non-contradiction for a set of arguments is defined as follows: a set $S = \bigcup_{i=1}^{n} \{\langle \mathcal{A}_i, Q_i, \alpha_i \rangle\}$ is *contradictory* wrt $\mathcal{P}$ iff $\Psi \cup \bigcup_{i=1}^{n} \mathcal{A}_i$ is contradictory.

2. **No circular argumentation:** No argument $\langle \mathcal{A}_j, Q_j, \alpha_j \rangle$ in λ is a sub-argument of an argument $\langle \mathcal{A}_i, Q_i, \alpha_i \rangle$ in λ, $i < j$.
3. **Progressive argumentation:** Every blocking defeater $\langle \mathcal{A}_i, Q_i, \alpha_i \rangle$ in λ is defeated by a proper defeater $\langle \mathcal{A}_{i+1}, Q_{i+1}, \alpha_{i+1} \rangle$ in λ.

An argumentation line satisfying the above restrictions is called *acceptable*, and can be proved to be finite. Given a program $\mathcal{P}$ and an argument $\langle \mathcal{A}_0, Q_0, \alpha_0 \rangle$, the set of all acceptable argumentation lines starting in $\langle \mathcal{A}_0, Q_0, \alpha_0 \rangle$ accounts for a whole dialectical analysis for $\langle \mathcal{A}_0, Q_0, \alpha_0 \rangle$ (i.e. all possible dialogs rooted in $\langle \mathcal{A}_0, Q_0, \alpha_0 \rangle$, formalized as a *dialectical tree*, denoted $\mathcal{T}_{\langle \mathcal{A}_0, Q_0, \alpha_0 \rangle}$. Nodes in a dialectical tree $\mathcal{T}_{\langle \mathcal{A}_0, Q_0, \alpha_0 \rangle}$ can be marked as *undefeated* and *defeated* nodes (U-nodes and D-nodes, resp.). A dialectical tree will be marked as an AND-OR tree: all leaves in $\mathcal{T}_{\langle \mathcal{A}_0, Q_0, \alpha_0 \rangle}$ will be marked U-nodes (as they have no defeaters), and every inner node is to be marked as *D-node* iff it has at least one U-node as a child, and as *U-node* otherwise. An argument $\langle \mathcal{A}_0, Q_0, \alpha_0 \rangle$ is ultimately accepted as valid (or *warranted*) iff the root of $\mathcal{T}_{\langle \mathcal{A}_0, Q_0, \alpha_0 \rangle}$ is labeled as *U-node*.

Definition 12. [Warrant][11] *Given a program $\mathcal{P}$, and a literal Q, Q is warranted wrt $\mathcal{P}$ iff there exists a warranted argument $\langle \mathcal{A}, Q, \alpha \rangle$ than can be built from $\mathcal{P}$.*

To answer a query for a given literal we should see if there exists a warranted argument supporting this literal. Nevertheless, in OP-DeLP there may be different arguments with different certainty degrees supporting a given query. This fact was not considered in [11], but we are clearly interested in finding the warranted argument with the highest certainty degree.

Definition 13. [Strongest Warrant] *Given a program $\mathcal{P}$, and a literal Q, we will say that α is the* strongest warrant degree *of Q iff (1) there exists a warranted argument $\langle \mathcal{A}, Q, \alpha \rangle$ than can be built from $\mathcal{P}$ and (2) no warranted argument $\langle \mathcal{B}, Q, \beta \rangle$ such that $\beta > \alpha$ can built from $\mathcal{P}$.*

Note that to find out the strongest warrant degree for a given literal Q we need to find the strongest warranted argument supporting it, that is, the warranted argument supporting Q with the higher certainty degree. Then, to find the strongest warrant degree for a literal Q we must first build the argument $\mathcal{A}$ that supports the query Q with the highest possible certainty degree and see if $\mathcal{A}$ is a warrant for Q. Otherwise we must find another argument $\mathcal{B}$ for Q with the highest certainty degree among the remaining ones, see if it is a warrant for Q, and so on, until a warranted argument is found or there are no more arguments supporting Q. This procedure is detailed in algorithm 1.

Example 2. Consider the program shown in Example 1 and let `move_inbox(d)` be a query wrt this program. The search for a warrant for `move_inbox(d)` will result in an argument $\langle \mathcal{A}, \texttt{move\_inbox(d)}, 0.6 \rangle$, with

$$\mathcal{A} = \{(\texttt{move\_inbox(d)} \prec \sim\texttt{filters(d)}, 0.6)\}$$

Algorithm 1. `StrongestWarrantDegree`

```
Input: P = ⟨Ψ, Δ⟩, Q
Output: warranted (true/false), α {if warranted is true}

FindArguments(Q,P,L)
{Finds the arguments supporting Q and returns them in L}
L ← Sort(L)
{Orders the arguments according to their certainty degree}
warranted ← false
While L is not empty and not warranted
   ⟨A, Q, α⟩ ← Pop(L)
   If Warrant(⟨A, Q, α⟩,P)
   {returns true if ⟨A, Q, α⟩ is warranted wrt P, false otherwise}
      then warranted ← true
```

allowing to conclude that message d should be moved to the folder Inbox, as it has no associated filter with a certainty degree of 0.6. However, there exists a defeater for $\langle \mathcal{A}, \texttt{move\_inbox(d)}, 0.6\rangle$, namely $\langle \mathcal{B}, \sim\texttt{move\_inbox(d)}, 0.7\rangle$, as there are reasons to believe that message d is spam:

$$\mathcal{B} = \{(\sim\texttt{move\_inbox(d)} \prec \texttt{move\_junk(d)}, 0.8)$$
$$(\texttt{move\_junk(d)} \prec \texttt{spam(d)}, 1), (\texttt{spam(d)} \prec \texttt{black\_list(d)}, 0.7)\}$$

Using the preference criterion, $\langle \mathcal{B}, \sim\texttt{move\_inbox(d)}, 0.7\rangle$ is a proper defeater for $\langle \mathcal{A}, \texttt{move\_inbox(d)}, 0.6\rangle$. However, two counterarguments can be found for $\langle \mathcal{B}, \sim\texttt{move\_inbox(d)}, 0.7\rangle$, since message d comes from the local host, and the sender is in the user's contacts list:

- $\langle \mathcal{C}, \sim\texttt{spam(d)}, 0.6\rangle$, where $\mathcal{C} = \{(\sim\texttt{spam(d)} \prec \texttt{contacts(d)}, 0.6)\}$.
- $\langle \mathcal{D}, \sim\texttt{spam(d)}, 0.9\rangle$, where $\mathcal{D} = \{(\sim\texttt{spam(d)} \prec \texttt{local(d)}, 0.9)\}$.

$\mathcal{B}$ defeats $\mathcal{C}$ but is defeated by $\mathcal{D}$. There are no more arguments to consider, and the resulting dialectical tree has only one argumentation line: $\mathcal{A}$ is defeated by $\mathcal{B}$ who is in turn defeated by $\mathcal{D}$. Hence, the marking procedure determines that the root node $\langle \mathcal{A}, \texttt{move\_inbox(d)}, 0.6\rangle$ is a `U-node` and the original query is warranted.

Having defined the inference mechanism, it is interesting to analyze its properties. Consistency of the set of deductions (warranted literals in OP-DeLP) is a fundamental property of any inference system. To show this property in OP-DeLP we demonstrate that the set of warranted arguments is conflict free, that is, there is no pair of warranted arguments (A, B) such that attack A attacks B (or the other way around). This is summed-up in the following lemma, but its proof is not included here for space reasons.

Lemma 1. *Let $\mathcal{P} = \langle \Psi, \Delta\rangle$ be an OP-DeLP program, and let Warr($\mathcal{P}$) be the set of arguments warranted from $\mathcal{P}$. For any pair of arguments $\langle \mathcal{A}, Q, \alpha\rangle$, $\langle \mathcal{B}, R, \beta\rangle$, such that $\langle \mathcal{A}, Q, \alpha\rangle \in$ Warr($\mathcal{P}$) and $\langle \mathcal{B}, R, \beta\rangle \in$ Warr($\mathcal{P}$) it holds that $\langle \mathcal{A}, Q, \alpha\rangle$ is not a counterargument for $\langle \mathcal{B}, R, \beta\rangle$.*

3 A Design Pattern for OP-DeLP Applications

In this section we present an architectural pattern that can be applied to design applications that use the OP-DeLP system. Such applications will be engineered for contexts where: (1) information is uncertain and heterogeneous, (2) handling of great volume of data flows is needed, and (3) data may be incomplete, vague or contradictory. In this scenario data will generally be obtained from multiple sources. Nowadays the availability of information trough the Internet has shifted the issue of information from quantitative stakes to qualitative ones [4]. For this reason, new information systems also need to provide assistance for judgind and examining the quality of the information they receive.

We have chosen to use a multi-source perspective into the characterization of data quality[4]. In this case the quality of data can be evaluated by comparison with the quality of other homologous data (i.e. data from different information sources which represent the same reality but may have contradictory values). The approaches usually adopted to reconcile heterogeneity between values of data are : (1) to prefer the values of the most reliable sources, (2) to mention the source ID for each value, or (3) to store quality meta-data with the data.

For our proposed architecture we have chosen to use the second approach. In multi-source databases, each attribute of a multiple source element has multiple values with the ID of their source and their associated quality expertise. Quality expertise is represented as meta-data associated with each value. We have simplified this model for an easy and practical integration with the OP-DeLP system. In our case, data sources are assigned a unique certainty degree. For simplicity sake, we assume that different sources have different values. All data from a given source will have the same certainty degree. This degree may be obtained weighting the plausibility of the data value, its accuracy, the credibility of its source and the freshness of the data.

OP-DeLP programs basically have a set of observations Ψ and a set of rules Δ. The set of rules is chosen by the knowledge engineer and remains fixed. The observation set may change according with new perceptions received from the multiple data sources. Nevertheless, inside the observation set we will distinguish a special kind of perceptions, those with certainty degree 1. Those perceptions are also codified by the knowledge engineer and cannot be modified in the future by the perception mechanism. To assure this, we assume that every data source has a certainty value γ such that $0 < \gamma < 1$.

Example 3. Consider the program in Example 2. In this case data stablishing a given message is from the local host cames from the same data source and can be given a certainty degree of 1. The same applies for `contacts(X)`. The algorithm that decides whether to filter a given message is another data source with a degree of 0.9, the filter that classifies a message as a virus is another data source with a degree of 0.7, and the algorithm that checks if the message came from some server in the blacklist is a different source that has a degree of 0.75. Note that we could have different virus filters with different associated certainty degrees if we wanted to build higher trust on this filter mechanism.

The scenario just described requires an updating criterion different to the one presented in [6], given that the situation regarding perceptions in OP-DeLP is much more complex. To solve this, we have devised Algorithm 2, that summarizes different situations in two conditions. The first one acts when the complement of the literal Q is already present in the set Ψ. Three different cases can be analyzed in this setting: (1) If both certainty degrees are equal it means that both Q and its complement proceed from the same data source. Then the only reason for the conflict is a change in the state of affairs, thus an update is needed and the new literal is added. (2) If $\alpha > \beta$ it means that the data sources are different, Thus we choose to add (Q, α) since it has the higher certainty degree. (3) If $\alpha < \beta$ we keep $(\overline{Q}, \beta)$. Note that (1) is an update operation [17] while (2) and (3) are revisions over Ψ. The difference between updating and revision is foundamental. Updating consists in bringing the knowledge base up to date when the world changes. Revision allows us to obtain new information about a static scenario [17].

The second condition in Algorithm 2 considers the case when Q was in Ψ with a different certainty degree. Then it chooses the weighted literal with the highest degree possible. Note that the observations initially codified that have a certainty degree of 1 cannot be deleted or modified by algorithm 2.

Algorithm 2. `UpdateObservationSet`

`Input:` $\mathcal{P} = \langle \Psi, \Delta \rangle$, (Q, α)
`Output:` $\mathcal{P} = \langle \Psi, \Delta \rangle$ {*With Ψ updated*}

`If there exists a weighted literal` $(\overline{Q}, \beta) \in \Psi$ `such that` $\beta \leq \alpha$ `Then`
  `delete(`$(\overline{Q}, \beta)$
  `add(`(Q, α)`)`
`If there exists a weighted literal` $(Q, \beta) \in \Psi$ `such that` $\alpha \leq \beta$ `Then`
  `delete(`(Q, β)
  `add(`(Q, α)`)`

Associating a certainty degree with each data source is an interesting choice for many applications, but it is not hard coded in the OP-DeLP system. The certainty degrees in the observation set can be assigned by means of any reasonable criteria, and this decision is ultimately in hands of the application developer, who can understand better the particular needs of a given context.

Finally, fig. 2 summarizes the main elements of the O-DeLP-based architecture. Knowledge is represented by an OP-DeLP program $\mathcal{P}$. Perceptions from multiple sources may result in changes in the set of observations in $\mathcal{P}$, handled by the updating mechanism defined in algorithm 2. To solve queries the OP-DeLP inference engine is used. This engine is assisted by the dialectical graph (Def. 15) to speed-up the argumentation process. The final answer to a given query Q will be *yes*, with the particular certainty degree of the warranted argument supporting Q, or *no* if the system could not find a warrant for Q from $\mathcal{P}$.

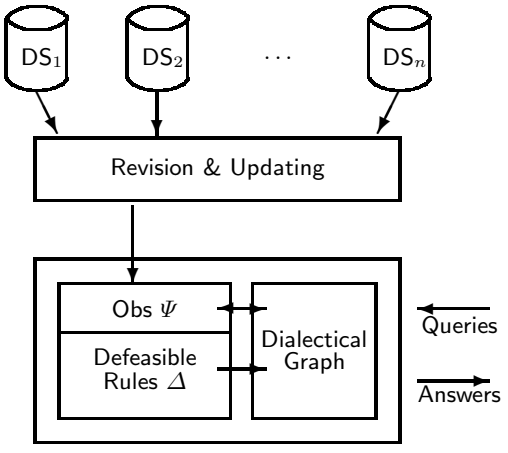


Fig. 2. Architecture for applications using OP-DeLP as underlying framework

4 Dialectical Graphs: An Optimization Based on Pre-compiled Knowledge

To obtain faster query processing in the OP-DeLP system we integrate pre-compiled knowledge to avoid the construction of arguments which were already computed before. The approach follows the proposal presented in [6] where the pre-compiled knowledge component is required to: (1) minimize the number of stored arguments in the pre-compiled base of arguments (for instance, using one structure to represent the set of arguments that use the same defeasible rules); and (2) maintain independence from the observations that may change with new perceptions, to avoid modifying also the pre-compiled knowledge when new observations are incorporated.

Considering these requirements, we define a database structure called *dialectical graph*, which will keep a record of all possible *arguments* in an OP-DeLP program $\mathcal{P}$ (by means of a special structure named potential argument) as well as the counterargument relation among them. Potential arguments, originally defined in [6] contain non-grounded defeasible rules, depending thus only on the set of rules Δ in $\mathcal{P}$ and are independant from the set of observations Ψ.

Potential arguments have been devised to sum-up arguments that are obtained using *different* instances of the *same* defeasible rules. Recording every generated argument could result in storing many arguments which are structurally identical, only differing on the constants being used to build the corresponding derivations. Thus, a potential argument stands for several arguments which use the same defeasible rules. Attack relations among potential arguments can be also captured, and in some cases even defeat can be pre-compiled. In what follows we introduce the formal definitions, adapted from [6] to fit the OP-DeLP system.

Definition 14. [Weighted Potential argument] *Let Δ be a set of defeasible rules. A subset A of Δ is a* potential argument *for a literal Q with an upper bound γ for its certainty degree, noted as $\langle\langle \mathsf{A}, Q, \gamma \rangle\rangle$ if there exists a non-contradictory set of literals Φ and an instance $\mathcal{A}$ that is obtained finding an intance for every rule in A, such that $\langle \mathcal{A}, Q, \alpha \rangle$ is an argument wrt $\langle \Phi, \Delta \rangle (\alpha \leq \gamma)$ and there is no instance $\langle \mathcal{B}, Q, \beta \rangle$ of A such that $\beta > \gamma$.*

Definition 14 does not help to obtain the set of potential arguments from a given program. The interested reader may consult [6] for a constructive definition and its associated algorithm. The calculation of the upper bound γ deserves a special mention, since the algorithm in [6] was devised for a different system, without uncertainty management. This element will be used later on to speed-up the extraction of the dialectical tree for a given query from the dialectical graph. To calculate γ on a potential argument A we simply choose the lower certainty degree of the defeasible rules present in A.

The nodes of the dialectical graph are the potential arguments. The arcs of our graph are obtained calculating the counterargument relation among the nodes previously obtained. To do this, we extend the concept of counterargument for potential arguments. A potential argument $\langle\langle \mathsf{A}_1, Q_1, \alpha \rangle\rangle$ *counter-argues* $\langle\langle \mathsf{A}_2, Q_2, \beta \rangle\rangle$ at a literal Q if and only if there is a non-empty potential subargument $\langle\langle \mathsf{A}, Q, \gamma \rangle\rangle$ of $\langle\langle \mathsf{A}_2, Q_2, \beta \rangle\rangle$ such that Q_1 and Q are contradictory literals.[1] Note that potential counter-arguments may or may not result in a real conflict between the instances (arguments) associated with the corresponding potential arguments. In some cases instances of these arguments cannot co-exist in any scenario (*e.g.*, consider two potential arguments based on contradictory observations). Now we can finally define the concept of dialectical graph:

Definition 15. [Dialectical Graph] *Let $\mathcal{P} = \langle \Psi, \Delta \rangle$ be an OP-DeLP program. The* dialectical graph *of Δ, denoted as $\mathcal{G}_\Delta$, is a pair* $(\text{PotArg}(\Delta), C)$ *such that: (1)* $\text{PotArg}(\Delta)$ *is the set $\{\langle\langle \mathsf{A}_1, Q_1, \alpha_1 \rangle\rangle, \ldots, \langle\langle \mathsf{A}_k, Q_k, \alpha_k \rangle\rangle\}$ of all the potential arguments that can be built from Δ; (2) $\mathcal{C}$ is the counterargument relation over the elements of* $\text{PotArg}(\Delta)$.

Example 4. Consider the program given in Example 2. The associated graph is composed by the potential arguments shown in Figure 3, and $\mathcal{C} = \{(\mathsf{A}_1, \mathsf{A}_5), (\mathsf{A}_5, \mathsf{A}_1), (\mathsf{A}_2, \mathsf{A}_5), (\mathsf{A}_5, \mathsf{A}_2), (\mathsf{A}_4, \mathsf{A}_5), (\mathsf{A}_5, \mathsf{A}_4), (\mathsf{A}_3, \mathsf{A}_5), (\mathsf{A}_5, \mathsf{A}_3), (\mathsf{A}_6, \mathsf{A}_5), (\mathsf{A}_5, \mathsf{A}_6), (\mathsf{C}_1, \mathsf{C}_2), (\mathsf{C}_2, \mathsf{C}_1), (\mathsf{C}_1, \mathsf{C}_3), (\mathsf{C}_3, \mathsf{C}_1), (\mathsf{C}_2, \mathsf{B}_3), (\mathsf{C}_3, \mathsf{B}_3)\}$.

Having defined the dialectical graph we now present algorithm 3 to extract a particular dialectical tree rooted in a given potential argument.

To solve a query under the OP-DeLP system using pre-compiled knowledge we use algorithm 1 replacing the call to algorithm `Warrant` with a call to `WarrantFromGraph`. Then the inference process starts finding the potential argument in the graph corresponding to $\langle \mathcal{A}, Q, \alpha \rangle$ to follow the link to its counterarguments that are already precomputed in the dialectical graph. Before analyzing the counterarguments it cuts those whose upper bound δ is not greater

[1] Note that $P(X)$ and $\sim P(X)$ are contradictory literals although they are non-grounded. The same idea is applied to identify contradiction in potential arguments.

$\langle\langle \mathsf{A}_1, \sim\texttt{move\_inbox(X)}, 0.8\rangle\rangle$, where $\mathsf{A}_1 = \{(\sim\texttt{move\_inbox(X)} \prec \texttt{move\_junk(X)}, 0.8)\}$
$\langle\langle \mathsf{A}_2, \sim\texttt{move\_inbox(X)}, 0.8\rangle\rangle$, where $\mathsf{A}_2 = \{(\sim\texttt{move\_inbox(X)} \prec \texttt{move\_junk(X)}, 0.8); (\texttt{move\_junk(X)} \prec \texttt{virus(X)}, 1)\}$
$\langle\langle \mathsf{A}_3, \sim\texttt{move\_inbox(X)}, 0.8\rangle\rangle$, where $\mathsf{A}_3 = \{(\sim\texttt{move\_inbox(X)} \prec \texttt{move\_junk(X)}, 0.8); (\texttt{move\_junk(X)} \prec \texttt{spam(X)}, 1)\}$
$\langle\langle \mathsf{A}_4, \sim\texttt{move\_inbox(X)}, 0.7\rangle\rangle$, where $\mathsf{A}_4 = \{(\sim\texttt{move\_inbox(X)} \prec \texttt{move\_junk(X)}, 0.8); (\texttt{move\_junk(X)} \prec \texttt{spam(X)}, 1), (\texttt{spam(X)} \prec \texttt{black\_list(X)}, 0.7)\}$
$\langle\langle \mathsf{A}_5, \texttt{move\_inbox(X)}, 0.6\rangle\rangle$, where $\mathsf{A}_5 = \{(\texttt{move\_inbox(X)} \prec \sim\texttt{filters(X)}, 0.6)\}$
$\langle\langle \mathsf{A}_6, \sim\texttt{move\_inbox(X)}, 0.7\rangle\rangle$, where $\mathsf{A}_6 = \{(\sim\texttt{move\_inbox(X)} \prec \texttt{filters(X)}, 0.7)\}$
$\langle\langle \mathsf{B}_1, \texttt{move\_junk(X)}, 1\rangle\rangle$, where $\mathsf{B}_1 = \{(\texttt{move\_junk(X)} \prec \texttt{virus(X)}, 1)\}$
$\langle\langle \mathsf{B}_2, \texttt{move\_junk(X)}, 1\rangle\rangle$, where $\mathsf{B}_2 = \{(\texttt{move\_junk(X)} \prec \texttt{spam(X)}, 1)\}$
$\langle\langle \mathsf{B}_3, \texttt{move\_junk(X)}, 0.7\rangle\rangle$, where $\mathsf{B}_3 = \{(\texttt{move\_junk(X)} \prec \texttt{spam(X)}, 1); (\texttt{spam(X)} \prec \texttt{black\_list(X)}, 0.7)\}$
$\langle\langle \mathsf{C}_1, \texttt{spam(X)}, 0.7\rangle\rangle$, where $\mathsf{C}_1 = \{(\texttt{spam(X)} \prec \texttt{black\_list(X)}, 0.7)\}$
$\langle\langle \mathsf{C}_2, \sim\texttt{spam(X)}, 0.6\rangle\rangle$, where $\mathsf{C}_2 = \{(\sim\texttt{spam(X)} \prec \texttt{contacts(X)}, 0.6)\}$
$\langle\langle \mathsf{C}_3, \sim\texttt{spam(X)}, 0.7\rangle\rangle$, where $\mathsf{C}_1 = \{(\sim\texttt{spam(X)} \prec \texttt{local(X)}, 0.7)\}$

Fig. 3. Potential arguments for Example 4

or equal than the certainty degree of $\langle \mathcal{A}, Q, \alpha \rangle$. Then it instantiates the potential counterarguments and calculates its certainty degree, to calculate the defeat relation. The state of every one of $\mathcal{A}$'s defeaters must be obtained through the `State` algorithm. Finally the state of $\mathcal{A}$ (defeated or undefeated) is set according to the results obtained for its defeaters.

The code for the `State` algorithm is not presented here for space reasons. Basically, it takes as input an OP-DeLP program $\mathcal{P}$, an argument $\langle \mathcal{A}, Q, \alpha \rangle$ based on it, and the *interference* and *support* argumentation lines up to that point, denoted as `IL` and `SL`. Simply put, `IL` represents the set of arguments with an even level in the current path of the tree under construction, and `SL` the arguments with an odd level. Then the `State` algorithm works like algorithm `WarrantFromGraph` analyzing the defeaters for $\mathcal{A}$ to define its state by calling itself recursively.

Example 5. Consider program in Example 2 and its associated dialectical graph shown in Example 4. To see if a given message d must be placed in the inbox, the query `move_inbox(d)` must be solved. As shown in Example 2, solving this query

Algorithm 3. `WarrantFromGraph`

```
Input: P = ⟨Ψ, Δ⟩, ⟨A, Q, α⟩, G_Δ
Output: (true/false)

⟪A, Q, γ⟫ ← PotentialArgument(⟨A, Q, α⟩)
{finds the potential argument in the graph whose instance is ⟨A, Q, α⟩}
Defeat-state ← undefeated
{finds the defeaters for ⟨A, Q, α⟩ and sets their state}
For every ⟪B, X, delta⟫ in PotArg(Δ) such that (B, A) ∈ C and δ ≥ α
   For every instance ⟨B, R, β⟩ of ⟪B, X, δ⟫ such that β ≥ α
      {Sets the state of the main argument according to its defeaters}
      If state(⟨B, R, β⟩, P, ∅, {⟨A, Q, α⟩}) = undefeated
         then Defeat-state ← defeated
   {If any of the instances remains undefeated it is a warrant}
WarrantFromGraph ← Defeat-state
```

results in a dialectical tree with tree arguments. Let us analyze now how the same inference can be obtained using the dialectical graph. Following algorithm 3, potential argument $\langle\!\langle \mathsf{A}_5, \texttt{move\_inbox(X)}, 0.6 \rangle\!\rangle$ will be instantiated resulting in argument $\langle \mathcal{A}, \texttt{move\_inbox(d)}, 0.6 \rangle$, with $\mathcal{A} = \{(\texttt{move\_inbox(d)} \prec \sim\texttt{filters(d)}, 0.6)\}$.

From the dialectical graph it follows that $\langle\!\langle \mathsf{A}_5, \texttt{move\_index(X)}, 0.6 \rangle\!\rangle$ has counterarguments A_1, A_2, A_3, A_4, A_6, but only A_4 is active according to the current set of observations. This argument is instanciated to $\langle \mathcal{B}, \sim\texttt{move\_inbox(d)}, 0.7 \rangle$, where $\mathcal{B} = \{(\sim\texttt{move\_inbox(d)} \prec \texttt{move\_junk(d)}, 0.8); (\texttt{move\_junk(d)} \prec \texttt{spam(d)}, 1); (\texttt{spam(d)} \prec \texttt{black\_list(d)}, 0.7)\}$.

From the graph associated with the dialectical database, potential counterargument C_3 can be found and instanciated $\langle \mathcal{D}, \sim\texttt{spam(d)}, 0.9 \rangle$, where $\mathcal{D} = \{(\sim\texttt{spam(d)} \prec \texttt{local(d)}, 0.9)\}$. Note that from the information in the dialectical graph related to C_3 there are no more links in the graph to new counterarguments for these potential arguments that can be instanciated to defeat $\langle \mathcal{D}, \sim\texttt{spam(d)}, 0.9 \rangle$. As a consequence, a dialectical tree identical to the one shown in Example 2 has been computed from the dialectical graph.

5 Conclusions

In this work we have defined an argumentation-based formalism that integrates uncertainty management. This system was also provided with an optimization mechanism based on pre-compiled knowledge. Using this, the argumentation system can comply with real time requirements needed to administer data and model reasoning over this data in dynamic environments.

To use dialectical graphs in OP-DeLP we have introduced algorithms for the inference process. We have also developed new algorithms for the construction of dialectical bases that have not been included here for space reasons. We compared the obtained algorithms theoretically with standard argument-based

inference techniques (such as those used in P-DeLP). At the inference process, complexity is lowered from $O\left(2^{|\Delta'|^{3.(2^{|\Delta'|})/4}}\right)$ to $O(2^{|\Delta'|}.|\Delta'|)$.

Another contribution is the architectural model to integrate OP-DeLP in practical applications to administer and reason with data from multiple sources. In this model we incorporated a perception mechanism that subsumes belief revision and updating. We have also chosen to assign a unique certainty degree to each data source. Perceptions obtained from this source automatically acquire this degree. This is an interesting solution for many applications, since it frees the knowledge engineer from assigning degrees to perceptions in an *ad hoc* way. Currently we are developing a prototype based on the proposed architecture to extend the theoretical complexity analysis with empirical results and to test the integration of the OP-DeLP reasoning system in real world applications.

References

1. Alsinet, T., Chesñevar, C.I., Godo, L., Sandri, S., Simari, G.R.: Formalizing argumentative reasoning in a possibilistic logic programming setting with fuzzy unification. International Journal of Approximate Reasoning 48(3), 711–729 (2008)
2. Alsinet, T., Chesñevar, C.I., Godo, L., Simari, G.R.: R Simari. A logic programming framework for possibilistic argumentation: Formalization and logical properties. Fuzzy Sets and Systems 159(10), 208–228 (2008)
3. Alsinet, T., Godo, L.: A complete calculus for possibilistic logic programming with fuzzy propositional variables. In: Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence (UAI-2000), pp. 1–10. ACM Press, New York (2000)
4. Berti, L.: Quality and recommendation of multi-source data for assisting technological intelligence applications. In: Proc. of 10th International Conference on Database and Expert Systems Applications, Italy, pp. 282–291. AAAI, Menlo Park (1999)
5. Bryant, D., Krause, P.: An implementation of a lightweight argumentation engine for agent applications. In: Fisher, M., van der Hoek, W., Konev, B., Lisitsa, A. (eds.) JELIA 2006. LNCS (LNAI), vol. 4160, pp. 469–472. Springer, Heidelberg (2006)
6. Capobianco, M., Chesñevar, C.I., Simari, G.R.: Argumentation and the dynamics of warranted beliefs in changing environments. Journal of Autonomous Agents and Multiagent Systems 11, 127–151 (2005)
7. Carbogim, D., Robertson, D., Lee, J.: Argument-based applications to knowledge engineering. The Knowledge Engineering Review 15(2), 119–149 (2000)
8. Chesñevar, C.I., Maguitman, A.G., Loui, R.P.: Logical Models of Argument. ACM Computing Surveys 32(4), 337–383 (2000)
9. Chesñevar, C.I., Maguitman, A.G., Simari, G.R.: Argument-based critics and recommenders: A qualitative perspective on user support systems. Data & Knowledge Engineering 59(2), 293–319 (2006)
10. Chesñevar, C.I., Maguitman, A.G.: ArgueNet: An Argument-Based Recommender System for Solving Web Search Queries. In: Proc. of Intl. IEEE Conference on Intelligent Systems IS-2004, Varna, Bulgaria (June 2004)

11. Chesñevar, C.I., Simari, G.R., Alsinet, T., Godo, L.: A logic programming framework for possibilistic argumentation with vague knowledge. In: Proc. of Uncertainty in Artificial Intelligence Conference (UAI 2004), Banff, Canada (2004) (to appear)
12. Doyle, J.: A Truth Maintenance System. Artificial Intelligence 12(3), 231–272 (1979)
13. Dubois, D., Lang, J., Prade, H.: Possibilistic logic. In: Gabbay, D., Hogger, C., Robinson, J. (eds.) Handbook of Logic in Art. Int. and Logic Prog. (Nonmonotonic Reasoning and Uncertain Reasoning), pp. 439–513. Oxford Univ. Press, Oxford (1994)
14. García, A., Simari, G.: Defeasible Logic Programming: An Argumentative Approach. Theory and Practice of Logic Programming 4(1), 95–138 (2004)
15. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New Generation Computing, 365–385 (1991)
16. Gomez, S.A., Chesñevar, C.I.: A Hybrid Approach to Pattern Classification Using Neural Networks and Defeasible Argumentation. In: Proc. of Intl. 17th FLAIRS Conference, Palm Beach, FL, USA, May 2004, pp. 393–398. AAAI, Menlo Park (2004)
17. Katsuno, H., Mendelzon, A.: On the difference between updating a knowledge base and revising it. In: Gardenfors, P. (ed.) Belief Revision, pp. 183–203. Cambridge University Press, Cambridge (1992)
18. Prakken, H., Vreeswijk, G.: Logical systems for defeasible argumentation. In: Handbook of Philosophical Logic, vol. 4, pp. 219–318 (2002)
19. Rahwan, I., Ramchurn, S.D., Jennings, N.R., McBurney, P., Parsons, S., Sonenberg, L.: Argumentation-based negotiation. The Knowledge Engineering Review 18(4), 343–375 (2003)
20. Simari, G.R., Loui, R.P.: A Mathematical Treatment of Defeasible Reasoning and its Implementation. Artificial Intelligence 53(1–2), 125–157 (1992)

Making Sense of a Sequence of Events: A Psychologically Supported AI Implementation

Philippe Chassy and Henri Prade

IRIT, University of Toulouse,
31062 Toulouse Cedex 9, France
chassy@irit.fr, henri.prade@irit.fr

Abstract. People try to make sense of the usually incomplete reports they receive about events that take place. For doing this, they make use of what they believe the normal course of thing should be. An agent′s beliefs may be consonant or dissonant with what is reported. For making sense people usually ascribe different types of relations between events. A prototypical example is the ascription of causality between events. The paper proposes a systematic study of consonance and dissonance between beliefs and reports. The approach is shown to be consistent with findings in psychology. An implementation is presented with some illustrative examples.

1 Making Sense in Artificial Intelligence and Psychology

Whether robots, animals, or humans, autonomous agents have to make sense of their environment before undertaking any further goal-driven action. More specific to human agents is the fact that making sense may be only but a prerequisite for engaging in further task-oriented cognitive processing such as explaining, persuading, predicting, or diagnosing. The fact that making sense is an all-purpose psychological function makes it a central topic both in psychology and in artificial intelligence (AI).

The present paper focuses on human making sense and how to model it. As pointed out by many psychologists (See [1] and references therein), the making sense process is not mere information intake and requires that knowledge interacts with salient features of a situation. For example, human agents have at their disposal memory structures developed to encode knowledge about the events that typically occur in highly-structured situations (e.g. at the doctor, at the restaurant). These memory structures are called schemes or scripts [5,6] and are keys to interpret the world. Similarly, experts make rapid sense of complex situations by coordinating perceptual input and domain-specific knowledge [2]. In line with this view, psychological models, such as ACT- R [3] make use of declarative and procedural knowledge to account for human problem solving. The literature of AI is also rich in models that make sense of a situation by interpreting data with domain-specific knowledge. MYCIN [4] is one of the first examples of expert support systems that makes sense of a situation. Both AI

L. Godo and A. Pugliese (Eds.): SUM 2009, LNAI 5785, pp. 111–123, 2009.

and psychology have brought to light the role played by both declarative and rule-based knowledge.

In agreement with this view, a new approach of causality ascription has been recently proposed [7]: faced with a reported sequence of events, the model interprets the facts by using context-dependent knowledge. This model compares well with other models accounting for causal ascription [8]. We develop the approach further and term it KUMS (for Knowledge Use for Making Sense) theory in the following. Data from psychology have been used to justify the architecture of the model and to add several new features.

The remainder of this paper is structured as follows. In section 2, we show how a link is put to connect two events[1]. In Section 3, we provide a structural view of different making sense situations. Section 4 presents the computer model developed to test the validity of the model regarding human data. Section 5 reports the results of the simulations and illustrates which key psychological phenomena the model is able to simulate. Finally, in section 6, we discuss the limits and point to further studies.

2 Making Sense: Connecting Facts through Knowledge

To introduce the key idea of connection, let us consider the context of soccer, and the sequence : S (shooting, dribbling, Messi_running, goal). We can expect an agent to reorganize data and perceive: Messi_running $\rightarrow$ Dribbling $\rightarrow$ Shooting $\rightarrow$ Goal. The connector symbol $\rightarrow$ does not yet carry any specific meaning and is used here to show that statements are re-organized so as to integrate facts. This section addresses the question of the nature of such connectors.

AI and psychology agree on the fact that human agents have a wide variety of connectors that can bind events together [9,10]. Due to its historical importance in philosophy, psychology and AI, the causal connector is probably one of the most studied. We use this connector to illustrate how KUMS theory describes the process whereby one perceiver ascribes a connector so as to link two (or more) events. Causal ascription has been the focus of active research recently [7,8,9,11,12]. For a causal relationship to be perceived, time is to elapse between the putative cause and its effect [13]. KUMS theory is consistent with these findings and integrates time as a discrete variable. As outlined in the above, KUMS theory is based on the assumption that making sense emerges from the interplay between context, facts, and context-dependent knowledge. We examine each factor in turn. Context refers to the set of variables that define the situation under consideration as belonging to a class of situations. The context is of no use to the agent except that of constraining the knowledge that can be used to make sense of a situation. The context will be denoted by K. The facts are the sequence of events that are under consideration in a given context. There are two possible states about any statement E in context K at time t: either E is known

[1] We use the idea of a connector as a 'psychological operation linking two elements'. This notion should be distinguished from the common AI notion of connectives for that human connectors do not need to be logical.

as true (denoted E_t) , or E is known as false ($\neg E_t$). A sequence of events is thus a series of time-stamped statements reflecting the agent′s knowledge about what has occurred. Sequences may have more than one event that takes place in one time step (e.g., the sequence Messi_running_1, Dribbling_1, Shooting_2, Goal_3). It is worth noting that the relevant information is not supposed to be complete.

Now we turn our attention to context-dependent knowledge. Note that beliefs are not necessarily true, but are considered as such by the agent who holds them. We can distinguish two types of beliefs.

The first type refers to the putative relationship that links two events. We make the assumption that the pieces of knowledge that are used by an agent for interpreting reported facts are consistent. Namely, in a given context K, the agent cannot believe in the same time that E is normal and E is not normal. The belief connecting an event A to a statement E in context K will be viewed as a default rule *if A and K then normally E* denoted by $A \wedge K \mathrel{|\!\sim} E$; where the $\mathrel{|\!\sim}$ denotes a nonmonotonic consequence relation assumed to follow the postulates of system P [24]. In context K, given that A is true, an agent may (i) either believe that E should be true, i.e., $A \wedge K \mathrel{|\!\sim} E$, (ii) that E should be false, i.e., $A \wedge K \mathrel{|\!\sim} \neg E$, or (iii) that E may be true or false as well, i.e., $A \wedge K \not\mathrel{|\!\sim} E$ and $A \wedge K \not\mathrel{|\!\sim} \neg E$. Here, the symbol $\not\mathrel{|\!\sim}$ stands for the negation that the default holds.

The second type of knowledge that is crucial for making sense is normality. As AI scientists and psychologists have shown, events perceived as normal or abnormal play a central role in causal ascription [14,15]. This finding highlights the fact that agents hold context-specific beliefs about what is expected to be normal and what is not. For example, if we are attending a music festival it is normal to see guitars while this object is not normal in the context of a battlefield. The epistemic state of an agent with respect to an event E in context K could correspond to three mutually exclusive situations: (i) $K \mathrel{|\!\sim} E$, E is generally true in context K; (ii) $K \mathrel{|\!\sim} \neg E$, E is generally false in context K; (iii) $K \not\mathrel{|\!\sim} E$. E may or not be true in context K.

Let us now detail the mechanisms ascribing a causal connector between two events. Consider for instance the case of an agent informed that a friend of his, Paul, had an accident. The agent is also aware that Paul was drunk while driving. Finally, let us presume that the agent believes that drunkenness in the context of driving generates accidents. Then, the agent will conclude that Paul being drunk caused his accident. Formally:

Definition 1 (Causal Ascription). Typically, an agent learns that $\neg E$ and A were true at a time t and that at a later time $(t + 1)$ there has been a change and now E is true. If the agent holds the belief that $K \mathrel{|\!\sim} \neg E$ and $A \wedge K \mathrel{|\!\sim} E$, then A is perceived by the agent to be the cause of E in context K.

The above definition has been validated by psychological experiments [9]. Such a definition of causality allows deriving some properties associated to the perception of a causal connector. In particular it is not the case that A can be normal in context K (the agent cannot consistently hold the belief that $K \mathrel{|\!\sim} A$). Moreover

transitivity does not necessarily hold for causal ascription; i.e., believing that $A \wedge K \mathrel{|\!\sim} E$ and $E \wedge K \mathrel{|\!\sim} Z$ does not make believe that $A \wedge K \mathrel{|\!\sim} Z$; this property is due to the nonmonotonicity of the $\mathrel{|\!\sim}$ [24]. Through the example of causal ascription we have shown that connector ascription results from a specific combination of a set of items, namely: beliefs about what is normal, beliefs about possible influences, truth state of the possible antecedent A, and change of state of the supposed effect E. These factors can be set to different values and generate different ascriptions. These different connectors are the focus of the next section.

3 The Role of Knowledge in Making Sense

It has been explained in the above that the ascription of the causal link arises as the interplay between a specific report (pattern of events) and a specific pattern of beliefs. We present hereafter all the possible beliefs states and the possible report states. We show that the report-belief interaction generates many possible types of connections, among which cause is only but one case. For sake of simplicity, we consider the case wherein the agent is interested in making sense of the action A on the putative effect E. Let t and $t+1$ be two time steps. Action A is reported (or not) to take place at time step t. The state of E in time steps t and $t+1$ is known so that the agent can determine whether the state of E changes or persists in the course of time. We now detail how connectors are ascribed. By convention, and for the sake of simplicity, we set $t = 1$ and $t+1 = 2$ in the notation below.

Let first consider the role of beliefs. As already said, the belief regarding the status of normality of one object in a given context can have three different possible states (i.e., normal, abnormal, or unknown). Similarly, the conditional belief factor has also three states ($A \wedge K \mathrel{|\!\sim} \neg E$, $A \wedge K \mathrel{|\!\sim} \neg E$, or $A \wedge K \not\mathrel{|\!\sim} E$ and $A \wedge K \not\mathrel{|\!\sim} \neg E$). A factorial combination of those two types of belief yields nine possible belief states. The type of report also arises from a factorial combination of two factors. The first factor is whether the state of the putative effect E changes or not during the course of events. The second factor refers to whether action A is reported to the agent or not. The combination of these two variables yields four possible report cases. We introduce two key terms that are of great use to organize the result of a comparison between beliefs and facts. The term of consonance is used to describe a psychological state where cognitions do not contradict each other. For example, let consider an agent who believes that apples fall because of gravity. If an apple falls from a tree the fact will not contradict the knowledge and thus will be consonant with it. Dissonance appears whenever there is a inconsistency in cognitions. For example, observing that the apple does not fall would generate dissonance. Dissonance can also arise from beliefs inconsistencies. For example, an agent may believe that Paul is nice and that James is not. A dissonance can emerge between the two beliefs if the agent learns that Paul and James are good friends (assuming that normally a nice person has nice friends only). Human agents tend to reduce dissonance by

Table 1. Connector ascription as a function of state of the supposed effect (E) and action (A) as well as the beliefs held regarding the facts (A and E)

			FACTS			
			A reported		A not reported	
	(K)	(K ∧ A)	E changes (E1, ¬E2)	E persists (E1, E2)	E changes (E1, ¬E2)	E persists (E1, E2)
Context dependent BELIEFS	$\mid\sim E$	$\mid\sim E$	Dissonance	Consonant*	Dissonance	Consonant*
		$\not\mid\sim E$	Facilitation (Consonant**)	Consonant**	Dissonance	Consonant*
		$\mid\sim \neg E$	Cause (Consonant*)	Dissonance	Dissonance	Consonant*
	$\not\mid\sim E$	$\mid\sim E$	Dissonance	Consonant**	Consonant**	Consonant**
		$\not\mid\sim E$	Consonant**	Consonant**	Consonant**	Consonant**
		$\mid\sim \neg E$	Justification (Consonant**)	Dissonance	Consonant**	Consonant**
	$\mid\sim \neg E$	$\mid\sim E$	Dissonance	Dissonance	Dissonance	Dissonance
		$\not\mid\sim E$	Dissonance	Dissonance	Dissonance	Dissonance
		$\mid\sim \neg E$	Dissonance	Dissonance	Dissonance	Dissonance

reorganizing cognitions. Dissonance reduction is processed by considering events in a new light or by changing beliefs. Within the theoretical framework presented in the current paper the terms of consonance and dissonance will refers to a belief-facts comparison. The combination between beliefs and reports yields the 36 report-belief cases that appear in Table 1.

At the best of our knowledge, the approach presented in Table 1 is new to AI and psychology. We would like here to establish the various ways in which making sense is being carried out. Psychologically speaking, the beliefs that an agent holds regarding a given context yields expectations about what is likely to occur. As long as the situation is as expected (matches what is believed to be normal), the match between facts and belief is consonant. We distinguish between two subclasses of consonance. Note that the proposed taxonomy is made possible because of the use of $\mid\sim$ that encodes a qualitative theory of uncertain knowledge. The type of connector that is ascribed has been selected upon the base of results collected in AI and psychology [9,8].

Table 1 summarizes how an agent may react to a given report according to his beliefs. The considered sequence may report a change or a mere persistence, and it may include or not an event A. Thus, given the context K the agent may find i) *normal* that E is true at time 1 (if for him $K \mid\sim E$), ii) *abnormal* that E is true at time 1 (if $(K \mid\sim \neg E)$, iii) *contingent* (if $K \not\mid\sim E$ and $K \not\mid\sim \neg E$) that E is true at time 1. Similarly, the state of E at time 2 may be found normal, abnormal, or contingent w. r. t. the agent′s belief that involve A if A is reported, and the other

beliefs if A is not reported. Thus, one may have nine different pairs of perceived world states, ranging from $(normal_1 - normal_2)$ to $(abnormal_1 - normal_2)$ and to $(contingent_1 - abnormal_2)$. The pair $(normal_1 - normal_2)$ expresses a strong consonance (stamped by * in the table), the pairs $(normal_1 - contingent_2)$, $(contingent_1 - normal_2)$ and $(contingent_1 - contingent_2)$ sound weakly consonant (Stamped by ** in the table). The other pairs are felt dissonant. We distinguish three different forms of dissonance: dissonance is complete if abnormality takes place at times 1 and 2, it is softened (resp. resolved) if abnormality is only at time 1 and contingency (resp. normality) takes place at time 2, and it is final when the situation reported at time 2 is abnormal. Complete and final dissonance are the most uncomfortable for the agent, since the dissonance marks a lack of making sense.

Some consonant or dissonant situations when A takes place allow the ascription of some specific connector. Causation corresponds to a change from E_1 to $\neg E_2$ where normality w. r. t. beliefs is preserved. Facilitation is a weakened form where the final state after change is found contingent ($(K \wedge A) \not\sim E$ and $(K \wedge A) \not\sim \neg E$). In case $\neg E$ is normal in context K for the agent ($K \sim \neg E$) – which makes E_1 exceptional – while the agent believes that $\neg E$ is normal when A takes place ($(K \wedge A) \sim \neg E$), A *justifies* the arrival of $\neg E_2$ (perceived as a return to normality). If E_2 is reported, justification will correspond to the case where the agent believes $(K \wedge A) \sim E$. Note that A is felt as a justification when we are back to normality, after being initially in an abnormal or contingent state.

Note that the belief $K \sim E$ expresses that E tends to persist when nothing takes place. When $K \not\sim E$ or $K \sim \neg E$, it might be the case that the agent has the additional belief that $K \wedge E_t \sim \neg E_{t+1}$, i.e. if E takes place contingently or by exception, it is unstable. Then in such a situation (not considered in Table 1), if the agent also believes that $K \wedge A \sim E$, then the agent will consider that the occurrence of A has *prevented* $\neg E$ to take place, when E_1, A and E_2 are reported.

The idea that we process information as chunks has been put forward by Miller [17] and then empirically validated [18,19]. This chunking helps to compensate for the limit on human memory. Information is chunked on the base of association through meaning. Hence, KUMS theory reflects findings in psychology and provides a formal background to explain how chunking occurs. KUMS bridges the gap between memory and the reasoning abilities of human agents. To summarize, Table 1 can be used to make predictions about how humans are going to make sense of the situation at hand. That is, ascription will link events in a bigger psychological representation so that the individual can operate further cognitive processing (diagnosing for example). Furthermore, Table 1 offers a taxonomy of human reactions based on environmental and experiential factors. As such, it is a good tool to model and simulate human behavior as well as to develop human-like robots.

The following example illustrates the process of making sense within KUMS theory. Let consider an agent who holds the following beliefs: One is generally not drunk, and when driving, one has generally no accident: $\sim \neg$ Drunk, Drive

$\not\sim \neg$ Accident. The agent also believes that driving while drunk is not as safe and may lead to an accident: Drive $\wedge$ Drunk $\not\not\sim \neg$ Accident. The agent is convinced that being drunk and driving fast leads to accidents Drive $\wedge$ Fast $\wedge$ Drunk $\not\sim$ Accident. In the context of a bar, the agent holds the view that drinking is normal and that the presence of alcohol is normal as well. So the agent believes that Bar $\wedge$ Alcohol $\not\sim$ Drunk. Let now spell out how the agent is going to make sense of the following story: John spent his evening at the bar. John was wearing a red shirt. He drank alcohol for two hours. He eventually got drunk. When time has come he took his car to go back home. He drove fast as he was in a hurry. He had an accident.

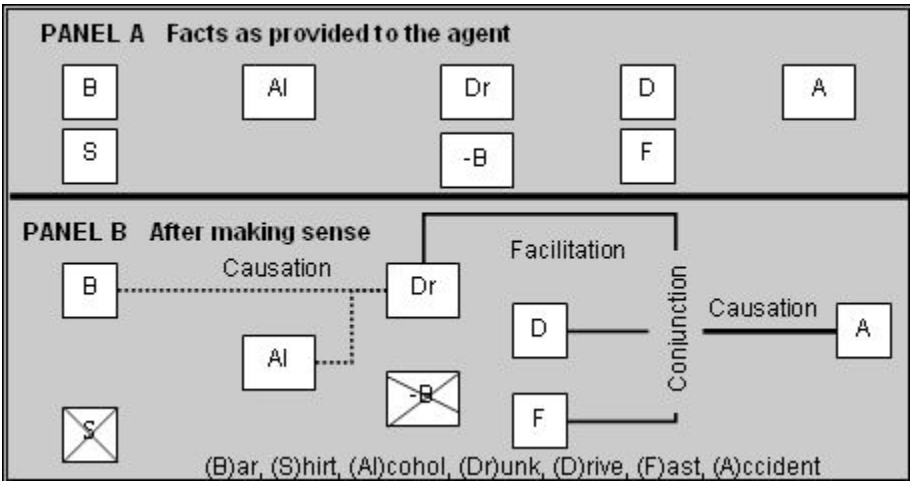


Fig. 1. Representation of the situation held by the agent before and after making sense

Let us code the facts according to the following sequence: S (Bar_1, Shirt_1, Alcohol_2, Drunk_3, $\neg\ \text{Bar}_3$, Drive_4, Fast_4, Accident_5). Let us know make a graph representation of the sequence. See Figure 1 Panel A to see a time dependent representation of the events.

Figure 1 illustrates the contents of the agent′s temporary buffer before and after that the making sense process has operated upon the internal representation. Once the agent is reported with the sequence of events, events are still unlinked (Panel A). Panel B shows how the agent has made sense of the sequence of events. In line with the approach we have adopted, we got the following ascriptions: {Drunk $\wedge$ Drive $\wedge$ Fast CAUSE Accident}; {Bar $\wedge$ Alcohol CAUSE Drunk}; {Drunk FACILITATE Accident}. It is worth noting that this illustrates the fact that some pieces of information (here, 'shirt') can have no importance in a given context, and as such can disappear during the process of connector ascription, as being irrelevant. This is true whenever the agent has no belief that

can be related with some reported fact X (there is no belief of the form $K \wedge S \not\sim X$, $K \wedge S \not\sim \neg X$, $K \wedge X \not\sim S$, or $K \wedge X \not\sim \neg S$ held by the agent).

4 Implementing a Connection Ascriptor for Making Sense

Figure 2 shows the architecture of the Connection Ascriptor for Making Sense (CAMS) module. CAMS is made of four components: a temporary buffer, a belief base, an interpreter, and a working buffer. We examine each component in turn. The temporary buffer is in charge of holding the sequence of events for the duration of the simulation. Psychologically, it is supposed to hold the information that is accessed by the agent′s consciousness. It is the place wherein data is held online while making sense processes are run. Events are marked by their time of occurrence so that time-stamped sequences of events can be entered.

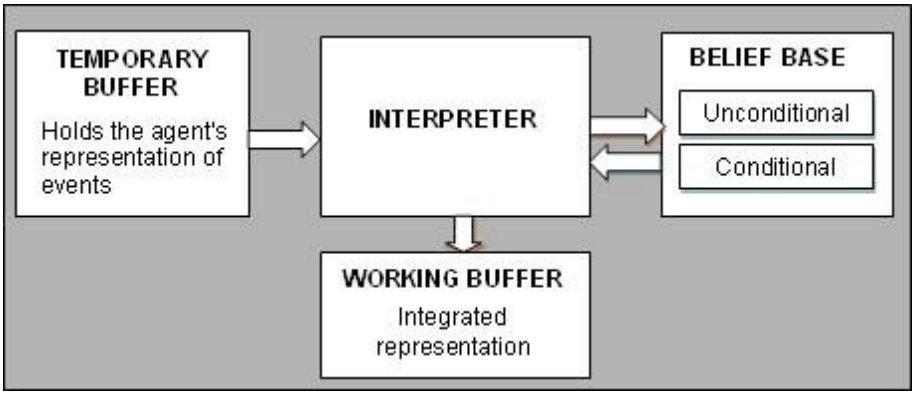


Fig. 2. Architecture of the CAMS program

The belief base encodes all the context-dependent beliefs held by an agent. The data stored in the belief base is indexed by two types of entries. The first type of entry is normality (unconditional beliefs). The records of the unconditional belief will inform the interpreter as to what is the normal state of the event under consideration. The second belief base, termed conditional, stores the records connecting the event under consideration with other pieces of knowledge. From the standpoint of psychology the belief base implements declarative long-term memory[2]. The query of searching into the database is made by the interpreter. The results that match the queries are forwarded to the interpreter.

[2] It is supposed here that the belief memories contain all the beliefs that the agent is aware of. In particular they should explicitly contain their respective deductive closure in the sense of System P [24], if the agent is supposed to be omniscient.

The interpreter is the core of the model. It completes the function of ascribing connectors between events. It stands at the interplay between the temporary buffer, the belief base, and the working buffer. It scans all the events stored in the temporary buffer and retrieves one fact at a time from it. For each fact, it applies the following 4-step procedure. First, the interpreter searches for the information concerning the epistemic state of normality. Then, the information is retrieved and stored in a local variable. The interpreter then searches into the conditional belief base to determine whether the event under consideration has one or more known beliefs associated to it. Finally, by a set of if-then rules the interpreter combines the type of report and the belief state to determine the type of connector that is to be ascribed. The interpreter is also in charge of updating the content of the working buffer at the end of the simulation by forwarding the chunks of connected data.

The working buffer receives the interpreter results. It holds the chunks of information reflecting the agents' understanding of the situation. This buffer does not play an active role in the simulation but helps distinguishing mere perception of facts (temporary buffer) from elaborated mental representations. The architecture of CAMS shares a lot of properties with Atkinson and Shiffrin′s [21,22] model of human cognition. These authors have posited that an executive system is controlling the flow of information between short-term memory and long-term memory.

5 Illustrative Examples

We present in this section some simulations illustrating how the model works and how sensitive it is to modifications in the belief base or in the report details. These features aims to reproduce the behavior of human agents when handling report-based information. In CAMS, we aimed to implement most of the connectors that are under study in psychology (causation, facilitation, justification). We chose not to show normal ascriptions and unconnected events as this would overload the output without adding much information as how people make sense of events. For the first two examples we selected excerpts from a database of actual accident reports.

The following table presents the conditional and unconditional beliefs that were used to simulate the data of examples 1 and 2. The table is a list of the system knowledge in no specific order. We detail the first example we have used so as to illustrate how the input and the output are coded. Let us consider the following story.

Example 1. *We were at Somewhere, I was surprised by the person who braked in front of me, not having the option of changing lane and the road being wet, I could not stop completely in time.*

For the simulations, statements were recoded as simple words; for example, the fact that the author brakes was coded as *A_brakes*. By using this procedure,

Table 2. Beliefs used to simulate the model on a selected set of examples

Unconditional beliefs		Conditional beliefs	
A_brakes	¬Accident	road_wet ∧ ¬A_r_distance |~ accident	A_surprised |/~ accident
A_moves	Obstacle	Rain wet |~ road	changelane |~ fullock
A_trajectory	A_distance	A_drives ∧ A_surprised |~ A_brakes	A_moves ∧ ¬A_controls |~ accident
¬B_stalls	¬A_surprised	A_moves ∧ A_surprised |~ accident	A_moves ∧ ¬A_controls |~ accident
A_controls	|/~ road_wet	A_brakes ∧ ¬A_r_distance |~ accident	A_trajectoire |~ accident
A_stops	B_moves	A_moves ∧ A_surprised |~ A_ brakes	¬A_controls |~ accident
		A_surprised |~ A_ brakes	A_brakes ∧ road_wet |~ ¬A_controls

Example 1 was recoded as a sequence: S (A_drives$_1$, obstacle$_2$, A_surprised$_2$, A_brakes$_3$, road_wet$_4$, ¬A_distance$_4$, accident$_5$). Events were entered sequentially into CAMS.

The interpreter forwarded the following output:

{A_brakes ∧ road_wet ∧ ¬ A_distance CAUSE accident} {A_drives ∧ A_surprised CAUSE A_brakes} {rain JUSTIFIES road_wet}

We can see that CAMS ascribed three different types of connectors to make sense of the situation. CAMS thus agrees with the fact that humans do have several ways of connecting events.

With the following example we aim at showing the importance of what are the beliefs involved in connector ascription.

Example 2. *I was behind the learner-driver car at the stop sign. The vehicle started up again, moved forwards and suddenly stalled. I didn't have the time to stop my vehicle; I hit the learner-driver car gently at the back. This car had a slight dent on the boot (trunk), hardly visible.*

The input was S (A_moves$_1$, B_moves$_1$, B_stall$_2$, A_surprised$_2$, A_brakes$_3$, ¬A_distance$_4$, accident$_4$). At the first simulation (S1) we used the knowledge base presented in Table 2. At the second run (S2) one piece of knowledge was changed from K $\mid\sim$ ¬ accident to K $\mid\sim$accident. The output of the two simulations was:

Simulation 1: {A_moves ∧ A_surprised CAUSE accident}
{A_brakes ∧ ¬A_distance CAUSE accident} {A_moves ∧ A_surprised CAUSE A_brakes} {A_surprised FACILITATE A_brakes}
Simulation 2: {A_moves ∧ A_surprised CAUSE A_brakes} {A_surprised FACILITATE A_brakes}

The results show that CAMS is sensitive to changes in the belief base. Even though the results may seem rather easy from the point of view of AI, they have far-reaching consequences when analyzed under the light of applied psychology. Let us consider the case where an agent has to make sense of a situation in a trial. We can see that in such a case a single belief can change the whole representation and as such have tremendous consequences on the jury. The next example shows that CAMS is sensitive to how the events are reported.

Example 3. *Billy and Suzy each throw a pebble at a window* [$Billy\_Stone$, $Suzy\_stone$] *and Suzy's pebble gets there first* [$Suzy\_Faster$]*, breaking the glass* [$Suzy\_wins$]*. Had not Suzy's pebble been thrown, Billy's pebble would have broken the glass, but Suzy's throw was nevertheless a cause of the breaking of the window.*

We equipped CAMS with the beliefs presented below.

Game $\mid\sim$ Billy_stone; Game $\mid\sim$ Suzy_stone; Game $\mid\sim$ Billy_wins; Game $\mid\sim$ Suzy_wins; Game $\wedge$ Suzy_faster $\mid\sim$ $\neg$Billy_wins; Game $\wedge$ Billy_faster $\mid\sim$ $\neg$Suzy_wins; Game $\wedge$ Billy_stone $\wedge$ Billy_faster $\mid\sim$ Billy_wins; Game $\wedge$ Suzy_stone $\wedge$ Suzy_faster $\mid\sim$ Suzy_wins; Game $\wedge$ Billy_wins $\mid\sim$ Shatter; Game $\wedge$ Suzy_wins $\mid\sim$ Shatter.

Two simulations were run. In the first simulation, the following sequence of events has been entered as an input into CAMS: S (Billy_stone$_1$, Suzy_stone$_1$, Suzy_faster$_2$, Suzy_wins$_3$, $\neg$ Billy_wins$_3$, Shatter$_4$). In the second simulation, we got rid of $Suzy\_faster$ in the sequence.

Simulation 1: {Suzy_stone$\wedge$Suzy_faster CAUSE Suzy_wins}; Simulation 2: Nil!

The results are striking: an unsuitable level of description can lead CAMS to be unable to make sense of the report. Similar to humans, CAMS is sensitive to the level of granularity at which events are reported. Should this level be not suitably chosen then the model cannot perceive the underlying structure of events. This last example illustrates the fact that sometimes the agent holds the beliefs necessary to make sense of the situation but due to a lack of appropriate level of details it cannot make sense of the situation. Hence, any further processing, such as explaining or persuading cannot carried out.

6 Concluding Remarks

By extending previous work on causal ascription, we provided a novel approach to making sense. The novelty of the theory presented in the current paper lies in the minimality of the assumptions required to make the model work and in the way the model simulates how events are connected. Various features of the model make it practical to test several psychological phenomena. CAMS consistently exhibited good performance in modeling different situations. The results are consistent with human findings. Interestingly, CAMS is not only able to simulate what humans do but also it displays the same sensitivity to changes in beliefs and reports. As such, CAMS represents a significant advance compared with the

previous version of the model [8]. Furthermore, for that CAMS integrates many types of connectors, it offers a better account of human cognition that models who limit their making sense of a situation to the analysis of causal relationships.

Although this research advances our understanding of making sense, it is not free of limitations. The system still need an experimental validation. Besides, CAMS does not implement intentional processes which are key in the human modelling of human making sense. In spite of its limits CAMS offers an implementation of the notion of making sense. As such, it may serve as a base to implement further, more elaborated notions. Responsibility is a promising avenue for that it is a notion closely related to the one of making sense. It also shares a lot of features with the ascription of causal connectors [20]. Finally, it has been posited that emotions influence how we build the representation of a situation. Models of emotional influence on experts' making sense have been put forward [23]. A new emotional module could be implemented to simulate how emotions modulate making sense. AI is interested in providing support systems with the capacity of predicting what is to occur or diagnosing what occurred. Such skills rely on the systems ability to trace back the course of events and analyze which events are connected to which. CAMS offers a tool to equip such systems.

References

1. Richard, J.-F.: Les activites mentales: comprendre, raisonner, trouver des solutions. Armand Colin, Paris (1998)
2. Gobet, F., Lane, P.C.R., Croker, S., Cheng, P.C.H., Jones, G., Oliver, I., Pine, J.M.: Chunking mechanisms in human learning. Trends in Cognitive Sciences 5, 236–243 (2001)
3. Anderson, J.R., Lebiere, C.: The atomic component of thought. Erlbaum, Mahwah, NJ (1998)
4. Shortliffe, E.H.: Computer-based medical consultations: MYCIN. Elsevier, New York (1976)
5. Abelson, R.P.: The Psychological status of the script concept. American Psychologist 36, 715–729 (1981)
6. Schank, R.C., Abelson, R.P.: Scripts, plans, goals, and understanding. Lawrence Erlbaum, Hillsdale (1977)
7. Dubois, D., Prade, H.: Modeling the role of (ab)normality in the ascription of causality judgements by agents. In: NRAC 2005, pp. 22–27 (2005)
8. Benferhat, S., Bonnefon, J.F., Chassy, P., Da Silva Neves, R.M., Dubois, D., Dupin de Saint-Cyr, F., Kayser, D., Nouioua, F., Nouioua-Boutouhami, S., Prade, H., Smaoui, S.: A comparative study of six formal models of causal ascription. In: Greco, S., Lukasiewicz, T. (eds.) SUM 2008. LNCS (LNAI), vol. 5291, pp. 47–62. Springer, Heidelberg (2008)
9. Bonnefon, J.F., Da Silva Neves, R.M., Dubois, D., Prade, H.: Predicting causality ascriptions from background knowledge: Model and experimental validation. Inter. J. of Approximate Reasoning 48, 752–765 (2001)
10. Goldvarg, Y., Johnson-Laird, P.N.: Naive causality: a mental model theory of causal meaning and reasoning. Cognitive Science 25, 565–610 (2001)

11. Mandel, D.R., Lehman, D.R.: Counterfactual thinking and ascriptions of cause and preventability. Journal of Personality and Social Psychology 71, 450–463 (1996)
12. McCloy, R., Byrne, R.M.J.: Semifactual "even if" thinking. Thinking and Reasoning 8, 41–67 (2002)
13. Lagnado, D.A., Sloman, S.A.: Time as a guide to cause. Journal of Experimental Psychology Learning Memory and Cognition 32, 451–460 (2006)
14. Kayser, D., Nouioua, F.: About Norms and Causes. International Journal of Artificial Intelligence Tools 14, 7–24 (2005)
15. Hilton, D.J., Slugoski, B.R.: Knowledge-based causal attribution: The abnormal conditions focus model. Psychological Review 93, 75–88 (1986)
16. Festinger, L.: A theory of cognitive dissonance. Standford University (1957)
17. Miller, G.A.: The magical number seven, plus or minus two: Some limits on our capacity for processing information. Psychological Review 63, 81–97 (1956)
18. Simon, H.A.: How big is a chunk? Science 183, 482–488 (1974)
19. Campitelli, G., Gobet, F., Head, K., Buckley, M., Parker, A.: Brain localization of memory chunks in chessplayers. Inter. J. of Neuroscience 117, 1641–1659 (2008)
20. Prade, H.: Responsibility judgments: steps towards a formalization. In: Magdelena, L., Ojeda- Aciego, M., Verdegay, J.L. (eds.) International Conference on Information Processing and Management of Uncertainty in Knowledge-based Systems, pp. 145–152. University of Malaga (2008)
21. Atkinson, R.C., Shiffrin, R.M.: Human memory: A proposed system and its control processes. In: Spence, K.W., Spence, J.T. (eds.) The psychology of learning and motivation, vol. 2. Academic Press, London (1968)
22. Shiffrin, R.M., Atkinson, R.C.: Storage and retrieval processes in long term memory. Psychological Review 76, 179–193 (1969)
23. Chassy, P., Gobet, F.: A model of emotional influence on memory processing. In: Cañamero, L. (ed.) AISB 2005: Symposium on Agents that Want and Like, pp. 21–25. University of Hertforshire, Hatfield (2005)
24. Kraus, S., Lehmann, D., Magidor, M.: Nonmonotonic reasoning, preferential models and cumulative logics. Artificial Intelligence 44, 167–207 (1990)

Explaining Inconsistencies in OWL Ontologies

Matthew Horridge, Bijan Parsia, and Ulrike Sattler

The University of Manchester
Oxford Road, Manchester, M13 9PL
{matthew.horridge,bparsia,sattler}@cs.man.ac.uk

Abstract. Justifications play a central role as the basis for explaining entailments in OWL ontologies. While techniques for computing justifications for entailments in consistent ontologies are theoretically and practically well-understood, little is known about the practicalities of computing justifications for inconsistent ontologies. This is despite the fact that justifications are important for repairing inconsistent ontologies, and can be used as a basis for paraconsistent reasoning. This paper presents algorithms, optimisations, and experiments in this area. Surprisingly, it turns out that justifications for inconsistent ontologies are more "difficult" to compute and are often more "numerous" than justifications for entailments in consistent ontologies: whereas it is always possible to compute *some* justifications, it is often not possible to compute *all* justifications for real world inconsistent ontologies.

1 Introduction

An ontology is a machine processable artefact that captures the relationships between concepts and objects in some domain of interest. A logic based ontology language allows ontologies to be specified as logical theories, meaning that is it possible to reason about the relationships between concepts and objects that are expressed in such an ontology. A widely used logic based ontology language is the the Web Ontology Language, OWL [1], the latest version of which is underpinned by a highly expressive Description Logic[1] called $\mathcal{SROIQ}$ [3].

An *inconsistent ontology*, is an ontology that, by virtue of what has been stated in the ontology, cannot have any models, and entails everything. The effect of this is that, using standard classic semantics, no meaningful conclusions can be drawn from an inconsistent ontology. All Description Logic based OWL reasoners merely report "ontology inconsistent" when fed such ontologies.

In practice, inconsistent ontologies can arise due to a number of reasons. For example, an ontology may become inconsistent during the editing process after the addition of some axioms. While the last set of axioms that were added may play a part in making the ontology inconsistent, it may be the case that they were the correct axioms to add from a modelling point of view, but it is the interplay of the newly added axioms with axioms already in the ontology that

[1] Description Logics [2] may be regarded as decidable fragments of First Order Logic.

L. Godo and A. Pugliese (Eds.): SUM 2009, LNAI 5785, pp. 124–137, 2009.

triggers the inconsistency. Inconsistencies can also be the result of *automated* ontology construction techniques. For example, in [4], ontologies are automatically constructed as an output from text mining, and it is possible for the resulting ontologies can be inconsistent.

There are a number of different ways of dealing with an inconsistent ontology. Broadly speaking, either some form of paraconsistent reasoning can be used to draw "meaningful" conclusions from the ontology, as in [5], or the ontology can be *repaired* so that it becomes consistent, thus meaningful conclusions can be drawn from the repaired ontology using standard semantics. The work presented in this paper focuses on this latter "repair strategy", with an assumed scenario that ontologies may become inconsistent during their construction process, but not wanting to publish such ontologies, their authors would repair the ontologies before publication.

Given the assumed "repair and publish" scenario, it is necessary for ontology authors to pin point the reasons for an inconsistent ontology. These reasons may be highly non-obvious, and manually browsing an ontology to determine the causes of inconsistency, if not impossible, can be a wretched, tedious and error prone task. In practice it is therefore necessary for people to have access to some form of automated explanation generation service that, given an inconsistent ontology, can identify *minimal subsets* of the ontology that cause it to be inconsistent. These minimal subsets are known as *justifications* [6,7,8], and a procedure for finding and exposing justifications for inconsistent ontologies is the subject of this paper.

In what follows, an algorithm for computing all justifications for inconsistent ontologies is presented. An investigation on real world, hand-crafted, ontologies is carried out to evaluate the effectiveness of the algorithm for computing justifications. Finally, the problems and challenges of computing justifications for inconsistent ontologies, along with the utility of these justifications, is discussed.

2 Preliminaries: OWL and Justifications

The work presented here focuses OWL 2 ontologies which correspond to $\mathcal{SROIQ}$ knowledge bases. $\mathcal{SROIQ}$ [3] is a highly expressive description logic and can be viewed as a decidable and 2NExpTime-complete [9] fragment of First Order Logic, a close relative of the 2-variable fragment with counting and modal logics [2]. In what follows,

- A and B are used as *class names*–they correspond in FOL to unary predicates,
- R and S are *properties*, i.e., *property names* or *inverse properties* (written R^- for R a property name)–they correspond to binary predicates,
- C and D are (possibly complex) *class expressions* that can be built using Boolean operators on classes ($\sqcap$, $\sqcup$, and $\neg$), value restrictions ($(\exists R.C)$, $(\forall R.C)$), number restrictions ($(\geq nR.C)$, $(\leq nR.C)$ for n a non-negative integer), and enumerations ($\{a\}$)–they correspond to formulae in one free variable,
- a and b are *individual names*–they correspond to constants.

A $\mathcal{SROIQ}$ *ontology* is a finite set of $\mathcal{SROIQ}$ *axioms*, which take the form of subsumptions between (possibly complex) class expressions $C \sqsubseteq D$ or properties $R \sqsubseteq S$, or the form of assertions $C(a)$ or $R(a, b)$.

A definition of the syntax and semantics of $\mathcal{SROIQ}$ can be found in [3]. In what follows a sketch is provided by a translation of the main class, property and axiom forming constructors into FOL. The translation of classes and properties is parametrised by two variables, x and y, and uses counting quantifiers.

$$\begin{aligned}
\pi(\mathcal{O}) &:= \bigwedge_{\alpha \in \mathcal{O}} \pi(\alpha), \\
\pi(C \sqsubseteq D) &:= \forall x.\pi_x(C) \Rightarrow \pi_x(D), \quad \pi(R \sqsubseteq S) := \forall x, y.\pi_{x,y}(R) \Rightarrow \pi_{x,y}(S), \\
\pi(C(a)) &:= \pi_a(C), \quad \pi(R(a, b)) := \pi_{a,b}(R), \\
\pi_{x,y}(R) &:= R(x, y), \quad \pi_{x,y}(R^-) := R(y, x), \\
\pi_x(A) &:= A(x), \quad \pi_x(\top) := \mathsf{true}, \quad \pi_x(\bot) := \mathsf{false}, \quad \pi_x(\{a\}) := (x = a), \\
\pi_x(C \sqcap D) &:= \pi_x(C) \wedge \pi_x(D), \quad \pi_x(C \sqcup D) := \pi_x(C) \vee \pi_x(D), \\
\pi_x(\neg C) &:= \neg \pi_x(C), \\
\pi_x(\exists R.C) &:= \exists y.\pi_{x,y}(R) \wedge \pi_y(C), \quad \pi_x(\forall R.C) := \forall y.\pi_{x,y}(R) \Rightarrow \pi_y(C), \\
\pi_x(\leq nR.C) &:= \exists^{\leq n} y.\pi_{x,y}(R) \wedge \pi_y(C), \quad \pi_x(\geq nR.C) := \exists^{\geq n} y.\pi_{x,y}(R) \wedge \pi_y(C)
\end{aligned}$$

Interpretations, models, and entailments are defined as usual in FOL. In particular, an interpretation $\mathcal{I}$ is said to be a *model of* an ontology $\mathcal{O}$ (written $\mathcal{I} \models \mathcal{O}$) iff $\mathcal{I} \models \pi(\alpha)$ for every axiom $\alpha \in \mathcal{O}$. A class expression C is *satisfiable* with respect to an ontology $\mathcal{O}$ if there exists a model $\mathcal{I}$ of $\mathcal{O}$ with $\mathcal{I} \models \exists x.\pi_x(C)$. An ontology is *inconsistent* if there does not exist a model of $\mathcal{O}$, i.e., if $\mathcal{O} \models \mathsf{false}(a)$. Given an ontology $\mathcal{O}$ and an axiom η, $\mathcal{O}$ entails η, written $\mathcal{O} \models \eta$ if, for every model $\mathcal{I}$ of $\mathcal{O}$, we have that $\mathcal{I} \models \pi(\eta)$.

Definition 1 (Justification). *Let $\mathcal{O}$ be an ontology such that $\mathcal{O} \models \eta$. $\mathcal{J}$ is a justification for η in $\mathcal{O}$ if $\mathcal{J} \subseteq \mathcal{O}$, and $\mathcal{J} \models \eta$, and for all $\mathcal{J}' \subsetneq \mathcal{J}$ $\mathcal{J}' \not\models \eta$.*

Intuitively, a justification for an entailment in an ontology is a minimal subset of the ontology that entails the entailment in question. A justification is a minimal subset in the sense that any proper subset of the justification does not entail the entailment in question. It is important to note that there may be multiple, potentially overlapping, justifications for any given entailment. Note that in the logics considered here, the number of justifications for an entailment in an ontology may be exponential in the size of the ontology [10].

Example: Let $\mathcal{O} = \{A \sqsubseteq B, B \sqsubseteq D, A \sqsubseteq \exists R.C, \exists R.\top \sqsubseteq D, A \sqsubseteq E\}$ so that $\mathcal{O} \models A \sqsubseteq D$. There are two justifications for $\mathcal{O} \models A \sqsubseteq D$, $\mathcal{J}_1 = \{A \sqsubseteq B, B \sqsubseteq D\}$ and $\mathcal{J}_2 = \{A \sqsubseteq \exists R.C, \exists R.\top \sqsubseteq D\}$.

Definition 2 (Simple Repair). *$\mathcal{S}$ is a simple repair for $\mathcal{O} \models \eta$ if $\mathcal{S} \subseteq \mathcal{O}$ and, for every justification $\mathcal{J}$ for $\mathcal{O} \models \eta$, $\mathcal{S} \cap \mathcal{J} \neq \emptyset$, and for all $\mathcal{S}' \subsetneq \mathcal{S}$ $(\mathcal{O} \setminus \mathcal{S}') \models \eta$*

Hence, in order to construct a repair for an entailment η in an ontology $\mathcal{O}$, it is necessary to choose one axiom from each justification for η in $\mathcal{O}$ and remove this axiom from $\mathcal{O}$. This implies that, in the general sense, all justifications for η in $\mathcal{O}$ are needed in order to construct a simple repair for $\mathcal{O} \models \eta$. The process of finding the justifications for entailments and repairing ontologies is frequently referred to as ontology *debugging*.

3 Related Work and Background

In recent years there has been a significant amount of effort and research into the explanation of entailments that arise in *consistent* ontologies. In a seminal paper in the area of debugging ontologies, Schlobach and Cornet [7] presented minimal unsatisfiable preserving sub-tboxes (MUPS) as a specific kind of justification for unsatisfiable classes in ontologies. Kalyanpur et al. coined the term justification as a form of explanation for any arbitrary entailment that holds in an ontology. In [11] it was shown that it is practical to compute *all* justifications for any given entailment in real, published (consistent) ontologies. The Propositional Logic reasoning community have long used the notion of *minimal conflict sets* for determining the minimal unsatisfiable subsets of a set of clauses for example in [12]. These conflict sets are used to gain insight into why a set of clauses is unsatisfiable. In essence minimal conflict sets are akin to justifications. On a practical level, justifications are used by many ontology development and debugging environments such as Protégé-4 [13], Swoop [14], and RaDON [15], as a popular form of explanation for entailments in ontologies.

3.1 Model Based Diagnosis and Techniques for Computing All Justifications

Techniques for computing all justifications for an entailment, and for computing all minimal conflict sets for a set of unsatisfiable clauses, borrow from techniques used in the field of *Model Based Diagnosis*. Model Based Diagnosis is the overarching name for the process of computing *diagnoses* for a *faulty system*. Briefly, a *diagnosis* is a subset minimal set of *components* from a faulty system, that if replaced would *repair* the system so that it functions as intended. The term *model* refers to the fact that a model is used to obtain the *predicted* behaviour of of a system, which is then compared the *observed* behaviour of the system. Any discrepancy between the predicted and observed behaviours indicates a fault in the system. Model based diagnosis techniques have a wide variety of applications, and for example, are typically used for diagnosing sets of faulty gates and connections in circuits. There is a direct correspondence between the ideas behind model based diagnosis, and the ideas behind justification based ontology debugging. Essentially, a diagnosis is the same as a repair (Definition 2), and *Minimal Conflict Sets*, which are subsets of the faulty system, are equivalent to justifications which are subsets of an ontology. A particularly pertinent

algorithm that is borrowed from the area of Model Based Diagnosis is "Reiter's Hitting Set Tree (HST) Algorithm" [16]. This algorithm essentially constructs a finite tree whose nodes are labelled with minimal conflict sets (justifications), and whose edges are labelled with components (axioms) from the system. In doing this the algorithm finds all minimal hitting sets for the conflict sets in a system. Since a minimal hitting set corresponds to a repair, and a conflict set corresponds to a justification, it is possible to use Reiter's algorithm to find all repairs, and in doing this find all justifications for an entailment in an ontology (proof in [11]).

Continuing with the previous example from Section 2, the hitting set tree algorithm is used to compute all justifications for $\mathcal{O} \models A \sqsubseteq D$ as follows. A subprocedure, computeJustification(η, $\mathcal{O}$), is used, which computes a single justification for the entailment η in an ontology $\mathcal{O}$. Suppose that $\mathcal{J}_1 = \{A \sqsubseteq B, B \sqsubseteq D\}$ is computed by computeJustification($A \sqsubseteq D$, $\mathcal{O}$). This justification is set as the label of the root of the tree. Next, one of the axioms from $\mathcal{J}_1$, say $A \sqsubseteq B$ is selected and removed from $\mathcal{O}$ to give $\mathcal{O}'$. This essentially repairs $\mathcal{J}_1$ and allows another justification (if one exists) to be computed for the entailment with respect to the modified ontology $\mathcal{O}'$. In this case, computeJustification($A \sqsubseteq D$, $\mathcal{O}'$) computes $\mathcal{J}_2 = \{A \sqsubseteq \exists R.C, \exists R.\top \sqsubseteq D\}$ as a justification. The hitting set tree is extended with an edge from the root, to a node labelled with $\mathcal{J}_2$ (the upper left most edge in Figure 3.1), and the edge is labelled with the axiom that was removed. Next, an axiom from $\mathcal{J}_2$, say $A \sqsubseteq \exists R.C$ is selected and removed from $\mathcal{O}'$ to give $\mathcal{O}''$, and computeJustification($A \sqsubseteq D$, $\mathcal{O}''$) is called to compute further justifications with respect to $\mathcal{O}''$. This time, no further justifications are found since the entailment does not hold in $\mathcal{O}''$, and the hitting set tree is extended with an edge from the node labelled with $\mathcal{J}_2$ to a node with an empty label which marks this branch as complete. The algorithm then backtracks up a level to remove a different axiom from $\mathcal{J}_2$ and the process continues. The algorithm terminates when all branches are marked as complete. It is noticeable, that the axioms labelling the edges in a complete branch form a repair, or a superset of a repair, since removing them from $\mathcal{O}$ results in $\mathcal{O}''$ such that $\mathcal{O}'' \not\models \eta$. A more formal description of this algorithm (with optimisations) is given later in Function-1R in Algorithm 1 (where the entailment that $\mathcal{O}$ is inconsistent is implicit) and is used in the computation of all justifications for inconsistent ontologies.

One of the issues with computing all justifications for an entailment is that the size of the hitting set tree can, in the worst case, be exponential in the size of the set of justifications. However, a suite of optimisations, detailed in [11] and presented in Algorithm 1, can be applied to prune the hitting set tree and minimise its size, so that for practical purposes this is rarely a problem. For example, in [11] it was shown that for real, published ontologies, the number of justifications per entailment is usually small—for the ontologies sampled in [11], the average number of justifications per entailment was two, with one ontology containing a maximum of twenty six justifications for an entailment.

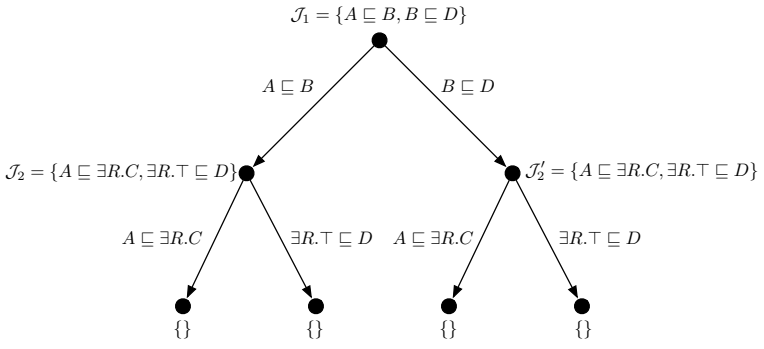


Fig. 1. An example of a hitting set tree (HST) that is constructed by Reiter's HST algorithm

3.2 Computing Single Justifications

The technique described in the previous section for computing *all* justifications for an entailment, uses a sub-procedure $\mathsf{computeJustification}(\eta, \mathcal{O})$ that computes a justification for an entailment η in an ontology $\mathcal{O}$. In this work, a black-box technique, which is not specific to a particular reasoner or proof strategy is used. Black-box techniques typically use an "expand-contract" strategy in combination with calls to an external reasoner. In order to compute a justification for an entailment η in an ontology $\mathcal{O}$, an initial, and small, subset $\mathcal{S}$ of axioms from $\mathcal{O}$ is selected based on the signature of η. The axioms in $\mathcal{S}$ are typically the axioms whose signature has a non-empty intersection with the signature of η, or axioms that "define"[2] terms in the signature of η. A reasoner is then used to check if $\mathcal{S} \models \eta$, if not, $\mathcal{S}$ is expanded by adding a few more axioms from $\mathcal{O}$. This process continues until $\mathcal{S}$ is large enough so that it entails η (this is the expansion phase). When a subset $\mathcal{S}$ of $\mathcal{O}$ that entails η is found, this subset is pruned until it is a minimal set of axioms that entails η i.e. until it is a justification (this is the contraction phase). Through careful selection of axioms in the expansion phase, which minimises the maximum size of $\mathcal{S}$, and various pruning techniques in the contraction phase, it has been found that this algorithm works well in practice for entailments in consistent ontologies [11].

4 A Black-Box Algorithm for Computing Justifications for Inconsistent Ontologies

As described in Section 3.2, the general approach taken by black-box justification finding algorithms is to use a two phase expand-contract procedure. The "seed" for selecting the initial small set axioms for the expansion phase is usually the signature of the entailment in question (the class, property, individual names

[2] For example, the axiom $A \sqsubseteq B$ defines the class name A.

appearing in the entailment). However, an issue when generating justifications for inconsistent ontologies using black-box techniques is "where to start". How should the initial set of axioms be selected? With inconsistent ontologies there is no seed signature that can be used as an input to start the expansion process. This is due to the fact that current OWL reasoners typically do not provide any information or clues as the possible causes of an inconsistent ontology—they just output that the ontology is inconsistent.[3] Therefore, the black-box algorithm that is presented here, consists of a very simple expansion phase, which immediately causes the expansion to contain *all* of the axioms in an ontology, followed by a "divide and conquer" contraction phase, which contracts the set of axioms in the ontology down to a justification. The algorithm for computing all justifications is a hybrid of the algorithm for computing single justifications taken from [17] and combined with the algorithm for computing all justifications (the HST algorithm) taken from [11]. Proof that the Hitting Set Tree algorithm finds all justifications is given in [11]. Algorithm 1 is the algorithm based on Reiter's Hitting Set Tree Algorithm and is used to compute all justifications for an inconsistent ontology. Algorithm 2 is used as a sub-procedure for Algorithm 1 for computing a justification for the inconsistent ontology.

Algorithm 1. ComputeAllJustifications

Function-1: ComputeAllJustifications($\mathcal{O}$)

1: $S, curpath, allpaths \leftarrow \emptyset$
2: ComputeAllJustificationsHST($\mathcal{O}$, S, $curpath$, $allpaths$)
3: **return** S

Function-1R: ComputeAllJustificationsHST($\mathcal{O}$, S, $curpath$, $allpaths$)

1: **for** $path \in allpaths$ **do**
2: **if** $curpath \supseteq path$ **then**
3: **return** //Path termination without consistency check
4: **if** IsConsistent($\mathcal{O}$) **then**
5: $allpaths \leftarrow allpath \cup \{curpath\}$
6: **return**
7: $J \leftarrow \emptyset$
8: **for** $s \in S$ **do**
9: **if** $s \cap path = \emptyset$ **then**
10: $J \leftarrow s$ //Justification reuse (saves recomputing a justification)
11: **if** $J = \emptyset$ **then**
12: $J \leftarrow$ ComputeSingleJustification($\mathcal{O}$)
13: $S \leftarrow S \cup \{J\}$
14: **for** $ax \in J$ **do**
15: $curpath \leftarrow curpaths \cup \{ax\}$
16: ComputeAllJustificationsHST($\mathcal{O} \setminus \{ax\}$, S, $curpath$, $allpaths$)

[3] The exception to this is the Pellet reasoner, which is able to output clash information—while this is a potential optimisation, it is not supported by all reasoners and is therefore not used here.

Algorithm 2. ComputeSingleJustification

Function-2: ComputeSingleJustification($\mathcal{O}$)

1: **return** ComputeSingleJustification($\emptyset$, $\mathcal{O}$)

Function-2R: ComputeSingleJustification(S, F)

1: **if** $|F| = 1$ **then**
2: **return**
3: $S_L, S_R \leftarrow \text{split}(F)$
4: **if** IsInconsistent($S \cup S_L$) **then**
5: **return** ComputeSingleJustification(S, S_L)
6: **if** IsInconsistent($S \cup S_R$) **then**
7: **return** ComputeSingleJustification(S, S_R)
8: $S'_L \leftarrow$ ComputeSingleJustification($S \cup S_R, S_L$)
9: $S'_R \leftarrow$ ComputeSingleJustification($S \cup S'_L, S_R$)
10: **return** $S'_L \cup S'_R$

5 Implementation and Evaluation

Algorithm 1, and its sub-routine Algorithm 2, were implemented in Java using the latest version of the OWL API [18]. The OWL API has excellent support for manipulating ontologies as sets of axioms and provides a common interface to various OWL reasoners. The algorithm was run on the set of ontologies shown in Table 1 using the FaCT++ [19], Pellet [20] and HermiT [21] reasoners. For each ontology, the time to perform a consistency check of the complete ontology was recorded. Then the time to compute a single justification for the inconsistency, and the time to compute as many justifications as possible within a time of 30 minutes was recorded.

5.1 Results Analysis

The ontologies shown in Table 1 are real ontologies in the sense that they were developed by end users using editing tools such as Protégé-4. Since it is rarely

Table 1. Inconsistent Ontologies Used In Experiments

ID	Ontology	Expressivity	Axioms	Cons. HermiT (ms)	Cons. Pellet (ms)	Cons. FaCT++ (ms)
1	Assignment4	$\mathcal{ALCIQ}$	88	-	1301	40
2	Boat	$\mathcal{ALCIF}(\mathcal{D})$	22	34	11	2
3	ComparaGrid	$\mathcal{ALCHIQ}(\mathcal{D})$	409	192	175	-
4	Country	$\mathcal{SROIQ}(\mathcal{D})$	5921	539	1454	333
5	Fish	$\mathcal{ALCH}$	49	36	78	13
6	IedbExport	$\mathcal{SHOIN}(\mathcal{D})$	2417	141	1117	67
7	Micro	$\mathcal{SROIQ}(\mathcal{D})$	1458	-	556	48
8	Spectro	$\mathcal{ALCON}(\mathcal{D})$	26612	1841	1763	378
9	Pizza	$\mathcal{SHOIQ}(\mathcal{D})$	179	25	45	5
10	ClassesAsValues	$\mathcal{ALCO}$	13	3	2	1
11	Travel	$\mathcal{SROIQ}(\mathcal{D})$	6437	443	701	135
12	Units	$\mathcal{ALCOIF}(\mathcal{D})$	2254	217	435	72

the case that inconsistent ontologies are published, all of the ontologies in the sample were obtained either from public mailing lists, where the ontologies had been posted to the mailing list with a request for help in debugging the ontology, or from ontologies that were directly received via email from colleagues and users of Protégé-4 and Swoop.

The rightmost three columns of Table 1 shows the time to load, and check the consistency of the ontologies using HermiT, Pellet and FaCT++. Even for the largest ontology, 8, that contains over twenty six thousand axioms, the consistency check is performed within two seconds using HermiT or Pellet and less that half a second using FaCT++. Note that FaCT++ failed to terminate on the ComparaGrid ontology, where it entered an infinite loop during preprocessing. HermiT incorrectly found Assignment4 consistent, and crashed on the Micro ontology. On the surface, it would appear that even for very large ontologies, the efficiency and speed at which ontologies can be loaded and checked for consistency seems to indicate the feasibility of using a reasoner as an oracle in a black-box based debugging algorithm.

Table 2 shows the performance results for computing justifications for inconsistent ontologies. Showing by ontology and reasoner, the time (ms) to compute one justification, the number of justifications found within 30 minutes, the time (ms) to compute all justifications if less than 30 minutes. A dash indicates that no justifications could be computed. A $\star$ indicates that the algorithm did not terminate within 30 minutes.

It should be noted that for all sampled ontologies, it was possible to compute *at least one* justification for the ontology being inconsistent using *at least one* of the three reasoners. With the exception of the Spectro ontology, it was possible to compute at least one justification *within ten seconds*. This is clearly acceptable performance in the context of generating on demand explanations in an ontology development environment such as Protégé-4. In the case of the Spectro ontology, Pellet was able to compute at least one justification within 11 seconds and HermiT within 31 seconds. Considering that this ontology contains over 26500 axioms, it is arguable that this is acceptable, and even impressive, performance for the purposes of interactive debugging and explanation in an ontology development environment.

It can be seen that for the Assignment4, Fish, Spectro, Travel, and Units ontologies, it was not possible to compute *all* justifications with any of the tested reasoners. However, for these ontologies, with the exception of Assignment4, a sizeable number of justifications could be computed. In all cases, the availability of one or more justifications can provide a vital insight into why an ontology is inconsistent, and allow a person to begin to repair the ontology. Not all justifications could be computed for all ontologies within 30 minutes. Broadly speaking, there are two reasons for this: (1) The reasoner being used found it *much* harder to check if *some subset of an ontology* was consistent, compared to checking if the *whole* ontology was consistent (in all cases, consistency checking performance for whole ontologies was good e.g. less than 2 seconds), and the running algorithm therefore "ground to a halt" during entailment checking. An exam-

Table 2. Times for computing single and all justifications

Ontology	Reasoner	Time for One (ms)	Number found in 30 mins.	Time for all (ms)
Assignment4	HermiT	-	-	⋆
	Pellet	-	-	⋆
	FaCT++	5711	1	⋆
Boat	HermiT	129	1	193
	Pellet	73	1	93
	FaCT++	11	1	41
ComparaGrid	HermiT	2252	9	15901
	Pellet	1515	9	13164
	FaCT++	-	-	⋆
Country	HermiT	4177	4	383963
	Pellet	4564	4	137348
	FaCT++	1726	4	78547
Fish	HermiT	134	162	⋆
	Pellet	-	-	⋆
	FaCT++	115	162	⋆
IedbExport	HermiT	855	2	2415
	Pellet	1257	2	3860
	FaCT++	765	2	2255
Micro	HermiT	4538	1	⋆
	Pellet	2326	1	59090
	FaCT++	809	1	1574
Spectro	HermiT	30930	76	⋆
	Pellet	10768	76	⋆
	FaCT++	-	-	⋆
Pizza	HermiT	114	3	592
	Pellet	7491	3	⋆
	FaCT++	37	3	329
ClassesAsValues	HermiT	21	1	23
	Pellet	4	2	15
	FaCT++	5	2	13
Travel	HermiT	7873	7	⋆
	Pellet	884	492	⋆
	FaCT++	521	163	⋆
Units	HermiT	3023	54	⋆
	Pellet	473	287	⋆
	FaCT++	285	287	⋆
Travel-semi-rep (semi repaired Travel ont.)	HermiT	8309	6	⋆
	Pellet	3975	6	25722
	FaCT++	1331	6	19280

ple of this is the Travel ontology, where using HermiT the procedure was only able to compute 7 justifications, but using Pellet, the procedure was able to compute nearly 500 justifications. (2) The number of justifications for the inconsistent ontology was very large. The runtime performance of the algorithm for computing justifications is in the worst case *exponential* with the number of justifications—computing all justifications is an inherently difficult problem.

6 Discussion

The usefulness of justifications as explanations for inconsistent ontologies. Despite the fact that, in *some cases*, it is not feasible to compute *all* justifications for real inconsistent ontologies, the ability to obtain just one or two justifications can lend a vital insight into why the ontology is inconsistent.

Some ontologies are inconsistent due to highly non-obvious reasons, and it is arguable that without automated explanation support, ontology authors would face a hopeless task of trying to figure out the reasons for an ontology being inconsistent in order to arrive at a manual repair. An example of a justification from the Country ontology is shown in Figure 2. The ontology authors, who made a request for help in trying to track down the reasons for the inconsistency, indicated that it was highly unlikely that they would have discovered the reason, without considerable difficulty, using manual debugging techniques.

Using justifications for inconsistent ontologies for repair. Even though it may not be possible in practice to compute all justifications for a given ontology, with the availability of at least one justification, it is still possible to carry out a manual interactive repair. For example, on closer inspection of the justifications for the travel ontology it was observed that many of them had the form $\{R(a, l), \top \sqsubseteq \forall R.C\}$, where l is a data value not in the value space of C[4]. In fact, structural inspection revealed that there were over 550 of these kinds of justifications for the travel ontology and over 12000 for the Spectro ontology. In order to test the hypothesis that even seeing a handful of justifications is helpful for interactive debugging and can support repair, the travel ontology was modified to produce a new version where all literals in property assertions were typed with the appropriate data type[5]. This new version of the ontology (Travel-semi-rep in Table 2) was still inconsistent, but had just six justifications for it being inconsistent. Thus, even though it may not be possible to compute all justifications is one go, being able to compute some justifications can provide enough insight in order to understand the reasons for an ontology being inconsistent so that is is possible to perform a manual repair.

The difficulty of finding *all* justifications for an inconsistent ontology compared to finding justifications for entailments in consistent ontologies. In other work [6,11] it has been shown that it is practical to compute *all* justifications for entailments in consistent ontologies using similar blackbox algorithms as the one presented here. However, the experiments carried out as part of this research clearly indicate that it is not possible to compute all justifications for some of the sampled inconsistent ontologies. As a means to understanding why this could be the case, consider the following *consistent* ontology $\mathcal{O} = \{B \sqsubseteq A, C \sqsubseteq A, D \sqsubseteq A, \dots\}$ such that B, C and D are entailed to be unsatisfiable. Suppose that there are ten justifications for each unsatisfiable class, meaning that there is a total of *thirty* justifications for the three unsatisfiable classes. In this case, *each entailment can be examined separately*, one after the other, so that the justifications for each entailment are computed separately. In other words, although the total number of justifications for all unsatisfiable classes is thirty, they are *computed in batches of ten*. Now consider $\mathcal{O}' = \mathcal{O} \cup \{B(a), C(b), D(c)\}$. Since B, C and D are unsatisfiable, $\mathcal{O}'$ is entailed to

[4] OWL 2 supports datatypes.

[5] Upon seeing the justifications for this ontology it was deemed that this was a natural repair.

$$\text{Island}(\text{ireland}) \tag{1}$$
$$\text{Island} \sqsubseteq \text{LandMass} \tag{2}$$
$$\text{LandMass} \sqsubseteq \text{GeographicalFeature} \sqcap \exists \text{hasCoastline}.\text{Coastline} \tag{3}$$
$$\text{GeographicalFeature} \sqsubseteq \text{NaturalPhysicalThing} \tag{4}$$
$$\text{NaturalPhysicalThing} \sqsubseteq \text{NaturalEntity} \tag{5}$$
$$\text{NaturalEntity} \sqsubseteq \text{PhysicalEntity} \tag{6}$$
$$\text{landBoundaryOf}(\text{unitedKingdomIrelandBorder}, \text{ireland}) \tag{7}$$
$$\text{landBoundaryOf} \sqsubseteq \text{hasLandBoundary}^{-} \tag{8}$$
$$\exists \text{hasLandBoundary}.\top \sqsubseteq \text{Country} \tag{9}$$
$$\text{Country} \sqsubseteq \text{AdministrativeDivision} \tag{10}$$
$$\text{AdministrativeDivision} \sqsubseteq \text{PoliticalEntity} \sqcap \exists \text{directPartOf}.\text{PoliticalDiv} \tag{11}$$
$$\text{PoliticalEntity} \sqsubseteq \neg \text{PhysicalEntity} \tag{12}$$

Fig. 2. An (small) example justification from the (large) country ontology. The individual ireland is entailed to be a PhysicalEntity (by axioms (1)(6)) and is also entailed to be a PoliticalEntity (by axioms (7)-(11)), PhysicalEntity and PoliticalEntity are disjoint with each other (axiom (12)).

be inconsistent, and there are thirty justifications for this entailment. Although the total number of justifications for the entailment of interest in $\mathcal{O}'$ is the same as the total number of justifications for the entailments of interest in $\mathcal{O}$, and the justifications are almost structurally the same, the key difference is that there are now *thirty* justifications for *one* entailment in $\mathcal{O}'$, compared to thirty justifications spread over three entailments in $\mathcal{O}$. Since the algorithm for computing all justifications for an entailment in an ontology is, in the *worst case*, exponential in the number of justifications for the entailment, it is clear to see that it can be *much* more time consuming to compute all justifications for an inconsistent ontology compared to computing all justifications for all entailments of interest in a consistent ontology. In essence, one of the factors that makes it practical to compute all justifications for entailments in consistent ontologies, is that for real ontologies, the average number of justifications per entailment tends to be fairly low. In the case of inconsistent ontologies, it is not possible, using classical reasoning, to take a more fine-grained approach, because it is not possible to recognise entailments other than the single entailment that the ontology is inconsistent.

7 Summary and Future Work

Computing a single justification, as a form of explanation, for real world inconsistent ontologies is possible in practice. However, computing *all* justifications is not possible in practice for some real world ontologies. This is primarily due to the fact that there can be a very large number of justifications for an ontology being inconsistent. Despite this, it is expected that having the ability to compute

just one or two justifications for an inconsistent ontology will be of enormous benefit to people trying to understand why the ontology is inconsistent. As future work, we plan to investigate the possibility of using clash information for further optimisation the current black-box algorithm. We also plan to investigate the use of paraconsistent reasoning as means for exploring justifications for meaningful entailments in inconsistent ontologies with a view to repairing these ontologies.

References

1. Horrocks, I., Patel-Schneider, P.F., van Harmelen, F.: From $\mathcal{SHIQ}$ and RDF to OWL: The making of a web ontology language. J. of Web Semantics (2003)
2. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F.: The Description Logic Handbook: Theory, Implementation and Applications (2003)
3. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible $\mathcal{SROIQ}$. In: KR 2006 (2006)
4. Dolby, J., Fokoue, A., Kalyanpur, A., Kershenbaum, A., Schonberg, E., Srinivas, K., Ma, L.: Scalable semantic retrieval through summarization and refinement. In: AAAI, pp. 299–304 (2007)
5. Ma, Y., Hitzler, P., Lin, Z.: Algorithms for paraconsistent reasoning with OWL. In: Franconi, E., Kifer, M., May, W. (eds.) ESWC 2007. LNCS, vol. 4519, pp. 399–413. Springer, Heidelberg (2007)
6. Kalyanpur, A.: Debugging and Repair of OWL Ontologies. PhD thesis, The Graduate School of the University of Maryland (2006)
7. Schlobach, S., Cornet, R.: Non-standard reasoning services for the debugging of description logic terminologies. In: IJCAI (2003)
8. Baader, F., Hollunder, B.: Embedding defaults into terminological knowledge representation formalisms. Journal of Automated Reasoning (1995)
9. Kazakov, Y.: $\mathcal{SRIQ}$ and $\mathcal{SROIQ}$ are harder than $\mathcal{SHOIQ}$. Description Logics (2008)
10. Baader, F., Peñaloza, R., Suntisrivaraporn, B.: Pinpointing in the description logic $\mathcal{EL}$. In: KI (2007)
11. Kalyanpur, A., Parsia, B., Horridge, M., Sirin, E.: Finding all justifications of OWL DL entailments. In: ISWC (2007)
12. Bailey, J., Stuckey, P.J.: Discovery of minimal unsatisfiable subsets of constraints using hitting set dualization. In: Hermenegildo, M.V., Cabeza, D. (eds.) PADL 2004. LNCS, vol. 3350, pp. 174–186. Springer, Heidelberg (2005)
13. Horridge, M., Tsarkov, D., Redmond, T.: Supporting early adoption of owl 1.1 with protégé-owl and fact++. In: OWL: Experiences and Directions, OWLED (2006)
14. Kalyanpur, A., Parsia, B., Hendler, J.: A tool for working with web ontologies. International Journal on Semantic Web and Information Systems 1 (January - March 2005)
15. Ji, Q., Haase, P., Qu, G., Hitzler, P., Stadtmoeller, S.: RaDON – repair and diagnosis in ontology networks. In: Aroyo, L., Traverso, P., Ciravegna, F., Cimiano, P., Heath, T., Hyvönen, E., Mizoguchi, R., Oren, E., Sabou, M., Simper, L.E. (eds.) ESWC 2009. LNCS, vol. 5554, pp. 863–867. Springer, Heidelberg (2009)
16. Reiter, R.: A theory of diagnosis from first principles. Artificial Intelligence 32, 57–95 (1987)
17. Baader, F., Suntisrivaraporn, B.: Debugging SNOMED CT using axiom pinpointing in the description logic $\mathcal{EL}^+$. In: KR-MED (2008)

18. Horridge, M., Bechhofer, S., Noppens, O.: Igniting the OWL 1.1 touch paper: The OWL API. In: OWLED (2007)
19. Tsarkov, D., Horrocks, I.: FaCT++ Description Logic Reasoner: System description. In: IJCAR (2006)
20. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A practical owl-dl reasoner. Journal of Web Semantics 5(2) (2007)
21. Motik, B., Shearer, R., Horrocks, I.: Optimized reasoning in description logics using hypertableaux. In: CADE 21 (2007)

On Improving the Scalability of Checking Satisfiability in Probabilistic Description Logics

Pavel Klinov and Bijan Parsia

The University of Manchester
Oxford Road, Manchester M13 9PL, UK
{pklinov,bparsia}@cs.man.ac.uk

Abstract. This paper presents an approach aimed at improving scalability reasoning in the probabilistic description logic P-$\mathcal{SHIQ}$(D). The satisfiability problem is currently intractable as the existing algorithm generates exponentially large linear optimization systems. To cope with this, we employ advanced optimization techniques that do not require the systems to be fully generated before starting to solve them. One such technique, namely, column generation, can be applied by exploiting the structure of the reasoning problems in P-$\mathcal{SHIQ}$(D) analogously to how it has been used for solving such large optimization problems as constrained shortest path, cutting stock and many others. We present the formulation of the linear systems for the satisfiability problems in the column generation form, describe methods for solving them, discuss their difference from the known column generation approaches to propositional probabilistic logic, and finally show preliminary experimental results that demonstrate a useful improvement of scalability.

1 Introduction

Description logics (DLs), especially those in the $\mathcal{SH}$ family, are the foundation for the popular logic based knowledge representation formalism, the Web Ontology Language (OWL). DLs have proven very effective in supporting large, complex bio-health ontologies, such as SNOMED-CT, Galen, and the NCI Thesaurus.

However, DLs are (decidable) fragments of first order logic and, as such, do not provide support for representing statistical knowledge or subjective confidence (or, for that matter, fuzzy or vague knowledge). Such knowledge is central to health care, even if we only consider mortality estimation.

There are many proposed formalisms for combining DLs with various sorts of uncertainty, although, to the best of our knowledge, none have been used for a production oriented ontology. We believe that this is due to two reasons: 1) there is comparatively little knowledge about how to use these formalisms effectively (or even, which are best suited for what purposes) and 2) there is a severe lack of tooling, in particular, there have been no sufficiently effective reasoners.

In our prior work [1,2], we tackled both of these problems for the probabilistic DL P-$\mathcal{SHIQ}$(D). On the reasoning side, we were able to develop an algorithm that localized the practical intractability of P-$\mathcal{SHIQ}$(D) entailment in a

L. Godo and A. Pugliese (Eds.): SUM 2009, LNAI 5785, pp. 138–149, 2009.

sub-problem, to wit, probabilistic satisfiability (PSAT) and presented a testing methodology for evaluating P-$\mathcal{SHIQ}$(D) reasoners. In this paper, we report on the progress we have made in taming PSAT, as well as compare our efforts to related work undertaken in the context of other probabilistic logics.

2 P-$\mathcal{SHIQ}$(D) Background

P-$\mathcal{SHIQ}$(D) [3,4] is a probabilistic generalization of the DL $\mathcal{SHIQ}$ [5]. It has a number of properties that make it an attractive formalism. First, it is very expressive: it supports probabilistic subsumptions between arbitrary $\mathcal{SHIQ}$ concepts and a certain class of probabilistic class assertions (but no form of probabilistic role assertions). Second, any $\mathcal{SHIQ}$ ontology can be used as a basis for a P-$\mathcal{SHIQ}$(D) ontology which facilitates transition from classical to probabilistic ontological models. Third, it combines probabilistic and default reasoning. This allows for a consistent treatment of exceptional individuals and subconcepts. Last but not least, its reasoning tasks are decidable.

P-$\mathcal{SHIQ}$(D) is extends the syntax of $\mathcal{SHIQ}$ with *conditional constraints* [6],that is, expressions of the form $(D|C)[l,u]$ where C and D are arbitrary $\mathcal{SHIQ}$ concepts. Conditional constraints can be used for representing uncertainty in both terminological (TBox) and assertional (ABox) knowledge. A probabilistic TBox (PTBox) is a 2-tuple $PT = (\mathcal{T}, \mathcal{P})$ where $\mathcal{T}$ is a $\mathcal{SHIQ}$ TBox and $\mathcal{P}$ is a finite set of default conditional constraints. Informally, a PTBox axiom $(D|C)[l,u]$ means that "generally, if a randomly chosen individual belongs to C, its probability of belonging to D is in $[l,u]$". A probabilistic ABox (PABox) is a finite set of strict conditional constraints pertaining to a single probabilistic individual o. All constraints in a PABox are of the restricted form $(D|\top)[l,u]$, and informally read as "o is a member of D with probability between $[l,u]$" [4].

The semantics of P-$\mathcal{SHIQ}$(D) is standardly given in terms of a *possible world* [4]. Let Φ be the set of all concepts that can be used in conditional constraints. Then a *world* is a subset of Φ. A world I is called *possible* if $\{a : C|C \in I\} \cup \{a : \neg C|C \notin I\}$ is satisfiable w.r.t. TBox for a fresh individual a. The set of all possible worlds with respect to Φ (the so-called *index set*) is denoted $\mathcal{I}_\Phi$. A world I satisfies a concept $C \in \Phi$ is denoted as $I \models C$ if $C \in I$. Satisfiability is inductively extended to complex concepts as usual.

A world I is said to be a model of a TBox axiom η denoted as $I \models \eta$ if $\{\eta\} \cup \{a : C|C \in I\} \cup \{a : \neg C|C \notin I\}$ is satisfiable for a fresh individual a. A world I is a model of a TBox T denoted as $I \models T$ if it is a model of all axioms of T. It has been shown that this way to providing semantics to DL is equivalent to the classical model-theoretic approach based on interpretation functions [4].

It is convenient to define probabilistic models in terms of possible worlds. A probabilistic interpretation Pr is a function $Pr : \mathcal{I}_\Phi \rightarrow [0,1]$ such that $\sum_{I \in \mathcal{I}_\Phi} Pr(I) = 1$. Pr is said to *satisfy* a TBox $\mathcal{T}$ denoted as $Pr \models \mathcal{T}$ iff $\forall I \in \mathcal{I}_\Phi, Pr(I) > 0 \Rightarrow I \models \mathcal{T}$. Next, probability of a concept $C \in \Phi$ denoted as $Pr(C)$ is defined as $\sum_{I \models C} Pr(I)$. $Pr(D|C)$ is used as an abbreviation for $Pr(C \cap D)/Pr(C)$ given $Pr(C) > 0$. A probabilistic interpretation Pr satisfies

a conditional constraint $(D|C)[l,u]$ (or Pr is a model of $(D|C)[l,u]$) denoted as $Pr \models (D|C)[l,u]$ iff $Pr(C) = 0$ or $Pr(D|C) \in [l,u]$. Finally, Pr satisfies a set of conditional constraints $\mathcal{P}$ iff it satisfies each of the constraints.

Reasoning in P-$\mathcal{SHIQ}$(D) can be expected to be computationally hard because the formalism is an extension of $\mathcal{SHIQ}$ which is EXPTIME-Complete in its main reasoning tasks [5]. Next we explain why although P-$\mathcal{SHIQ}$(D) reasoning procedures are still in EXPTIME, their implementations perform significantly worse in practice.

A PTBox is *satisfiable* if there exists a probabilistic interpretation that satisfies all the conditional constraints. Such an interpretation is a probability distribution built over the index set. As we see shortly typical size of index sets is the main reason of intractability.

Given PTBox $PT = (\mathcal{P}, \mathcal{T})$, PSAT is equivalent to solvability of the following system of linear inequalities (for each constraint $(C|D)[l,u]$ in $\mathcal{P}$) [4]:

$$\frac{\displaystyle\sum_{I \in I_\Phi, I \models \neg D \sqcap C} -l x_I + \sum_{I \in I_\Phi, I \models D \sqcap C} (1-l) x_I \geq 0}{\displaystyle\sum_{I \in I_\Phi, I \models \neg D \sqcap C} u x_I + \sum_{I \in I_\Phi, I \models D \sqcap C} (u-1) x_I \geq 0} \qquad (1)$$
$$\frac{}{x_I \geq 0, \displaystyle\sum_{I \in I_\Phi,} x_I = 1}$$

It is easy to see that the number of variables (x_I) in the linear system (1) is equal to the size of the index set which is exponential in the number of concepts that appear in conditional constraints. This is the main reason why PSAT is intractable. Experimental studies show that even for small ontologies (i.e. fewer than 20 constraints) the linear systems become unmanageable [1].

3 Incremental Construction of Linear Systems

The difficulty of having an unmanageable number of variables in linear systems is well known and is faced in many important problems reduced to linear programming, e.g., cutting stock. One very successful approach to coping with this problem is *column generation* (CG) [7]. CG is based on the fundamental property of linear programming which says that if the optimal solution is feasible then it can be achieved with only a small subset of variables, the so-called *basic* variables, being positive. All other variables can be safely restricted to zero.

In the context of P-$\mathcal{SHIQ}$(D) this forms a basis of the *small model theorem* which states that a PTBox is satisfiable (i.e. has *some* model) if and only if it has a *small* model, i.e. a model of polynomial size [6]. In other words, for a satisfiable PTBox there always exists a satisfying probability distribution built over only a small subset of the index set. All other index set items do not need to be generated as they will be represented by non-basic variables.

It is therefore highly desirable to compute only those parts of the index set that are sufficient for constructing a satisfying model. Our strategy, which is based on column generation, is to compute index sets *incrementally* trying to ensure that the desirable subset will be found reasonably soon.

The key aspect of employing column generation is defining and solving the problem of generating new columns which often appears to be an extra optimization problem (the so-called pricing-out problem (POP)). Its constraints must ensure that each produced solution is, in fact, a column in the context of the original linear system. As long as each variable in (1)[1] corresponds to a possible world, POP should be formulated in such a way that the set of generated columns can be put in one-to-one correspondence with the index set.

As defined in Section 2, each possible world can be represented as a satisfiable conjunctive concept expression in $\mathcal{SHIQ}$. Thus each generated column must satisfy two conditions: First, it should correspond to a certain conjunctive class expression. Second, that class expression must be satisfiable given the original TBox. Then and only then the column is *valid* (i.e. represents a member of I_Φ).

In order to enforce the above conditions we should look at the possible values of column components in (1). Rows of the matrix represent conditional constraints (each constraint is described by two inequalities). Each component of a generated column ($a_{i,j}$) should correspond to some constraint $(D|C)[l,u] \in P$. This observation allows us to restrict the values that $a_{i,j}$ can take on. It follows from (1) that if i^{th} component corresponds to the lower (resp. upper) bound inequality then domain of $a_{i,j}$ is $\{0, -l, 1-l\}$ (resp. $\{0, u, u-1\}$).

Furthermore, a closer inspection of (1) reveals that since expressions $\neg D \sqcap C$ and $D \sqcap C$ are disjoint, no I can be a subclass of both. That leads to the following binary extensional constraints on pairs of consecutive components:

$$R_{i,i+1} = \{\{0,0\}, \{-l,u\}, \{1-l, u-1\}\} \quad \text{for } i = 0, 2, \ldots, m.\ m = 2 \times |\mathcal{P}| \tag{2}$$

The constraints (2) guarantee that each column produced by solving POP will correspond to a *world*. The transformation (which we call η) of columns into $\mathcal{SHIQ}$ class expressions is straightforward: it proceeds by examining each pair of column's components and transforming it into conjunctions of literals. For example, if 0^{th} and 1^{st} components of a column are $-0.4, 0.6$ and these (0^{th} and 1^{st}) inequalities correspond to the conditional constraint $(X|Y)[0.4, 0.6]$ then the conjunct $\neg X \sqcap Y$ will be added to the class expression. The process terminates in time that is linear in the number of pairs (i.e., the number of conditional constraints - m). More formally:

Definition 1. *$\eta : \mathcal{C} \rightarrow \Phi$, where $\mathcal{C}$ is a set of all possible columns and Φ is a set of all (not necessarily possible) worlds, is a function that transforms a generated column into a conjunctive $\mathcal{SHIQ}$ class expression. It is defined as $\eta(c) = \sqcap_i(I_i)$*

[1] In fact, the system of inequalities 1 can always be transformed into a linear programming instance which always admits a solution. This makes column generation easier to apply. We skip the details for brevity.

where each index i is associated with either lower bound or upper bound linear inequality corresponding to some conditional constraint $(D|C)[l, u]$. If it is lower (resp. upper) bound inequality then I_i is defined as:

- $c[i] = -l$ *(resp.* u*) then* $I_i = \neg D \sqcap C$
- $c[i] = 1 - l$ *(resp.* $u - 1$*) then* $I_i = D \sqcap C$
- *Otherwise* $I_i = \top$

This makes η is a well-defined total function on the set of all columns.

The constraints (2) are necessary but not sufficient. They do not ensure that the corresponding class expression will be satisfiable w.r.t. TBox (or, equivalently, that the world will be *possible*). For example, if there are conditional constraints $(X|Y)[0.4, 0.6]$ and $(X|Z)[0.8, 1]$ then a column might be generated that will correspond to $\ldots \sqcap (X \sqcap Y) \sqcap \ldots \sqcap (\neg X \sqcap Y) \sqcap \ldots$. Clearly this expression is unsatisfiable so the column is invalid (it does not represent a variable in (1)).

In order to explicitly prohibit generation of invalid columns we must add an extra constraint (call it R^*) to the POP:

$$\forall c \in \mathcal{C}, \eta(c) \text{ must be satisfiable w.r.t. the original TBox} \tag{3}$$

Now we can give the formulation of the POP for PSAT column generation:

Definition 2. *The price-out problem for PSAT is a tuple $\langle X, D, C, F \rangle$, where:*

- *X is a set of variables where each variable represents a component of the column being generated.*
- *D is a set of variable domains. Each domain is associated with a conditional constraint $(D|C)[l, u]$ and contains values $\{0, -l, 1 - l\}$ or $\{0, u, u - 1\}$.*
- *C is a set of constraints. They include constraints $\{R_{i,i+1}\}$ of the form (2) and extra constraint R^* (3) that ensures satisfiability.*
- *$F = \sum w_i x_i$ is a linear objective function where $\{w_i\}$ are known coefficients.*

The goal is to minimize F subject to the constraints.

Note that in this formulation POP represents a *constraint optimization problem*. Importantly, its size is linear in the number of conditional constraints.

3.1 Constraint Optimization for Column Generation

Constraint optimization problems (COP) can be seen as generalizations of constraint satisfaction problems (CSP) that in addition to satisfying constraints also seek for a solution that optimizes a cost function. It is important to understand the class of COPs that the problem (2) falls into in order to select the right method for solving it.

Clearly POP is a *combinatorial optimization problem* since the domains of all variables are discrete. Its feasible solution space is discrete and finite (the upper bound is 3^{2m} since columns are $2m$-vectors and each domain consists of 3 elements). It can formulated in the weighted Max-CSP form with multiple hard constraints and one soft constraint (representing the objective function).

A classical method for solving such COP problems is based on *Branch-and-Bound* search. Solvers that implement this method proceed by assigning values to variables in the depth-first manner. On each step they compute the lower (or upper) *bound* of the objective function based on the current partial assignment and *backtrack* if the estimated value is guaranteed to be worse than previously found. The critically important factor for this search is the ability to compute good (i.e., tight) objective bounds which allows pruning of search subspaces.

In order to apply Branch-and-Bound or similar search methods we need to be able to evaluate constraints. This is tricky for the satisfiability constraint (3) because transforming each partial assignment into a $\mathcal{SHIQ}$ constraint and solving SAT is a very costly operation. Moreover it is unclear how this constraint can be propagated for ensuring local consistency during the search.

```
   Input: W - objective coefficients, T - TBox, {R_{i,i+1}} - initial constraints
   Output: Valid improving column c or null
 1 POP = initialize_POP_problem(R, W)
 2 while True do
 3 |   c := solve(POP);
 4 |   if c != null && reduced_cost(c) < 0 then
 5 |   |   if SAT(η(c), T) then
 6 |   |   |   return c;                  // Column is improving and valid
 7 |   |   end
 8 |   |   else
   |   |   |   // Column is improving but invalid
 9 |   |   |   j := compute_justifications(η(c), T);
10 |   |   |   prohibit_assignments(POP, j);
11 |   |   end
12 |   end
13 |   else
14 |   |   return null;            // There is no improving column, stop
15 |   end
16 end
```

Algorithm 1. Column generation algorithm

Instead we propose an alternative approach (see Algorithm 1). The idea is to generate a valid column in several optimization steps by gradually adding constraints that will eventually ensure satisfiability (or lead to the conclusion that no valid column is an improving one). The algorithm works as follows:

1. Initially the POP contains only constraints $R_{i,i+1}$ (line 1).
2. COP solver solves the POP (line 3).
3. If there exists an improving column then its validity is checked, otherwise *null* is returned (line 4).

4. The column is transformed into a $\mathcal{SHIQ}$ concept expression which is checked for satisfiability w.r.t. TBox (line 5).
5. If the expression is satisfiable then the column is returned (line 6).
6. Add extra constraints that prohibit all partial variable assignments responsible for the unsatisfiability (lines 9 and 10).
7. Step 2 is repeated.

Since search space is finite and the algorithm can never generate the same column twice it is guaranteed to terminate.

The most interesting step here is 6. Recall that each $\mathcal{SHIQ}$ concept expression is a set of conjuncts. Therefore any unsatisfiability can be *justified* by minimal sets of conflicting conjuncts. These are analogous to *precise justifications* in OWL [8] in the sense that removal of any conjunct from all of the subsets is sufficient to make the expression satisfiable. Each minimal subset of conjuncts (a justification) corresponds to the assignment of values to a subset of column's variables. Thus prohibiting such assignment is equivalent to ensuring that any generated column will not be invalid due to *that* particular justification. The goal of the entire process is to capture all reasons of potential unsatisfiabilities in terms of lists of prohibited assignments to ensure that any generated column will be valid. In other words, the constraint R^* is handled as a set of extensional n-ary constraints prohibiting assignments that make the current column invalid.

The algorithm involves two NP-hard subproblems: solving constraint optimization problem POP (step 2) and computing all precise justifications (step 6). Unfortunately OWL reasoners currently do not provide efficient means for obtaining precise justifications for long conjunctive concept expressions thus we employ an ad hoc pin-pointing method optimized for conjunctions of literals as opposed to axioms. This may become a bottleneck when dealing with very long expressions which suggests that better methods are needed.

Now we briefly present constraint optimization method used to solve POP. As mentioned earlier, the simplest method is Branch-and-Bound however it turns out to be inefficient in this case due to the presence of prohibited assignments. Such constraints significantly complicate computing of the lower bound because, in general, their objective values cannot be estimated until they are fully assigned. This makes them very different from normal extensional constraints which can be easily propagated using local consistency enforcing methods.

Instead our solver is based on the Russian Doll Search algorithm which solves the problem by splitting it onto a series of nested subproblems [9]. The idea of classical Russian Doll Search (RDS) is to start with the subproblem having only the last variable, then proceed to the subproblem having two last variables and so on, so that i^{th} subproblem contains variables from $n - i + 1$ to the last. Finally the n^{th} subproblem is equivalent to the entire problem. The idea of solving n problems instead of one might appear illogical but, in fact, given that the variable ordering is fixed the solver is able to use results of previously solved subproblems to compute better lower bounds for the current subproblems. This approach is known to be beneficial on overconstrained problems (which are also the hardest) which was the reason for us to choose it.

4 Comparison with Related Work

The intractability of the linear programming approach to probabilistic satisfiability has been tackled with two notable approaches: local inference methods, which attempt to minimize the size of the linear systems by clever construction methods, and methods based on the incremental construction the linear systems.

Local inference means that inference rules which consider only subsets of probabilistic knowledge are applied to obtain probability bounds. Unfortunately, resulting reasoning procedures are almost always globally incomplete. Even worse, it has been shown that no globally complete set of rules exists for *any* logic that allows conjunctive expressions (events) in probabilistic formulas [10]. Finally, local methods cannot ensure *any* fixed precision in general [11].

The goal of methods based on an informed construction of linear systems is to avoid unnecessary variables. This approach works well in specific cases, for example, linear systems for *chains* of probabilistic formulas will always have polynomial size [12]. However, in general polynomial size cannot be guaranteed, for instance, the above technique will generate exponential linear systems if constraints of the form $(C|\top)[l,u]$ are added to the chain. Since PABox constraints are always of that form, the method cannot be applied to P-$\mathcal{SHIQ}$(D).

The closest in spirit to our approach is the approach by Kavvadias and Papadimitriou [13], and Jaumard, Hansen and de Aragão [14]. Similar to them we employ column generation methods of build linear systems incrementally. Their methods allowed them to solve PSAT for propositional probabilistic knowledges bases of substantial size (up to 300 unconditional formulas over 140 atoms).

Nevertheless there are substantial differences between our column generation method and the one used by Jaumard et al. [14]. The key difference is that they were able to encode the entire knowledge base into specific optimization problems (nonlinear 0-1 programs) solved to generate columns. While this is arguably more efficient than our generate-and-validate approach based on constraint graphs (see Algorithm 1) it is less generic and has some limitations. Most importantly, it assumes that the entire knowledge base is just a set of propositional probabilistic formulas. Obviously, DL knowledge bases are not propositional. While we might hope to reduce our DL knowledge bases to a propositional one in order to apply Jaumard's method, it's clear that any straightforward reduction will involve an exponential blow up. Part of our future work is to investigate whether this blow up can be mitigated in practice and whether the subsequent application of this method fares better than our more generic approach.

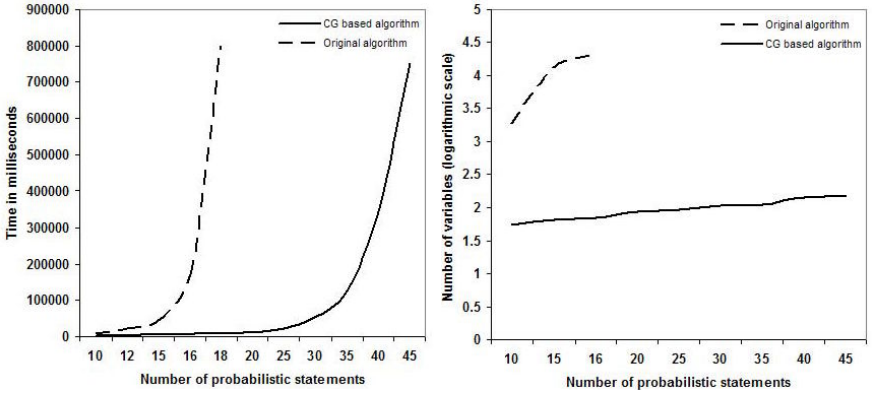
5 Experimental Results

We have conducted two series of experiments on different datasets. The first series is based on random sampling of the previously created BRCA ontology [2,1]. In the other series we use random P-$\mathcal{SHIQ}$(D) generated as in the experiments of Jaumard et al. [14]. The time limit was 10 minutes. We increased the problem size as long as the ratio of successfully solved instances stayed above 50% over 10 runs. The test machine was a laptop with 2.13MHz CPU and 1G RAM.

5.1 Subsets of the BRCA Ontology

We start with the experiments on random fragments of the BRCA ontology, which is, to the best of our knowledge, the only public realistic P-$\mathcal{SHIQ}$(D) ontology. Random samples are generated following our recently developed methodology [1]. We vary their size and measure the time and space requirements of the PSAT algorithms. The former is measured in milliseconds while the latter is defined as the number of variables (columns) of the largest linear system constructed during the solving process. (We attend to space as the original PSAT algorithm is known to run out of space much sooner than out of time.)

Results for the original and the CG based algorithms are shown on figures 1a and 1b. There are couple of important things to note. First, the CG based algorithm requires drastically less memory since it does not need to generate the entire linear system to check its solvability. Furthermore, the amount of needed space increases very gently with the size of the problem. As a consequence, it is capable of solving problems that are beyond the limit of the original algorithm. Finally, it runs faster even on those problems that the original algorithm can handle because it requires considerably fewer $\mathcal{SHIQ}$ SAT tests (recall from Section 2 that the number of SAT tests corresponds to the number of variables).



(a) (b)

Fig. 1. a) Time results for BRCA ontology b) Space results for BRCA ontology (logarithmic scale)

5.2 Random P-$\mathcal{SHIQ}$(D) Ontologies and Modularity

Next we present results obtained on random P-$\mathcal{SHIQ}$(D) ontologies, i.e. where both classical and probabilistic parts are randomly generated. The generation procedure follows the one used in [14]. First, we generate conjunctive concept expressions with random distribution over length (from one to four literals). Second, we generate conditional constraints using names assigned to previously generated concept expressions. The ratio of the number of atomic concepts to

the number of concept expressions was kept near 0.75. Since satisfiability is typically harder to prove than unsatisfiability we make a special effort to generate satisfiable sets of constraints. As in [14], we generate random vectors of possible world with random probability distribution over them (basically probabilistic models) and then set probabilities to conditional constraints based on those models. This guarantees that a satisfying probability distribution will exist.

The results are represented on Figure 2. Figure 2a shows results for unconditional constraints (i.e. with $\top$ used as evidence) which allows us to compare our results to those of Jaumard's while Figure 2b shows more realistic results involving conditional constraints. We note that the results are similar to the BRCA case except that the unconditional knowledge bases are substantially easier. It was also observed (although not depicted due to the lack of space) that PSAT tends to be easier as the number of atoms increases. The reason is that more atoms generally means fewer relationships between concepts in conditional constraints. Therefore the POP problem used to generate columns becomes easier because there are fewer constraints in the constraint graph (since constraints are direct consequences of subsumption/disjointness relationships). Interestingly the increase in the number of atoms is a negative factor for both the original PSAT algorithm and the Jaumard's method (see [14]).

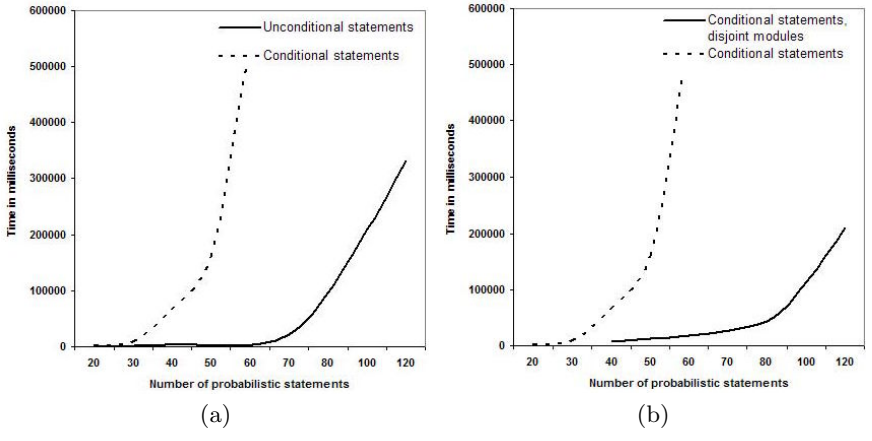


Fig. 2. a) Time results for ontologies with random conditional and unconditional statements b) Time results for ontologies with and without disjoint modules

We conclude that while our algorithm for P-$\mathcal{SHIQ}$(D) is by far superior to the original one, it is still less scalable than the algorithm for propositional PSAT esp. since our hardware is orders of magnitude faster than used by Jaumard et al. in 1991. As mentioned above, the difference probably comes from the fact that their column generation problem belongs to a more tractable and specific class of optimization problems than POP. It remains to be seen whether this advantage will hold in non-propositional cases.

The results of random PSATs suggest that the CG algorithm is implicitly exploiting internal structure of PTBox and, thus, if we can encourage more appropriately structured PTBoxes we might see better performance. For example, assuming that PTBox contains conditional constraints on concepts that come from disjoint modules with signatures, what will the impact on scalability be? Such disjointness seems related to the concept of *independence* which plays rather fundamental role in many probabilistic formalisms but not yet in P-$\mathcal{SHIQ}$(D). It's possible that introducing independence to P-$\mathcal{SHIQ}$(D) will not only allow modelers more expressivity, but improve the typical performance of reasoners.

The experimental setup remains almost the same except that two sets of random concept expressions with disjoint signatures are generated. One half of conditional constraints are generated on concepts from the first set and another—from the second. This corresponds to having a combined P-$\mathcal{SHIQ}$(D) ontology containing strictly independent (in all possible senses) modules. Also, only conditional probabilistic formulas are used.

Figure 2b compares the performance of the CG algorithm on random modular and non-modular ontologies. Indeed the performance and scalability are by far better on modular ontologies. The improvements come from the fact that there can never be constraints in the POP whose scope includes variables that correspond to unrelated conditional constraints. Given that conditional constraints built on concepts from two hierarchies are totally unrelated, we can always decompose any POP instance onto two sub-instances that can be solved independently. It follows that the complexity of generating each column is now equal to the complexity of a POP of half the size.

It may seem that given that columns can be generated exponentially faster the positive impact on PSAT performance should be more significant. However, in addition to the column generation there is also an NP-hard column validation step which amounts to computing all justifications of concept unsatisfiability. Unfortunately, as we observed during the experiments, the latter dominates. This suggests that it might be needed to tune the algorithm for computing laconic justifications to the kind of expressions arising in Algorithm 1.

6 Conclusion

In this paper we proposed an approach to coping with exponential size of linear systems generated by the probabilistic satisfiability algorithm for P-$\mathcal{SHIQ}$(D). It is substantially different from the known applications of column generation to the satisfiability problem in propositional probabilistic logic. It does not require to encode the entire classical and probabilistic knowledge in the optimization problem solved to generate columns. As a consequence, it can be used with ontologies that contain very extensive bodies of classical knowledge, e.g. the NCI Thesaurus augmented with probabilistic statements.

Our approach has a number of advantages over the original PSAT algorithm [4]. The most important one is that it reduces the problem of managing exponentially large linear systems to a constraint optimization problem of generating

columns. The difference is that the latter is restricted in its size and better investigated with a lot of different methods already developed. Our experiments demonstrate substantial improvements in reasoning performance and exponential space savings.

Another advantage is related to exploiting potential modularity of P-$\mathcal{SHIQ}$(D) ontologies. The experiments show that our algorithm performs substantially better on ontologies that contain disjoint probabilistic modules while the original algorithm cannot cope with modularity at all. This fact may encourage modular design of P-$\mathcal{SHIQ}$(D) ontologies and further research on introducing independence into the language.

References

1. Klinov, P., Parsia, B.: Optimization and evaluation of reasoning in probabilistic description logic: Towards a systematic approach. In: International Semantic Web Conference, pp. 213–228 (2008)
2. Klinov, P., Parsia, B.: Probabilistic modeling and OWL: A user oriented introduction into P-$\mathcal{SHIQ}$(D). In: OWL: Experiences and Directions (2008)
3. Giugno, R., Lukasiewicz, T.: $P{-}\mathcal{SHOQ}(D)$: A probabilistic extension of $\mathcal{SHOQ}$(D) for probabilistic ontologies in the semantic web. Technical Report Nr. 1843-02-06, Institut fur Informationssysteme, Technische Universitat Wien (2002)
4. Lukasiewicz, T.: Expressive probabilistic description logics. Artificial Intelligence 172(6-7), 852–883 (2008)
5. Horrocks, I., Sattler, U., Tobies, S.: Practical reasoning for very expressive description logics. Journal of the IGPL 8(3) (2000)
6. Lukasiewicz, T.: Probabilistic logic programming with conditional constraints. ACM Transactions on Computational Logic 2(3), 289–339 (2001)
7. Gilmore, P.C., Gomory, R.E.: A linear programming approach to the cutting stock problem. Operations Research 9, 849–859 (1961)
8. Horridge, M., Parsia, B., Sattler, U.: Laconic and precise justifications in OWL. In: International Semantic Web Conference, pp. 323–338 (2008)
9. Verfaillie, G., Lematre, M., Schiex, T.: Russian doll search for solving constraint optimization problems. In: Advances in Artificial Intelligence Conference, pp. 181–187 (1996)
10. Lukasiewicz, T.: Probabilistic deduction with conditional constraints over basic events. Journal of Artificial Intelligence Research 10, 199–241 (1999)
11. Lukasiewicz, T.: Local probabilistic deduction from taxonomic and probabilistic knowledge-bases over conjunctive events. International Journal of Approximate Reasoning 21(1), 23–61 (1999)
12. Lukasiewicz, T.: Efficient global probabilistic deduction from taxonomic and probabilistic knowledge-bases over conjunctive events. In: 6th International Conference on Information and Knowledge Management, pp. 75–82. ACM Press, New York (1997)
13. Kavvadias, D.J., Papadimitriou, C.H.: A linear programming approach to reasoning about probabilities. Annals of Mathematics and Artificial Intelligence 1, 189–205 (1990)
14. Jaumard, B., Hansen, P., Aragão, M.P.d.: Column generation methods for probabilistic logic. In: Integer Programming and Combinatorial Optimization Conference, pp. 313–331 (1990)

Towards Relational Schema Uncertainty

Matteo Magnani[1] and Danilo Montesi[2]

[1] University of Bologna, Italy
matteo.magnani@cs.unibo.it
[2] University of Bologna, Italy
danilo.montesi@unibo.it

Abstract. In this paper we introduce the problem of managing uncertainty on the schema of a relational database. We formalize this problem using a possible world semantics, identify the challenges posed by the co-existence of many alternative schemata, and define some basic properties of a system with uncertain schemata. Finally, we describe a query rewriting system, i.e., the technique used to produce valid queries for each alternative schema, and study its properties.

Keywords: schema, uncertainty, possible worlds, soundness, completeness.

1 Introduction

Uncertainty is a state of limited knowledge about a past, current or future state of the world. Uncertainty plays a fundamental role in many disciplines, like physics (e.g., Heisenberg's uncertainty principle), finance (prediction of future stock prices), insurance (determination of insurance rates based on the likelihood of events), and in several fields of computer science, like artificial intelligence and database management. In the following, we will focus on the management of **uncertain databases**.

Uncertainty has been studied in the field of data management since the early Nineties, and during the last half-decade there have been significant research efforts to develop working systems to manage uncertain data. As an example, consider the relation illustrated in Figure 1: it expresses uncertainty about Charles' age, which can be 42 (with a probability of .3) or 43. Similarly, we are uncertain about Herman's surname. In both cases uncertainty affects values, e.g., we are not sure about Charles' age, but *we are sure that 42 and 43 represent ages*: there is no uncertainty on the metadata.

PERSON

Name	Surname	Age
Charles	Dickens	42: .3 || 43: .7
Herman	Melville: .5 || Melvil: .5	54
Emily	Brontë	25

Fig. 1. A relation with a certain schema containing uncertain data. The symbol || separates alternative choices, each one annotated with its probability.

L. Godo and A. Pugliese (Eds.): SUM 2009, LNAI 5785, pp. 150–164, 2009.

In general, a relational database can be modeled as a pair (S, I), where S is the schema (or intension) and I is the instance (or extension). In the previous example, PERSON(Name, Surname, Age) is the schema and (Emily, Brontë, 25) is part of the instance. So far uncertain data management systems have been developed to represent and manipulate *uncertain instances*: attributes or tuples are annotated with probabilities (or possibility/necessity degrees) and, in some models, with additional information to represent their probabilistic dependencies. However, uncertainty may also affect the metadata: in this work we provide the first results concerning *uncertainty management on schemata*. This problem is particularly relevant with regard to the topic of scalable uncertainty management, because it can be caused by the presence of large amounts of data, and the consequent need for automated analysis techniques, as shown in the following motivating scenarios.

1.1 Motivating Scenarios

Uncertain instances have many possible application fields, e.g., sensor and scientific databases, and for this reason they have been object of almost all studies on uncertain data so far. However, recent work on data integration has shown that when we want to merge different data sources, in general we cannot produce a single mediated schema, because of the uncertainty generated during the comparison of the data sources [1,2]. Data integration is one of the most relevant and studied problems in the field of database management, with applications ranging from company information system merging and data warehousing to Web and scientific meta-search engines. Therefore, data integration activities affect a very large number of databases, with a consequent very large number of potential applications of uncertain schemata.

As an example, consider Figure 2, where we have illustrated two schemata P1 and P2, each one made of a single table. If we want to merge these schemata, we must decide if `Office Phone` and `Home Phone` are two distinct concepts or not, and depending on this choice the integrated schema will look like the one represented on the left hand side of Figure 3 (Name, Office Phone, Home Phone) or the one on the right hand side (Name, Phone). This example shows how data integration activities may generate uncertainty in the form of alternative schemata.

P1

Name	Office phone
Charles	01-2345
Herman	01-3456

P2

Name	Home phone
Herman	01-4567
Emily	01-5678

Fig. 2. Two local data sources, P1 and P2

Although data integration is certainly a very important application field of uncertain schemata, we may find uncertain schemata also in centralized systems, and in particular inside legacy systems. If a database is not well documented, after many years of activity, or when its administrators change, there may be a decreasing understanding of its

P1⊕P2 (1)

Name	Office phone	Home phone
Charles	01-2345	
Herman	01-3456	01-4567
Emily		01-5678

P1⊕P2 (2)

Name	Phone
Charles	01-2345
Herman	01-3456
Herman	01-4567
Emily	01-5678

Fig. 3. Two alternative ways to merge the data sources P1 and P2

content. For example, if we find a column `telephone`, will it contain private or public (office) telephone numbers? This uncertainty does not necessarily affect only the data (we may know telephone numbers with certainty), but may also concern the metadata, i.e., the schema, and can be modeled considering different schemata with alternative semantics.

As another example, consider a hospital database where one column of a table with patient records contains dates, and its name is just `Date`. Are these hospitalization dates? Do they indicate the date of an exam? In this case, we have uncertainty on the schema, and alternative schemata can be used to impose different semantics to the data.

1.2 Possible World Semantics and Its Implications on User Interaction

As we have aforementioned, when there is no uncertainty a database can be seen as a pair (S, I). To extract information from it we can ask a query defined on schema S (notated q^S) and produce a new database $q^S(S, I) = (S^{q^S}, I^{q^S})$. For example, we may ask a relational query:

$$\pi_{\text{Office phone}}(\sigma_{\text{Name='Charles'}}(\text{P1}))$$

on the relation P1 represented in Figure 2, obtaining a single relation with one attribute `Office phone` containing the value `01-2345`.

When a database contains uncertain information, like in Figure 1, this denotes the contemporary existence of multiple possible worlds. For example, the tuple:

- $(\text{Charles}, \text{Dickens}, 42||43)$

corresponds to two possible databases (or worlds), one where Charles is 42 years old and one where he is 43 (if this is the only tuple in the only relation of the database):

- $(\text{Charles}, \text{Dickens}, 42)$
- $(\text{Charles}, \text{Dickens}, 43)$

Therefore, we may state that a database with uncertain instance can be modeled as a set of certain databases $(S, I_1), \ldots, (S, I_N)$. This creates a mismatch: a single user must deal with multiple databases at the same time. However, despite the presence of many possible databases, we can still easily define queries that can be applied to all of them, because they share the same schema. For example, a query on the database illustrated in Figure 1 like:

$$\pi_{\text{Age}}(\sigma_{\text{Name='Charles'}}(\text{PERSON}))$$

is valid in both the alternative databases above, and can be computed on each of them, producing 42 and 43 as alternative results. Then, these results may be aggregated and annotated with their probability (we do not provide additional details, as this is not the main object of this paper). In Figure 4 we have illustrated this more complex process, where we can nevertheless easily define queries.

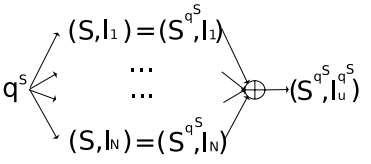


Fig. 4. Query execution with a certain schema and uncertain instances

However, if we allow each possible world to have a specific schema, different from the others, a query that is valid on one schema may not be the same on another. For example, with regard to Figure 3, the query:

$$\sigma_{\text{Office phone = Home phone}}(P1 \oplus P2)$$

is valid for the first relation (1), but not for the second, where there is just a single `Phone` column.

A solution to this problem is to provide a *consolidated schema* (S_c), containing a sort of summary of all the alternative schemata $S_1, \ldots, S_N$, and a way to rewrite a query on S_c for each alternative schema S_i. This scenario, which we call *query rewriting system*, is illustrated in Figure 5.

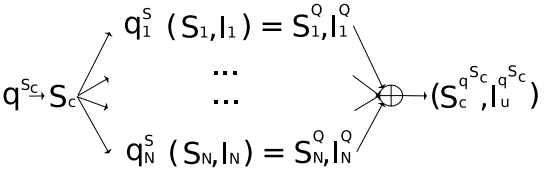


Fig. 5. Query execution with an uncertain schema, using a consolidated schema

To see how this scenario applies to the aforementioned context of data integration, and to point out the differences between query rewriting in data integration systems and in our query rewriting systems, consider Figure 6. The left hand side of the figure represents a traditional data integration setting, with three local data sources and one mediated schema (M). Queries are defined on the mediated schema, and then translated and executed on each data source. The integrated database corresponding to the mediated schema contains data from the first *and* the second *and* the third data source. On the right, we have illustrated our scenario: there are many mediated schemata, and the correct mediated schema is M_1 *or* M_2 *or* M_3, exclusive. A query expressed on the

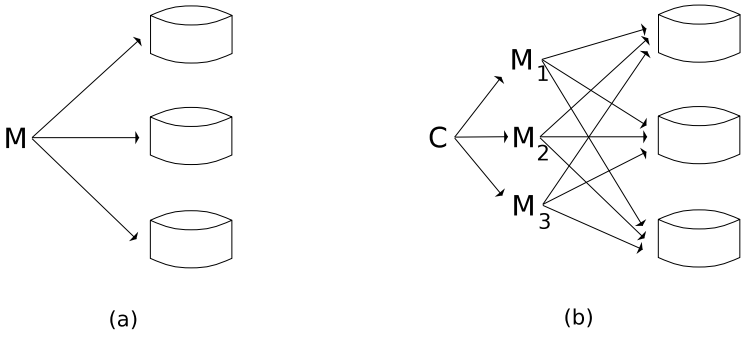


Fig. 6. Traditional (a) and uncertain (b) data integration

consolidated schema C is first translated to match each alternative mediated schema. Then, it is rewritten to be executed on each data source for each mediated schema.

Before proceeding with the formalization of this last scenario and of the properties we may expect from a consolidated schema and a query rewriting system, it is worth noticing that (although related) the management of uncertain instances and uncertain schemata are two separate problems: we may have cases with only uncertain tuples, as in Figure 1, and cases with certain tuples and uncertain schema, e.g., if we consider table $P2$ in Figure 2 and state that the first column may contain names or surnames. In general, both kinds of uncertainty may coexist, and uncertainty on schemata may induce uncertainty on the answer of queries.

2 Query Rewriting Systems (QRS)

Uncertainty may affect several features of a schema, as suggested in [3]. In this paper, we focus on one aspect: the aggregation of attributes (that we will also call columns). In particular, we assume that all alternative schemata contain the same attributes, but these attributes may constitute different relations — a scenario that can be found in data integration applications where all alternative mediated schemata contain all the attributes found in all the local data sources. As a working example, we will use the set of attributes represented in Figure 7 (notice that underlined names are used to distinguish different attributes with equal names, and not to indicate keys). Over this set, we can define the three alternative schemata represented in Figure 8. If these are the only alternatives, we can see that we are sure that the `Age` attribute contains ages of people whose surnames are contained inside the `Surname` attribute — in fact, these columns belong to a single relation in all the three alternative schemata. However, we are not sure that the people described by the underlined attributes (Name, Surname, Age) are in some relationship with the others — in fact, they are aggregated together only in the possible world represented in Figure 8(b) (PW2).

Following the previous discussion, we define a schema as a set of relations aggregating some basic attributes, i.e., a partition of these attributes where each partition (relation) has a name. In particular, we assume to work with un-normalized databases,

Name
Charles
Herman
Lewis

Surname
Dickens
Melville
Carroll

Age
42
54
13

Name
Emily
Jane
Joanne

Surname
Brontë
Austin
Rowling

Age
25
60
12

Fig. 7. A set of attributes $\mathcal{A}$

MAN

Name	Surname	Age
Charles	Dickens	42
Herman	Melville	54
Lewis	Carroll	13

WOMAN

Name	Surname	Age
Emily	Brontë	25
Jane	Austin	60
Joanne	Rowling	12

(a) Possible world PW1 — P(PW1)=.3

COUPLE

Name	Surname	Age	Name	Surname	Age
Charles	Dickens	42	Emily	Brontë	25
Herman	Melville	54	Jane	Austin	60
Lewis	Carroll	13	Joanne	Rowling	12

(b) Possible world PW2 — P(PW2)=.2

WOMAN

Name	Surname	Age
Emily	Dickens	42
Jane	Melville	54
Joanne	Carroll	13

MAN

Name	Surname	Age
Charles	Brontë	25
Herman	Austin	60
Lewis	Rowling	12

(c) Possible world PW3 — P(PW3)=.5

Fig. 8. An uncertain schema, made of three possible worlds

where there are no overlapping columns used to reference tuples from one table to the other. If we consider a single stand-alone database, this would usually result in a single table, whenever all tables are connected to the others through foreign keys. However, when we merge two or more heterogeneous databases we may obtain disjoint tables containing unrelated data.

Definition 1 (Schema). *Let $\mathcal{A}$ be a set of attribute names. A* schema *over $\mathcal{A}$ is a named partition of $\mathcal{A}$.*

In this paper, we do not consider the partition names while checking for schema equivalence, consistently with the fact that we focus only on attribute aggregation. The definition of uncertain schema is now straightforward, as an uncertain schema is just a set of alternative schemata, in this case over the same set of basic attributes.

Definition 2 (Uncertain schema). *Let $\mathcal{A}$ be a set of attribute names. An* uncertain schema *is a set of schemata $S^u = \{S_i \mid i \in [1, n]\}$ over $\mathcal{A}$.*

Given an uncertain schema, we may use several functions to represent their likelihood, our preference or our belief in them. In the remaining of the paper, we will focus on probabilistic uncertainty, and will thus assign a probability mass to each alternative schema. This choice does not prevent the definition of uncertain schemata annotated using other uncertainty representation formalims. In addition, we do not deal with the problem of how probabilities are generated.

Definition 3 (Probabilistic uncertain schema). *A* probabilistic uncertain schema *is a pair* $\langle S^u, P\rangle$*, where* S^u *is an uncertain schema and* P *is a probability distribution over* S^u*, i.e., a function* $P : S^u \to [0, 1]$ *such that* $\sum_{S \in S^u} P(S) = 1$.

Example 1 (Probabilistic uncertain schema). Consider the possible worlds represented in Figure 8, and the function P indicated in the figure. Then, $\langle\{\text{PW1, PW2, PW3}\}, P\rangle$ is a probabilistic uncertain schema.

As we have aforementioned, to produce queries that are valid for each alternative possible world, we need a *consolidated schema*. According to our previous definitions, all schemata contain the same attributes. Therefore, the same attributes should be present also in the consolidated schema used to express queries on the alternative databases.

Definition 4 (Consolidated schema). *A* consolidated schema *over an uncertain schema* S^u *is a schema over the same set of attributes* $\mathcal{A}$ *of* S^u.

Once we have a consolidated schema, we can express queries on it, that can be translated into queries on each alternative schema. In the following definition, $\text{sort}(q(S))$ indicates the set of attributes of the relation generated by the query q when applied to a schema S.

Definition 5 ((Probabilistic) query rewriting system). *Let* Q^S *be the set of queries we can express on schema* S*. A* query rewriting system *is a tuple* $\langle \mathcal{A}, S^u, P, S_c^u, \mathcal{R}\rangle$ *where* $S^u = \{S_i \mid i \in [1, n]\}$ *is an uncertain schema over* $\mathcal{A}$*,* $\langle S^u, P\rangle$ *is a probabilistic uncertain schema,* S_c^u *is a consolidated schema over* $\mathcal{A}$*,* $\mathcal{R} = \{r_i \mid i \in [1, n]\}$ *is a set of functions* $\{r_i : Q^{S_c^u} \to Q^{S_i} \mid i \in [1, n]\}$ *and* $\forall i, j \in [1, n], \forall q \in Q^{S_c^u}$*,* $sort(r_i(q)(S_i)) = sort(r_j(q)(S_j))$.

2.1 Properties of QRSs

The previous definitions are very general: any schema over $\mathcal{A}$ can be a consolidated schema. However, we are interested in schemata that represent *well* the information contained in the alternative possible worlds. Therefore, we need to define some properties to characterize *good* consolidated schemata: schema soundness (or s-soundness) and schema completeness (or s-completeness).

Let us start by defining schema completeness. Assume that two attributes are aggregated together (belong to the same relation) in an alternative schema. This expresses the existence of a relationship among the two attributes, i.e., the association of some values from one column with some values from the other. If the consolidated schema is complete, this relationship must be present in it as well.

Definition 6 (s-completeness). *Let* S^u *be an uncertain schema. A consolidated schema* S_c^u *is s-complete iff* $\forall S \in S^u, \forall R \in S \nexists R_1, R_2 \in S_c^u \mid R_1 \neq R_2, A_i \in R_1, A_j \in R_2, A_i, A_j \in R$.

The concept of schema soundness is dual: a sound consolidated schema does not contain aggregations that are not present in some alternative possible worlds:

Definition 7 (s-soundness). *Let S^u be an uncertain schema. A consolidated schema S^u_c is s-sound iff* $\forall S \in S^u, \forall R \in S^u_c \nexists R_1, R_2 \in S \mid R_1 \neq R_2, A_i \in R_1, A_j \in R_2, A_i, A_j \in R$.

These properties could also be graded, to take into account the probabilities attached to the alternative schemata. However, we leave this extension to future work.

Example 2. Consider the uncertain schema illustrated in Figure 8. The consolidated schema:

– R(Name, Surname, Age, Name, Surname, Age)

is s-complete, but not s-sound, while the consolidated schema:

– R_a(Name), R_b(Surname, Age), R_c(Name), R_d(Surname), R_e(Age)

is s-sound, but not s-complete.

Notice that the concepts of s-completeness and s-soundness are very strong: we require, respectively, that the consolidated schema contains all the information present in all possible worlds, but no information that is not present in some of them. It follows that having both conditions satisfied implies the existence of a single possible world identical to the consolidated schema.

Proposition 1. *Given an uncertain schema with cardinality strictly greater than one, there is no consolidated schema which is both sound and complete.*

So far, we have discussed the concepts of soundness and completeness with regard to the consolidated schema. While this is certainly important, because we query the alternative databases through it, the amount of information that we can extract depends also on the query language used to query the consolidated schema and the way in which we translate queries. It is worth noticing that these two basic properties alone do not define the set of *good* languages and translations. They only indicate how much information can be extracted from the consolidated schema with regard to the alternative schemata. Therefore, other additional properties can be defined to further characterize languages and translations — we leave this aspect to future works.

The concepts of soundness and completeness for QRSs are analogous to the ones defined for consolidated schemata. A QRS is complete if we can express on the consolidated schema all queries that can be expressed on each alternative schema:

Definition 8 (q-completeness). *Let $\langle \mathcal{A}, S^u, S^u_c, R \rangle$ be a query rewriting system.* $\forall S_i \in S^u, \forall q^{S_i} \in Q^{S_i} \exists q^{S^u_c} \in Q^{S^u_c} \mid r_i(q^{S^u_c}) = q^{S_i}$.

Similarly, a QRS is sound if we cannot express queries that cannot be expressed on the alternative schemata:

Definition 9 (q-soundness). *Let $\langle \mathcal{A}, S^u, S^u_c, R \rangle$ be a query rewriting system.* $\forall S_i \in S^u, \forall q^{S^u_c} \in Q^{S^u_c} \exists q^{S_i} \in Q^{S_i} \mid r_i(q^{S^u_c}) = q^{S_i}$.

3 Coarsest Refinement (CR) Rewriting Systems

Given an uncertain schema, to define a specific query rewriting system we need to specify:

- A consolidated schema.
- The query languages for both the consolidated schema and the alternative schemata.
- The rewriting functions.

In this section we describe a specific QRS, which ensures s-soundness. In fact, we have seen that we cannot be both s-sound and s-complete. However, there are several possible sound consolidated schemata and query rewriting systems. Among these, we will choose the one which maximizes completeness.

3.1 Consolidated Schema

Definition 10. *A schema S' is a* refinement *of a schema S (both over the same set of attributes $\mathcal{A}$) iff $\forall R' \in S' \; \exists R \in S \mid R' \subseteq R$.*

If a schema is not a refinement of another, it introduces new relationships. For example, the schema PW2 is not a refinement of PW1, because it introduces the relationship between the attributes related to men and those related to women, while PW1 refines PW2. If we have an uncertain schema $S^u = \{S_i \mid i \in [1, n]\}$ over $\mathcal{A}$, the set of consolidated schemata containing only relationships which are present in *all* the consolidated schemata corresponds to the set of refinements of all the alternative schemata in S^u (notated $\mathcal{R}(S^u)$). Importantly, for each uncertain schema there is always at least one schema in $\mathcal{R}(S^u)$. However, we are interested in exposing all the relationships that are defined in all the alternative possible worlds. This corresponds to a coarsest common refinement of $S_1, \ldots, S_n$.

Definition 11 (Coarsest Common Refinement). *If $S^u = \{S_i \mid i = 1 \ldots n\}$ is an uncertain schema, a schema S^u_c is a coarsest common refinement of $S_1, \ldots, S_n$ iff*

- *$S^u_c \in \mathcal{R}(S^u)$.*
- *$\forall S \in \mathcal{R}(S^u)$, S is a refinement of S^u_c.*

Proposition 2. *If $S^u = \{S_i \mid i = 1 \ldots n\}$ is an uncertain schema, there is always a coarsest common refinement of $S_1, \ldots, S_n$.*

Notice that also in [2] the authors look for the coarsest refinement of a partition, but to solve a completely different problem — they consider a single relation, and partitions indicate a *union of attributes* in a mediated schema.

Example 3. Consider the uncertain schema illustrated in Figure 8. The coarsest refinement consolidated schema is:

R_1(Name), R_2(Surname, Age), R_3(Name), R_4(Surname, Age)

Coarsest refinement consolidated schemata have the property that each relation corresponds to at most one relation in each alternative schema. Finally, notice that coarsest refinement consolidated schemata are sound but not complete, except if the uncertain schema contains only one possible world (in which case it is not really uncertain).

3.2 Query Languages

In this example of query rewriting system we will use the same query language for both the consolidated and alternative schemata. In particular, we will focus on the relational operators π, σ and $\times$. In addition, we will deal with queries in a normal form with projections, selections and products: $\pi_{A_1,\ldots,A_k}(\sigma_{A_i=c_i \wedge \cdots \wedge A_j=c_j}(R_1 \times \cdots \times R_m))$.

3.3 Rewriting Functions

Using a coarsest refinement, every relation used in a query on the consolidated schema corresponds to exactly one relation in each alternative schema. Therefore, a simple way of rewriting a query consists in replacing each consolidated relation with it, projected on their common attributes. In the following definition, sort(R) indicates the set of attributes of R.

Definition 12 (Rewriting function (1)). *Let $S^u = \{S_i \mid i = 1 \ldots n\}$ be an uncertain schema over $\mathcal{A}$, S^u_c a consolidated schema over $\mathcal{A}$ and $q \in Q^{S^u_c}$ a query on the consolidated schema. A rewriting function r_i replaces all the relations R referenced in q with $\pi_{sort(R)}(R')$ where R' corresponds to R in S_i.*

As an example, consider the query $\pi_{\text{Surname},\underline{\text{Surname}}}(R_2 \times R_4)$. This is rewritten as follows in the alternative possible worlds:

PW1 $\pi_{\text{Surname},\underline{\text{Surname}}}(\pi_{\text{Surname,Age}}(\text{MAN}) \times \pi_{\underline{\text{Surname}},\underline{\text{Age}}}(\text{WOMAN}))$

PW2 $\pi_{\text{Surname},\underline{\text{Surname}}}(\pi_{\text{Surname,Age}}(\text{COUPLE}) \times \pi_{\underline{\text{Surname}},\underline{\text{Age}}}(\text{COUPLE}))$

PW3 $\pi_{\text{Surname},\underline{\text{Surname}}}(\pi_{\text{Surname,Age}}(\text{WOMAN}) \times \pi_{\underline{\text{Surname}},\underline{\text{Age}}}(\text{MAN}))$

In Figure 9 we have represented the result of this query in each possible world, and the aggregated result (it is worth noticing that the probability of each tuple cannot be greater than one in the aggregated result, but we do not prove it here because of space reasons). However, from this result we miss the fact that in one possible world (PW2) there is

Surname	Surname	P
Dickens	Brontë	.3
Dickens	Austin	.3
Dickens	Rowling	.3
Melville	Brontë	.3
Melville	Austin	.3
Melville	Rowling	.3
Carroll	Brontë	.3
Carroll	Austin	.3
Carroll	Rowling	.3

(a) PW1

Surname	Surname	P
Dickens	Brontë	.2
Dickens	Austin	.2
Dickens	Rowling	.2
Melville	Brontë	.2
Melville	Austin	.2
Melville	Rowling	.2
Carroll	Brontë	.2
Carroll	Austin	.2
Carroll	Rowling	.2

(b) PW2

Surname	Surname	P
Dickens	Brontë	.5
Dickens	Austin	.5
Dickens	Rowling	.5
Melville	Brontë	.5
Melville	Austin	.5
Melville	Rowling	.5
Carroll	Brontë	.5
Carroll	Austin	.5
Carroll	Rowling	.5

(c) PW3

Surname	Surname	P
Dickens	Brontë	1
Dickens	Austin	1
Dickens	Rowling	1
Melville	Brontë	1
Melville	Austin	1
Melville	Rowling	1
Carroll	Brontë	1
Carroll	Austin	1
Carroll	Rowling	1

(d) Aggregated

Fig. 9. The result of $\pi_{\text{Surname},\underline{\text{Surname}}}(R_2 \times R_4)$ for the rewriting function (1)

an explicit relationship between the surnames, which does not include, for instance, the pair Carroll/Brontë (annotated with probability 1 in the result). In fact, this rewriting system is not q-complete.

Proposition 3. *The query rewriting system with the rewriting function (1) is q-sound but not q-complete.*

Therefore, we define another rewriting function which is both sound and complete:

Definition 13 (Rewriting function (2)). *Let $S^u = \{S_i \mid i = 1 \dots n\}$ be an uncertain schema over $\mathcal{A}$, S^u_c a consolidated schema over $\mathcal{A}$ and $q \in Q^{S^u_c}$ a query on the consolidated schema. Let $A_1, \dots, A_k$ the attributes of the relations referenced in q, and $\{R_1, \dots, R_m\}$ the smallest set of relations in S_i such that $\forall j \in \{1, \dots, k\}\ \exists t \in \{1, \dots, m\} \mid A_j \in R_t$. A rewriting function r_i replaces the product of relations in q with $\pi_{A_1,\dots,A_k}(R_1 \times \dots \times R_m)$.*

Let us consider again the query $\pi_{\text{Surname},\underline{\text{Surname}}}(R_2 \times R_4)$. This query is rewritten as follows:

PW1 $\pi_{\text{Surname},\underline{\text{Surname}}}(\pi_{\text{Surname,Age}}(\text{MAN}) \times \pi_{\underline{\text{Surname}},\underline{\text{Age}}}(\text{WOMAN}))$
PW2 $\pi_{\text{Surname},\underline{\text{Surname}}}(\pi_{\text{Surname,Age},\underline{\text{Surname}},\underline{\text{Age}}}(\text{COUPLE}))$
PW3 $\pi_{\text{Surname},\underline{\text{Surname}}}(\pi_{\text{Surname,Age}}(\text{WOMAN}) \times \pi_{\underline{\text{Surname}},\underline{\text{Age}}}(\text{MAN}))$

Lemma 1. *Let Sys = $\langle \mathcal{A}, S^u, S^u_c, \mathcal{R} \rangle$ be a query rewriting system, and I be the instance of schema S. If $\forall S \in S^u, \forall R_i \in S\ \exists q \in Q^{S^u_c} \mid q(S, I) = (sort(R_i), R_i)$, then Sys is q-complete.*

Basically, for each relation in any uncertain schema (and its corresponding instance) there is a query on the consolidated schema returning exactly that relation.

Proposition 4. *The query rewriting system with the rewriting function (2) is q-sound and q-complete.*

Surname	Surname	P
Dickens	Brontë	.3
Dickens	Austin	.3
Dickens	Rowling	.3
Melville	Brontë	.3
Melville	Austin	.3
Melville	Rowling	.3
Carroll	Brontë	.3
Carroll	Austin	.3
Carroll	Rowling	.3

(a) PW1

Surname	Surname	P
Dickens	Brontë	.2
Melville	Austin	.2
Carroll	Rowling	.2

(b) PW2

Surname	Surname	P
Dickens	Brontë	.5
Dickens	Austin	.5
Dickens	Rowling	.5
Melville	Brontë	.5
Melville	Austin	.5
Melville	Rowling	.5
Carroll	Brontë	.5
Carroll	Austin	.5
Carroll	Rowling	.5

(c) PW3

Surname	Surname	P
Dickens	Brontë	1
Dickens	Austin	.8
Dickens	Rowling	.8
Melville	Brontë	.8
Melville	Austin	1
Melville	Rowling	.8
Carroll	Brontë	.8
Carroll	Austin	.8
Carroll	Rowling	1

(d) Aggregated

Fig. 10. The result of $\pi_{\text{Surname},\underline{\text{Surname}}}(R_2 \times R_4)$ for the rewriting function (2)

In Figure 10 we have represented the result of this query in each possible world, and the aggregated result. Now, it can be appreciated how the pairs not supported by the possible world PW2 do not receive the corresponding probability mass.

4 Related Work

Uncertain schemata have been suggested as an important component of data integration systems in [1,4,2,5]. [2] provides the first treatment of this problem, but applying different assumptions with respect to our work, in particular single-table data sources.

Also the literature on uncertain databases is related to the problem discussed in this paper, because uncertain schemata may generate probabilistic answers [6,7,8,9,10,11]. However, as we have mentioned in the introduction, the two problems can be considered independently one from the other.

Works on uncertain semi-structured (XML) data management are related to the management of uncertain schemata, because in semi-structured data schemata are part of the data [12,13]. However, these works have not treated the problem of XML documents with sub-trees representing alternative worlds, nor uncertainty on XML schemata.

Finally, our query rewriting system has some references to works on heterogenous data querying using views [14,15], but in our setting we deal with alternative (i.e., disjunctive) databases, and not with separate data sources each providing part of a single database.

5 Conclusion and Future Work

In this paper we have presented some results on the management of uncertainty in schemata, a problem recently emerged in the field of data integration [1,2,5] but with potential applications also in centralized database systems. In particular, we have shown that uncertainty in metadata can be used to express doubtful relationships between the data and also alternative semantics of a single data source.

The main problem with uncertain schemata is the general impossibility to express a single query that is valid for all alternative databases. Therefore, a consolidated schema (as it has been called in [2]) is used to allow the user to express queries on a single schema, that are then translated into queries for the alternative databases. In this context, we have defined some properties (soundness and completeness) to evaluate both the consolidated schema and the query rewriting system. Finally, we have presented some specific systems, exemplified query answering, and studied their properties.

This matter is still in its infancy, thus many sub-problems are still awating for treatment. Important but not exahustive topics are: missing attributes, compositions of attributes, new languages to represent and query consolidated schemata, as well as practical issues regarding the implementation of uncertain schemata over existing traditional or probabilistic database management systems.

References

1. Magnani, M., Rizopoulos, N., Mc.Brien, P., Montesi, D.: Schema integration based on uncertain semantic mappings. In: Delcambre, L.M.L., Kop, C., Mayr, H.C., Mylopoulos, J., Pastor, Ó. (eds.) ER 2005. LNCS, vol. 3716, pp. 31–46. Springer, Heidelberg (2005)

2. Sarma, A.D., Dong, X., Halevy, A.: Bootstrapping pay-as-you-go data integration systems. In: Proceedings of the 2008 ACM SIGMOD international conference on Management of data, pp. 861–874. ACM, New York (2008)
3. Magnani, M., Montesi, D.: Dimensions of ignorance in a semi-structured data model. In: DEXA Workshop, pp. 933–937. IEEE Computer Society, Los Alamitos (2004)
4. Magnani, M., Montesi, D.: Uncertainty in data integration: current approaches and open problems. In: VLDB Workshop on Management of Uncertain Data (2007)
5. Sarma, A.D., Dong, X., Halevy, A.: Uncertainty in data integration. In: Managing and Mining Uncertain Data. Springer, Heidelberg (2008)
6. Re, C., Dalvi, N.N., Suciu, D.: Efficient top-k query evaluation on probabilistic data. In: Proceedings of the 23rd International Conference on Data Engineering, pp. 886–895. IEEE, Los Alamitos (2007)
7. Sarma, A.D., Benjelloun, O., Halevy, A.Y., Widom, J.: Working models for uncertain data. In: Proceedings of the 22nd International Conference on Data Engineering, p. 7. IEEE Computer Society, Los Alamitos (2006)
8. Boulos, J., Dalvi, N.N., Mandhani, B., Mathur, S., Ré, C., Suciu, D.: Mystiq: a system for finding more answers by using probabilities. In: SIGMOD Conference, pp. 891–893 (2005)
9. Cheng, R., Singh, S., Prabhakar, S.: U-dbms: A database system for managing constantly-evolving data. In: Proceedings of the 31st International Conference on Very Large Data Bases, pp. 1271–1274. ACM, New York (2005)
10. Widom, J.: Trio: A system for integrated management of data, accuracy, and lineage. In: CIDR, pp. 262–276 (2005)
11. Agrawal, P., Benjelloun, O., Sarma, A.D., Hayworth, C., Nabar, S.U., Sugihara, T., Widom, J.: Trio: A system for data, uncertainty, and lineage. In: Proceedings of the 32nd International Conference on Very Large Data Bases, pp. 1151–1154. ACM, New York (2006)
12. van Keulen, M., de Keijzer, A., Alink, W.: A probabilistic XML approach to data integration. In: 21st International Conference on Data Engineering, pp. 459–470 (2005)
13. Magnani, M., Montesi, D.: Management of interval probabilistic data. Acta Informatica 45(2), 93–130 (2008)
14. Lenzerini, M.: Data integration: a theoretical perspective. In: PODS Conference, pp. 233–246 (2002)
15. Halevy, A.Y., Rajaraman, A., Ordille, J.J.: Data integration: The teenage years. In: VLDB, pp. 9–16 (2006)

A Proofs

Proof. (Proposition 1) To prove this proposition we assume that a consolidated schema S_c^u is s-complete, and show that it cannot be s-sound.

- The uncertain schema is not a singleton, as stated in the proposition, therefore it contains at least two schemata S_1 and S_2.
- S_1 and S_2 are different (being elements of a set), therefore there must be a relation R in S_1 containing two attributes A and B that belong to two different relations in S_2 (in case S_1 contains only relations with one attribute, the proof is still valid swapping S_1 and S_2).
- As the consolidated schema is s-complete, it must contain a relation $R^{S_c^u}$ with A and B.
- This relation violates the s-soundness condition with regard to S_2.

Therefore, S_c^u cannot be both s-sound and s-complete. □

Proof. (Proposition 2) We provide a constructive proof, by showing how to generate a coarsest common refinement.

- Build a complete graph $G = (\mathcal{A}, E)$, where $\mathcal{A}$ is the set of attributes over which the uncertain schema S^u is made. An arc between two nodes/attributes means that they can belong to the same relation in the consolidated schema.
- For each relation in each alternative schema, remove from G the arcs between attributes belonging to different relations.
- Build a consolidated schema S_c^u such that each relation is made by the attributes in a connected subgraphs of G.

Notice that:

1. If two attributes are not together in all alternative schemata, they are disconnected in the graph. Therefore, *they cannot be together in the consolidated schema*.
2. Consider a consolidated schema S_c which is not a refinement of S_c^u. By definition, this implies that it contains two attributes in a relation which belong to two different relations in S_c^u, that is, two attributes whose arc in G has been removed because they belong to different relations in some alternative schema. Therefore, S_c is not a refinement of all the alternative schemata.

From (1), S_c^u is a common refinement. From (2), there are no other schemata that are common refinements of S^u but not refinements of S_c^u. Therefore, S_c^u is the coarsest common refinement. □

Proof. (Proposition 3)

Completeness: To show that the system is not q-complete, we provide a counterexample: we cannot express on the consolidated schema the query in PW1:

$$\pi_{\text{Name,Surname}}(\text{MAN})$$

because these two attributes (Name and Surname) involve two distinct relations in the consolidated schema.

Soundness: We prove by structural induction that all valid queries on the consolidated schema:

1. Are valid on a generic alternative schema S_i.
2. Produce a relation with the same attributes.

- R is rewritten as $\pi_{\text{sort}(R)}(r_i(R))$. Both conditions are satisfied, because $r_i(R) \supseteq R$.
- $q_1 \times q_2$, where q_1 and q_2 are valid queries on the consolidated schema satisfying both conditions, satisfy both conditions.
- $\sigma_{A=c}(q)$, where q is a valid query on the consolidated schema satisfying both conditions, satisfies both conditions if it is itself valid on the consolidated schema.

– $\pi_{A_1,\ldots,A_k}(q)$, where q is a valid query on the consolidated schema satisfying both conditions, satisfies both conditions if it is itself valid on the consolidated schema.

□

Proof. (Lemma 1) Let $q(R)$ be the query that retrieves relation R from S. A generic query on a consolidated schema can be expressed on the consolidated schema by substituting all occurrences of relation R with $q(R)$. □

Proof. (Proposition 4)

Completeness: Let $R(A_1, \ldots, A_k)$ be a generic relation in S, and $R_1, \ldots, R_q$ the smallest set of relations in S_c^u such that $R_1 \cup \cdots \cup R_q \supseteq \{A_1, \ldots, A_k\}$. A query $\pi_{A_1,\ldots,A_k}(R_1 \times \cdots \times R_q)$ is rewritten, by definition, as $\pi_{A_1,\ldots,A_k}(R)$. Then, this proposition follows directly from Lemma 1.

Soundness: Analogous to the proof of Proposition 3. □

Aggregation of Trust for Iterated Belief Revision in Probabilistic Logics*

Pere Pardo

Institut d'Investigació en Intel·ligència Artificial (IIIA - CSIC)
Campus UAB, E-08193 Bellaterra, Catalonia, Spain

Abstract. In this paper it is shown how communication about trust in a multi-agent system may be used to endow agents with belief change capabilities, in a probabilistic logical framework. Belief change operators are obtained in an intuitive, principled way using aggregation operators for trust-values. Under additional conditions, such change operators may be proved to be maxichoice. The present approach constitutes a sound method for autonomous uncertainty management in multi-agent systems.

Introduction

The purpose of this paper is to show how aggregation of trust-values[1] may be used, in a probabilistic logical framework, to define (iterable) belief change operations in an intuitive, principled way for multi-agent systems.

With more detail, we propose (using a probabilistic approach) to model, from an agent x perspective, how does the belief state of x change due to communications with other agents. Since other agents y may eventually change their minds, and these revisions may be relevant to x's own beliefs (due to previous interactions), x is interested in keeping an up-to-date version of y's belief state; the revised belief state of y is then used as an input to revise x's own beliefs by. A second part involves a comparison with announcements from other agents which are logically incompatible with this input, and the subsequent revision process based upon trust. Briefly, in each step lexicographical orderings (over possible revisions) or aggregation of trust-values will be used to define revision operators; in some cases, these operators may be shown to be maxichoice. The novelty of our approach lies in combining the following features: *iterability* (suggested criteria being reusable[2]), *merging-like* (to accept -register- any

* The author wishes to thank Pilar Dellunde, Lluís Godo and anonymous reviewers for helpful comments and Agreement Technologies project (Consolider CSD2007-022, INGENIO 2010) from the Spanish Ministry of Science and Innovation for financial support.

[1] We define *trust* as a measure of confidence or credibility. In the literature, see e.g. [23], credibility is understood as *trust upon an agent as informant*.

[2] In this sense, our proposal complies with Hansson's *categorial matching* principle in [16], stating that the output of a change operation should contain the same kind of elements used earlier in order to face, for arbitrary new inputs, future change operations in a self-sufficient manner.

L. Godo and A. Pugliese (Eds.): SUM 2009, LNAI 5785, pp. 165–179, 2009.

communication does not mean to accept its content), *trust-communicability* (enabling the whole system to be truly dynamic), and *meaningfulness* (trust-based aggregation providing intuitive criteria, whose adequacy for particular problems may be discussed about). After a general presentation of this revision-theoretic procedure for the *probabilistic* case within the field of (many-valued modal) uncertainty logics, we identify some weak conditions making the induced revision operator maxichoice, and illustrate the effect of properties of aggregation operators (or lexicographical orderings) in the dynamic behavior of revision operators induced by them.

Related Work: There are different presentations combining modal logic and probability in the literature, including Ognjanovic and Raškovic [20], Halpern and Fagin [15]. We opted for Hájek's fuzzy approach because truth-constants can be combined with propositions by using (fuzzy) logical connectives[3]. One may find in the literature several proposals related to the trust-based multi-agent aspects of our proposal; among these we may point out: Liau's BIT logic [18] (a crisp multi-modal logic for belief, communication and trust), Liau and Demolombe's [7] (a graded extension of the latter for static level-wise management), Dragoni and Giorgini [8] (combining Dempster-Shafer belief functions with Bayesian inference), Flaminio, Pinna and Tiezzi [11] (nested modal fuzzy predicate logic with trust-based information management under a static hierarchy of sources), Maynard-Reid and Shoham [19] (fusion of multi-source belief sets), Konieczny, Lang and Marquis [17] (aggregation functions for distance-based belief merging) and Delgrande, Dubois and Lang [5] (iterated revision as prioritized merging, based in e.g. lexicographic orderings).

The paper is structured as follows. First, we set out the preliminaries including aggregation operators, t-norm based fuzzy logics and belief change. Then we describe the method which combines results in these areas. Finally, some results showing how to (iteratively) induce (possibly maxichoice) revision operators are proved. As a result, this method is shown to endow agents with autonomous capabilities to manage arbitrary belief change problems in classical propositional logic, only assuming some initial trust information.

1 Preliminaries

In this section we introduce the definitions and results (we will need in later sections) for: aggregation operators, t-norm (graded) fuzzy logics and their (modal) uncertainty versions, and screened, *deg*-closed, maxichoice belief revision operators.

1.1 Aggregation Operators and t-norms

The field of aggregation operators [3] studies quantitative functions mapping (finitely-many) input values to a single output value. Such functions are assumed

[3] In contrast, modal approaches for probabilistic modal logics regard truth-constants as *parameters* r attached to modal operators, $\Box_i^r$. This is also the case for graded trust-belief-communication logics e.g. [7] listed below.

to be defined in the real unit interval $[0,1]$. Formally, a family of functions $\mathbf{A}_{(n)} : [0,1]^n \to [0,1]$ for arbitrary $n \in \omega$ is identified with function $\mathbf{A} : \bigcup_{n\in\omega}[0,1]^n \to [0,1]$. Properties considered to define the general class of aggregation operators are listed next.

Definition 1. *[3] A function* $\mathbf{A}_{(n)} : [0,1]^n \to [0,1]$ *is an* aggregation operator *iff it satisfies:*

(Boundary) $A_{(n)}(0,\ldots,0) = 0, \quad \mathbf{A}_{(n)}(1,\ldots,1) = 1$ *and* $\mathbf{A}_{(1)}(x) = x$
(Monotonicity) For each $n \in \omega$, *if* $x_1 \leq y_1, \ldots, x_n \leq y_n$ *then*
$\mathbf{A}(x_1,\ldots,x_n) \leq \mathbf{A}(y_1,\ldots,y_n)$

For a given $\mathbf{A}$ *function satisfying additionally some of the following conditions*

(Idempotence) $\mathbf{A}_{(n)}(x,\ldots,x) = x$, *or*
(Symmetry) $\mathbf{A}_{(n)}(x_0,\ldots,x_n) = \mathbf{A}_{(n)}(x_{\pi(0)},\ldots,x_{\pi(n)})$,
for each permutation π *of* $\{1,\ldots n\}$
(Associativity) $\mathbf{A}(x_1,\ldots,x_n,y_1,\ldots,y_m) = \mathbf{A}(\mathbf{A}(x_1,\ldots,x_n),\mathbf{A}(y_1,\ldots,y_m))$

then we say $\mathbf{A}$ *is, respectively, an* idempotent, symmetric *or* associative *aggregation operator. An element* $e \in [0,1]$ *is a* neutral element *of* $\mathbf{A}$ *iff for all* $n \in \omega$ *and* $x_1,\ldots,x_n \in [0,1]$, $x_i = e$ *(for some* $i \in \{1,\ldots,n\}$ *implies* $\mathbf{A}(x_1,\ldots,x_n) = \mathbf{A}(x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_n)$.

In terms of the preceding definition, a triangular norm, or *t-norm*, is a binary associative[4] associative symmetric aggregation function with neutral element 1. On the other side, the class of idempotent aggregation operators $\mathbf{A}$ lies between max and min functions: $\min \leq \mathbf{A} \leq \max$. (The min function also happens to be the only idempotent t-norm.) See [24] for a study of idempotent symmetric aggregation operators; in this work, operators in this subclass are simply called *aggregation operators*; henceforth, we adopt this notation for convenience, and refer to each other subclass by its specific name, e.g. *t-norms*.

1.2 Logical Change Operators

Belief change is the study of how some theory T (non-necessarily closed, as we use the term) in a given language L can adapt to new incoming information $\varphi \in L$ (inconsistent with T, in the interesting case). The main operations are: *revision*, where the new input must follow from the revised theory in a consistent way and *contraction* where the input must not follow from the contracted theory. In the classical paper [1], by Alchourrón, Gärdenfors and Makinson, partial meet revision and contraction operations were characterized for closed theories in, essentially, monotonic compact logics with the deduction property. Their work opened the way for other studies involving new objects of change, operations

[4] Any associative aggregation function can be simply presented as a binary function; that is, the corresponding n-ary functions may easily be obtained from the binary case using associativity.

(see [22] for a comprehensive list) or logics[5]. Change operators can be defined by the next method, from [1] (there exist other -classically equivalent- methods as well, which are based on rankings rather than on selections). *Partial meet* consists in (i) generating all logically maximal ways to adapt T to the new sentence (those subtheories of T making further information loss logically unnecessary), (ii) selecting some of these possibilities, (iii) forming their meet, and, optionally, (iv) performing additional steps (if required by the operation). Then a set of axioms is provided to capture these partial meet operators, by showing equivalence between satisfaction of these axioms and being a partial meet operator. (Observe step (ii) assumes some non-logical components to define a solution for a given revision problem; this paper is mostly devoted to finding such (a class of) non-logical components). A *base* is an arbitrary set of formulas, the original requirement of logical closure being dropped (for convenience, we indistinctively use *base* and *theory*). We make use of definitions and results proved in [21] (see also [2]):

Definition 2. *([2]) Given some monotonic logic $\mathcal{S}$, let T_0, T_1 be theories. We say T_0 is* consistent *if $T_0 \nvdash_{\mathcal{S}} \overline{0}$, and define the set of* subsets *of T_0* maximally consistent with T_1 *as follows: $X \in Con(T_0, T_1)$ iff:*

(i) $X \subseteq T_0$,
(ii) $X \cup T_1$ *is consistent, and*
(iii) for any X' such that $X \subsetneq X' \subseteq T_0$, we have $X' \cup T_1$ is inconsistent

The axioms to characterize (multiple) base revision operators $\circledast : \mathcal{P}(\mathbf{Fm}) \to \mathcal{P}(\mathbf{Fm})$ for some T_0 (in a given finitary monotonic logic $\mathcal{S}$) are the following:

(F1) $T_1 \subseteq T_0 \circledast T_1$ (Success)
(F2) If T_1 is consistent, then $T_0 \circledast T_1$ is also consistent. (Consistency)
(F3) $T_0 \circledast T_1 \subseteq T_0 \cup T_1$ (Inclusion)
(F4) For all $\psi \in \mathbf{Fm}$, if $\psi \in T_0 \smallsetminus T_0 \circledast T_1$ then, there exists T' with $T_0 \circledast T_1 \subseteq T' \subseteq T_0 \cup T_1$ and such that $T' \nvdash_{\mathcal{S}} \overline{0}$ but $T' \cup \{\psi\} \vdash_{\mathcal{S}} \overline{0}$) (Relevance)
(F5) If for all $T' \subseteq T_0$ ($T' \cup T_1 \nvdash_{\mathcal{S}} \overline{0} \Leftrightarrow T' \cup T_2 \nvdash_{\mathcal{S}} \overline{0}$) then $T_0 \cap (T_0 \circledast T_1) = T_0 \cap (T_0 \circledast T_2)$ (Uniformity)

Definition 3. *For a given set $\mathbb{X}$ of families of sets, a* selection function γ *is a function selecting a non-empty subset of each family in $\mathbb{X}$.*

$$\gamma : \mathbb{X} \longrightarrow \bigcup_{X \in \mathbb{X}} \mathcal{P}(X), \text{ with } \emptyset \neq \gamma(X) \subseteq X$$

[5] Following [12], we define a logic as a finitary and structural consequence relation $\vdash_{\mathcal{S}} \subseteq \mathcal{P}(\mathbf{Fm}) \times \mathbf{Fm}$, for some algebra of formulas $\mathbf{Fm}$. That is, $\mathcal{S}$ satisfies (1) If $\varphi \in \Gamma$ then $\Gamma \vdash_{\mathcal{S}} \varphi$, (2) If $\Gamma \vdash_{\mathcal{S}} \varphi$ and $\Gamma \subseteq \Delta$ then $\Delta \vdash_{\mathcal{S}} \varphi$, (3) If $\Gamma \vdash_{\mathcal{S}} \varphi$ and for every $\psi \in \Gamma$, $\Delta \vdash_{\mathcal{S}} \psi$ then $\Delta \vdash_{\mathcal{S}} \varphi$ (*consequence relation*); (4) If $\Gamma \vdash_{\mathcal{S}} \varphi$ then for some finite $\Gamma_0 \subseteq \Gamma$ we have $\Gamma_0 \vdash_{\mathcal{S}} \varphi$ (*finitarity*); (5) If $\Gamma \vdash_{\mathcal{S}} \varphi$ then $e[\Gamma] \vdash_{\mathcal{S}} e(\varphi)$ for all substitutions $e \in Hom(\mathbf{Fm}, \mathbf{Fm})$ (*structurality*).

In particular, we will make use of selection functions being defined for $\mathrm{Con}(T_0, T_1)$ sets, for at least some T_0 and arbitrary T_1. This suffices to define, given some theory $T_0 \subseteq \mathbf{Fm}$ and selection function γ for T_0, a *partial meet revision operator* $\circledast_\gamma$ for T_0 as follows:

$$T_0 \circledast_\gamma T_1 = \bigcap \gamma(\mathrm{Con}(T_0, T_1)) \cup T_1$$

Definition 4. *Let $\mathcal{S}$ be some finitary logic, and T_0 a theory. Then $\circledast : \mathcal{P}(\mathbf{Fm}) \to \mathcal{P}(\mathbf{Fm})$ is a* revision operator *for T_0 iff for any $T_1 \subseteq \mathbf{Fm}$, $T_0 \circledast T_1 = T_0 \circledast_\gamma T_1$ for some selection function γ for T_0.*[6]

Theorem 1. *[21] Let $\mathcal{S}$ be a finitary monotonic logic. For any $T_0, T_1 \subseteq \mathbf{Fm}$ and function $\circledast : \mathcal{P}(\mathbf{Fm}) \to \mathcal{P}(\mathbf{Fm})$ for T_0:*

$$\circledast \textit{ satisfies } (\mathrm{F1}) - (\mathrm{F5}) \text{ iff } T_0 \circledast T_1 = T_0 \circledast_\gamma T_1, \textit{ for some } \gamma$$

In the limiting case where a revision operator is induced by some function γ selecting a unique element X, i.e. $\gamma(X) = \{x\}$, this operator is called *maxichoice*.

Theorem 2. *[21] Let $\mathcal{S}, T_0$ and T_1 be as above, then $\circledast$ is maxichoice iff* (F1) – (F5) *plus* (MC) *hold:*

(MC) *For all $\psi \in \mathbf{Fm}$ with $\psi \in T_0 \setminus T_0 \circledast T_1$ we have* $T_0 \circledast T_1 \cup \{\psi\} \vdash_{\mathcal{S}} \overline{0}$

In the results shown in the next section, a variant of belief revision, *screened* belief revision is used. This operation of belief change consists, as usual, in selecting subsets of the base T_0 consistent with the new input T_1 *with the restriction* that for some previously chosen consistent subset $T^* \subseteq T_0$, it must be the case that $T^* \subseteq T_0 \circledast T_1$. Sentences in T^* are, then, preserved under revision and can be considered irrevocable; thus, it is assumed that (acceptable) inputs T_1 will be consistent with T^*. This is called *T^*-screened revision operator for T_0*. Screened revision can be characterized in terms of standard belief revision as follows:

$$T_0 \circledast^{T^*} T_1 = T_0 \circledast (T_1 \cup T^*) = \bigcap \gamma'(\mathrm{Con}(T_0 \setminus T^*, T_1 \cup T^*) \cup (T_1 \cup T^*)$$

for some suitable γ'. We will also consider a natural variant, in graded logics, of basehood, where bases are deductively closed by lower truth-degrees (*deg*-closed), as in [2]. This is unproblematic, since, as it was proved in [21], revision of *deg*-closed bases by *deg*-closed inputs results in *deg*-closed bases. Moreover, this can be generalized to finite subsets[7] of the set of truth-values, preserving this result in the finitely-closed case (i.e for f*deg*-closed bases).

[6] Thus, revision operators are defined relative to some fixed base T_0. Nonetheless, instead of $\circledast^{T_0} T_1$, we keep the traditional infix notation $T_0 \circledast T_1$.

[7] That is, if truth-constants $\overline{r}$ occurring in base and input are taken from values $r = \frac{m}{n}$ for $0 \leq m \leq n$ (i.e. dividing $[0, 1]$ into n equal parts).

1.3 Uncertainty Logics Based on t-norms

We recall basic definitions about t-norm based fuzzy logics and some definitions and results for their uncertainty modal extensions. T-norm based logics are residuated logical systems capturing the validities of formulas whose conjunction & and implication $\rightarrow$ connectives are interpreted respectively by classes of (left-)continuous t-norms and their residua. A left-continuous t-norm $*$ and its residuum $\Rightarrow_*$, then, define a pair of fuzzy truth-functions in the real unit interval $[0,1]$, which extend their classical counterparts in $\{0,1\}$ and satisfy the residuation law: $a * b \leq c$ iff $a \leq b \Rightarrow_* c$, for all $a, b, c \in [0,1]$ The logics of the three basic continuous t-norms, Łukasiewicz Ł, product Π and Gödel G, are axiomatic extensions of Hájek's Basic logic BL [14] (this logic capturing the set of tautologies common to any logic of a continuous t-norm) with the following corresponding axioms:

$$(\text{Ł})\ \neg\neg\varphi \rightarrow \varphi \qquad (\Pi)\ \neg\varphi \vee ((\varphi\ \rightarrow\ \varphi\&\psi) \rightarrow \psi) \qquad (G)\ \varphi \rightarrow \varphi\&\varphi$$

which are shown to be complete with respect to the semantic calculus defined by the corresponding t-norm: $a *_{\text{Ł}} b = \max(0, a + b - 1)$, $a *_{\Pi} b = a \cdot b$ and $a *_G b = \min(a, b)$.

Hájek introduced in [14] the logic **RPL** as a graded expansion of Łukasiewicz logic Ł, adding into the language a truth-constant $\overline{r}$ for each rational number in $[0,1]$ together with the so-called *book-keeping axioms*: $\overline{r *_{\text{Ł}} s} \equiv \overline{r}\&\overline{s}$ and $\overline{r} \rightarrow_{\text{Ł}} \overline{s} \equiv \overline{r \Rightarrow_{\text{Ł}} s}$, where $a \Rightarrow_{\text{Ł}} b = \min\{1, 1 - a + b\}$. **RPL** logic was proved to satisfy Pavelka-style completeness. Given a theory T and a sentence φ in the language of **RPL**, we define the *provability-* and the *truth-degrees* of φ in T by, respectively,

$$|\varphi|_T\ =\ \sup\{r \in [0,1] \mid T \vdash_{\mathcal{S}} \overline{r} \rightarrow_{\text{Ł}} \varphi\} \qquad \|\varphi\|_T\ =\ \inf\{e(\varphi) \mid e \text{ is a model of } T\}$$

Theorem 3. *[14, Thm 3.3.5] (In* **RPL***) For each theory T and formula φ, $|\varphi|_T = \|\varphi\|_T$.*

Finitely Strong Completeness was also proved to hold in **RPL**.

Theorem 4. *[14, Thm 3.3.14] (In* **RPL***) Let T be a finite theory and φ a formula. If φ is 1-true in all models of T, then $T \vdash_{\mathbf{RPL}} \varphi$.*

Other graded expansions of continuous t-norm based logics have been studied, see e.g. [10]. One of the motivations after the introduction of (t-norm) fuzzy logics was to capture (in a sound way) reasoning about sentences having a truth-degree other than 0 or 1. It was realized, e.g. [9], that (a) *gradual truth* of some worldly sentence was to be distinguished from (b) *uncertainty* about some crisp (worldly) sentence. T-norm fuzzy logics capture reasoning of type (a), but to model non-truth-functional uncertainty, some modifications are required. In the literature, logical models of uncertainty have been presented (so far) as fuzzy logics over a simple modal logic for two-valued Boolean propositions. This is the case of probability and possibility [14], and Dempster-Shafer functions [13] among others.

We briefly describe next Hájek's $FP(\mathbf{RPL})$ logic [14], a modal-like logic to reason about probability built over **RPL**. This logic deals with classical Boolean sentences being qualified by a modality here denoted B (for belief). Indeed, atomic modal formulas of $FP(\mathbf{RPL})$ are of the form $B\varphi$, to be read as φ *is believed*, where φ is a classical Boolean formula φ (nested modal operators are not allowed); finally, these basic modal formulas and truth-constants $\overline{r}$ can be arbitrarily combined by means of Łukasiewicz connectives $\&, \to_{\text{Ł}}$ in the usual way.

For our purposes in this paper, we will need to deal. not with a single modal operator B, but with a set of modal operators $\{[B_x]\}_{x\in\mathsf{Ag}}$, each one modelling the belief of each agent x in a given (finite) set of agents Ag. The axioms and rules for our multi-modal version of $FP(\mathbf{RPL})$ are listed in Fig. 1.3. The semantics

Axioms of Classical Propositional Logic (for non-modal formulas)
Axioms of **RPL** (for modal formulas): **BL** + Ł and book-keeping axioms
Axioms of fuzzy probability for each modality $[B_x]$
  $(FP1)$: $[B_x](\varphi) \to_{\text{Ł}} ([B_x](\varphi \to \psi) \to_{\text{Ł}} [B_x]\psi)$
  $(FP2)$: $[B_x](\neg\varphi) \equiv \neg[B_x]\varphi$
  $(FP3)$: $[B_x](\varphi \vee \psi) \equiv [([B_x]\varphi \to_{\text{Ł}} [B_x](\varphi\&\psi)) \to_{\text{Ł}} [B_x]\psi]$
Rules of inference:
  modus ponens (for modal and non-modal formulas) and
  $[B_x]$-*necessitation*: from $\vdash \varphi$ infer $\vdash [B_x]\varphi$, for each modality $[B_x]$

Fig. 1. Axioms and rules of multi-modal $FP(\mathbf{RPL})$-logic

of our language is given by means of Kripke models of the following form $\mathcal{K} = (W, e, \langle\mu_x\rangle_{x\in\mathsf{Ag}})$ where $W \neq \emptyset$, e is an evaluation (that is, $e(w,p) \in \{0,1\}$ for $w \in W$ and p a propositional variable), and for every modality $[B_x]$, μ_x is a finitely additive probability on some Boolean subalgebra $F \subseteq 2^W$ such that the sets $\{w \mid e(w,\varphi) = 1\}$, for every classical propositional formula ψ, are μ_x-measurable. The semantics of non-modal formulas, also denoted by e, is defined as usual from atomic evaluations in each $w \in W$. For every modality $[B_i]$ the truth value of an atomic modal formula $[B_i]\varphi$ in a model $\mathcal{K}$ is defined as

$$\|[B_x]\varphi\|_{\mathcal{K}} = \mu_x(\{w \mid e(w,\varphi) = 1\})$$

and the truth-value of compound modal formulas are computed from the atomic (modal) ones using the truth-functions of Łukasiewicz logic.

As before, a set of sentences T is called a *theory* and it is said that a theory T is *closed* iff T is closed under the $\vdash_{FP(\mathbf{RPL})}$ relation. Finally, a theory T is called *modal* iff T only contains modal formulas. Borrowing the notions of provability and truth-degrees for **RPL**, the following completeness results for the multi-modal version of $FP(\mathbf{RPL})$ hold.

Theorem 5. *[14]* [Pavelka Completeness] *For any modal theory T and modal formula Φ, it holds that:* $|\Phi|_T = \|\Phi\|_T$.

Theorem 6. *[14]* [Finite Strong Completeness] *For any finite modal theory T, and modal formula Φ,* $T \vdash_{FP(\mathbf{RPL})} \Phi$ *iff* $\|\Phi\|_T = 1$.

2 From Aggregation Functions to Change Operators

Let Ag be a set of agents. Given a set of information communications among the agents, our aim is to model, from an agent x's perspective, changes in x's beliefs about the world and about the trust in other agents.

To this end, we assume that, at a given moment, each agent $x \in \mathsf{Ag}$ maintains a belief base $\mathbf{T}_x$ which is composed of

(i) a *trust subbase* $\mathbf{Tr}_x$, gathering information about the degree of trust x has on other agents. This information will be represented by $FP(\mathbf{RPL})$-formulas of the kind $\overline{r} \rightarrow_{\mathrm{L}} [B_x]Tr_{xy}$, where B_x is the agent x's probability modality and Tr_{xy} is a Boolean atomic formula. The intended reading of such a formula is: x believes with probability at least r that x *can trust on* y.

(ii) a *knowledge subbase* $\mathbf{K}_x$, gathering (graded) beliefs about the world, expressed by means of modal formulas of $FP(\mathbf{RPL})$ built with the modality $[B_x]$. Typical formulas will be of the form $\overline{s} \rightarrow_{\mathrm{L}} [B_x]\varphi$, denoting that agent x believes φ holds, with probability at least s. But arbitrary, more complex $FP(\mathbf{RPL})$-formulas are also allowed[8].

(iii) a *communication subbase* $\mathbb{T}_{y \triangleright x}$ for each agent $y \neq x$ containing information that agent y has communicated to agent x, i.e. what agent x has received from agent y's discourse. Such information will be again obtainable from y's trust, knowledge and communication subbases. All this information coming from agent y will be again expressed by $FP(\mathbf{RPL})$ formulas built over agent y's modality $[B_y]$.

Therefore each agent belief state can be expressed as

$$\mathbf{T}_x = \mathbf{Tr}_x \cup \mathbf{K}_x \cup \bigcup_{y \neq x} \mathbb{T}_{y \triangleright x}$$

If the context makes it clear, we define $\mathbb{T} = \bigcup_{y \neq x} \mathbb{T}_{y \triangleright x}$ as the set of x's knowledge of all agents' discourses.

Let us describe informally which kind of belief change processes we envisage when an agent x receives some communication from agent y of form $\Phi = \overline{r} \rightarrow_{\mathrm{L}} [B_y]\varphi$, expressing that agent y believes φ (which may denote an information about the world or about the trust on some other agent z) to the degree r.

In a first step, agent x revises (her record of) y's discourse, $\mathbb{T}_{y \triangleright x}$, by Φ, to keep it up-to-date. The idea here to guide this revision is to consider that *newest*

[8] For the sake of a simpler presentation, in this paper we will assume agent x's knowledge base to be empty.

information prevails over old one (in $\mathbb{T}_{y \triangleright x}$). In a second step, once $\mathbb{T}_{y \triangleright x}$ has been revised and taking into account the trust she has on agent y, agent x transforms the information provided by agent y in $\mathbb{T}_{y \triangleright x}$ into x's own beliefs by means, building a new base $F[\mathbb{T}_{y \triangleright x}]$ by means of the following simple transformation:

$$\text{if } \Phi([B_y]\varphi) \in \mathbb{T}_{y \triangleright x}, \text{ then } [B_x]Tr_{xy} \rightarrow_{\mathrm{L}} \Phi^* \in F[\mathbb{T}_{y \triangleright x}],$$

where Φ^* is the result of replacing each occurrence of $[B_y]$ by $[B_x]$. That is, this transformation in fact conditionalizes Φ to how much agent x trusts on agent y. Note that if agent x does not trust y at all, i.e. $\neg[B_x]Tr_{xy} \in \mathbf{Tr}_x$, then from $\mathbf{Tr}_x \cup F[\mathbb{T}_{y \triangleright x}]$ nothing can be inferred about the belief on Φ.

Then the question which remains is how to revise the agent x's belief trust subbase $\mathbf{Tr}_x$ in the light of the communicated information to agent x by the other agents. The idea[9] that guides this revision process is a principle of *minimizing the loss of trust* on the other agents, i.e. we look for maximal subsets of $\mathbf{Tr}_x$ consistent with $F[\mathbb{T}]$ which yield a minimal aggregated loss of trust (given a suitable aggregation function A) of agent x on the rest of agents $y \in \mathsf{Ag} \setminus \{x\}$.

In the following subsections we provide details on the revision procedures used in each step described above.

2.1 (i) Enforcing Consistency in an Agent's Discourse

At this step agent x adjusts her beliefs about an agent's discourse to this agent's announcements. This revision will be non-trivial (i.e. not an expansion) when y tells x something contradicting some of y's previous announcements[10]. For our modeling purposes, we need to consider the communications subbase $\mathbb{T}_{y \triangleright x}$ as a temporally ordered sequence with newest information first:

$$\overrightarrow{\mathbb{T}_{y \triangleright x}} = \langle \Phi_{n+1}, \Phi_n, \ldots, \Phi_0 \rangle$$

This enables agent y to univocally guide (if necessary) x's revision process in $\mathbb{T}_{y \triangleright x}$ according to the *prevalence of new information* convention[11]. This is implemented by letting the lexicographic ordering determine a selection function for $\mathbb{T}_{y \triangleright x}$. Let $\overrightarrow{X} = \langle \varphi_{n_i} \rangle_{i \in I}$ and $\overrightarrow{Y} = \langle \varphi_{n_j} \rangle_{j \in J}$ be the temporally reverse orderings of subsets X, Y of $\mathbb{T}_{y \triangleright x}$. We define a lexicographical ordering based on the subindexes of formulas in $\overrightarrow{X}, \overrightarrow{Y}$:

$$X <_{\text{lex}} Y, \text{ if } \langle n_i \rangle_{i \in I} \prec_{\text{lex}} \langle n_j \rangle_{j \in J}$$

[9] Other principles can also be taken into account. See the examples at the end of the paper.

[10] This is how we prevent diachronic inconsistencies (y telling p first and $\neg p$ later) from being attributed $p \wedge \neg p$ to an agent. By definition of revision operators, synchronic inconsistencies (i.e. telling $p \wedge \neg p$) are simply ignored.

[11] After being told by y (1) that φ, (2) that ψ and (3) that $\neg(\varphi \wedge \psi)$ agent x will prefer ψ to φ, unless y tells x that φ between just before (3) to guide x's revision of her own discourse. Similarly, if y told. Observe it is in benefit of y to make her position clear.

where $I, J \subseteq \{0, \ldots, n+1\}$ and $\prec_{\text{lex}}$ is the usual lexicographic ordering of $\omega^{<\omega}$ (the set of finite sequences of natural numbers).

Proposition 1. *Any function of the following form is a selection function for* $\mathbb{T}_{y \triangleright x}$ *and induces a maxichoice revision operator for* $\mathbb{T}_{y \triangleright x}$.

$$\gamma_0(\text{Con}(\mathbb{T}_{y \triangleright x}, \Phi)) = \{X\}, \text{ if } X \text{ is maximum w.r.t. } <_{\text{lex}}.$$

Proof. Obviously γ_0 selects a subset of $\text{Con}(\mathbb{T}_{y \triangleright x}, \Phi)$. We show that γ_0 selects a unique element (hence it is non-empty). Since $\prec_{\text{lex}}$ is a total ordering of $\omega^{<\omega}$, $<_{\text{lex}}$ is also total in $\text{Con}(\mathbb{T}_{y \triangleright x}, \Phi)$; hence, it has a $<_{\text{lex}}$-maximum element, X, which is the only element of $\gamma_0(\text{Con}(\mathbb{T}_{y \triangleright x}, \Phi))$. Thus, γ_0 is non-empty, hence a selection function for $\mathbb{T}_{y \triangleright x}$. This γ_0 also induces a maxichoice revision operator, since it is proved above that γ selects a single element.

2.2 (ii) Multi-source Conflict Resolution

Managing conflicts between sources is the main part of the method proposed. We generalize the previous step to the case where more than one discourse is being revised. Let $\mathbb{T} = \bigcup_{y \neq x}(\mathbb{T}'_{y \triangleright x} \circledast_{\gamma_0} \Phi_y)$, where $\mathbb{T}'_{y \triangleright x}$ is the old subbase and Φ_y its revision input. Analogously, we define $\mathbb{T}' = \bigcup_{y \neq x} \mathbb{T}'_{y \triangleright x}$. The set $F[\mathbb{T}]$, the F-translation of step one output $\mathbb{T}$, is the revision input for this part.

Only sentences in the trust-subbase are eligible[12] for withdrawal when revising by $F[\mathbb{T}]$. Withdrawing old beliefs in the trust-base $\mathbf{Tr}_x$ will be needed only when $F[\mathbb{T}]$ turns out to be inconsistent with the trust-base $\mathbf{Tr}_x$. This motivates the following definitions.

Consider the family of maxichoice revision operator outputs, for fixed $\mathbf{Tr}_x$ and $F[\mathbb{T}]$, defined as the family $\mathbb{X}$ of sets $X^* = X \cup F[\mathbb{T}]$, for some $X \in \text{Con}(\mathbf{Tr}_x, F[\mathbb{T}])$. We want to compare these values with those from the old, pre-revision base $\mathbf{Tr}_x \cup F[\mathbb{T}']$, denoted by $F[\mathbf{T}_x]$. To do so, in each revised set X^* we compute to which degree $\alpha_{xy}^{X^*}$ agent x trusts on y as the provability degree over $FP(\mathbf{RPL})$ (see Section 1.3) of the formula $[B_x]Tr_{xy}$ in the theory X^*. Formally,

Definition 5. *For each* $y \neq x$, *we define* $\alpha_{xy}^{X^*} = |[B_x]Tr_{xy}|_{X^*}$. *We define* $\beta_{xy}^{X^*} = |[B_x]Tr_{xy}|_{F[\mathbf{T}_x]} - |[B_x]Tr_{xy}|_{X^*}$. *In the following* $\mathbb{P}$ *stands for* α *or* β *(with corresponding scripts).*

Parameters α measure the total amount of agents' trust (either preserved or gained), while β parameters measure the amount of trust lost from the old belief

[12] The present approach might be improved into more sophisticated versions. (1) A first possibility would be to turn trust recommendation $\overline{r} \rightarrow_{\text{L}} [B_y]Tr_{xz}$ (which is conditional on x's trust upon y) into (unconditional) trust information to be added to the trust subbase $\mathbf{Tr}_x$. This would have the effect of making y no longer responsible for her recommendation of z. (2) Another option is to enlarge the trust subbase so as to cover derived trust information too (e.g. trust information derived from unconditional trust, and a trust recommendation). To simplify the exposition neither of these options is considered in the present approach.

state, both after some revision (would that particular revision take place). We pointed out that revision processes often require non-logical components. In our proposal, α or β values will be the first element used to define this non-logical component. Other elements are: (ii) an aggregation operator A to aggregate the agents' $\mathbb{P}$-values for a fixed $X^* \in \mathbb{X}^*$ (that is, for y varying over $\mathsf{Ag} \smallsetminus \{x\}$); thus, from a sequence $\langle \mathbb{P}^{X^*}_{xy} \rangle_{y \neq x}$ we obtain a value $\mathsf{A}(\langle \mathbb{P}^{X^*}_{xy} \rangle_{y \neq x}) \in [0,1]$. Finally, we introduce (iii) a closed[13] aggregation operator f (typically, max or min) to select among $\mathbb{X}^*$ sets according to the aggregated A-value supplied in (ii). That is, given $\mathsf{A}(\langle \mathbb{P}^{X^*_0}_{xy} \rangle_{y \neq x}), \ldots, \mathsf{A}(\langle \mathbb{P}^{X^*_n}_{xy} \rangle_{y \neq x})$, function f will choose, say, the maximum of these values. Now, the advantage of requiring f to be closed is that we can recover the particular set(s) X^* whose A-value gets selected by f. We define the selection function induced by $(f, \mathsf{A}, \mathbb{P})$ as follows:

$$\gamma_{(f,\mathsf{A},\mathbb{P})}(\mathrm{Con}(T_0, T_1)) = \{X^* \cap T_0\}$$

for X^* such that $\mathsf{A}(\langle \mathbb{P}^{X^*}_{xy} \rangle_{y \neq x}) = f(\{\mathsf{A}(\langle \mathbb{P}^{Y^*}_{xy} \rangle_{y \neq x}) \mid Y^* \in \mathbb{X}^*\})$. As shown next, this $\gamma_{(f,\mathsf{A},\mathbb{P})}$ is a selection function for any base T_0 and input T_1. Observe this method must compute first the values of each output of some maxichoice revision operator, in order to decide for the $(f, \mathsf{A}, \mathbb{P})$-best selections.

Proposition 2. *Let T_0 be a base and T_1 an input. For any $(f, \mathsf{A}, \mathbb{P})$ as defined above $\gamma_{(f,\mathsf{A},\mathbb{P})}$ is a selection function inducing a $(F[\mathbf{T}_x] \smallsetminus \mathbf{Tr}_x)$-screened revision operator for $\mathbf{Tr}_x$.*

Proof. The bijection between $\mathbb{X}^*$ elements (under $\cap \mathbf{Tr}_x$) and $\mathrm{Con}(T_0, T_1)$ elements is obvious (by definition of $\mathbb{X}^*$). By definition of $\gamma_{(f,\mathsf{A},\mathbb{P})}$, this function selects a subset of $\mathrm{Con}(T_0, T_1)$. Now, we prove that $\gamma_{(f,\mathsf{A},\mathbb{P})}$ is non-empty: since f is closed there exists a subset $\{X^*_0, \ldots, X^*_k\}$ such that, for $0 \leq i \leq k$, $\mathsf{A}(\langle \mathbb{P}^{X^*_i}_{xy} \rangle_{y \neq x}) = f(\{\langle \mathsf{A}(\mathbb{P}^{X^*}_{xy}) \rangle_{y \neq x} \mid X^* \in \mathbb{X}^*\})$. By the previous bijection $\{X^*_0 \cap T_0, \ldots, X^*_k \cap T_0\}$ is a subset of $\mathrm{Con}(T_0, T_1)$. The latter set is non-empty whenever T_1 is consistent.

Observe that F-translated step one inputs satisfy the consistency condition for T_1: the set $F[\mathbb{T}]$ is always consistent: let e be an assignment with $e(Tr_{xy}) = 0$ for each $y \neq x$. Then, e is a model of $F[\mathbb{T}]$ Thus, when T_1 is obtained as an step one revision, the above result applies. This result also shows F to preserve consistency, but not necessarily to preserve consistency with $\mathbf{Tr}_x$. As usual, the revision operator induced by such $\gamma_{(f,\mathsf{A},\beta)}$ is defined by

$$F[\mathbf{T}_x] \circledast_{(f,\mathsf{A},\mathbb{P})} F[\mathbb{T}] = (\bigcap \gamma_{(f,\mathsf{A},\mathbb{P})}(\mathrm{Con}(\mathbf{Tr}_x, F[\mathbb{T}])) \cup F[\mathbb{T}]$$

Under additional conditions, we can enforce the defined revision operators to be maxichoice.

[13] An aggregation operator f is closed if $f(X) \in X$. In set-theoretic terminology, a closed aggregation operator is also a choice function.

Proposition 3. *Let* $(f, \mathsf{A}, \mathbb{P})$ *be as above and assume all values* $\mathsf{A}(\langle \mathbb{P}_{xy}^{X^*} \rangle_{y \neq x})$ *are pairwise different for* $X^* \in \mathbb{X}^*$. *Then* $\gamma_{(f,\mathsf{A},\beta)}$ *induces a maxichoice revision operator. The same holds if the above quantification ranges over* $X \in \mathrm{Con}(\mathbf{Tr}_x, F[\mathbb{T}])$ *instead of* $X \in \mathbb{X}^*$.

Proof. We prove the last, stronger, claim. Since the aggregated A-values of $\mathbb{P}$-parameters for any X^*, Y^* are pairwise different, they can be totally ordered, so closed f will select a unique $\mathbb{P}$-value, r. Let X^* be the unique element with $\mathsf{A}(\langle \mathbb{P}_{xy}^{X^*} \rangle_{y \neq x}) = r$, that is $\mathsf{A}(\langle \mathbb{P}_{xy}^{X^*} \rangle_{y \neq x}) = f(\{\mathsf{A}(\langle \mathbb{P}_{xy}^{Y^*} \rangle_{y \neq x}) \mid Y^* \in \mathbb{X}^*\})$. Finally, $X^* \cap \mathbf{Tr}_x$ is the unique maximal consistent theory selected by $\gamma_{(f,\mathsf{A},\mathbb{P})}$, so this function induces a maxichoice revision operator.

Alternatively, if we are given a lexicographical ordering $<_{\mathrm{lex}}$ (in place of previous A), then $f \in \{\max_{\mathrm{lex}}, \min_{\mathrm{lex}}\}$ may be introduced to induce selection functions which will give rise, as before, to maxichoice revision operators. We will define lexicographical orderings from reference base Z and a class of $\mathbb{P}$-parameters. Then, the resulting selection function (to be defined) will be called $\gamma_{(f',\emptyset,\mathbb{P})_Z}$. This function may be shown to induce a maxichoice revision operator. Again, a natural choice for Z, according to the *minimize information loss* principle, is the pre-revision belief state $Z = F[\mathbf{T}_x]$.

Proposition 4. *For* $f \in \{\min, \max\}$ *and a base* Z, *we have that* $\gamma_{(f',\emptyset,\mathbb{P})_Z}$ *induces a maxichoice revision operator.*

Proof. Let $(f, \emptyset, \mathbb{P})_Z$ be as above for some base Z; observe that for any X, Y maximal consistent subsets, by their maximality, exist $y_0, y_1 \in \mathsf{Ag} \smallsetminus \{x\}$ such that $\mathbb{P}_{xy_0}^{X^*} > \mathbb{P}_{xy_0}^{Y^*}$ while $\mathbb{P}_{xy_1}^{X^*} < \mathbb{P}_{xy_1}^{Y^*}$. Now, we order each $\langle \mathbb{P}_{xy}^{(\cdot)} \rangle$, matching this order (upon agents) with that of $\mathbb{P}_{xy}^{Z}$, when the latter is decreasingly ordered, the result being possibly non-decreasing. In any case, the above consequence of maximality induces a lexicographic ordering of X^* sets in $\mathbb{X}^*$; this lexicographic ordering is translated to maximal consistent subset by means of the mapping $X^* \longmapsto X = X^* \cap \mathbf{Tr}_x$. Let this second lexicographic ordering be denoted by $<_{\mathrm{lex}}^*$. Obviously, taking the f-value on $<_{\mathrm{lex}}^*$ makes $(f', \emptyset, \mathbb{P})_Z$ to select a unique element: consider the $f = \max$ case first; for $X, Y \in \mathrm{Con}(\mathbf{Tr}_x, F[\mathbb{T}])$, we have

$$\{X^*\} = \gamma_{(\max,\emptyset,\mathbb{P})_Z}(\mathrm{Con}(\mathbf{Tr}_x, F[\mathbb{T}])), \quad \text{where } X^* \text{ is maximum w.r.t. } <_{\mathrm{lex}}^*$$

This $\gamma_{(\max,\emptyset,\mathbb{P})}$ is a selection function (it is non-empty, since $\mathrm{Con}(\mathbf{Tr}_x, F[\mathbb{T}])$ is not empty either), and since the element selected is unique this selection function induces a maxichoice revision operator. The proof of case $f = \min$ is similar: simply replace *maximum* by *minimum* in the previous set.

2.3 Examples of Aggregation-Based Revision Operators

We consider the next criteria (1) $(\max, \max, \alpha)$ i.e. $f = \mathsf{A} = \max$, and $\alpha_{xy}^X = |[B_x]Tr_{xy}|_X$ inducing revision operators that opt for maximizing the trust upon the maximally trusted agent in the resulting revision; and (2) $(\min, \mathrm{mean}, \beta)$, for

$\beta^X_{xy} = |[B_x]Tr_{xy}|_T - |[B_x]Tr_{xy}|_X$, to generate revision operators that minimize the mean of the differences in trust before and after revision. Observe that max is associative but *mean* is not: (1) induces revision operators unsensitive to order and simultaneity of inputs; while this need not to be the case for (2), in principle; although to properly implement (2) agent x should not revise her belief states after each new input is received. Doing so would make $(\min, \text{mean}, \beta)$ to collapse into the former $(\max, \max, \alpha)$ [14]. This is shown in the first example. As mentioned before, we will assume in the next examples that bases are closed under (a finite but sufficient set of) truth-degrees: if $\overline{r} \to_{\text{Ł}} [B_x]\varphi \in T$ then for each $s < r$, $\overline{s} \to_{\text{Ł}} [B_x]\varphi \in T$. (Doing otherwise just leads to more drastic revision operators.) In these examples, there is no internal conflict in the discourse of any agent $y \neq x$. Hence we can forget about the first revision step depicted above.

Example 1. Consider the base

$$T_0 = \{\ [B_x]Tr_{xy} \to_{\text{Ł}} [B_x]p,\ \overline{0.9} \to_{\text{Ł}} [B_x]Tr_{xy},\ \overline{0.8} \to_{\text{Ł}} [B_x]Tr_{xz_0},\ \overline{0.7} \to_{\text{Ł}} [B_x]Tr_{xz_1}\ \}, \text{ and inputs}$$

$$\varphi_0 = [B_x]Tr_{xz_0} \to_{\text{Ł}} [B_x]\neg p \text{ and } \varphi_1 = [B_x]Tr_{xz_1} \to_{\text{Ł}} [B_x]\neg p$$

(a) Observe criterion $\mathbb{C} = (\max, \max, \alpha)$ leads to $|\psi|_{(T_0 \circledast_{\mathbb{C}} \varphi_0) \circledast_{\mathbb{C}} \varphi_1} = |\psi|_{T_0}$, for ψ not being a trust sentence about z_0 or z_1; i.e. revision preserves the rest of the base. This is so because neither z_0 nor z_1 can individually beat y. As a consequence, trust upon z_0 and z_1 is lowered to $\overline{0.1}$.

(b) The same holds for $\mathbb{C}' = (\min, \text{mean}, \beta)$, if x revises as soon as a new input is received. Let $T' := T \circledast_{\mathbb{C}'} \varphi_0$. We have T' makes x's trust upon z_0 be set to $\overline{0.1} \to_{\text{Ł}} [B_x]Tr_{xz_0}$. On second input, $T' \circledast_{\mathbb{C}'} \varphi_1$, x joint trust upon z_0 and z_1 cannot beat that upon y.

(c) Finally, consider (b)'s criterion $\mathbb{C}'$ but now with simultaneous (i.e. multiple) revision of T by input $\{\varphi_0, \varphi_1\}$. In this case, x must choose between a loss of $\frac{.9+0+0}{3}$ units of trust or a loss of $\frac{0+(.8-.1)+(.7-.1)}{3}$ units of trust. Since the later means a minimum loss, z_0 and z_1 beat y, who looses credibility from x's point of view.

Another important question is how does a new communication of trust affect old communications. Assume z_0 and z_1 happen to disagree about p. The next example shows there is an important difference between $\psi = y$ *recommending* z_1 (to x) and $\psi' = y$ *disrecommending* z_0 (to x). The latter, as a revision input, does not make x to recover an old high degree of trust in z_1 (this is case $(T \circledast \varphi) \circledast \psi'$) *unless* y's disrecommendation of z_0 takes place before z_1's conflicting message about p (case $(T \circledast \psi') \circledast \varphi$). On the other side, this example makes no difference between $\mathbb{C}$ and $\mathbb{C}'$.

[14] Instead, in order to implement revision under $(\min, \text{mean}, \beta)$ it is recommendable to wait until several inputs have been gathered so as to have better grounds for deciding.

Example 2. Let

$$T = \{\ \overline{0.9} \to_{\text{Ł}} [B_x]Tr_{xy}, \overline{0.7} \to_{\text{Ł}} [B_x]Tr_{xz_0},\ \overline{0.6} \to_{\text{Ł}} [B_x]Tr_{xz_1}, [B_x]Tr_{xz_0} \to_{\text{Ł}} [B_x]\neg p\ \}, \text{ and inputs}$$

$$\varphi = [B_x]Tr_{xz_1} \to_{\text{Ł}} [B_x]p,\ \psi = [B_x]Tr_{xy} \to_{\text{Ł}} (\overline{0.9} \to_{\text{Ł}} [B_x]Tr_{xz_1}),$$
$$\text{and } \psi' = [B_x]Tr_{xy} \to_{\text{Ł}} ([B_x]Tr_{xz_0} \to_{\text{Ł}} \overline{0.4})$$

In each of the next sequences of revisions, x's trust upon z_0, z_1 becomes

$$\begin{array}{llll} |[B_x]Tr_{xz_1}|_{T\circledast\varphi} & = 0.3 & & \\ |[B_x]Tr_{xz_1}|_{(T\circledast\varphi)\circledast\psi} & = 0.8 & |[B_x]Tr_{xz_0}|_{(T\circledast\varphi)\circledast\psi} & = 0.2 \\ |[B_x]Tr_{xz_0}|_{(T\circledast\varphi)\circledast\psi'} & = 0.5 & |[B_x]Tr_{xz_1}|_{(T\circledast\varphi)\circledast\psi'} & = 0.3 \\ |[B_x]Tr_{xz_1}|_{(T\circledast\psi')\circledast\varphi} & = 0.6 & |[B_x]Tr_{xz_0}|_{(T\circledast\psi')\circledast\varphi} & = 0.4 \end{array}$$

3 Conclusions and Future Work

We studied revision-theoretic properties of a probabilistic (uncertainty) graded logic, with propositional atoms for trust and modalities for (communicated) belief. This was but an example of how can we define a trust-based method of iterated belief revision (on top of some uncertainty logic) by imposing an aggregation operator to generate revision operators guiding belief change in an automated way.

We hope this work can throw some light into the area of iterated belief revision by providing classes of examples from which axioms for iterated change (other than those from Darwiche and Pearl's [4]) can be extracted. Several improvements of our proposal can also be devised for future work, including updates and topic-sensitive trust. The study of protocols for sending messages to other agents would make the system fully autonomous in its implementation. Allowing for nested communications or splitting Trust into Sincerity and Credibility could improve agents' epistemic capabilities as well as their performance.

References

1. Alchourrón, C., Gärdenfors, P., Makinson, D.: On the Logic of Theory Change: Partial Meet Contraction and Revision Functions. The Journal of Symbolic Logic 50, 510–530 (1985)
2. Booth, R., Richter, E.: On Revising Fuzzy Belief Bases. Studia Logica 80, 29–61 (2005)
3. Calvo, T., Mayor, G., Mesiar, R. (eds.): Aggregation Operators. Physica-Verlag; Springer-Verlag (2002)
4. Darwiche, A., Pearl, J.: On the logic of iterated belief revision. Artificial Intelligence 89(1,2), 1–29 (1997)
5. Delgrande, J., Dubois, D., Lang, J.: Iterated revision as prioritized merging. In: Proceedings of KR-2006, pp. 210–220 (2006)
6. Dellunde, P., Godo, L.: Introducing grades in deontic logics. In: van der Meyden, R., van der Torre, L. (eds.) DEON 2008. LNCS (LNAI), vol. 5076, pp. 248–262. Springer, Heidelberg (2008)

7. Demolombe, R., Liau, C.J.: A Logic of Graded Trust and Belief Fusion. In: Proc. 4th Workshop on Deception, Fraud and Trust in Agent Societies, Montréal (2001)
8. Dragoni, A.F., Giorgini, P.: Revising beliefs received from multiple sources. In: Williams, M.A., Rott, H. (eds.) Frontiers in belief revision, vol. 22. Kluwer, Dordrecht (2001)
9. Dubois, D., Prade, H.: Possibility theory, probability theory and muliple-valued logics: A clarification. Annals of Math. and Art. Intel. 32, 35–66 (2001)
10. Esteva, F., Gispert, J.: Adding truth-constants to logics of continuous t-norms: axiomatization and completeness results. Fuzzy Sets and Systems 185, 597–618 (2007)
11. Flaminio, T., Pinna, G.M., Tiezzi, E.B.P.: A complete fuzzy logical system to deal with trust management systems. Fuzzy Sets and Systems 159, 1191–1207 (2008)
12. Font, J.M., Jansana, R.: A General Algebraic Semantics for Sentential Logics. Lecture Notes in Logic, vol. 7. Springer, Heidelberg (1996)
13. Godo L.l., Hájek P. and Esteva F.: A Fuzzy Modal Logic for Belief Functions. Fundamenta Informaticae, 1001–1020 (2001)
14. Hájek, P.: Metamatematics of Fuzzy Logic. Trends in Logic, vol. 4. Kluwer, Dordrecht (1998)
15. Halpern, J., Fagin, R.: Reasoning about Knowledge and Probability. Journal of the ACM 41(2), 340–367 (1994)
16. Hansson, S.o.: Ten philosophical problems in belief revision. Journal of Logic and Computation 13(1), 37–49 (2003)
17. Konieczny, S., Lang, J., Marquis, P.: DA^2 merging operators. Artificial Intelligence 157, 49–79 (2004)
18. Liau, C.J.: Belief, information acquisition, and trust in multi-agent systems - A modal logic formulation. Artificial Intelligence 149, 31–60 (2003)
19. Maynard-Reid II, P., Shoham, Y.: Belief Fusion: Aggregating Pedigreed Belief States. Journal of Logic, Language and Information 10, 183–209 (2001)
20. Ognjanovic, Z., Raškovic, M.: Some first-order probability logics. Theoretical Computer Science 247, 191–212 (2000)
21. Pardo, P.: Base Belief Revision for Finitary Monotonic logics. In: Proceedings of the ESSLLI 2009 Student Session (2009)
22. Peppas, P.: Belief Revision. In: van Harmelen, F., Lifschitz, V., Porter, B. (eds.) Handbook of Knowledge Representation. Elsevier, Amsterdam (2007)
23. Sierra, C., Sabater-Mir, J.: Review on Computational Trust and Reputation Systems. Artificial Intelligence Review 24, 33–60 (2005)
24. Torra, V., Narukawa, Y.: Information Fusion and Aggregation Operators. Cognitive Technologies Series. Springer, Heidelberg (2007)

Fast and Accurate Prediction of the Destination of Moving Objects

Austin Parker[1], V.S. Subrahmanian[1], and John Grant[1,2]

[1] University of Maryland, College Park, MD 20742, USA
[2] Towson University, Towson, MD 21252, USA

Abstract. Companies and organizations that track moving objects are interested in predicting the intended *destination* of these moving objects. We develop a formal model for *destination prediction problems* where the agent (Predictor) predicting a destination may not know anything about the route planning mechanism used by another agent (Target) nor does the agent have historical information about the target's past movements nor do the observations about the agent have to be complete (there may be gaps when the target was not seen). We develop axioms that any destination probability function should satisfy and then provide a broad family of such functions guaranteed to satisfy the axioms. We experimentally compare our work with an existing method for destination prediction using Hidden Semi-Markov Models (HSMMs). We found our algorithms to be faster than the existing method. Considering prediction accuracy we found that, when the Predictor knows the route planning algorithm the target is using, the HSMM method is better, but without this assumption our algorithm is better.

1 Introduction

Numerous applications require the prediction of the destinations of various moving objects. Maritime security organizations are interested in tracking maritime traffic, determining where they are going, and rapidly making an assessment of the threat posed by those vessels. Recent events involving Somali pirates in the Gulf of Aden, and the arrival by sea of terrorists in Mumbai further highlight the need to rapidly predict where vessels are headed and then make decisions based on those determinations. In a similar vein, the ability to predict destinations of cell phones is critical to determining and balancing loads across multiple cell towers. Each of these situations requires a solution to the simpler problem of determining an agent's intended destination based on information about their current path. Throughout this paper, we will refer to the entity whose destination we seek to predict as the *target* and the entity performing the prediction as the *predictor*. We will further assume the target is unaware of the predictor's operation and is not willfully trying to deceive the predictor (such an adversarial treatment of the problem is left as future work).

All of the above examples have three important characteristics. First, little if any information might be available to the predictor on the actual planning mechanisms used by a target. Does a specific target just make up a route manually? Does the target use a route planning service (such as Google Maps or Via Michelin for land vehicles) for planning purposes? *The important point is that the route planning mechanism used*

L. Godo and A. Pugliese (Eds.): SUM 2009, LNAI 5785, pp. 180–192, 2009.

by the target is usually unknown to the predictor. Second, the observations of the target may be spotty or incomplete: the target may be spotted only at irregular intervals. Third, there may be little if any historical data about the target's past movements from which one can build a model of the target's preferred movement.

In this paper, we develop a formal, axiomatic theory to address the destination prediction problem under these assumptions. We start with a simple motivating example and some basic definitions in Section 2, Then, in Section 3, we formally define a destination probability function as any function that satisfies certain axioms. We explain why these axioms make sense. In order to define specific destination probability functions, we often need to examine the distance between an *observed* route that the target has taken, and some other route. In Section 4, we also define axioms that any function to measure distance between routes should satisfy in order to be applicable to destination prediction, and then we go on to provide definitions of four route distance functions that satisfy these very simple axioms. In Section 5, we use the route distance functions to define a class of destination prediction functions that satisfy all the destination prediction axioms from Section 3. Finally, in Section 6, we report on a prototype implementation and a suite of experiments comparing these methods — not only with each other, but also with the only other algorithm we are aware of in the literature for destination prediction under our assumptions. We show that several of our algorithms significantly outperform these algorithms when the predictor does not know the route planning algorithm used by the target — however, when the predictor knows the route planner used by the target, [1]'s algorithm outperforms ours. As a consequence, a combination of the two methods could provide a powerful predictive mechanism depending on whether or not the route planner used by the target is known.

2 Preliminary Definitions

Throughout this paper, we assume that all objects are present in an $N \times N$ space S where N is the set of all non-negative integers. We assume that S is discrete – thus S includes the point $(3, 5)$ but not $(2.5, 2.7)$. We assume the existence of a distance function $d : S \times S \rightarrow \mathbb{R}$ specifying the physical distance which must be traveled to get from one point to another. We say that two points are *adjacent* if the Manhattan distance between them is one[1]. We also assume that there is a set $\mathcal{D} \subseteq S$ of *possible destinations*. Each point in $\mathcal{D}$ may represent a location deemed to be worth visiting from the point of view of the target. For example, if the target appears to be engaged in tourism, $\mathcal{D}$ could include hotels, restaurants, tourist sights, airports, bus stations, train stations, ports, etc. We assume that the target has a destination in $\mathcal{D}$ unknown to the predictor; however, the predictor knows $\mathcal{D}$.

Definition 1 (Route). *A* route $\wp$ *is a sequence* $\langle p_1, p_2, ..., p_n \rangle$ *where all* $p_i \in S$. *We use* $\mathcal{R}$ *to denote the set of all routes. Route* $\wp$ *is said to be* full *iff* p_i *and* p_{i+1} *are adjacent for all* i, $1 \leq i < n$.

Basically a *route* is a list of points in space through which the target travels. Note that $\wp = \langle (1, 1), (3, 1), (4, 3) \rangle$ is a route, even though consecutive points are not adjacent.

[1] Manhattan distance between (x_1, y_1) and (x_2, y_2) is the L_1 norm, or $|x_1 - x_2| + |y_1 - y_2|$.

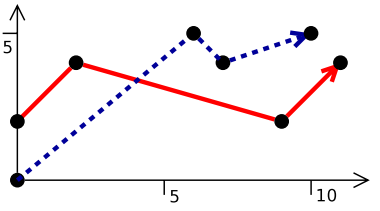


Fig. 1. Pictured here are two example routes: $\wp_1$ in the solid line is $\langle(0,2),(4,4),(9.2),(11,4)\rangle$, and $\wp_2$ in the dotted line is $\langle(0,0),(6,5),(7,4),(10,5)\rangle$

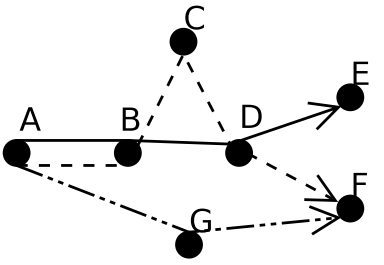


Fig. 2. Illustration for counter-example to using the Monotonicity properties as axioms

This may simply reflect known information about the target's route, since not all points the target has visited may be recorded in a route. We specify complete ignorance as to the target's location with the empty route: $\langle\rangle$ – this route is used when there is no information about the target's past or present location.

We now define the concept of a subroute.

Definition 2 (Subroute). *A route $\wp = \langle p_1, p_2, \ldots, p_n\rangle$ is a subroute of a route $\wp' = \langle p'_1, p'_2, \ldots, p'_m\rangle$ iff there is a function $\alpha : \{1,\ldots,n\} \rightarrow \{1,\ldots,m\}$ such that for all i, $p'_i = p_{\alpha(i)}$ and $\alpha(i) < \alpha(i+1)$.*

For example, the route $\langle(1,1),(3,1),(4,3)\rangle$ is a subroute of the (full) route $\langle(1,1),(2,1),(3,1),(3,2),(3,3),(4,3)\rangle$.

Definition 3 (Prefix Route). *A route $\wp = \langle p_1, p_2, \ldots, p_n\rangle$ is a prefix of a route $\wp' = \langle p'_1, p'_2, \ldots, p'_m\rangle$ iff for $i = 1$ to n, $p_i = p'_i$.*

Thus far, a route has merely been defined as a sequence of points. But clearly not just any route is *feasible* for a given target. For instance, if the target is a boat, it is infeasible for it to traverse the middle of the Sahara desert. We assume the existence of a feasibility predicate feas that takes a route as input and either returns true or false. feas($\wp$) returns "true" iff the route is considered feasible. The feasibility predicate is required to satisfy the axiom: If $\wp$ is a subroute of $\wp'$ and feas($\wp'$) = true, then feas($\wp$) =true. Throughout this paper, we assume that the definition of feas is arbitrary, but fixed. We now define route planners as programs that return routes that the given target might use.

Definition 4 (Route Planner). *A route planner is a function $\mathcal{P} : S \times S \rightarrow 2^{\mathcal{R}}$ which takes an origin and a destination from space and returns a set of feasible full routes from the origin to the destination. If there is no feasible route from the origin to the destination, $\emptyset$ is returned. $\mathcal{P}$ must satisfy the following condition: if $\mathcal{P}(p_1,p_2) \neq \emptyset$ and $\mathcal{P}(p_2,p_3) \neq \emptyset$ then $\mathcal{P}(p_1,p_3) \neq \emptyset$.*

Some route planners return only a subset of the feasible routes. Examples of such route planners are Google Maps, Mapquest, Via Michelin, and other automated route planners

that return a single route (which can be interpreted as a singleton route set). One can imagine more and more inclusive route planners, which return larger and larger sets of feasible full routes. We assume all route planners always return a finite set of full routes.

Intuitively, $\mathcal{P}((x_1, y_1), (x_2, y_2))$ returns feasible routes from (x_1, y_1) to (x_2, y_2). The transitivity requirement in the definition makes sense because if the target has a way of getting from p_1 to p_2 (making $\mathcal{P}(p_1, p_2)$ non-empty) and a way of getting from p_2 to p_3 (making $\mathcal{P}(p_2, p_3)$ non-empty), then it should have a way of getting from p_1 to p_3 (making $\mathcal{P}(p_1, p_3)$ non-empty). Route planners can be implemented in many ways — the above definition is rich enough to allow us to think of Google Maps, Mapquest, Via Michelin and other online route planning sites as route planners. More traditional route planners such A* search and variants [2] also fit our definition. Likewise, flight planning programs would be route planners as well (if we were to extend our treatment to 3-dimensional spaces). Even a person constructing routes by hand can be considered a route planner!

In this paper, we consider the *Destination Probability Function* problem.

DPF Problem: Given a route $\wp$ that a target has been seen taking, and given a potential destination ℓ, what is the probability that the target is going to destination ℓ?

3 Destination Prediction Axioms

As we have already seen, there are many different route planners. There are also many different ways in which one could try to define a destination probability function.

Definition 5 (Destination Probability Function). $\mu : \mathcal{R} \times \mathcal{D} \rightarrow [0, 1]$ *is a destination probability function.*

$\mu(\wp, \ell)$ is the probability that location ℓ is the destination of an object given that the object has traveled the route $\wp$. When applied to the null route $\langle\rangle$, μ gives an *a priori* distribution over possible destinations, specifying where an object is headed given no information on the object's location or path.

We want to identify "reasonable" properties that any destination probability function should satisfy.

DP1 (PDF Axiom) For all feasible routes $\wp$, $\sum_{\ell \in \mathcal{D}} \mu(\wp, \ell) = 1$.
DP2 (*a priori* Axiom) For all destinations $\ell \in \mathcal{D}$, $\mu(\langle\rangle, \ell) > 0$.
DP3 (Reachability Axiom) For all routes $\wp$ with first location p_1 and all $\ell \in \mathcal{D}$, $\mu(\wp, \ell) > 0$ iff $\wp$ is a feasible route from p_1 to ℓ.

Axiom **DP1** ensures that for every route that may be observed, μ is a probability distribution over $\mathcal{D}$. Axiom **DP2** ensures the exclusion of trivial destinations from the list of potential destinations i.e. only destinations that are possible *a priori* are allowed in the destination set. Axiom **DP3** ensures that any location that can be reached by a feasible route gets a nonzero probability, while when a location cannot be reached by a feasible route, the probability must be zero.

It turns out that destination probability functions satisfying these axioms do not always exist as we now demonstrate. Let $\mathcal{D} = \{\ell\}$ and suppose there exists no feasible

route from any $p \in S$, $p \neq \ell$ to ℓ (maybe ℓ is an island and the target cannot travel on water). Next suppose that there exists a destination probability function μ that satisfies the axioms. For any route $\wp$ not containing ℓ, $\mu(\wp, \ell)$ must be zero by **DP3**. Thus $\sum_{\ell' \in \mathcal{D}} \mu(\wp, \ell') = 0$ (since $\mathcal{D} = \{\ell\}$), and that violates **DP1**.

In the future we will avoid such an anomaly when there is no way to get to a destination and assume that for all $\mathcal{D}$ we encounter, there exists a DPF that satisfies **DP1**-**DP3**.

We now introduce the monotonicity properties that assert that a destination probability function must not become less sure of the correct destination as it is given more information about the agent's movement. These properties involve the intuition that the probability of a given destination being an agent's destination should not decrease as the agent travels a path towards that destination.

Definition 6 (Monotonicity Properties)

Weak Monotonicity. *For all points p and all locations ℓ if $\wp = \langle p_1, \ldots, p_n \rangle \in \mathcal{P}(p_1, \ell)$, then for all i s.t. $1 \leq i < n$, $\mu((p_1, \ldots, p_i), \ell) \leq \mu((p_1, \ldots, p_{i+1}), \ell)$.*

Strong Monotonicity. *For all points p and all locations ℓ if $\wp = \langle p_1, \ldots, p_n \rangle \in \mathcal{P}(p, \ell)$, then for all subroutes $\wp'$ and $\wp''$ of $\wp$, where $\wp'$ is also a subroute of $\wp''$, $\mu(\wp', \ell) \leq \mu(\wp'', \ell)$.*

The monotonicity properties are segregated from the DPF axioms because there are sensible DPFs that do not satisfy these properties. Consider Figure 2 to see an example case where we may not want either monotonicity axiom to hold. According to both axioms, the probability of traveling to destination F given route $\langle A, B \rangle$ must be equal or higher than the route $\langle A \rangle$ (since $\langle A \rangle$ is a subroute of $\langle A, B \rangle$ and both are prefix routes of $\langle A, B, C, D, F \rangle$). However, it may be that the agent is known to be very likely to travel route $\langle A, G, F \rangle$ when traveling to F, implying that when given the route $\langle A, B \rangle$ the destination F is actually less likely than when given the route $\langle A \rangle$.

4 Route Distance Functions

One general method of finding the probability that a given location ℓ is the destination of a target that has been observed to travel route $\wp = (p_1, \ldots, p_n)$ is the following. We use some route planner $\mathcal{P}$ to get a set of possible routes from p_1 to ℓ. Let us suppose that $\mathcal{P}(p_1, \ell) = \{\wp_1, \ldots, \wp_k\}$. At this stage, we define the "similarity" between the observed route $\wp$ taken by the target and then aggregate the similarities between $\wp$ and each of $\wp_1, \ldots, \wp_k$ to identify a net probability that the target is going towards ℓ.

To define similarity, we define the dissimilarity between routes with a "route distance function" that gives a numeric distance between two routes, and use fractions composed of these numeric distances as similarity. We define a route distance function to be something that tells the distance between two routes and satisfies some simple axioms.

Definition 7 (Route Distance Function). *A* route distance function *is a function $\psi : \mathcal{R} \times \mathcal{R} \rightarrow \mathbb{R}$ satisfying the following axioms. (B is a constant.)*

RD1 *For all routes* $\wp$, $\psi(\wp, \wp) = 0$.
RD2 *For all routes* $\wp_1$ *and* $\wp_2$, $\psi(\wp_1, \wp_2) \geq 0$.
RD3 *For all routes* $\wp_1, \wp_2$, $\psi(\wp_1, \wp_2) \leq B$.

Axiom **RD1** states that a route's distance from itself is 0. Axiom **RD2** enforces the non-negativity of the route distance function. Finally, axiom **RD3** states that the distance function is bounded. This boundedness axiom is reasonable because our space S is finite.

More notable than the axioms we chose to include are the axioms we left out. In particular, one might expect a distance function to satisfy both symmetry and the triangle inequality. For the purposes of destination prediction, both properties may be undesirable, as we now show.

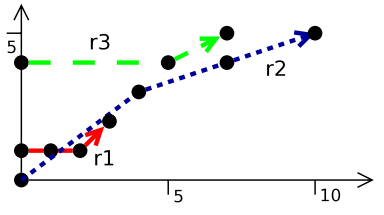


Fig. 3. Example routes demonstrating why both symmetry and the triangle inequality may be undesirable properties for a route distance function

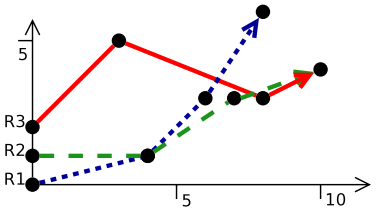


Fig. 4. A set of example routes where, for the purposes of destination prediction, one would want to consider R2 to be closer to R3 than to R1

Symmetry: Suppose we have seen an object travel route $r1$ in Figure 3, and suppose we want to compare it to a path $r2$ from route planner $\mathcal{P}$. In this case $r1$ should be fairly "close" to $r2$: the entirety of the $r1$ path is near to $r2$. However, if we have seen the object travel the route $r2$, and the route planner gives us $r1$ as a potential route we want the distance from $r2$ to $r1$ to be quite large. An object traveling $r2$ is pretty unlikely to have the last point in $r1$ as its intended destination.

Triangle Inequality: Again using the routes in Figure 3, consider the triangle inequality: $\psi(r1, r3) \leq \psi(r1, r2) + \psi(r2, r3)$. However, the distance from $r1$ to $r2$ should be small (it is reasonable for an object traveling $r1$ to be intending a route like $r2$), and the distance from $r2$ to $r3$ should be small (they run parallel to one another towards the end). But again intuitively, the distance from $r1$ to $r3$ should be large: they seem the most "distant" two routes in the diagram. So we do not include the triangle inequality as an axiom. It may, however, be used in a discretionary manner depending upon the characteristics of an application.

In the rest of this section, we will define some alternative methods to compute the distance between routes.

Naive Route Distance Function: The first route distance function is naive and is included as a baseline example. The distance between a point p_i in the first route and the

entire second route is defined to be the distance between p_i and the closest point to it in the second route. The distance between the first route and the second route is then obtained by averaging the distances between each point in the first route and the entire second route.

Definition 8 (Naïve Route Distance). *For* $\wp = \langle p_1, \ldots, p_n \rangle$ *and* $\wp' = \langle p'_1, \ldots, p'_m \rangle$ *the* naive route distance *is:*

$$NRD(\wp, \wp') \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^{n} \min_{j \in \{1,\ldots,m\}} \left(d(p_i, p'_j)\right)$$

Weighted Distance Function: One potential improvement on the naïve route distance function involves weighting the summation according to the fraction of the path the point is responsible for.

Definition 9 (Weighted Route Distance). *For routes* $\wp = (p_1, \ldots, p_n)$ *and* $\wp' = (p'_1, \ldots, p'_m)$ *the weighted route distance is:*

$$WRD(\wp, \wp') \stackrel{\text{def}}{=} \frac{1}{2 \cdot d(\wp)} \sum_{i=1}^{n} \left[\begin{pmatrix} d(p_{i-1}, p_i) \\ +d(p_i, p_{i+1}) \end{pmatrix} \min_{j \in \{1,\ldots,m\}} (d(p_i, p'_j) \right]$$

(for $p_i = p_1$ *when* $i < 1$*, and* $p_i = p_n$ *when* $i > n$*, and* $d(\wp) = \sum_{i=1}^{n} d(p_i, p_{i+1})$*).*

In order to understand this, note that $(d(p_{i-1}, p_i) + d(p_i, p_{i+1}))$ is the amount of distance in route $\wp$ in which p_i is involved. As each p_i participates in two terms, dividing this quantity by $2 \cdot d(\wp)$ gives the contribution of p_i (in terms of distance) to the total distance in route $\wp$. This serves to weight the distance from that point appropriately. If p_1 is responsible for very little of $\wp$, then its distance should be taken into account proportionally less than if it is responsible for a large piece of $\wp$.

Discounted Route Distance: So far we have only considered route distance functions that treat the distance between paths regardless of where the distances between the paths occur. For instance, consider the routes in Figure 4. In this figure we have three routes, R1, R2, and R3. R2 and R3 have the same destination, however, according to all the route distance functions considered thus far, R1 is "closer" to R2 than R3. We employ the notion of a discount factor to alleviate this issue, and define a route distance function where early similarity between routes is worth less than late similarity.

Definition 10 (Discounted Route Distance). *For routes* $\wp = (p_1, \ldots, p_n)$ *and* $\wp' = (p'_1, \ldots, p'_m)$ *and* $\delta \leq 1$ *(a discount factor):*

$$DRD_\delta(\wp, \wp') = \frac{1}{n} \sum_{i=1}^{n} \delta^{n-i} \min_{j \in \{1,\ldots,m\}} d(p_i, p'_j).$$

Example 1. In Figure 4 we see three routes: $R1 = ((0,0),(4,1),(6,3),(8,6))$, $R2 = ((0,1),(4,1),(7,3),(10,4))$, and $R3 = ((0,2),(3,5),(8,3),(10,4))$. The discounted route distance from $R1$ to $R2$ is: $\frac{1}{4}\left(\delta^3 \cdot 1 + \delta^2 \cdot 0 + \delta^1 \cdot 1 + \delta^0 \cdot 2.83\right)$, and the discounted route distance from $R3$ to $R2$ is: $\frac{1}{4}\left(\delta^3 \cdot 1 + \delta^2 \cdot 4.12 + \delta^1 \cdot 1 + \delta^0 \cdot 0\right)$. When

$\delta = 0.9$ the first value is 1.11, and the second value is 1.24, and when $\delta = 0.6$ the first value is 0.91 and the second value is 0.57. So with the higher δ, $R1$ is closer to $R2$ than $R3$ is to $R2$; however, with lower $\delta = 0.6$, $R3$ is closer to $R2$ than $R1$ is to $R2$, since with lower δ, the initial portion of the routes count proportionally less towards the total distance.

We can also weight the points in the discounted route distance as in the weighted route distance function.

Definition 11 (Discounted Weighted Route Distance). *For routes $\wp = (p_1, \ldots, p_n)$ and $\wp' = (p'_1, \ldots, p'_m)$ and $\delta \leq 1$ (a discount factor):*

$$DWRD_\delta(\wp, \wp') \stackrel{\text{def}}{=} \frac{1}{2d(\wp)} \sum_{i=1}^{n} \left[\delta^{n-i} \begin{pmatrix} d(p_{i-1}, p_i) \\ +d(p_i, p_{i+1}) \end{pmatrix} \min_{j \in \{1,\ldots,m\}} (d(p_i, p'_j) \right]$$

(for $p_i = p_1$ when $i < 1$, and $p_i = p_n$ when $i > n$, and $d(\wp) = \sum_{i=1}^{n} d(p_i, p_{i+1})$).

The following result states that all the route distance functions provided thus far satisfy the axioms for a function to be considered a route distance function.

Proposition 1. *Suppose $0 \leq \delta \leq 1$. Then: NRD, WRD, DRD_δ and $DWRD_\delta$ are all valid route distance functions, i.e. they satisfy axioms **RD1**, **RD2**, and **RD3**.*

5 Destination Probability Function

Given a route distance function ψ and a route planner $\mathcal{P}$, we are now in a position to construct a destination probability function. We define one family of such functions, parameterized by a specific distance function, in this section. We point out that many different destination probability functions exist, and we do not mean to suggest that the construction below is best (though it will be shown to satisfy the destination prediction axioms).

Given a route $\wp$, we can find the minimum route distance to routes returned by $\mathcal{P}$ for a given destination. By normalizing that value over all destinations, we now have what we call a *route distance destination probability function*.

Definition 12 (Route Distance Destination Probability Function). *Suppose ψ is a route distance function, $\mathcal{P}$ is a route planner, and route $\wp = \langle p_1, \ldots, p_n \rangle$. We define a route distance destination probability function $\mu_{\psi,\mathcal{P}} : \mathcal{P} \times \mathcal{D} \to [0, 1]$ as follows:*

$$\mu_{\psi,\mathcal{P}}(\wp, \ell) \stackrel{\text{def}}{=} \begin{cases} 0 & \text{if } \mathcal{P}(\wp, \ell) = \emptyset \\ \frac{\max_{\wp' \in \mathcal{P}(p_1,\ell)} (\psi(\wp,\wp')^{-1})}{\sum_{\ell' \in \mathcal{D}} \max_{\wp' \in \mathcal{P}(p_1,\ell')} (\psi(\wp,\wp')^{-1})} & \text{otherwise} \end{cases}$$

First, if there is no $\wp' \in \mathcal{P}(p_1, \ell)$, then the expression equals zero and the probability of ℓ being the intended destination likewise becomes zero. This makes sense since in this case there is no route from p_1 to ℓ, so how could ℓ be the destination? Otherwise, since $\psi(\wp, \wp')$ gives the distance between routes $\wp, \wp'$, $\psi(\wp, \wp')^{-1}$ is one way of measuring

similarity (inverse of distance) between these two routes[2]. Thus, $\mu_{\psi,\mathcal{P}}$ computes the similarity between the target's observed route and the best route to destination ℓ and divides this by the sum of the same similarity for every possible destination. The simple example below illustrates this.

Example 2. Let $\mathcal{D} = \{\ell, \ell'\}$ be a set of just two destinations and suppose $\mathcal{P}$ returns two routes r_1, r_2 between p_1 and ℓ where the target's observed route is $\wp = p_1, p_2 \ldots$. Suppose the distances between the routes are given as follows: $\psi(\wp, r_1) = 100, \psi(\wp, r_2) = 600$. In this case, r_1 is closer to $\wp$, so the numerator in the definition of μ becomes $(100)^{-1}$. This is a measure of the similarity between route $\wp$ and the best route to destination ℓ. We likewise compute the similarity between route $\wp$ and the best route to ℓ'. Suppose there are two routes r_3, r_4 from p_1 to ℓ' and that $\psi(\wp, r_3) = 300, \psi(\wp, r_4) = 400$. Then the similarity between route $\wp$ and the best route to ℓ' is given by $(300)^{-1}$. The destination probability function $\mu_{\psi,\mathcal{P}}$ computes the probability of the target going to ℓ as the ratio of the similarity of the target's observed route to the best route to ℓ to the sum of the similarities of the target's observed route to the best route to each possible destination. We obtain:

$\mu_{\psi,\mathcal{P}}(\wp, \ell) = \frac{(100)^{-1}}{(100)^{-1}+(300)^{-1}} = 0.75$, while $\mu_{\psi,\mathcal{P}}(\wp, \ell') = \frac{(300)^{-1}}{(100)^{-1}+(300)^{-1}} = 0.25$

The following result states that μ always satisfies the axioms to be a destination probability function.

Proposition 2. *For any route distance function ψ and any route planner $\mathcal{P}$ that returns at least one route between any two points where a feasible route exists, the route similarity destination probability function $\mu_{\psi,\mathcal{P}}$ satisfies Axioms* **DP1**-**DP3**.

The reader can imagine a weighted version of the above destination probability function which weights the minimum distance to each location by an *a prior* probability of the location being the intended destination. Examination of such functions and the increased predictive capacity are reserved for future work.

6 Experiments

We built a prototype of this system on top of the HOG system [2] that provides 54 different maps as well as several pathfinding algorithms. In our experiments, we select a number of destinations on a map, and create a target that travels to the destination using A* search and then try to measure the predictive accuracy of our different algorithms.

After every move by the agent, we computed the expected distance from the predicted destination to the actual destination as follows. Let ℓ_d be the agent's actual destination and μ be the probability distribution returned by the destination predictor. *The expected distance from prediction* is then defined as: $dis(\mu, \ell_d) = \sum_{\ell \in \mathcal{D}} d(\ell, \ell_d) \cdot \mu(\ell)$. The smaller this measure, the better the algorithm's predictive power.

The running times reported are the average per-step running time. The algorithms in [1] require some pre-computation time that is not required by the algorithms presented

[2] To make this method work in practice, we enforce some small lower bound ϵ on the distance between two routes, so as to avoid division-by-zero errors when, for instance, $\wp$ equals $\wp'$.

Table 1. The expected distance of prediction and running time of various destination prediction techniques when all techniques are given access to the same planner as the agent uses

Algorithm	exp. dis. of prediction	running time (ms)
HSMM	4.629726	34
WRD	5.928825	8
DWRD (0.9)	6.041613	8
DWRD (0.6)	6.283690	8
NRD	7.830972	8
DRD (0.9)	9.041494	8
DRD (0.6)	11.788955	8

Table 2. The expected distance of prediction and running time of various destination prediction techniques when all techniques are given access to a different planner than the agent uses

Algorithm	exp. dis. of prediction	running time (ms)
DWRD (0.6)	10.301018	6
DWRD (0.9)	10.641678	6
WRD	11.097869	6
NRD	12.467360	6
DRD (0.9)	12.613394	6
HSMM	13.075787	34
DRD (0.6)	14.068911	6

in this paper. We therefore include pre-computation time in the running time of the first prediction, but include only the time needed to query the HSMM for subsequent predictions so that the average times reported reflect the total computational cost of both methods.

Experiment 1 (Predictor and Target Use Same Route Planner). We first assumed that the predictor uses the same A* search algorithm as the target. These results are shown in Table 1, and are an average of over 1000 runs. They show the HSMM algorithm of [1] doing best, though taking longer to compute. The longer computation time of the HSMM approach is due to the cost of generating the HSMM, which is done as rarely as is possible in our experiments. If this generation cost is left out, the HSMM has a runtime competitive with the route distance approaches.

Experiment 2. (Target and Predictor use Different Route Planners). In a second experiment, we let $\mathcal{P}(p, \ell)$ be the result of PRA* search (from [2]), while the agent still uses standard A* search. We take this as an example of how the prediction algorithms can perform when we do not have access to exactly the same pathfinding algorithm used by the moving agent. We see from Table 2 that in this case, discounted weighted route distance provides the best prediction with a discount factor of 0.6.

One may note that the running times in experiment 2 are consistently smaller than those for experiment 1. This is because the destination predictors in experiment 2 use PRA*, which is a faster route planner than A*.

To investigate the difference between when the predictor knows the target's route planner and when it does not, we narrowed our focus to two individual maps. The first is the GRID map, shown in Figure 5. This map contains a grid of obstacles, and allows many potentially quite different routes from one point to another. The second is the CSC2F map, which resembles the inside of an office buildling. In the CSC2F map, there is more similarity between paths from point to point than in GRID, as the target must travel through doors and down specific corridors.

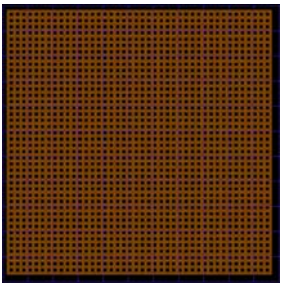


Fig. 5. The GRID map

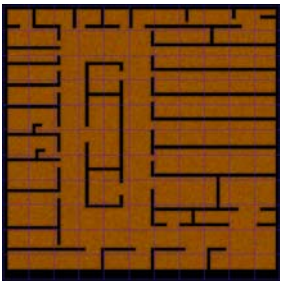


Fig. 6. The CSC2F map

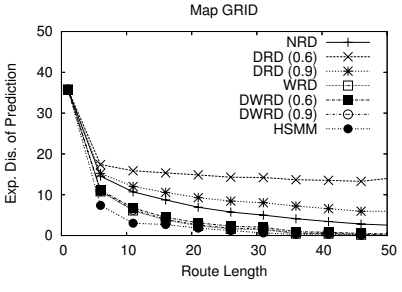


Fig. 7. Prediction accuracy by route length on GRID map when the Predictor knows Target's route planner

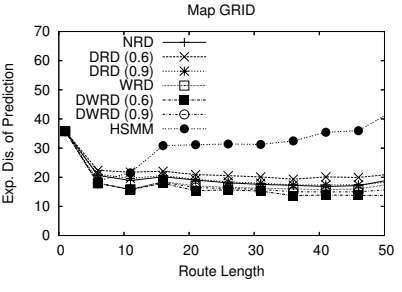


Fig. 8. Prediction accuracy by route length on GRID map when the Predictor does not know Target's route planner

In Figure 7 we see the expected distance of prediction as a function of the length of the input route for the GRID map when the Predictor is using the Target's route planner. Comparing this graph to Figure 8, where the Predictor uses a different route planner, we see a case where the HSMM predictor does substantially worse than the predictors investigated in this paper. This appears to be due to the large number of routes possible in the grid world between one point and another.

In CSC2F the variety of routes is limited by the existence of doorways and hallways, and in Figures 9 and 10 we see that for this map, HSMM remains competitive with the techniques provided in this paper even when it is not given the same route planner. This supports the hypothesis that HSMM is robust whenever the routes between points tend to be very similar.

7 Related Work and Conclusions

In this paper, we have developed a framework for predicting where a target might be going. We do this under (i) no knowledge by the predictor of the route planning program used by the target, (ii) routes that are potentially incomplete in the sense that observations of the target may be intermittent, and (iii) no information about past movements

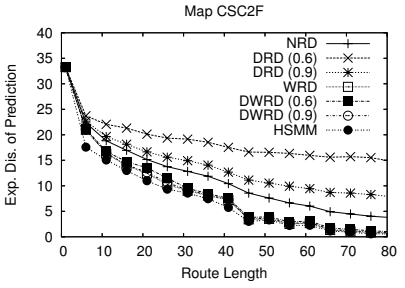


Fig. 9. Prediction accuracy by route length on CSC2F map when the Predictor knows the Target's route planner

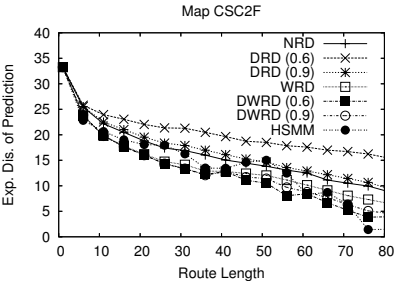


Fig. 10. Prediction accuracy by route length on CSC2F map when the Predictor does not know the Target's route planner

of the target. This latter part would make our methods suitable to predicting destinations on a one-time basis for a new target. We have provided a set of axioms defining a destination prediction function. In order to define specific destination prediction functions, we first came up with axioms to define distances between routes (measuring how dissimilar two routes are), and then proposed four such route distance functions. We showed how any route distance function (including the four we proposed) can be used to define a destination probability function satisfying the axioms. Finally, we implemented our algorithms and combined them with the only other system we know that does not require historical data about the moving object: [1]. Present results showing that when the route planner used by the target is known in advance by the predictor, [1] is slower, but more accurate than our algorithms; otherwise, our algorithms are both faster and more accurate.

[1] proposed an HSMM based algorithm for assigning probabilities to destinations. In the same paper, Southey *et al.* further address important issues relating to destination prediction such as multiple target identification and the use of graph abstraction techniques in prediction. Our work expands upon theirs by providing an axiomatic foundation for destination prediction and by introducing general algorithms that do not require pre-computation and are more robust in environments with dissimilar routes.

There are several works using past GPS data for specific users in a probabilistic model to predict future locations or destinations for the given user [3,4,5,6]. Unlike our work, these techniques assume the existence of past GPS data for the specific individual to learn their respective models, and are therefore solving a different problem.

In [3], a hierarchical Markov model is developed to model a user's daily movements where different levels of the hierarchy describe different modes of transportation or user destinations. The parameters for the model can be learned from a given user's GPS logs. In [4], a system called *comMotion* is created for recognizing locations of import to a given user. Their system recognizes important locations and allows users to tag these locations and attach data (such as reminders, to-do lists and the like) for retrieval upon arrival at a given location. The technique in [5] involves clustering GPS data into "places" via a variant of k-means clustering, then creating a Markov model where the nodes are "places" and the transitions are defined by data-derived probabilities for

moving between the associated locations. They show their algorithm to consistently predict agent location in a real-world pilot study.

Other work centers on predicting where a mobile user might be in a wireless network, based on the base stations the user has contacted recently. Cheng *et al.* [7] survey some of the work in this area.

Acknowledgements

Researchers funded in part by ARO grant W911NF0910206, ONR grant N000140910685, and AFOSR grant FA95500510298. We thank the referees for their helpful comments.

References

1. Southey, F., Loh, W., Wilkinson, D.F.: Inferring complex agent motions from partial trajectory observations. In: Veloso, M.M. (ed.) IJCAI, pp. 2631–2637 (2007)
2. Bulitko, V., Sturtevant, N., Lu, J., Yau, T.: Graph abstraction in real-time heuristic search. Journal of Artificial Intelligence Research (JAIR) 30, 51–100 (2007)
3. Liao, L., Patterson, D.J., Fox, D., Kautz, H.: Learning and inferring transportation routines. Artificial Intelligence 171(5-6), 311–331 (2007)
4. Marmasse, N., Schmandt, C.: A User-Centered location model. Personal and Ubiquitous Computing 6(5), 318–321 (2002)
5. Ashbrook, D., Starner, T.: Using GPS to learn significant locations and predict movement across multiple users. Personal and Ubiquitous Computing 7(5), 275–286 (2003)
6. Krumm, J., Horvitz, E.: Predestination: Inferring destinations from partial trajectories. In: Dourish, P., Friday, A. (eds.) UbiComp 2006. LNCS, vol. 4206, pp. 243–260. Springer, Heidelberg (2006)
7. Cheng, C., Jain, R., van den Berg, E.: Location prediction algorithms for mobile wireless systems. In: Wireless internet handbook: technologies, standards, and application, pp. 245–263. CRC Press, Inc., Boca Raton (2003)

Weighted Description Logics Preference Formulas for Multiattribute Negotiation

Azzurra Ragone[1], Tommaso Di Noia[1], Francesco M. Donini[2], Eugenio Di Sciascio[1], and Michael P. Wellman[3]

[1] SisInfLab, Politecnico di Bari, Bari, Italy
{a.ragone,t.dinoia,disciascio}@poliba.it
[2] Università della Tuscia , Viterbo, Italy
donini@unitus.it
[3] Artificial Intelligence Laboratory–University of Michigan, Ann Arbor, USA
wellman@umich.edu

Abstract. We propose a framework to compute the utility of an agreement w.r.t. a preference set in a negotiation process. In particular, we refer to preferences expressed as weighted formulas in a decidable fragment of First-order Logic and agreements expressed as a formula. We ground our framework in Description Logics (DL) endowed with disjunction, to be compliant with Semantic Web technologies. A logic based approach to preference representation allows, when a background knowledge base is exploited, to relax the often unrealistic assumption of additive independence among attributes. We provide suitable definitions of the problem and present algorithms to compute utility in our setting. We also validate our approach through an experimental evaluation.

1 Introduction

The problem of user preference specification have always been a particularly challenging research problem in knowledge representation. Expressing preferences in an effective and expressive way is especially important in negotiation contests. Indeed, in negotiation processes preference representation is of capital importance in order to allow users to express preferences on issues characterizing the bargaining object. One common approach to represent preferences on multiattribute negotiation rely on *multiattribute utility theory* [1], where utility functions assign a real value to different combinations of attributes. However, the number of possible combinations exponentially grow in the size of the attribute domain, therefore even with a small number of attributes, the number of possible combinations could be really high. For this reason, usually, independence relations among the attributes are exploited, and often attributes are assumed to be additively independent, so that the multiattribute utility function is a weighted sum of single-attribute utility functions. However, in real-world domains such an assumption is often not entirely true, as there are preferential dependencies among attributes that cannot be simple ruled out. For example, referring to a negotiation on the characteristics of a laptop, a user should be able to express a preference not only on a single attribute, *e.g.*, a particular operating systems, but on a combination of attributes: a particular operating system with a minimum amount of memory. This mean that a particular operating

L. Godo and A. Pugliese (Eds.): SUM 2009, LNAI 5785, pp. 193–205, 2009.

system does not have a value per se, so we cannot simple use an additive value function. Some recent approaches support relaxation of the fully additive assumption, for example by providing generalized versions [2] or exploiting graphical models of dependence structure [3,4,5], while remaining within the multiattribute framework.

On the other hand, logical languages seem to be the ideal candidates to express interdependencies among preferences. Moreover they can also model in an ontology the knowledge about the domain, in order to express interdependencies among attributes, (*e.g.*, *a Centrino is an Intel processor with a 32-bit CPU*), as well as the fact that some combination of features may be infeasible due to constraints in the ontology itself (*e.g.*, *a Centrino processor is not compatible with a processor with a 64-bit architecture*). In a negotiation it is important to compute the utility value of different outcomes (agreements) w.r.t. the set of preferences expressed by the agent, in order to rank them.

The main contribution of this paper is an approach that, given a set of preferences, represented as weighted Description Logics formulas w.r.t. a shared ontology, computes the utility of a formula (agreement) based on its possible models (interpretations). To our knowledge, the only prior method proposed in the literature for this problem is subsumption, which has some limitations, as shown in [6].

The problem is particularly challenging as if preferences are expressed using Propositional Logic, then the utility can be computed considering a particular propositional model (representing the agreement), taking into account formulas satisfied by that model. While for Propositional Logic it is possible to refer directly to models (interpretations) in order to compute utility, this computation for First-order Logic (FOL) is less straightforward, as the number of possible models is infinite.

The results presented in the paper remain valid for whatever decidable logic with a model-theoretic semantics, but we ground our approach on DLs because of their importance in the development of the Semantic Web.

The remainder of the paper proceeds as follows. Firstly we brief introduce the notation used in the paper and the problem of preference representation in the field of logic languages. Then we present the framework to compute the utility of a DL preference set w.r.t. a formula, illustrating a novel algorithm to compute a preference closure in our set. Finally we perform some experiments in order to evaluate the approach. Conclusion closes the paper.

2 Notation

Hereafter, we assume the reader be familiar with DLs syntax and semantics. We use symbols $A, B, C, D, \ldots, \top, \bot$ to denote concepts. Given a generic concept C, we use the notation $\mathcal{I} \models C$ to say that $C^{\mathcal{I}} \neq \emptyset$. Interpretation functions can also apply to concept expressions and axioms: $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$, $(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$, $(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \backslash C^{\mathcal{I}}$, $(C \sqsubseteq D)^{\mathcal{I}} = C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$.

Here we use $\mathcal{T}$ (for Terminology) to indicate a DL ontology, *i.e.*, a set of axioms of the form $C \sqsubseteq D$ (inclusion) and $C \equiv D$ (definition) with C and D being concepts. We say C is subsumed by D w.r.t. $\mathcal{T}$ when $\mathcal{T} \models C \sqsubseteq D$, or equivalently $C \sqsubseteq_{\mathcal{T}} D$; C is not satisfiable w.r.t. the ontology $\mathcal{T}$ when it is subsumed by the most specific concept $\mathcal{T} \models C \sqsubseteq \bot$, or equivalently $C \sqsubseteq_{\mathcal{T}} \bot$; C is not subsumed by D w.r.t. $\mathcal{T}$ when

$\mathcal{T} \not\models C \sqsubseteq D$, or equivalently $C \not\sqsubseteq_{\mathcal{T}} D$. We write $\mathcal{I} \models \mathcal{T}$ to denote that for each axiom $C \sqsubseteq D$ in $\mathcal{T}$ it results $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ and for each axiom $C \equiv D$ in $\mathcal{T}$ it results $C^{\mathcal{I}} = D^{\mathcal{I}}$. Similarly $\mathcal{I} \models C \sqsubseteq_{\mathcal{T}} D$, with $C \sqsubseteq D \notin \mathcal{T}$, denotes that both $\mathcal{I} \models \mathcal{T}$ and $\mathcal{I} \models C \sqsubseteq D$.

3 Preference Representation Using Description Logics

The problem of preference representation deals with the expression and evaluation of preferences over a set of different alternatives (outcomes). This problem can be challenging even for a small set of alternatives, involving a moderate number of features, as the user has to evaluate all possible configurations of feature values in the domain.

In this work, we deal with this problem by a combination of expressive language to facilitate preference specification, and preference structure exploitation, justified by multiattribute utility theory.

Several approaches to negotiation have exploited logic languages in order to express preferences, most of them using propositional logic [7,8,9], however only few of the approaches proposed in the literature have explored the possibility to use also an Ontology to model relations among attributes [10] or the use of more expressive logics as DLs [11].

We point out the importance to refer to a background knowledge, *i.e.*, having an Ontology $\mathcal{T}$, in order to model not only interdependencies among attributes in preference statements, but also to model inner relations among attributes that cannot be disregarded. We extend the well-known approach of weighted propositional formulas [7,8,9], representing preferences as DL formulas, where at each formula we associate a value v representing the relative importance of that formula.

Definition 1. *Let $\mathcal{T}$ be an ontology in a DL. A **Preference** is a pair $\phi = \langle P, v \rangle$ where P is a concept such that $P \not\sqsubseteq_{\mathcal{T}} \bot$ and v is a positive real number assigning a worth to P. We call a finite set $\mathcal{P}$ of preferences a **Preference Set** iff for each pair of preferences $\phi' = \langle P', v' \rangle$ and $\phi'' = \langle P'', v'' \rangle$ such that $P' \equiv_{\mathcal{T}} P''$ then $v' = v''$ holds.*

Example 1 (Desktop Computer Negotiation). Imagine a negotiation setting where buyer and seller are negotiating on the characteristics of a *desktop computer*. The buyer will have some preferences, while the seller will have some different configurations to offer to the buyer in order to satisfy her preferences. *Let us hence suppose the buyer is looking for a desktop PC endowed with an AMD CPU. Otherwise, if the desktop PC has an Intel CPU, it should only be a Centrino one. The buyer also wants a desktop PC supporting wireless connection.* Following Definition 1 the buyer's Preference set is $\mathcal{P} = \{\langle P_1, v_1 \rangle, \langle P_2, v_2 \rangle, \langle P_3, v_3 \rangle\}$, with:

$P_1 = \forall \texttt{hasCPU.Centrino} \sqcap \exists \texttt{hasCPU}$
$P_2 = \exists \texttt{hasCPU.AMD}$
$P_3 = \exists \texttt{netSupport.WiFi}$

On the other side, the seller could offer *a Desktop computer supporting either a wireless connection or an AMD CPU, specifically a Sempron one, and he does not have a desktop PC endowed with a Centrino CPU*.

$A = \texttt{DesktopComputer} \sqcap \neg\exists\texttt{hasCPU.Centrino} \sqcap (\exists\texttt{netSupport.WiFi} \sqcup \forall\texttt{hasCPU.Sempron})$

Therefore, given a preference set $\mathcal{P}$ and a proposal A, how to evaluate the utility of this proposal w.r.t. buyer's preferences? Intuitively, the utility value should be the sum of the value v_i of the preferences satisfied by the seller's proposal. Next sections will adress this problem, showing a computation method for weighted DL-formulas. Δ

Definition 2 (Minimal Models – Minimal Utility Value). *Given an ontology $\mathcal{T}$, a concept A, such that $\mathcal{T} \not\models A \sqsubseteq \bot$, and a Prefernce Set $\mathcal{P}$, a **Minimal Model** w.r.t. A, $\mathcal{P}$ and $\mathcal{T}$ is an interpretation $\mathcal{I}$ such that:*

1. *both $\mathcal{I} \models A$ and $\mathcal{I} \models \mathcal{T}$*
2. *the value $u^c(A) = \sum\{v \mid \langle P, v\rangle \in \mathcal{P}$ and $\mathcal{I} \models P\}$ is minimal*

*We call $u^c(A)$ a **Minimal Utility Value** for A w.r.t. to $\mathcal{P}$.*

4 Computation of Minimal Utility Value

In this section we show how the computation of the *minimal utility value* for a preference set $\mathcal{P}$ w.r.t. a concept A can be turned out in solving an optimization problem.

Definition 3 (Preference Clause). *Given a preference set $\mathcal{P} = \{\langle P_i, v_i\rangle\}, i = 1 \ldots n$, an ontology $\mathcal{T}$ and a concept A such that $A \not\sqsubseteq_{\mathcal{T}} \bot$, we say that $\mathcal{P}$ is constrained if the following condition holds:*

$$A \sqsubseteq_{\mathcal{T}} \hat{P}_1 \sqcup \ldots \sqcup \hat{P}_n \tag{1}$$

*Where $\hat{P}_i \in \{P_i, \neg P_i\}$. We call $\hat{P}_1 \sqcup \ldots \sqcup \hat{P}_n$ a **Preference Clause** if there is no strict subset $\mathcal{Q} \subset \mathcal{P}$ such that $\mathcal{Q}$ is constrained.*

We may say that a *Preference Clause* contains the minimal set of preferences such that Equation (1) holds. Note that Equation (1) can be rewritten as $A \sqcap \neg\hat{P}_1 \sqcap \ldots \sqcap \neg\hat{P}_n \sqsubseteq_{\mathcal{T}} \bot$ (see [12] for more details).

Definition 4 (Preference Closure). *Given a Preference set $\mathcal{P} = \{\phi_i\}, i = 1 \ldots n$, an ontology $\mathcal{T}$ and a concept $A \not\sqsubseteq_{\mathcal{T}} \bot$, we call **Preference Closure**, denoted as $\mathcal{CL}$, the set of Preference Clauses built, if any, for each set in $2^{\mathcal{P}}$.*

In other words, a *Preference Closure* represents the set of all possible *Preference Clauses* over $\mathcal{P}$. It represents all possible (minimal) interrelations occurring between A and preference descriptions in $\mathcal{P}$ w.r.t. an ontology $\mathcal{T}$.

Proposition 1. *Given a concept A, a **Preference Closure** $\mathcal{CL}$ and an ontology $\mathcal{T}$, if $\mathcal{I}^m$ is a Minimal Model of A then*

$$\mathcal{I}^m \models \mathcal{CL} \tag{2}$$

Proof. By Definition 2 since $\mathcal{I}^m$ is a Minimal Model then $\mathcal{I}^m \models \mathcal{T}$ and $\mathcal{I}^m \models A$. As for all models $\mathcal{I}$ such that $\mathcal{I} \models \mathcal{T}$, including $\mathcal{I}^m$, also $\mathcal{I} \models \mathcal{CL}$ is satisfied, then the proposition holds. $\square$

4.1 Preference Clauses and Theory Prime Implicates

It is easy to see that the notion of Preference Closure shares many characteristics with the one of prime implicates [13] of a formula. Nevertheless the former extends, to some extent, the definition of prime implicates of a formula. For the sake of completeness we rewrite here the definition of theory prime implicates [14]. This formulation was originally proposed for a propositional setting but it remains consistent also if we move to First Order Logic or to whathever logical language for which we have the notion of: clause (the language allows the use of Boolean disjunction); logical entailment; logical equivalence w.r.t. entailment.

Definition 5 (Theory Prime Implicates [14]). *Let $\mathcal{T}$ be a propositional knowledge base and both A and CL be two propositional formulas. A **theory prime implicat**e of A w.r.t. $\mathcal{T}$ is a clause CL such that:*

- $A \models_{\mathcal{T}} CL$ *holds;*
- *for every clause CL', if both $A \models_{\mathcal{T}} CL'$ and $CL' \models_{\mathcal{T}} CL$ hold, then $\models_{\mathcal{T}} CL' \equiv CL$ holds.*

Noteworthy is that in the classical definition of theory of prime implicates there is no restriction of the symbols to be used in CL. In order to (re)define Definition 3 in terms similar to Theory Prime Implicates we introduce the notion of concept signature of a formula in DL. Given a DL formula C we call concept signature of C, denoted as $sign(C)$, the set of concept names occurring in C. It is straight to extend concept signature to axioms and set of axioms.

Definition 6 (Preference Clause - Preference Name). *Let $\mathcal{L}$ be a decidable Description Logic and let $\mathcal{T}$ and A be a consistent knowledge base in $\mathcal{L}$ and a concept in $\mathcal{L}$ such that $A \not\sqsubseteq_{\mathcal{T}} \bot$. Given a set of preferences $\mathcal{P} = \{\langle P_i, v_i\rangle\}$ we define a TBox $\mathcal{T}_{\mathcal{PN}} = \{PN_i \equiv P_i\}$ where PN_i is a concept name such that $PN_i \notin \bigcup_{\langle P_i, vi\rangle \in \mathcal{P}} sign(P_i) \cup sign(\mathcal{T}) \cup sign(A)$. We say that PN_i is a **preference name** for P_i and we use $\mathcal{PN}$ to denote the set of all preference names. A **preference clause** of A w.r.t. $\mathcal{T}$ is a clause CL such that:*

- $sign(CL) \subseteq \mathcal{PN}$*;*
- $A \sqsubseteq_{\mathcal{T} \cup \mathcal{T}_{\mathcal{P}}} CL$ *holds;*
- *for every clause CL', such that $sign(CL') \subseteq \mathcal{PN}$, if both $A \models_{\mathcal{T} \cup \mathcal{T}_{\mathcal{P}}} CL'$ and $CL' \models_{\mathcal{T} \cup \mathcal{T}_{\mathcal{P}}} CL$ hold, then $\models_{\mathcal{T}} CL' \equiv CL$ holds.*

In other words, a preference clause can be seen as a theory prime implicate where CL has to be rewritten in terms of Preference Names. In fact, prime implicates for Propositional Logic are more similar to preference clauses than the ones proposed for for Description Logics [15].

4.2 From Preference Closure to Minimal Utility Value

In order to compute *Minimal Utility Value* $u^c(A)$ we reduce to an optimization problem (OP). Usually, in an OP we have a set of constrained numerical variables and a function

to be maximized/minimized. In our case we will represent constraints as a set χ of linear inequalities over binary variables, *i.e.*, variables whose value is in $\{0, 1\}$, and the function to be minimized as a weighted combination of such variables. In order to represent χ we need some pre-processing steps.

1. Compute the *Preference Closure* $\mathcal{CL}$ for $\mathcal{P}$;
2. For each *Preference Clause* $A \sqcap \hat{P}_1 \sqcap \ldots \hat{P}_n \sqsubseteq_{\mathcal{T}} \bot \in \mathcal{CL}$, compute a corresponding preference constraint set $\overline{\mathcal{CL}} = \{\neg\hat{P}_1, \ldots, \neg\hat{P}_n\}$. We denote with $\overline{\mathcal{CL}}^c = \{\overline{\mathcal{CL}}\}$ the set of all preference constraint sets.

Example 2 (Desktop Computer Negotiation cont'd). Consider again the Desktop Computer negotiation of Example 1. Given the set of preference $\mathcal{P} = \{\langle P_1, v_1\rangle, \langle P_2, v_2\rangle, \langle P_3, v_3\rangle\}$ and the proposal A, if we compute the **Preference Closure** $\mathcal{CL}$ we find:

$$\mathcal{CL} = \left\{ \begin{array}{l} A \sqcap P_1 \sqsubseteq_{\mathcal{T}} \bot; \\ A \sqcap \neg P_2 \sqcap \neg P_3 \sqsubseteq_{\mathcal{T}} \bot \end{array} \right\}$$

hence, the two corresponding preference constraint sets in $\overline{\mathcal{CL}}^c$ are: $\overline{\mathcal{CL}}_1 = \{\neg P_1\}$, $\overline{\mathcal{CL}}_2 = \{P_2, P_3\}$ Δ

Based on well-known encoding of clauses into linear inequalities (*e.g.*, [16, p.314]) we transform each set $\overline{\mathcal{CL}} \in \overline{\mathcal{CL}}^c$ in a set of linear inequalities χ and then define a function to be minimized in order to solve an OP.

Definition 7 (Minimal Utility Value OP). *Let $\mathcal{P}$ be a set of preferences and $\overline{\mathcal{CL}}^c$ be the set of all preference constraint sets. We define a* Minimal Utility Value OP*, represented as $\langle \chi, u(\mathbf{p})\rangle$, the optimization problem built as follows:*

1. ***numerical variables*** *– for each preference $\langle P_i, v_i\rangle \in \mathcal{P}$, with $i = 1, \ldots, n$ introduce a binary variable $p_i \in \{0, 1\}$ and define the corresponding array $\mathbf{p} = (p_1, \ldots, p_n)$ (see Example 3);*
2. ***set χ of linear inequalities*** *– pick up each set $\overline{\mathcal{CL}} \in \overline{\mathcal{CL}}^c$ and build the linear inequalities*

$$\sum\{(1-p) \mid \neg P \in \overline{\mathcal{CL}}\} + \sum\{p \mid P \in \overline{\mathcal{CL}}\} \geq 1$$

3. ***function to be minimized*** *– given the array $\mathbf{p}$ of binary variables*

$$u(\mathbf{p}) = \sum\{v \cdot p \mid p \text{ is the variable mapping } \langle P, v\rangle\}$$

Observation 1 *If we considered also $\neg A$ when computing the sets $\overline{\mathcal{CL}} \in \overline{\mathcal{CL}}^c$ we would have had inequalities in the form:*

$$(1-a) + \sum\{(1-p) \mid \neg P \in \overline{\mathcal{CL}}\} + \sum\{p \mid P \in \overline{\mathcal{CL}}\} \geq 1$$

Since we are interested in models where $A^{\mathcal{I}}$ is interpreted as nonempty, then variable a has to be equal to 1. Hence the first element of the above summation is always equal to 0. In other words, we can omit $\neg A$ when computing a preference constraint set $\overline{\mathcal{CL}}$.

The solution to a *Minimal Utility Value OP* will be an assignment $\mathbf{p}_s$ for $\mathbf{p}$, *i.e.*, an array of $\{0,1\}$-values, minimizing $u(\mathbf{p})$.

Example 3 (Desktop Computer Negotiation cont'd). Back to the Desktop Computer negotiation of Example 1, after the computation of Preference Closures and set $\overline{\mathcal{CL}}^c$, we build the corresponding optimization problem in order to find the model with the minimal utility value:

$$\mathbf{p} = (p_1, p_2, p_3)$$
$$\chi = \begin{cases} 1 - p_1 \geq 1 \\ p_2 + p_3 \geq 1 \end{cases}$$
$$u(\mathbf{p}) = v_1 \cdot p_1 + v_2 \cdot p_2 + v_3 \cdot p_3$$

Possible solutions are:

$$\mathbf{p}'_s = (0,1,0)\ ,\ u(\mathbf{p}'_s) = v_2$$
$$\mathbf{p}''_s = (0,0,1)\ ,\ u(\mathbf{p}''_s) = v_3$$
$$\mathbf{p}'''_s = (0,1,1)\ ,\ u(\mathbf{p}'''_s) = v_2 + v_3$$

The minimal solution will be either $\mathbf{p}'_s$ or $\mathbf{p}''_s$, depending of the value of v_2 and v_3. $\bigtriangleup$

Given a a solution $\mathbf{p}_s$ to a *Minimal Utility Value OP* $\langle \chi, u(\mathbf{p}) \rangle$, we call *Minimal Preference Set* $\overline{\mathcal{P}}^m$ and *Minimal Assignment* A^m, respectively, the set and the formula built as in the following[1]:

$$\overline{\mathcal{P}}^m = \{\langle P_i, v_i \rangle \mid p_i = 1 \textit{ in the solution } \mathbf{p}_s\}$$
$$A^m = \bigsqcap\{P_i \mid p_i = 1 \textit{ in the solution } \mathbf{p}_s\} \sqcap \bigsqcap\{\neg P_i \mid p_i = 0 \textit{ in the solution } \mathbf{p}_s\}$$

Theorem 1. *Given a solution* $\mathbf{p}_s$ *to a* Minimal Utility Value OP $\langle \chi, u(\mathcal{P}) \rangle$ *and a* Minimal Assignment A^m*:*

1. *if* $\mathcal{I}^m$ *is a* Minimal Model *then* $\mathcal{I}^m \models A^m$*;*
2. $u(\mathbf{p}_s)$ *is a* Minimal Utility Value*.*

Proof. First we show that there exists at least one model $\mathcal{I}^m \models \mathcal{T}$ *such that both* $\mathcal{I}^m \models A$ *and* $\mathcal{I}^m \models A^m$*. If* $\mathcal{I}^m$ *did not exist, then* $A^{\mathcal{I}^m} \cap (A^m)^{\mathcal{I}^m} = \emptyset$*. We can easily rewrite the latter relation as* $A \sqcap A^m \sqsubseteq_{\mathcal{T}} \bot$ *which is equivalent to* $A \sqsubseteq_{\mathcal{T}} \neg A^m$*. But this is not possible. Indeed, if* A *and* A^m *were inconsistent with each other w.r.t.* $\mathcal{T}$ *then, by Proposition 1 we should have the corresponding Preference Clause in* $\mathcal{CL}$ *and the related inequality in* χ*:*

$$\sum\{(1-p) \mid P \textit{ appears in } A^m\} + \sum\{p \mid \neg P \textit{ appears in } A^m\} \geq 1$$

In order to be satisfied, the latter inequality must have either (a) at least one variable assigned to 0 *in the first summation or (b) at least one variable assigned to* 1 *in the second one. Case (a) means that the corresponding preference is not satisfied by* A^m

[1] with $\bigsqcap\{\cdot\}$ we denote the conjunction of all the concepts in the set $\{\cdot\}$.

while case (b) means that the corresponding preference is satisfied by A^m. Both cases are conflicting with the definition of A^m.
By construction of χ, we have that if $\mathcal{I}^m \models A^m$ then $\mathcal{I}^m \models A$ (see Observation 1). Since A^m comes from the minimization of $u(\mathbf{p}_s)$ then $\mathcal{I}^m \models A^m$ represents a model of A^m (and then of A) such that

$$\sum\{v \mid \langle P, v\rangle \in \mathcal{P} \text{ and } \mathcal{I}^m \models P\}$$

is minimal.
It is straightforward to show that $u(\mathbf{p}_s)$ is a Minimal Utility Value. □

4.3 An Algorithm to Compute a Preference Closure

Although the above shown similarities, it is quite hard to adapt the algorithm already proposed in the literature for the theory of prime implicates to compute a Preference Clousure. Indeed, algorithms for Propositional Logic rely on the propositional structure of formulas and axioms in the theory $\mathcal{T}$. In [17], Ngair needs the knowledge base to be a conjunction of DNF formulas while Dechter and Rish [18] uses a knowledge base in CNF. Marquis and Sadaoui [14] rewrite propositional formulas using a Shannon expansion which, to our knowledge, is not applicable to non-propositional languages. Moreover known algorithms for Description Logics [15] do not take into accounts the knowledge base (theory). In this section we propose an algorithm to compute a Preference Closure for a Description Logic $\mathcal{L}$. The main idea behind the algorithm we are going to describe bases on the following two simple observations:

1. since Boolean disjunction enjoys the commutativity property, if $A \sqsubseteq_{\mathcal{T}} P_i \sqcup P_j$ then $A \sqsubseteq_{\mathcal{T}} P_j \sqcup P_i$. Hence, once we know that $A \sqsubseteq_{\mathcal{T}} P_i \sqcup P_j$ holds we do not check $A \sqsubseteq_{\mathcal{T}} P_j \sqcup P_i$ also.
2. given a clause CL with $sign(CL) \subseteq \mathcal{PN}$, if $A \sqsubseteq_{\mathcal{T}} CL$ holds then also $A \sqsubseteq_{\mathcal{T}} CL \sqcup PN'$, with $PN' \in \mathcal{PN}$, holds (the same may be for $\neg PN'$).

To compute all the clauses belonging to the Preference Closure, we build a tree where each node is labeled with a preference name or a negated preference name and edges represent a Boolean disjunction between the two preferences (positive or negated) represented by the two nodes connected by the edge. We denote with $L(n)$ the label of a node n in the tree and with N the set containing only the leaf nodes of the tree. In order to build the tree, we impose a total order $<$ between preference names. We call PN_{last} the preference name such that for each preference name PN_i the relation $PN_i < PN_{last}$ always holds.

The following functions will be useful to build the Preference Closure starting from the tree.

$$abs \ : \ N \rightarrow \mathcal{PN}$$
$$abs(n) = \begin{cases} PN, if \ \ L(n) = PN \\ PN, if \ L(n) = \neg PN \end{cases}$$

$$branch\_clause \ : \ N \rightarrow \mathcal{L}$$
$$branch\_clause(n) = \bigsqcup\{L(m) \mid m \text{ is a node in the branch containing } n\}$$

Given a leaf node n, the tree is expanded according to Algorithm $expand(n)$.

Algorithm:$expand(n)$
foreach *preference name* PN *such that* $abs(L(n)) < PN$ **do**
  create two nodes m' and m'' such that $L(m') = PN$ and $L(m'') = \neg PN$;
  add the edges (n, m') and (n, m'') to the tree ;
end

We say a branch is **open** if it is possible to further expand the branch starting from its leaf node. A branch is closed if it is not open. We impose the root node of the tree, denoted as ϵ, has no label. The tree for Preference Closure computation is computed according to Algorithm 1. Note that the subsumption check in Line 2 of Algorithm 1, both does not involve the TBox and it compares only clauses. Then it can be easily reduced to set comparison.

In Figure 1 is depicted the tree built to compute the Preference Closure in Example 2. The number below the leaf nodes represents the reason why the corresponding branch closed w.r.t. lines in Algorithm 1. As an example, number 2 (respectively 2 and 2) says taht the brach closed because of line 2 (respectively 2 and 2) in Algorithm 1.

```
1  Algorithm:preference_closure(ε, A, T, P, PN, T_PN)
2  CL = ∅;
3  foreach preference name PN_i ∈ PN do
4      create two nodes m'_i and m''_i such that L(m'_i) = PN and L(m''_i) = ¬PN;
5      add the edges (ε, m'_i) and (ε, m''_i) to the tree;
6  end
7  while there is an open branch in the tree do
8      foreach open branch β_j with n_j being its leaf node do
9          if CL ⊨ A ⊑ branch_clause(n_j) then
10             close β_j;
11         end
12         else if A ⊑_{T ∪ T_PN} branch_clause(n_j) then
13             CL = CL ∪ {A ⊑ branch_clause(n_j)} ;
14             close β_j;
15         end
16         if label(n_j) = PN_last then
17             close β_j;
18         end
19         else
20             expand(n_j);
21         end
22     end
23 end
```

Algorithm 1. An algorithm to compute a Preference Closure

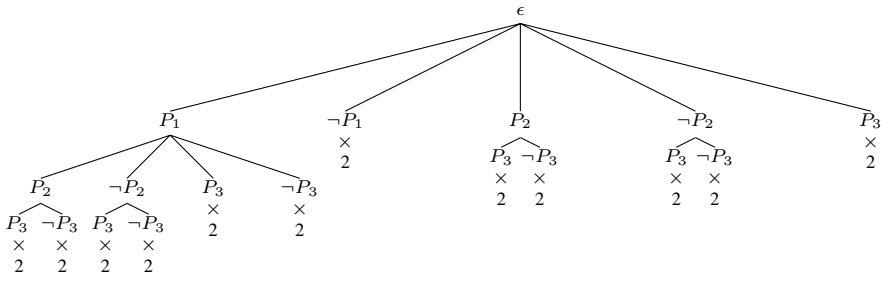


Fig. 1. The tree built using $preference\_closure$ to compute the Preference Closure of Ex. 2

The complexity of Algorithm 1 is dominated by the number of open nodes in the tree. Although one may optimize the search by expanding first nodes with shortest and longest paths, in the worst case a constant fraction of the powerset of the preferences must be taken into account. Hence the subsumption test of Line 2 is repeated $2^{|\mathcal{P}|}$ times. Let $\|x\|$ be a measure of the size of x. Denoting by $s(\|\mathcal{T}\|, \|C\|, \|D\|)$ a function bounding the time complexity of a subsumption test $C \sqsubseteq_{\mathcal{T}} D$ in the chosen DL, the overall complexity is bounded by $2^{|\mathcal{P}|} \cdot s(\|\mathcal{T} \cup \mathcal{T}_{\mathcal{PN}}\|, \|A\|, |\mathcal{P}|)$. Observe that the third argument of s is just $|\mathcal{P}|$, since the size of preferences is hidden in $\mathcal{T}_{\mathcal{PN}}$. It is reasonable to assume that the number of preferences is bounded; in this case, the overall complexity of Algorithm 1 is dominated by the subsumption test in the chosen DL.

4.4 Performance Evaluation

In order to evaluate what is the impact, from a performance perspective, of the computation of a Preference Clause in a real world application, we built some testbeds and evaluated execution time w.r.t. the parameters highlighted in the previous section. We selected some OWL DL ontologies with different expressivity (considering the corresponding DL language) from the publicly available repository of TONES project[2]. Based on these ontologies we randomly generated concept expressions which resulted consistent w.r.t. the source ontology. From these sets of concepts, one for each ontology, we selected 150 different subsets of 11, 10, 9, 8, 7, 6, 5, 4, 3, 2 concepts. Hence, we generated 150 tests with 1 concept representing A and 10 concepts representing preferences $P_1, \ldots, P_{10}$ and similarly for the other subsets. In Figure 2 in Appendix we report results for three of the ontologies we tested, i.e Pizza, Movie and BuildingsAndPlaces (see Table 1). For Pizza ontology we removed inconsistent classes. For the sake of presentation we report only average execution time w.r.t. number of axioms in the original ontology (*i.e.*, without $\mathcal{T}_{\mathcal{PN}}$) as the number of preferences changes. Complete testbeds and test results are available at `http://sisinflab.poliba.it/dinoia/preference_tests.zip`. The reference architecture is a Intel Core Duo Quad CPU @ 2.66 Ghz processor, 3.2 Gigabytes of RAM, Ubuntu Linux 9.04, Kernel 2.6.28, Sun Java Runtime Environment 1.6.0.

[2] `http://owl.cs.manchester.ac.uk/repository/browser`

Table 1. Reference Ontologies

Ontology Name	DL Expressivity	Number of Axioms
Pizza	$\mathcal{SHOIN}$	712
Movie	$\mathcal{ALCN}$	140
BuildingsAndPlaces	$\mathcal{ALCHOQ}$	1204

5 Conclusion

Logic languages have been proposed here as a natural and powerful way to express preferences in negotiation contests. We presented a framework to compute the utility value of a formula (agreement), when preference are expressed as weighted DL formulas w.r.t. a shared ontology. We remark that, even if we use DLs, the framework is completely general and suitable for whatever decidable fragment of FOL. We propose a novel algorithm to compute a Preference closure and discuss also its complexity. An experimental evaluation showed the applicability and suitability of our framework. Currently, we are studying how to combine this approach with graphical models, and in particular GAI (Generalized Additive Independence) [19,3], in order to model multiattribute auctions.

Acknowledgment

We are very grateful to Michelantonio Trizio for developing the system used to test and evaluate the approach. We wish to thank anonymous reviewers for useful comments and suggestions. This research is partially sponsored by *Distributed Production as Innovative System (DIPIS)* and *Telecommunication Facilities and Wireless Sensor Networks in Emergency Management projects*.

References

1. Keeney, R.L., Raiffa, H.: Decisions with Multiple Objectives: Preferences and Value Tradeoffs. John Wiley & Sons, New York (1976)
2. Gonzales, C., Perny, P.: GAI networks for utility elicitation. In: KR, pp. 224–234 (2004)
3. Bacchus, F., Grove, A.: Graphical models for preference and utility. In: UAI, pp. 3–10 (1995)
4. Boutilier, C., Brafman, R.I., Domshlak, C., Hoos, H.H., Poole, D.: CP-nets: A tool for representing and reasoning about conditional ceteris paribus preference statements. JAIR 21, 135–191 (2004)
5. Engel, Y., Wellman, M.P.: CUI networks: A graphical representation for conditional utility independence. JAIR 31, 83–112 (2008)
6. Ragone, A., Di Noia, T., Di Sciascio, E., Donini, F.M., Wellman, M.: Computing utility from weighted description logic preference formulas. In: Declarative Agent Languages and Technologies VII Workshop Notes (2009), `http://www.di.unito.it/~baldoni/DALT-2009/DALT-WorkshopNotes.pdf`
7. Pinkas, G.: Propositional non-monotonic reasoning and inconsistency in symmetric neural networks. In: IJCAI, pp. 525–531 (1991)

8. Lafage, C., Lang, J.: Logical representation of preferences for group decision making. In: KR, pp. 457–468 (2000)
9. Chevaleyre, Y., Endriss, U., Lang, J.: Expressive power of weighted propositional formulas for cardinal preference modelling. In: KR, pp. 145–152 (2006)
10. Ragone, A., Di Noia, T., Di Sciascio, E., Donini, F.: A logic-based framework to compute Pareto agreements in one-shot bilateral negotiation. In: ECAI, pp. 230–234 (2006)
11. Ragone, A., Di Noia, T., Di Sciascio, E., Donini, F.M.: Description logics for multi-issue bilateral negotiation with incomplete information. In: AAAI, pp. 477–482 (2007)
12. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.): The Description Logic Handbook. Cambridge Univ. Press, Cambridge (2002)
13. Cadoli, M., Donini, F.M.: A survey on knowledge compilation. AI Comm 10(3-4), 137–150 (1997)
14. Marquis, P., Sadaoui, S.: A new algorithm for computing theory prime implicates compilations. In: AAAI, pp. 504–509 (1996)
15. Bienvenu, M.: Prime implicate normal form for $\mathcal{ALC}$ concepts. In: AAAI, pp. 412–417 (2008)
16. Papadimitriou, C.H., Steiglitz, K.: Combinatorial Optimization: Algorithms and Complexity. Prentice-Hall, Englewood Cliffs (1982)
17. Ngair, T.-H.: A new algorithm for incremental prime implicate generation. In: IJCAI, pp. 46–51 (1993)
18. Dechter, R., Rish, I.: Directional resolution: The Davis-Putnam procedure, revisited. In: KR, pp. 134–145 (1994)
19. Fishburn, P.C.: Interdependence and additivity in multivariate, unidimensional expected utility theory. International Economic Review 8, 335–342 (1967)

Appendix

N. of prefrences	Movie	Pizza	BuildingsAndPlaces
1	6,87	5,07	10,27
2	5,3	5,78	9,58
3	9,87	7,4	11,16
4	13,32	9,74	15,7
5	23,96	14,74	21,59
6	42,75	26,04	39,73
7	102,08	64,76	95,78
8	241,05	155,17	294,86
9	597,52	398,22	1055,96
10	1484,68	1317,39	2319,38

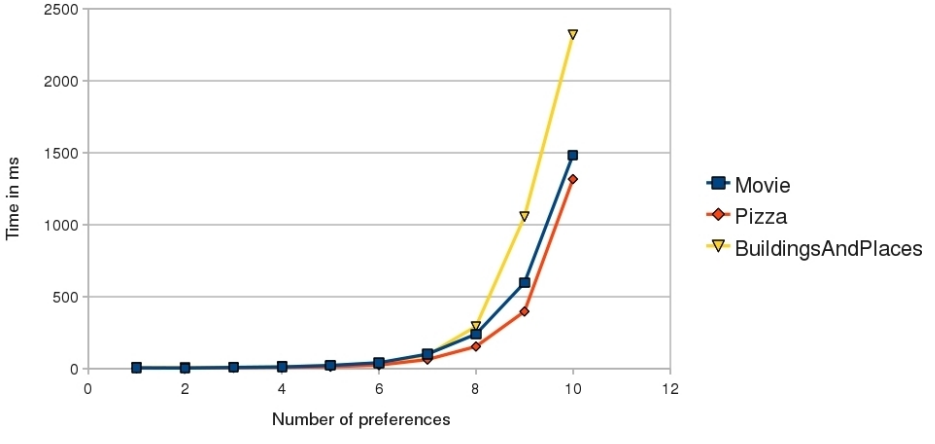


Fig. 2. Execution times (in ms) of $preference\_closure$ using generated testbeds for Pizza, Movie and BuildingAndPlaces ontologies

Probabilistic Planning with Imperfect Sensing Actions Using Hybrid Probabilistic Logic Programs

Emad Saad

Department of Computer Science
Gulf University for Science and Technology
West Mishref, Kuwait
saad.e@gust.edu.kw

Abstract. Effective planning in uncertain environment is important to agents and multi-agents systems. In this paper, we introduce a new logic based approach to probabilistic contingent planning (probabilistic planning with imperfect sensing actions), by relating probabilistic contingent planning to normal hybrid probabilistic logic programs with probabilistic answer set semantics [24]. We show that any probabilistic contingent planning problem can be encoded as a normal hybrid probabilistic logic program. We formally prove the correctness of our approach. Moreover, we show that the complexity of finding a probabilistic contingent plan in our approach is NP-complete. In addition, we show that any probabilistic contingent planning problem, $\mathcal{PP}$, can be encoded as a classical normal logic program with answer set semantics, whose answer sets corresponds to valid trajectories in $\mathcal{PP}$. We show that probabilistic contingent planning problems can be encoded as SAT problems. We present a new high level probabilistic action description language that allows the representation of sensing actions with probabilistic outcomes.

1 Introduction

Agents with sensing actions produce conditional plans under the assumption that the agent has incomplete knowledge about the world. However, classical conditional planning assumes the agent sensors are perfect, which is unrealistic assumption and limits its applicability to many real-world domains. To deal with the agent's imperfect sensors, classical conditional planning has been extended with sensing actions with probabilistic outcomes, as well as actions with probabilistic conditional effects, forming *probabilistic contingent planning* [5] that allows agents to cope with the uncertainty that stems form dynamic and changing environments as well as the imperfection of the actions execution.

An approach to probabilistic contingent planning has been introduced in [5] along with a representation of sensing actions with probabilistic outcomes and non-sensing actions with probabilistic conditional effects. However, there are two drawbacks in the sensing actions representation of [5]. In [5], the sensing actions outcomes are represented by arbitrary strings called observation labels that do

L. Godo and A. Pugliese (Eds.): SUM 2009, LNAI 5785, pp. 206–222, 2009.

not relate to the fluents that describe the world [19]. In addition, [5] treats sensing and non-sensing actions equally in the sense that, like non-sensing actions, a sensing action has preconditions and effects as well as outcomes resulting from observing the environment. However, [25] proved that sensing actions affect only knowledge fluents and has no effect on the other fluents.

Based on the success of classical planning as propositional satisfiability (SAT planning) [13], a probabilistic contingent planning approach has been presented in [18]. The approach of [18] is developed by converting a probabilistic contingent planning problem into a stochastic satisfiability problem and solving the stochastic satisfiability problem instead to generate probabilistic contingent plans. However, the problems with SAT planning in general are that [18] translating a planning problem as propositional satisfiability (SAT) problem causes an explosion in the size of the problem representation. Moreover, encoding a planning problem as a SAT problem affects the planning problem structure which makes it not obvious to clearly understand the planning process. Moreover, solving a probabilistic contingent planning problem as a stochastic satisfiability problem is $\mathrm{NP}^{PP}-$complete [18]. But, on the other hand, SAT planning has a number of advantages. These include that [18] SAT problem is a central widely studied problem in computer science, therefore, many techniques have been developed to solve the problem. The existence of many efficient solvers that solve the SAT problem.

Based on another successful logic based approach to classical planning, using normal logic programs with answer set semantics (answer set planning) [27], a probabilistic planning approach has been presented in [22], namely, probabilistic answer set planning. Probabilistic answer set planning [22] is a logical approach to probabilistic planning in which a probabilistic planning problem, $\mathcal{PP}$, is solved by translating $\mathcal{PP}$ into a normal hybrid probabilistic logic program with probabilistic answer set semantics [24] whose probabilistic answer sets correspond to trajectories in $\mathcal{PP}$ with associated probabilities. The complexity of finding a plan in probabilistic answer set planning is NP-complete [22]. Moreover, it has been shown in [22] that probabilistic planning problems can be encoded as SAT, hence, efficient SAT solvers can be used to solve probabilistic planning problems. However, the applicability of normal hybrid probabilistic logic programs with probabilistic answer set semantics to probabilistic contingent planning has not been yet investigated.

In this paper we relate probabilistic contingent planning to normal hybrid probabilistic logic programs (NHPP) with probabilistic answer set semantics, introducing a novel logical probabilistic contingent planning approach that is a probabilistic extension to both conditional planning using answer set programming [28] to account for probabilistic domains and probabilistic answer set planning [22] to account for sensing action with probabilistic outcomes. In addition, we introduce a new high level probabilistic action description language, called $\mathcal{P}$, that overcomes the drawbacks in the representation of sensing actions with probabilistic outcomes described in [5].

2 Syntax and Semantics of NHPP

We present a subset of the language NHPP [24], denoted by NHPP$_\mathcal{P}$, that is expressive enough and sufficient to encode probabilistic contingent planning problems. The syntax and semantics of the full version of NHPP can be found in [24].

2.1 The Language of NHPP$_\mathcal{P}$

Let $\mathcal{L}$ be a first-order language with finitely many predicate symbols, constants, and infinitely many variables. The Herbrand base of $\mathcal{L}$ is denoted by $\mathcal{B}_\mathcal{L}$. Probabilities are assigned to atoms in $\mathcal{B}_\mathcal{L}$ as values from $[0, 1]$. An *annotation*, μ, is either a constant in $[0, 1]$, a variable (*annotation variable*) ranging over $[0, 1]$, or $f(\mu_1, \ldots, \mu_n)$ (called *annotation function*) where f is a representation of a computable total function $f : ([0, 1])^n \rightarrow [0, 1]$ and $\mu_1, \ldots, \mu_n$ are annotations. Let $a_1, a_2 \in [0, 1]$. Then we say that $a_1 \leq_t a_2$ iff $a_1 \leq a_2$. A probabilistic logic program (*p-program*) in NHPP$_\mathcal{P}$ is a pair $P = \langle R, \tau \rangle$, where R is a finite set of normal probabilistic rules (p-rules) and τ is a mapping $\tau : \mathcal{B}_\mathcal{L} \rightarrow S_{disj}$, where S_{disj} is a set of disjunctive probabilistic strategies (p-strategies) whose composition functions, c, are mappings $c : [0, 1] \times [0, 1] \rightarrow [0, 1]$. A composition function of a disjunctive p-strategy returns the probability of a disjunction of two atoms given the probability values of its components. A p-rule is an expression of the form

$$A : \mu \leftarrow A_1 : \mu_1, \ldots, A_n : \mu_n, not\ (B_1 : \mu_{n+1}), \ldots, not\ (B_m : \mu_{n+m})$$

where A, $A_1, \ldots, A_n, B_1, \ldots, B_m$ are atoms and μ, μ_i $(1 \leq i \leq m + n)$ are annotations. Intuitively, the meaning of a p-rule is that if for each $A_i : \mu_i$, the probability value of A_i is at least μ_i (w.r.t. $\leq_t$) and for each *not* $(B_j : \mu_j)$, it is not *believable* that the probability values of B_j is at least μ_j, then the probability of A is μ. The mapping τ associates to each atom A a disjunctive p-strategy that will be employed to combine the probability values obtained from different p-rules having A in their heads. A p-program is ground if no variables appear in any of its p-rules.

2.2 Probabilistic Answer Set Semantics of NHPP$_\mathcal{P}$

A probabilistic interpretation (p-interpretation) is a mapping $h : \mathcal{B}_\mathcal{L} \rightarrow [0, 1]$. Let $P = \langle R, \tau \rangle$ be a ground p-program, h be a p-interpretation, and r be

$$A : \mu \leftarrow A_1 : \mu_1, \ldots, A_n : \mu_n, not\ (B_1 : \beta_1), \ldots, not\ (B_m : \beta_m) \in R.$$

Then, we say

- h satisfies $A_i : \mu_i$ (denoted by $h \models A_i : \mu_i$) iff $\mu_i \leq_t h(A_i)$.
- h satisfies *not* $(B_j : \beta_j)$ (denoted by $h \models not\ (B_j : \beta_j)$) iff $\beta_j \not\leq_t h(B_j)$.
- h satisfies $Body \equiv A_1 : \mu_1, \ldots, A_n : \mu_n, not\ (B_1 : \beta_1), \ldots, not\ (B_m : \beta_m)$ (denoted by $h \models Body$) iff $\forall (1 \leq i \leq n), h \models A_i : \mu_i$ and $\forall (1 \leq j \leq m), h \models not\ (B_j : \beta_j)$.

- h satisfies $A : \mu \leftarrow Body$ iff $h \models A : \mu$ or h does not satisfy $Body$.
- h satisfies P iff h satisfies every p-rule in R and for every atom $A \in \mathcal{B}_{\mathcal{L}}$, we have

$$c_{\tau(A)}\{\!\{\mu | A : \mu \leftarrow Body \in R \text{ such that } h \models Body\}\!\} \leq_t h(A).$$

The probabilistic reduct P^h of P w.r.t. h is a p-program $P^h = \langle R^h, \tau \rangle$ where:

$$R^h = \left\{ A : \mu \leftarrow A_1 : \mu_1, \ldots, A_n : \mu_n \;\middle|\; \begin{array}{l} A : \mu \leftarrow A_1 : \mu_1, \ldots, A_n : \mu_n, \\ \quad not\ (B_1 : \beta_1), \ldots, not\ (B_m : \beta_m) \in R \text{ and} \\ \forall (1 \leq j \leq m),\ \beta_j \not\leq_t h(B_j) \end{array} \right\}$$

Intuitively, for any $not\ (B_j : \beta_j)$ in the body of $r \in R$ with $\beta_j \not\leq_t h(B_j)$ is simply satisfied by h, and $not\ (B_j : \beta_j)$ is removed from the body of r. If $\beta_j \leq_t h(B_j)$ then the body of r is not satisfied and r is trivially ignored. A probabilistic model (*p-model*) of a p-program P is a p-interpretation of P that satisfies P. A p-interpretation h of a p-program P is said to be a probabilistic answer set of P if h is the minimal p-model of the probabilistic reduct of P w.r.t. h.

Definition 1 (Probabilistic Answer Set). *A probabilistic interpretation h is a probabilistic answer set of a p-program P if h is the least p-model of P^h.*

3 Probabilistic Action Language with Imperfect Sensing Actions

In this section we develop the syntax and semantics of a novel probabilistic action language called $\mathcal{P}$ that allows the representation of sensing actions with probabilistic outcomes.

3.1 Language Syntax of $\mathcal{P}$

A fluent is an atomic proposition that describes a property of the world. Let $\mathcal{F}$ be a set of fluents and $\mathcal{A}$ be a set of actions. A fluent literal is either a fluent f or $\neg f$. Conjunctive fluent formula is a conjunction of fluent literals of the form $l_1 \wedge \ldots \wedge l_n$. Sometimes we abuse the notation and refer to a conjunctive fluent formula as a set of fluent literals ($\emptyset$ denotes *true*). A *probabilistic action theory* in $\mathcal{P}$ is a set of probabilistic propositions of the form:

$$\textbf{initially } \{\psi_i \ : \ p_i \mid 1 \leq i \leq n\} \tag{1}$$

$$l \textbf{ if } \psi \tag{2}$$

$$\textbf{executable } a \textbf{ if } \psi \tag{3}$$

$$a \textbf{ causes } \{\phi_i \ : \ p_i \textbf{ if } \psi_i \mid 1 \leq i \leq n\} \tag{4}$$

$$a \textbf{ determines } \{\phi_i \ : \ p_i \textbf{ sensing } \psi_i \mid 1 \leq i \leq n\} \tag{5}$$

where l is a fluent literal and for all $(1 \leq i \leq n)$, ψ, ψ_i, ϕ_i are conjunctive fluent formulas, $p_i \in [0, 1]$, and $a \in \mathcal{A}$ is an action. The set of all ψ_i must be exhaustive

and mutually exclusive, where $\forall\ i\ \sum_s p_i\ Pr(\psi_i|s) = 1$ and $\forall\ i, j, s,\ \psi_i \neq \psi_j \Rightarrow Pr(\psi_i \wedge \psi_j|s) = 0$ (given s is a state defined later).

The initial situation is presented by probabilistic proposition (1). Probabilistic proposition (1) states that the possible initial state ψ_i holds with probability p_i. *Indirect effect of action* is described by proposition (2), which says that l holds in every state in which ψ also holds. *Executability condition* is represented by (3), which states that an action a is executable in any state in which ψ holds. Probabilistic proposition of the form (4) represents *the probabilistic conditional effects of a non-sensing action* a. It says that for all $1 \leq i \leq n$, a causes ϕ_i to hold with probability $p_i \in [0, 1]$ in a successor state to a state in which a is executed and ψ_i holds. The set of all ψ_i must be mutually exclusive and exhaustive. Probabilistic proposition of the form (5) describes sensing action with probabilistic outcomes. It states that executing a sensing action a in a state causes any of ϕ_i to be known true with probability p_i whenever a correlated ψ_i is known to be true in a successor state to a state in which a is executed. Each ψ_i is called *the sensed-literals* (*the sensor reading*), where the literals in ψ_i determine what the sensor is observing, ϕ_i is called *the sensor report*, where the literals in ϕ_i determine what the sensor reports on. p_i is the probability that ϕ_i holds whenever ψ_i holds after executing a. The set of all ψ_i (ϕ_i) must be mutually exclusive and exhaustive. Similar to [5], when a property of the world cannot be directly sensed by the sensor, another correlated property of the world, that can be sensed by the sensor, can be used instead.

A probabilistic action theory is a pair $\mathcal{D} = \langle \tilde{s}_I, \mathcal{AD} \rangle$, where $\tilde{s}_I$ is a probabilistic proposition of the form (1) and $\mathcal{AD}$ is a set of probabilistic propositions from (2)-(5). We call $\mathcal{AD}$ a probabilistic action description. For convenience, we present an action a by the set $a = \{a_1, \ldots, a_n\}$, where for each $1 \leq i \leq n$, a_i corresponds to ϕ_i and ψ_i.

Example 1 ([5]). Assume that a robot is processing a widget. The goal of the robot is to paint (pa) and process (pr) the widget without errors ($\neg er$) by determining if it is flawed (fl) or not flawed ($\neg\ fl$), then deciding to reject or ship the widget by performing the *reject* or *ship* actions respectively. The flawed (fl) property is not directly observable. Therefore, the robot determines whether the widget is flawed by performing the sensing action *inspect* that senses whether the widget is blemished (bl) (a correlated property). But, the robot's sensor is not always perfect. If the widget is blemished (bl) then the robot reports that the widget is flawed (fl) with 0.9 probability, however, it erroneously reports that the widget is not flawed ($\neg\ fl$) with 0.1 probability. Performing the *paint* action in the state of the world in which the widget is not processed ($\neg\ pr$) causes the widget to be painted (pa) and all blemishes removed ($\neg\ bl$) with 0.95 probability, and causes no change in the state of the world with 0.05 probability. However, an error is caused (er) if *paint* is performed in the state of the world in which widget is being processed (pr). The effects of *ship* and *reject* are certain. Consider that initially the widget is blemished and flawed with probability 0.3 and it is not blemished and not flawed with probability 0.7. Consider also that the target is to find a probabilistic contingent plan that achieves its goal with

probability at least 0.95. This can be represented by the probabilistic action theory $\mathcal{D} = \langle \tilde{s}_I, \mathcal{AD} \rangle$ where

$$\tilde{s}_I = \textbf{initially} \left\{ \begin{array}{ll} \{bl, fl, \neg pa, \neg pr, \neg er\} & : 0.3 \\ \{\neg bl, \neg fl, \neg pa, \neg pr, \neg er\} & : 0.7 \end{array} \right\}$$

and $\mathcal{AD}$ consists of: **executable** AC **if** $\emptyset$, where $AC \in \{paint, inspect, ship, reject\}$.

$$paint \textbf{ causes} \left\{ \begin{array}{lll} \{pa, \neg bl\} & : 0.95 & \textbf{if } \{\neg pr\} \\ \emptyset & : 0.05 & \textbf{if } \{\neg pr\} \\ \{er\} & : 1 & \textbf{if } \{pr\} \end{array} \right\} \quad ship \textbf{ causes} \left\{ \begin{array}{lll} \{pr\} & : 1 & \textbf{if } \{\neg pr, \neg fl\} \\ \{pr, er\} & : 1 & \textbf{if } \{\neg pr, fl\} \\ \{er\} & : 1 & \textbf{if } \{pr\} \end{array} \right\}$$

$$reject \textbf{ causes} \left\{ \begin{array}{lll} \{pr, er\} & : 1 & \textbf{if } \{\neg pr, \neg fl\} \\ \{pr\} & : 1 & \textbf{if } \{\neg pr, fl\} \\ \{er\} & : 1 & \textbf{if } \{pr\} \end{array} \right\} \quad inspect \textbf{ determines} \left\{ \begin{array}{lll} \{fl\} & : 0.9 & \textbf{sensing } \{bl\} \\ \{\neg fl\} & : 0.1 & \textbf{sensing } \{bl\} \\ \{\neg fl\} & : 1 & \textbf{sensing } \{\neg bl\} \end{array} \right\}$$

3.2 Semantics of $\mathcal{P}$

We say a set of literals ϕ is consistent if it does not contain a pair of complementary literals. We say that a set of literals ϕ satisfies an indirect effect of action proposition of the form (2), if l belongs to ϕ whenever ψ is contained in ϕ or ψ is not contained in ϕ. Let $\mathcal{D}$ be a probabilistic action theory and ϕ be a set of literals. Then $\mathcal{C}_\mathcal{D}(\phi)$ is the smallest set of literals that contains ϕ and satisfies all the indirect effects of actions propositions in $\mathcal{D}$. A state s is a complete and consistent set of literals that satisfies all the indirect effects of actions propositions appearing in $\mathcal{D}$. Let s be a state and $\mathcal{G}$ be conjunctive fluent formula. The probability of $\mathcal{G}$ w.r.t. s is given by $Pr(\mathcal{G}|s) = 1$ if $\mathcal{G} \subseteq s$, otherwise, $Pr(\mathcal{G}|s) = 0$.

Definition 2. *Let $\mathcal{D}$ be a probabilistic action theory, s be a state, a* **causes**$\{\phi_i : p_i$ **if** $\psi_i \mid 1 \leq i \leq n\}$ *and a'* **determines**$\{\phi'_i : p'_i$ **sensing** $\psi'_i \mid 1 \leq i \leq n\}$ *be probabilistic propositions, and $a = \{a_i \mid (1 \leq i \leq n)\}$, $a' = \{a'_i \mid (1 \leq i \leq n)\}$, where each $a_i(a'_i)$ corresponds to ϕ_i (ϕ'_i) and ψ_i (ψ'_i). Then, $\mathcal{C}_\mathcal{D}(\Phi(a_i, s))$ ($\mathcal{C}_\mathcal{D}(\Phi(a'_i, s))$) is the state resulting from executing a (a') in s, where $\Phi(a_i, s)$ is defined as follows:*

1. *$l \in \Phi(a_i, s)$ and $\neg\, l \notin \Phi(a_i, s)$ if $l \in \phi_i$ and $\psi_i \subseteq s$.*
2. *$\neg\, l \in \Phi(a_i, s)$ and $l \notin \Phi(a_i, s)$ if $\neg\, l \in \phi_i$ and $\psi_i \subseteq s$.*
3. *Otherwise, $l \in \Phi(a_i, s)$ iff $l \in s$ and $\neg\, l \in \Phi(a_i, s)$ iff $\neg\, l \in s$.*

Moreover, $\Phi(a'_i, s)$ is defined as:

1. *$l \in \Phi(a'_i, s)$ and $\neg\, l \notin \Phi(a'_i, s)$ iff $l \in \phi_i$ and $\psi_i \subseteq s$.*
2. *$\neg\, l \in \Phi(a'_i, s)$ and $l \notin \Phi(a'_i, s)$ iff $\neg\, l \in \phi_i$ and $\psi_i \subseteq s$.*
3. *Otherwise, $l \in \Phi(a'_i, s)$ iff $l \in s$ and $\neg\, l \in \Phi(a'_i, s)$ iff $\neg\, l \in s$.*

We call Φ probabilistic transition function. An action in a probabilistic action theory $\mathcal{D}$ causes a transition from a probability distribution over possible states

to another probability distribution. Probability is assigned to a possible world state, resulting from executing an action in a given state, in which literals are deterministically true or false. The probability of a state s' resulting from executing $a(a')$ in a state s is given by $Pr'(s'|s,a) = p_i$ if a **causes**$\{\phi_i : p_i$ **if** $\psi_i \,|\, 1 \leq i \leq n\}$ and $s' = \mathcal{C}_{\mathcal{D}}(\Phi(a_i, s))$. Otherwise, $Pr'(s'|s,a) = 0$. $Pr'(s'|s,a') = p_i$ if a' **determines**$\{\phi_i : p_i$ **sensing** $\psi_i \,|\, 1 \leq i \leq n\}$ and $s' = \mathcal{C}_{\mathcal{D}}(\Phi(a'_i, s))$. Otherwise, $Pr'(s'|s,a') = 0$.

Definition 3 (Probabilistic Contingent Plan). *An empty sequence of actions $\langle\rangle$ is a probabilistic contingent plan. $\langle a, c\rangle$ is a probabilistic contingent plan, where a is a non-sensing action with probabilistic effects and c is a probabilistic contingent plan. $\langle a, case\ \{\phi_i \rightarrow c_i\}_{i=1}^{n}\rangle$ is a probabilistic contingent plan where a is a sensing action with probabilistic outcomes and c_i is a probabilistic contingent plan.*

The definition of probabilistic contingent plan is inspired from [28]. The probability that a conjunctive fluent formula $\mathcal{G}$ holds after executing a non-empty probabilistic contingent plan is given by the following definition.

Definition 4. *Let s, s' be states, $\mathcal{G}$ be a conjunctive fluent formula, $\tilde{s}_I$ be a random variable over the initial states, and $q = \langle a, c\rangle$ ($q = \langle a, case\ \{\phi_i \rightarrow c_i\}_{i=1}^{n}\rangle$) be probabilistic contingent plan. Then:*

- *The probability that s' holds after executing q in s is given by:*
  - $Pr(s'|s, \langle a, c\rangle) = \sum_{s''} Pr'(s''|s,a)\, Pr(s'|s'', c)$.
  - $Pr(s'|s, \langle a, case\ \{\phi_i \rightarrow c_i\}_{i=1}^{n}\rangle) = \sum_{s'' \models \phi_i} Pr'(s''|s,a) Pr(s'|s'', c_i)$.
- *The probability that $\mathcal{G}$ is true after executing q in s is given by:*
  - $Pr(\mathcal{G}|s, \langle a, c\rangle) = \sum_{s'} Pr(s'|s, \langle a, c\rangle)\, Pr(\mathcal{G}|s')$.
  - $Pr(\mathcal{G}|s, \langle a, case\ \{\phi_i \rightarrow c_i\}_{i=1}^{n}\rangle) = \sum_{s'} Pr(s'|s, \langle a, case\ \{\phi_i \rightarrow c_i\}_{i=1}^{n}\rangle)$ $Pr(\mathcal{G}|s')$.
- *The probability that $\mathcal{G}$ is true after executing q in the initial states $\tilde{s}_I$ is given by:*
  - $Pr(\mathcal{G}|\tilde{s}_I, \langle a, c\rangle) = \sum_{s} Pr(\mathcal{G}|s, \langle a, c\rangle)\, Pr(\tilde{s}_I = s)$.
  - $Pr(\mathcal{G}|\tilde{s}_I, \langle a, case\ \{\phi_i \rightarrow c_i\}_{i=1}^{n}\rangle) = \sum_{s} Pr(\mathcal{G}|s, \langle a, case\ \{\phi_i \rightarrow c_i\}_{i=1}^{n}\rangle)$ $Pr(\tilde{s}_I = s)$.

Definition 5. *A probabilistic contingent planning problem is a 4-tuple $\mathcal{PP} = \langle \tilde{s}_I, \mathcal{AD}, \mathcal{G}, \mathcal{T}\rangle$, where $\tilde{s}_I$ is a random variable over states represents the initial agent knowledge about the world at the time of execution, $\mathcal{AD}$ is a probabilistic action description, $\mathcal{G}$ is conjunctive fluent formula represents the goal to be satisfied, and $0 \leq \mathcal{T} \leq 1$ is the probability threshold for the goal $\mathcal{G}$ to be achieved. We say q is a probabilistic contingent plan for $\mathcal{PP}$ iff each a in q appears in $\mathcal{AD}$ and $Pr(\mathcal{G}|\tilde{s}_I, q) \geq \mathcal{T}$.*

4 Probabilistic Answer Set Contingent Planning

In this section we describe a logic based probabilistic continent planning approach called *probabilistic answer set contingent planning* that employs NHPP$_{\mathcal{P}}$ with probabilistic answer set semantics. Probabilistic answer set contingent planning is developed by providing a translation from a probabilistic contingent planning problem in $\mathcal{P}$ into a p-program. The p-program translation of a probabilistic contingent planning problem is adapted from the probabilistic answer set planning described in [22]. We assume that the length of a probabilistic contingent plan we are looking for is known. We use the predicates (i) $holds(L, T)$ to represent the fact that a literal L holds at time moment T, (ii) $occ(AC, T)$ to describe that an action AC executes at time moment T, and (iii) $state(T)$ to represent a possible state of the world at time moment T. We use lower case letters to represent constants and upper case letters to represent variables.

Let $\Pi_{\mathcal{PP}} = \langle R, \tau \rangle$ be the p-program translation of a probabilistic contingent planning problem, $\mathcal{PP}$, where τ is any arbitrary assignment of disjunctive p-strategies and R is the set of p-rules described below. To simplify the presentation, any atom appearing in a p-rule in R with no annotation is assumed to be associated with the annotation 1. In addition, given p is a predicate and $\psi = \{l_1, \ldots, l_n\}$, we use $p(\psi) : V$ to denote $p(l_1) : V_1, \ldots, p(l_n) : V_n$, where $V, V_1, \ldots, V_n$ are annotations.

– For each action $a = \{a_1, \ldots, a_n\} \in \mathcal{A}$ is represented in R by the set of facts

$$action(a_i) \leftarrow \tag{6}$$

Furthermore, literals that form the states of the world are represented in R by

$$literal(A) \leftarrow atom(A) \tag{7}$$
$$literal(\neg A) \leftarrow atom(A) \tag{8}$$

$atom(A)$ is a set of facts that describe the properties of the world. In addition, p-rules that specify that A and $\neg A$ are contrary literals are encoded as

$$contrary(A, \neg A) \leftarrow atom(A) \tag{9}$$
$$contrary(\neg A, A) \leftarrow atom(A) \tag{10}$$

– The initial probability distribution over the possible initial states is encoded as follows. For **initially** $\{\psi_i : p_i \mid 1 \leq i \leq n\}$ and $Pr(\tilde{s}_I = \psi_i) = p_i$, let each $\psi_i : p_i$ be represented as $\{l_{i,1} : p_{i,1}, l_{i,2} : p_{i,2}, \ldots, l_{i,m_i} : p_{i,m_i}\}$, where $\psi_i = \{l_{i,1}, l_{i,2}, \ldots, l_{i,m_i}\}$ is a set of literals and $p_i = \prod_{j=1}^{m_i} p_{i,j}$. $\forall (1 \leq j \leq m_i)$, the meaning of $l_{i,j} : p_{i,j}$ is that the literal $l_{i,j}$ holds with probability $p_{i,j}$. Moreover, let $\psi = \psi_1 \cup \psi_2 \cup \ldots \cup \psi_n$, $\psi' = \psi_1 \cap \psi_2 \cap \ldots \cap \psi_n$, and $\widehat{\psi} = \psi - \psi'$. Let $\psi^{report} = \{\, l \mid l \in \psi_i$ and l is a sensor report literal $\}$ be the set of all sensor report literals in all ψ_i. We denote $\psi'' = \{\, l \mid l \in (\widehat{\psi} - \psi^{report}) \vee \neg l \in (\widehat{\psi} - \psi^{report})\}$. Intuitively, ψ'' is the same as $\widehat{\psi}$ after excluding the set of sensor

report literals ψ^{report} from $\widehat{\psi}$. Let ψ^{sense} be the set of all pairs (δ_i, γ_i), where δ_i and γ_i are sets of literals contained in $\psi_i, \forall\ (1 \leq i \leq n)$, such that δ_i is the set of sensor reading literals and γ_i is the set of sensor report literals appearing in ψ_i. Let us denote a literal in $\psi_i, \psi', \psi'', \delta_i, \gamma_i$ by $l_{i,j}$ for any i and j. Notice also that for any literal $l_{i,j}$, there is an associated probability $p_{i,j}$. To generate the set of all possible initial states we add to R the following set of p-rules. For each literal $l_{i,j} \in \psi'$, we add to R the fact

$$holds(l_{i,j}, 0) : p_{i,j} \leftarrow \tag{11}$$

where $p_{i,j}$ is the probability $l_{i,j}$ holds in the initial situation. This fact says that the literal $l_{i,j}$ holds at time moment 0 with probability $p_{i,j}$. This set of facts represents the set of literals that hold in every possible initial state along with their associated probability. For each literal $l_{i,j} \in \psi''$,

$$holds(l_{i,j}, 0) : p_{i,j} \leftarrow not\ holds(\neg l_{i,j}, 0) : \overline{p}_{i,j} \tag{12}$$

$$holds(\neg l_{i,j}, 0) : \overline{p}_{i,j} \leftarrow not\ holds(l_{i,j}, 0) : p_{i,j} \tag{13}$$

are added to R, where $p_{i,j}$ and $\overline{p}_{i,j}$ are the probabilities that $l_{i,j}$ and $\neg l_{i,j}$ hold respectively. The above p-rules describe that the literal l (similarly $\neg l$) holds with probability $p_{i,j}$ at time moment 0, if $\neg l$ (similarly l) does not hold with probability $\overline{p}_{i,j}$ at the time moment 0. For each $(\delta_i, \gamma_i) \in \psi^{sense}$, let $\delta_i = \{l_{i,1}, l_{i,2}, \ldots, l_{i,m_i}\}$, then for each $l_{i,j}$ in γ_i, we have

$$holds(l_{i,j}, 0) : p_{i,j} \leftarrow holds(l_{i,1}) : p_{i,1}, holds(l_{i,2}) : p_{i,1}, \ldots, holds(l_{i,m_i}) : p_{i,1} \tag{14}$$

belongs to R. The initial probability distribution over the possible initial states is encoded in R as

$$state(0) : p_i \leftarrow holds(l_{i,1}, 0) : p_{i,1}, holds(l_{i,2}, 0) : p_{i,2}, \ldots, holds(l_{i,m_i}, 0) : p_{i,m_i} \tag{15}$$

for all $(1 \leq i \leq n)$ and $p_i = \prod_{j=1}^{m_i} p_{i,j}$. The above p-rule says that the probability of a state at time moment 0 (a possible initial state) is p_i if $l_{i,1}, l_{i,2}, \ldots, l_{i,m_i}$ hold at the time moment 0 with probability $p_{i,1}$, $p_{i,2}, \ldots, p_{i,m_i}$ respectively.

– Let V, V' be annotation variables act as place holders, then for each indirect effect of an action proposition of the form (2), R contains

$$holds(l, T) : V' \leftarrow holds(\psi, T) : V \tag{16}$$

– For each action $a = \{a_1, \ldots, a_n\}$, each executability condition of the form (3) is encoded as

$$exec(a_i, T) \leftarrow holds(\psi, T) : V \tag{17}$$

– For each probabilistic conditional effects of a non-sensing action of the form a **causes** $\{\phi_i : p_i$ **if** $\psi_i \mid 1 \leq i \leq n\}$, let for each $\phi_i : p_i$ be represented as $\{l_{i,1} : p_{i,1}, l_{i,2} : p_{i,2}, \ldots, l_{i,m_i} : p_{i,m_i}\}$, where $\phi_i = \{l_{i,1}, l_{i,2}, \ldots, l_{i,m_i}\}$ and $p_i = \prod_{j=1}^{m_i} p_{i,j}$. Then for each $(1 \leq i \leq n)$ and $(1 \leq j \leq m_i)$, we have

$$holds(l_{i,j}, T+1) : p_{i,j} \leftarrow occ(a_i, T), exec(a_i, T), holds(\psi_i, T) : V \tag{18}$$

This p-rule states that if the action a occurs at time moment T and ψ_i holds at the same time moment with probability V, then the literal $l_{i,j}$ holds at time moment $T+1$ with probability $p_{i,j}$. The probability distribution over states resulting from executing a is represented in R by:

$$state(T+1) : p_i \times V \leftarrow state(T) : V, occ(a_i, T), exec(a_i, T), holds(\psi_i, T) : V', \\ holds(\phi_i, T+1) : p_i \quad (19)$$

where V, V' are annotation variables ranging over $[0, 1]$ acts as place holders. The above p-rule says that if ψ_i is true with probability V' in a state of the world at time moment T, whose probability is V, and in which the action a is executable, then the probability of a successor state at time moment $T+1$, after executing an action a in the state at time T, is $p_i \times V$, in which the effect ϕ_i is true with probability p_i.

– For each sensing action with probabilistic outcomes of the form a **determines** $\{\phi_i : p_i$ **sensing** $\psi_i \,|\, 1 \leq i \leq n\}$, let each $\phi_i : p_i$ be represented as $\{l_{i,1} : p_{i,1}, l_{i,2} : p_{i,2}, \ldots, l_{i,m_i} : p_{i,m_i}\}$, where $\phi_i = \{l_{i,1}, l_{i,2}, \ldots, l_{i,m_i}\}$ and $p_i = \prod_{j=1}^{m_i} p_{i,j}$. Then for each $(1 \leq i \leq n)$ and $(1 \leq j \leq m_i)$ and for each $l'_{i,j} \in \psi_i$, we have in R

$$sense(l'_{i,j}, T) : V' \leftarrow occ(a_i, T), exec(a_i, T), holds(\psi_i, T) : V \quad (20)$$

$$holds(l_{i,j}, T+1) : p_{i,j} \leftarrow occ(a_i, T), exec(a_i, T), sense(\psi_i, T) : V \quad (21)$$

The p-rule (20) says that executing the sensing action a at time moment T in which ψ_i holds with probability V causes ψ_i to be sensed to be known true at the same time moment T. The p-rule (21) states that if a sensing action a occurs at time moment T and the sensed-literals ψ_i is sensed to be known true at the same time moment with probability V, then the literal $l_{i,j}$ is known to hold at the time moment $T+1$ with probability $p_{i,j}$. The probability distribution over states resulting from executing the sensing action a is represented in R by

$$state(T+1) : p_i \times V \leftarrow state(T) : V, occ(a_i, T), exec(a_i, T), sense(\psi_i, T) : V', \\ holds(\phi_i, T+1) : p_i \quad (22)$$

The above p-rule says that the probability of a state at time moment $T+1$ is $p_i \times V$ if ϕ_i become known to be true at the same time moment with probability p_i, after executing the sensing action a in a state of the world at time T, whose probability is V, in which ψ_i is sensed to be true with probability V'.

– The frame axioms are encoded in R as follows. For any literal L we have

$$holds(L, T+1) : V \leftarrow holds(L, T) : V, not\, holds(L', T+1) : V', contrary(L, L') \quad (23)$$

The above p-rule says that L continues to hold at the time moment $T+1$ with probability V if L holds at the time moment T with the same probability and its contrary does not hold at the time moment $T+1$ with probability V'.

– To encode that a literal A and its negation $\neg A$ cannot hold at the same time

$$inconsistent \leftarrow not\ inconsistent, holds(A, T) : V, holds(\neg A, T) : V' \quad (24)$$

is added to R, where *inconsistent* is a special literal that does not appear in $\mathcal{P}$.
– Actions are generated using the p-rules

$$occ(AC^i, T) \leftarrow action(AC^i), not\ abocc(AC^i, T) \quad (25)$$
$$abocc(AC^i, T) \leftarrow occ(AC^j, T), AC^i \neq AC^j \quad (26)$$

The above two p-rules generate action occurrences once at a time, where AC^i and AC^j are variables representing actions.
– A goal $\mathcal{G} = g_1 \wedge \ldots \wedge g_m$ is encoded in R as a p-rule of the form

$$goal \leftarrow holds(g_1, T) : V_1, \ldots, holds(g_m, T) : V_m \quad (27)$$

Example 2. The p-program encoding, $\Pi_{\mathcal{PP}} = \langle R, \tau \rangle$, of the probabilistic contingent planning problem $\mathcal{PP} = \langle \tilde{s}_I, \mathcal{AD}, \mathcal{G}, \mathcal{T} \rangle$ presented in Example 1 is given as follows, where τ is any arbitrary assignment of disjunctive p-strategies and R consists of the following p-rules, in addition to the p-rules (7), (8), (9), (10), (23), (24), (25), (26):

$$action(paint_i) \leftarrow \quad action(ship_i) \leftarrow \quad action(reject_i) \leftarrow \quad action(inspect_i) \leftarrow$$

for $1 \leq i \leq 3$. Properties of the world are described by the atoms *bl* (widget blemished), *fl* (widget flawed), *pa* (widget painted), *pr* (widget processed), and *er* (error occurred), which are encoded in R by the p-rules

$$atom(bl) \leftarrow \quad atom(fl) \leftarrow \quad atom(pa) \leftarrow \quad atom(pr) \leftarrow \quad atom(er) \leftarrow$$

The set of possible initial states are encoded by the p-rules:

$$holds(\neg pa, 0) \leftarrow \qquad holds(\neg pr, 0) \leftarrow \qquad holds(\neg er, 0) \leftarrow$$
$$holds(bl, 0) \leftarrow not\ holds(\neg bl, 0) \qquad holds(\neg bl, 0) \leftarrow not\ holds(bl, 0)$$
$$holds(fl, 0) : 0.3 \leftarrow holds(bl, 0) \qquad holds(\neg fl, 0) : 0.7 \leftarrow holds(\neg bl, 0)$$

The initial probability distribution over the possible initial states is encoded as

$$state(0) : 0.3 \leftarrow holds(bl, 0), holds(fl, 0) : 0.3, holds(\neg pa, 0), holds(\neg pr, 0), holds(\neg er, 0)$$
$$state(0) : 0.7 \leftarrow holds(\neg bl, 0), holds(\neg fl, 0) : 0.7, holds(\neg pa, 0), holds(\neg pr, 0), holds(\neg er, 0)$$

The executability conditions of actions are encoded by the following p-rules

$$exec(paint_i) \leftarrow \quad exec(ship_i) \leftarrow \quad exec(reject_i) \leftarrow \quad exec(inspect_i) \leftarrow$$

for $1 \leq i \leq 3$. Effects of the *paint* action are encoded by the p-rules

$$holds(pa, T+1) \qquad \leftarrow occ(paint_1, T), exec(paint_1, T), holds(\neg pr, T) : V$$
$$holds(\neg bl, T+1) : 0.95 \leftarrow occ(paint_1, T), exec(paint_1, T), holds(\neg pr, T) : V$$
$$holds(er, T+1) \qquad \leftarrow occ(paint_3, T), exec(paint_3, T), holds(pr, T) : V$$

The probability distribution resulting from executing the *paint* action is given by

$$\begin{aligned}
&state(T+1):0.95\times V \leftarrow occ(paint_1,T), exec(paint_1,T), state(T):V, holds(\neg pr,T):V',\\
&\qquad\qquad holds(pa,T+1), holds(\neg bl,T+1):0.95\\
&state(T+1):0.05\times V \leftarrow occ(paint_2,T), exec(paint_2,T), state(T):V, holds(\neg pr,T):V'\\
&state(T+1):V \leftarrow occ(paint_3,T), exec(paint_3,T), state(T):V, holds(pr,T):V',\\
&\qquad\qquad holds(er,T+1)
\end{aligned}$$

Effects of the *ship* action are encoded by the p-rules

$$\begin{aligned}
&holds(pr,T+1) \leftarrow occ(ship_1,T), exec(ship_1,T), holds(\neg pr,T):V_1, holds(\neg fl,T):V_2\\
&holds(pr,T+1) \leftarrow occ(ship_2,T), exec(ship_2,T), holds(\neg pr,T):V_1, holds(fl,T):V_2\\
&holds(er,T+1) \leftarrow occ(ship_2,T), exec(ship_2,T), holds(\neg pr,T):V_1, holds(fl,T):V_2\\
&holds(er,T+1) \leftarrow occ(ship_3,T), exec(ship_3,T), holds(pr,T):V
\end{aligned}$$

The probability distribution resulting from executing the *ship* action is given by

$$\begin{aligned}
&state(T+1):V \leftarrow occ(ship_1,T), exec(ship_1,T), state(T):V, holds(\neg pr,T):V_1,\\
&\qquad\qquad holds(\neg fl,T):V_2, holds(pr,T+1):V_3\\
&state(T+1):V \leftarrow occ(ship_2,T), exec(ship_2,T), state(T):V, holds(\neg pr,T):V_1,\\
&\qquad holds(fl,T):V_2, holds(pr,T+1):V_3, holds(er,T+1):V_4\\
&state(T+1):V \leftarrow occ(ship_3,T), exec(ship_3,T), state(T):V, holds(pr,T):V_1,\\
&\qquad\qquad holds(er,T+1):V_2
\end{aligned}$$

Effects of the *reject* action are encoded by the p-rules

$$\begin{aligned}
&holds(pr,T+1) \leftarrow occ(reject_1,T), exec(reject_1,T), holds(\neg pr,T):V_1, holds(\neg fl,T):V_2\\
&holds(er,T+1) \leftarrow occ(reject_1,T), exec(reject_1,T), holds(\neg pr,T):V_1, holds(\neg fl,T):V_2\\
&holds(pr,T+1) \leftarrow occ(reject_2,T), exec(reject_2,T), holds(\neg pr,T):V_1, holds(fl,T):V_2\\
&holds(er,T+1) \leftarrow occ(reject_3,T), exec(reject_3,T), holds(pr,T):V
\end{aligned}$$

The probability distribution resulting from executing the *reject* action is given by

$$\begin{aligned}
&state(T+1):V \leftarrow occ(reject_1,T), exec(reject_1,T), state(T):V, holds(\neg pr,T):V_1,\\
&\qquad holds(\neg fl,T):V_2, holds(pr,T+1):V_3, holds(er,T+1):V_4\\
&state(T+1):V \leftarrow occ(reject_2,T), exec(reject_2,T), state(T):V, holds(\neg pr,T):V_1,\\
&\qquad holds(fl,T):V_2, holds(pr,T+1):V_3\\
&state(T+1):V \leftarrow occ(reject_3,T), exec(reject_3,T), state(T):V, holds(pr,T):V_1,\\
&\qquad holds(er,T+1):V_2
\end{aligned}$$

Effects of the *inspect* action are encoded by the p-rules

$$\begin{aligned}
&sense(bl,T):V &&\leftarrow occ(inspect_1,T), exec(inspect_1,T), holds(bl,T):V\\
&sense(bl,T):V &&\leftarrow occ(inspect_2,T), exec(inspect_2,T), holds(bl,T):V\\
&sense(\neg bl,T):V &&\leftarrow occ(inspect_3,T), exec(inspect_3,T), holds(\neg bl,T):V\\
&holds(fl,T+1):0.9 &&\leftarrow occ(inspect_1,T), exec(inspect_1,T), sense(bl,T):V\\
&holds(\neg fl,T+1):0.1 &&\leftarrow occ(inspect_2,T), exec(inspect_2,T), sense(bl,T):V\\
&holds(\neg fl,T+1) &&\leftarrow occ(inspect_3,T), exec(inspect_3,T), sense(\neg bl,T):V
\end{aligned}$$

The probability distribution resulting from executing the *inspect* action is given by

$$\begin{aligned} state(T+1):0.9\times V \leftarrow\ & occ(inspect_1,T), exec(inspect_1,T), state(T):V, sense(bl,T):V_1,\\ & holds(fl,T+1):0.9\\ state(T+1):0.1\times V \leftarrow\ & occ(inspect_2,T), exec(inspect_2,T), state(T):V, sense(bl,T):V_1,\\ & holds(\neg fl,T+1):0.1\\ state(T+1):V \leftarrow\ & occ(inspect_3,T), exec(inspect_3,T), state(T):V, sense(\neg bl,T):V_1,\\ & holds(\neg fl,T+1):V_2 \end{aligned}$$

The goal is encoded by the p-rule

$$goal \leftarrow holds(pa,T):V_1, holds(pr,T):V_2, holds(\neg er,T):V_3$$

5 Correctness

In this section we prove the correctness of the probabilistic answer set contingent planning. Let the domain of T be $\{0,\ldots,n\}$. Let $\mathcal{PP}=\langle \tilde{s}_I, \mathcal{AD}, \mathcal{G}, \mathcal{T}\rangle$ be a probabilistic contingent planning problem, Φ be a probabilistic transition function associated with $\mathcal{PP}$, s_0 be a possible initial state, and $a_0,\ldots,a_n$ be a collection of (sensing and non-sensing) actions in $\mathcal{A}$. We say that $s_0\ a_{j_0}\ s_1\ldots a_{j_n}\ s_{n+1}$ is a trajectory in $\mathcal{PP}$ if $s_{i+1}=\mathcal{C}_{\mathcal{D}}(\Phi(a_{j_i},s_i))$, where $\forall(0\leq i\leq n)$, s_i is a state, a_i is an action, and $a_{j_i}\in a_i=\{a_{1_i},\ldots,a_{m_i}\}$. A trajectory $s_0\ a_{j_0}\ s_1\ldots a_{j_n}\ s_{n+1}$ in $\mathcal{PP}$ is said to achieve a conjunctive fluent formula $\mathcal{G}$ if $\mathcal{G}\subseteq s_{n+1}$. Moreover, let $\mathcal{R}_{\mathcal{G}}$ be the set of all trajectories $s_0\ a_{j_0}\ s_1\ldots a_{j_n}\ s_{n+1}$ in $\mathcal{PP}$ that achieve $\mathcal{G}$. We say that a probabilistic contingent plan q achieves $\mathcal{G}$ if the execution of q in the initial states will yield a non-empty set of trajectories $\mathcal{R}_{\mathcal{G}}$ each of which achieves $\mathcal{G}$.

Theorem 1. *Let $\mathcal{PP}=\langle \tilde{s}_I, \mathcal{AD}, \mathcal{G}, \mathcal{T}\rangle$ be a probabilistic contingent planning problem and $\mathcal{G}\equiv g_1\wedge\ldots\wedge g_m$. Then, $\mathcal{G}$ is achievable from $\mathcal{PP}$ iff $\mathcal{G}'(t)\equiv holds(g_1,t),\ldots,holds(g_m,t)$ is true (satisfied) in some probabilistic answer set of $\Pi_{\mathcal{PP}}$, for some $0\leq t\leq n$.*

Intuitively, the probabilistic answer sets of the p-program translation, $\Pi_{\mathcal{PP}}$, of a probabilistic contingent planning problem, $\mathcal{PP}$, are equivalent to the trajectories of $\mathcal{PP}$. Let $\mathcal{OCC}$ be a set such that $s_0\ a_{j_0}\ s_1\ldots a_{j_n}\ s_{n+1}\in\mathcal{R}_{\mathcal{G}}$ iff $occ(a_{j_0},0),\ldots,occ(a_{j_n},n)\in\mathcal{OCC}$, where $a_{j_0},\ldots,a_{j_n}$ correspond to actions appearing in q

Lemma 1. *Let h be a probabilistic answer set of $\Pi_{\mathcal{PP}}$ and $q=\langle a, case\ \{\phi_i\rightarrow c_i\}_{i=1}^m\rangle$ (similarly $q=\langle a,c\rangle$) (possibly empty) be a probabilistic contingent plan for $\mathcal{PP}$. Then, $\sum_{h\models\mathcal{G}'(n+1)\ and\ h\models occ(a_{j_0},0),\ldots,occ(a_{j_n},n)\in\mathcal{OCC}} h(state(n+1))=Pr(\mathcal{G}|\tilde{s}_I,q)$, where $\mathcal{G}=g_1\wedge\ldots\wedge g_m$ and $\mathcal{G}'(n+1)\equiv holds(g_1,n+1):p_1,\ldots,holds(g_m,n+1):p_m$.*

The probability a goal $\mathcal{G}$ is true, after executing a probabilistic contingent plan q of $\mathcal{PP}$ in the possible initial states $\tilde{s}_I$, is equivalent to the summation of

the probabilities $h(state(n+1))$ over the probabilistic answer sets h of $\Pi_{\mathcal{PP}}$ that satisfy $\mathcal{G}'(n+1)$ (the goal $\mathcal{G}$ encoding in $\Pi_{\mathcal{PP}}$), where h contains actions appearing in q. The following theorem follows directly from Lemma 1.

Theorem 2. *Let h be a probabilistic answer set of $\Pi_{\mathcal{PP}}$ and q (possibly empty) be a probabilistic contingent plan for $\mathcal{PP}$. Then, $Pr(\mathcal{G}|\tilde{s}_I, q) \geq \mathcal{T}$ iff $\sum_{h \models \mathcal{G}'(n+1) \text{ and } h \models occ(a_{j_0},0),\ldots,occ(a_{j_n},n) \in \mathcal{OCC}} h(state(n+1)) \geq \mathcal{T}$.*

Probabilistic answer set contingent planning produces acyclic plans using flat representation of the probabilistic contingent planning domains. According to the classification of [16], acyclic plan, also referred to as totally ordered conditional (contingent) plan, is a total order plan with conditional (contingent) execution of actions. Flat representation of probabilistic planning domains is the explicit enumeration of the world states [16]. Hence, Theorem 4 follows directly from Theorem 3.

Theorem 3 ([16]). *The plan existence problem in the flat representation of probabilistic planning domains is NP-complete.*

Theorem 4. *The plan existence problem in probabilistic answer set contingent planning for probabilistic planning domains is NP-complete.*

6 Probabilistic Contingent Planning Using Answer Sets

In [22], it has been shown that probabilistic planning problems can be encoded as classical normal logic programs with classical answer set semantics. In the rest of this section, we follow the steps of [22]. In this section we show that any probabilistic contingent planning problem can be encoded as a classical normal logic program with classical answer set semantics. Excluding p-rules (15), (19), and (22) and replacing all annotations in the p-program translation, $\Pi_{\mathcal{PP}}$, of $\mathcal{PP}$ with 1 yields a p-program, denoted by $\Pi_{\mathcal{PP}}^{normal}$, with only annotations of the form 1. As shown in [24], the syntax and semantics of this class of p-programs is equivalent to classical normal logic programs with classical answer set semantics.

Theorem 5. *Let $\Pi_{\mathcal{PP}}^{normal}$ be the* normal logic program *resulting from $\Pi_{\mathcal{PP}}$ after deleting the p-rules (15), (19) and (22) and replacing all annotations in $\Pi_{\mathcal{PP}}$ with* 1*. Then, a trajectory $s_0\ a_{j_0}\ s_1 \ldots a_{j_n}\ s_{n+1}$ in $\mathcal{PP}$ achieves $\mathcal{G} = g_1 \wedge \ldots \wedge g_m$ iff $\mathcal{G}'(n+1) \equiv holds(g_1, n+1), \ldots, holds(g_m, n+1)$ is true in some answer set of $\Pi_{\mathcal{PP}}^{normal}$.*

Theorem 5 shows that classical normal logic programs with classical answer set semantics can be used to solve probabilistic contingent planning problems in two steps. The first step is to translate a probabilistic contingent planning problem, $\mathcal{PP}$, into a classical normal logic program whose answer sets corresponds to valid trajectories in $\mathcal{PP}$. From the answer sets of the normal logic program translation of $\mathcal{PP}$, we can determine the trajectories $\mathcal{R}_q$ of a plan q in $\mathcal{PP}$ that achieve the

goal $\mathcal{G}$. The second step is to calculate the probability that the goal is satisfied by $\sum_{s_0\ a_{j0}\ s_1 \ldots a_{jn}\ s_{n+1} \in \mathcal{R}_q} Pr(s_0)\ \prod_{i=0}^{n} p_{j_i}$.

Now, we show that any probabilistic contingent planning problem can be encoded as a SAT problem. Hence, state-of-the-art SAT solvers can be used to solve probabilistic contingent planning problems. Any normal logic program, Π, can be translated into a SAT problem, $\mathcal{S}$, where the models of $\mathcal{S}$ are equivalent to the answer sets of Π [15]. Therefore, the normal logic program encoding of a probabilistic contingent planning problem $\mathcal{PP}$ can be translated into an equivalent SAT problem, $\mathcal{S}$, where the models of $\mathcal{S}$ correspond to valid trajectories in $\mathcal{PP}$.

Theorem 6. *Let $\mathcal{PP}$ be a probabilistic contingent planning problem and $\Pi_{\mathcal{PP}}^{normal}$ be a normal logic program translation of $\mathcal{PP}$. The models of the SAT encoding of $\Pi_{\mathcal{PP}}^{normal}$ are equivalent to valid trajectories in $\mathcal{PP}$.*

However, similar to [22], in encoding probabilistic contingent planning problems into normal logic programs or SAT, explicit representation and assignment of probabilities to states rely on an external mechanism to normal logic programs syntax and semantics and not on normal logic programs syntax and semantics themselves. These issues are overcome naturally by encoding probabilistic contingent planning problems in NHPP.

7 Conclusions and Related Work

We presented a new high level probabilistic action language $\mathcal{P}$ that allows the representation of imperfect sensing actions, in addition, we introduced a new probabilistic contingent planning approach by relating probabilistic contingent planning to NHPP.

The syntax and semantics of the probabilistic action language $\mathcal{P}$ is built on top of the action language $\mathcal{A}_K$ [28] and the probabilistic action representation of [5]. The major difference between $\mathcal{A}_K$ and $\mathcal{P}$ is that $\mathcal{P}$ allows the specification of sensing actions with probabilistic outcomes and non-sensing actions with probabilistic conditional effects and the presentation of the initial probability distribution over the possible initial states. Unlike the probabilistic action representation of [5,19], $\mathcal{P}$ is a high level language and allows the representation of indirect effects of actions and executability conditions of actions. In addition, in $\mathcal{P}$, the sensing actions outcomes are represented by fluents that describe the world. Contrary to [5] and [19], we treat non-sensing and sensing actions differently in the sense that a sensing action has no preconditions and effects. The action language $\mathcal{E}^+$ [11] allows sensing actions, actions with probabilistic effects, and actions with non-deterministic effects. In $\mathcal{P}$ actions with non-deterministic effects are not allowed. However, similar to $\mathcal{A}_K$, $\mathcal{E}^+$ assumes that the agent's sensors are perfect. In addition, the semantics of $\mathcal{E}^+$ is based on description logic. Other high level probabilistic action description languages are described in [2,6], but they do not allow sensing actions.

Probabilistic answer set contingent planning extends probabilistic answer set planning [22] to deal with sensing actions with probabilistic outcomes. The translation from a probabilistic contingent planning problem in $\mathcal{P}$ into a p-program is mainly adapted from [22]. In [5,19] a probabilistic contingent planning approach based on partial order planning is presented. [5,19] extends [14], a probabilistic partial order planner, with imperfect sensing actions to generate contingent plans. Based on planning as satisfiability approach [13] for classical planning, a probabilistic contingent planning approach is developed [18]. A probabilistic contingent planning problem $\mathcal{PP}$ in [18] is solved by converting $\mathcal{PP}$ into a stochastic satisfiability problem and solving the stochastic satisfiability problem instead. [18] extends [17], a stochastic satisfiability based probabilistic planning approach, with sensing action with probabilistic outcomes to generate probabilistic contingent plans. Our approach is similar in spirit to [18] in the sense that both approaches are logic based approaches. However, solving stochastic satisfiability problem is NP^{PP}-complete, but, probabilistic answer set contingent planning is NP-complete. The probabilistic contingent planning approach in [4] is based on another classical planning approach (heuristic based planner) [3]. [4] produces probabilistic contingent plans, similar to [18,5].

References

1. Baral, C., Gelfond, M., Rushton, N.: Probabilistic reasoning with answer sets. In: Logic Programming and Non-monotonic Reasoning. Springer, Heidelberg (2004)
2. Baral, C., et al.: Reasoning about actions in a probabilistic setting. In: AAAI (2002)
3. Blum, A., Furst, M.: Fast planning through planning graph analysis. Artificial Intelligence 90(1-2), 297–298 (1997)
4. Blum, A., Langford, J.: Probabilistic planning in the Graphplan framework. In: ECP (1999)
5. Draper, D., Hanks, S., Weld, D.: Probabilistic planning with information gathering and contingent execution. In: Proc. of the 2nd AIPS, pp. 31–37 (1994)
6. Eiter, T., Lukasiewicz, T.: Probabilistic reasoning about actions in nonmonotonic causal theories. In: 19th Conference on Uncertainty in Artificial Intelligence, UAI 2003 (2003)
7. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In: ICSLP (1988)
8. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New Generation Computing 9(3-4), 363–385 (1991)
9. Gelfond, M., Lifschitz, V.: Representing action and change by logic programs. Journal of Logic Programming 17, 301–321 (1993)
10. Giunchiglia, E., Lierler, Y., Maratea, M.: Answer set programming based on propositional satisfiability. Journal of Automated Reasoning 36(4), 345–377 (2006)
11. Iocchi, L., Lukasiewicz, T., Nardi, D., Rosati, R.: Reasoning about actions with sensing under qualitative and probabilistic uncertainty. In: 16th ECAI (2004)
12. Kaelbling, L., Littman, M., Cassandra, A.: Planning and acting in partially observable stochastic domains. Artificial Intelligence 101, 99–134 (1998)
13. Kautz, H., Selman, B.: Pushing the envelope: planning, propositional logic, and stochastic search. In: Proc. of 13th National Conference on Artificial Intelligence (1996)

14. Kushmerick, N., Hanks, S., Weld, D.: An algorithm for probabilistic planning. Artificial Intelligence 76(1-2), 239–286 (1995)
15. Lin, F., Zhao, Y.: ASSAT: Computing answer sets of a logic program by SAT solvers. Artificial Intelligence 157(1-2), 115–137 (2004)
16. Littman, M., Goldsmith, J., Mundhenk, M.: The computational complexity of probabilistic planning. Journal of Artificial Intelligence Research 9, 1–36 (1998)
17. Majercik, S., Littman, M.: MAXPLAN: A new approach to probabilistic planning. In: Proc. of the 4th International Conference on Artificial Intelligence Planning (1998)
18. Majercik, S., Littman, M.: Contingent planning under uncertainty via stochastic satisfiability. Artificial Intelligence 147(1–2), 119–162 (2003)
19. Onder, N., Pollack, M.: Contingency Selection in plan generation. In: 4th ECP (1997)
20. Saad, E.: Incomplete knowlege in hybrid probabilistic logic programs. In: JELIA (2006)
21. Saad, E.: A logical approach to qualitative and quantitative reasoning. In: 9th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (2007)
22. Saad, E.: Probabilistic planning in hybrid probabilistic logic programs. 1st SUM (2007)
23. Saad, E., Pontelli, E.: Towards a more practical hybrid probabilistic logic programming framework. Practical Aspects of Declarative Languages (2005)
24. Saad, E., Pontelli, E.: A new approach to hybrid probabilistic logic programs. Annals of Mathematics and Artificial Intelligence Journal 48(3-4), 187–243 (2006)
25. Scherl, R., Levesque, H.: The frame problem and knowledge producing actions. In: AAAI (1993)
26. Son, T., Baral, C.: Formalizing sensing actions - a transition function based approach. Artificial Intelligence 125(1-2), 19–91 (2001)
27. Subrahmanian, V.S., Zaniolo, C.: Relating stable models and AI planning domains. In: International Conference of Logic Programming, pp. 233–247 (1995)
28. Tu, P., Son, T., Baral, C.: Reasoning and planning with sensing actions, incomplete information, and static causal laws using logic programming. TPLP 7(4), 377–450

Extended Fuzzy Logic Programs with Fuzzy Answer Set Semantics

Emad Saad

Department of Computer Science
Gulf University for Science and Technology
Mishref, Kuwait
saad.e@gust.edu.kw

Abstract. This paper extends fuzzy logic programs [12, 42] to allow the explicit representation of classical negation as well as non-monotonic negation, by introducing the notion of extended fuzzy logic programs. We present the fuzzy answer set semantics for the extended fuzzy logic programs, which is based on the classical answer set semantics of classical extended logic programs [7]. We show that the proposed semantics is a natural extension to the classical answer set semantics of classical extended logic programs [7]. Furthermore, we define fixpoint semantics for extended fuzzy logic programs with and without non-monotonic negation, and study their relationship to the fuzzy answer set semantics. In addition, we show that the fuzzy answer set semantics is reduced to the stable fuzzy model semantics for normal fuzzy logic programs introduced in [42]. The importance of that is computational methods developed for normal fuzzy logic programs can be applied to the extended fuzzy logic programs. Moreover, we show that extended fuzzy logic programs can be intuitively used for representing and reasoning about actions in fuzzy environment.

1 Introduction

Managing uncertainty is vital for real-world applications including those in AI domains. The literature is rich of proposals with extensions to logic programming with different notions of uncertainty. According to the way certainty values are attached to the rules and facts in logic programs, the frameworks for dealing with uncertainty in logic programming can be classified into two main approaches: annotation based (AB) approaches and implication based (IB) approaches. The semantics of logic programs under either approaches depends on the underlying formalisms which both approaches are based on. These formalisms include *fuzzy set theory* [10, 22, 25, 39, 40, 43], *possibilistic logic* [5], *multi-valued logic* [11, 12], and formalisms based on *probability theory* [2, 15, 26–30, 35–37].

In the implication-based approach, certainty values are associated to the rules as a whole as well as to the facts present in a program [10, 17, 22, 25, 39, 40, 43]. A rule in the IB approach is an expression of the form

$$(a \leftarrow a_1, \ldots, a_n, \theta)$$

L. Godo and A. Pugliese (Eds.): SUM 2009, LNAI 5785, pp. 223–239, 2009.

where that $a, a_1, \ldots, a_n$ are atoms and θ is the *certainty* associated to the rule. Intuitively, the certainty of the rule head a is determined by propagating the certainty of the rule body $a_1, \ldots, a_n$ to the rule head using the certainty θ attached to the rule. While in the annotation based approach, certainty values are attached to every sub-goal and head of rules of the logic program [2, 11, 12, 26–30, 35–37]. A rule in the AB approach is an expression of the form

$$a : \mu \leftarrow a_1 : \mu_1, \ldots, a_n : \mu_n$$

where $a, a_1, \ldots, a_n$ are atoms and $\mu, \mu_1, \ldots, \mu_n$ are annotations. The intuitive meaning of that rule is that if the certainty of a_1 is at least $\mu_1, \ldots$, and the certainty of a_n is at least μ_n, then the certainty of a is at least μ.

The main difference between the annotation based and the implication based approaches is that the annotation based approach is strictly more expressive than the implication based approach [12, 27]. This makes AB approach more suitable for complex applications than IB approach. Moreover, it has been shown that the AB approach of [12] subsumes van Emden IB approach for fuzzy logic programming [43]. In addition, as argued in [17], an appropriately defined AB approach can subsumes any IB approach which uses multisets as the basis of its semantics. Another difference between the IB and the AB approaches is that unlike IB, the way a rule is fired in AB approach is close to classical logic programming [12]. This is an important feature in the AB approach because it makes any possible extension to more expressive forms of logic programming with uncertainty in AB approach, including the addition of negation as failure, classical negation, and disjunctions, is more intuitive and more flexible than the IB approach as proposed in [28–30, 36, 42].

It is known that non-monotonic negation is vital to capture the principles of commonsense reasoning [1]. Moreover, it is important to provide the ability to derive negative conclusions in the absence of positive information [28]. In addition, classical negation is important to allow reasoning in the presence of incomplete knowledge [7]. Therefore, different versions of annotated logic programming have been extended with non-monotonic negation including multi-valued annotated logic programming [42] and probabilistic annotated logic programming [28–30, 36]. Although good progress has been made in probabilistic annotated logic programming with both non-monotonic negation and classical negation [28–30, 36] and its application to real-world applications [31–34], less progress and attention has been given to fuzzy annotated logic programming with both non-monotonic negation and classical negation and its application to real-world applications.

One of the earliest work to develop semantics for logic programs with uncertainty in the implication-based approach under the fuzzy set theory has been introduced in [43]. This work was inspired by the need for reasoning in the presence of uncertainties in rule-based expert systems, instead of reasoning over only two values true and false as proposed by classical logic programming. In addition, the use of fuzzy set theory provides a mathematical formalism for reasoning under uncertainty in rule-based expert systems. Hence, a notion of fuzzy logic

programs has been proposed, which generalizes definite logic programs (logic programs without either classical or non-monotonic negation).

A general semantics for annotated logic programs (without negation) based on lattice theory has been introduced in [12]. This has been done by permitting annotation variables to take certainty values over arbitrary lattices of truth values, instead of using the special lattice of [11]. Programs in the proposed framework are called *generalized annotated programs*. A model theoretic semantics of generalized annotated programs has been defined in [12]. Due to its expressive power, it has been shown that fuzzy logic programs proposed in [43] is subsumed by the generalized annotated programs of [12].

A stable model semantics and well-founded semantics (based on alternating fixpoint semantics [8]) have also been presented in [42] for annotated logic programs with non-monotonic negation. An annotation in [42] is a tuple of truth values generated from a cross product of complete lattices of arbitrary truth values. If one complete lattice is used that is formed from the set of values in $[0, 1]$ and the partial order $\leq$, the annotated logic programs with non-monotonic negation in [42] reduce to annotated logic programs with non-monotonic negation whose underlying semantics is the fuzzy set theory. I.e., [42] reduces to generalized annotated programs with non-monotonic negation with the fuzzy set theory as the underlying formalism (forming the notion of normal fuzzy logic programs (NFLP) whose semantics is the stable fuzzy model semantics and the alternating fixed point well-founded fuzzy model semantics). In this view, [42] generalizes [12] with non-monotonic negation under the fuzzy set theory.

Furthermore, in [3], the semantics of [42] has been extended to allow classical negation as well as non-monotonic negation by proposing alternating fixpoint like semantics. However, neither answer set semantics nor declarative well-founded semantics has been given for annotated logic programs of [3].

In this paper we extend annotated logic programs under the fuzzy set theory [12, 42] to allow both classical negation and non-monotonic negation introducing the notion of *extended fuzzy logic programs (EFLP)* and define the *fuzzy answer set semantics* of EFLP. We show that fuzzy answer set semantics of EFLP is a natural extension to the classical answer set semantics of classical extended logic programs [7]. Moreover, we show that the fuzzy answer set semantics of EFLP is reduced to stable fuzzy model semantics of normal fuzzy logic programs (NFLP) of [42]. The importance of that is computational methods developed for NFLP can be applied to EFLP. Furthermore, we define fixpoint semantics for EFLP with and without non-monotonic negation, and study their relationship to the fuzzy answer set semantics of EFLP. We show that a fuzzy answer set of an extended fuzzy logic program is a minimal fixpoint.

2 Related Work

In [18, 19], a well-founded like semantics extension to implication-based fuzzy logic programming framework of [43] was introduced along with well-founded like semantics [8] extension to various probabilistic and non-probabilistic logic

programming frameworks with uncertainty [17]. The notion of non-monotonic negation in [28–30, 36, 42] is more natural and closer to the classical notion of non-monotonic negation, however the notion of non-monotonic negation in [18, 19] is closer to classical negation (since fuzzy value of $\neg A$ is equal to 1 minus the fuzzy value of A, where $\neg$ is the non-monotonic negation in the sense of [18, 19]). An alternating fixpoint semantics was introduced in [18] to describe the well-founded like semantics. In [19], a framework to approximate the well-founded semantics for [17], including the fuzzy logic programming framework of [43], was described.

Another extension to [43] that allows disjunctions in the head of rules and arbitrary connector functions was presented in [40]. A logic program in [40] is a set of rules of the form $g(B_1, \ldots, B_m) \leftarrow f(A_1, \ldots, A_n)$, where $B_1, \ldots, B_m, A_1, \ldots, A_n$ are atoms and f,g are any arbitrary monotone computable connector functions over a complete lattice of truth values. This form of rules are expressive enough to subsume any monotone multi-valued implication-based logic programming approach including the implication-based fuzzy logic programming without any form of negation. However, the multi-valued logic programming approach of [40], although allows disjunctions in the head of rules, it does not allow either classical or non-monotonic negation.

[22] proposed an extension to the implication-based fuzzy logic programs with both classical negation and non-monotonic negation. A fuzzy logic program in [22] is a set of weighted rules, where negative literals (a negated atom with classical negation) are allowed in the head and the body of rules, and a weight is a certainty value taken form a residuated lattice of truth values. In addition, a fuzzy answer set semantics was defined for the fuzzy logic programs of [22]. However, similar to [18, 19], non-monotonic negation in [22] is treated like a classical negation, where the negation function in the residuated lattice is used to evaluate a literal negated with the non-monotonic negation. In addition, it is not clear how negative literals themselves are evaluated in [22], since although negative literals are allowed in the rules, the fuzzy answer set semantics of [22] is computed by transforming a fuzzy logic program with negative literals and non-monotonic negation into a positive form with positive literals (atoms without classical negation) and non-monotonic negation.

In [10, 25] another implication-based fuzzy logic programming approach with only non-monotonic negation was presented, in which [10] extended [25] with any arbitrary monotone fuzzy connector functions. Unlike other approaches to fuzzy logic programming, [10, 25] considered satisfying fuzzy rules to a maximum extent rather than completely satisfying the rules. However, in [10, 25], similar to [18, 19, 22], non-monotonic negation is treated as a classical negation using a negator function.

An implication-based semantics, based on the possible world semantics, for disjunctive logic programs with non-monotonic negation has been presented in [21]. The semantics of [21] is based on multi-valued logic and a stable model semantics has been described. However, similar to [18, 19], non-monotonic negation is treated as the classical negation in [21].

An annotation based framework based on multi-valued logic [11] has been developed to provide a formal semantics to rule based systems with uncertainty, which uses certainty values as a representation of uncertainty. The purpose for the development of this approach is to overcome some of the drawbacks of implication based-approaches, such as fuzzy logic programs of [43], and make the framework suitable to larger application domains. This is achieved by allowing complex certainty functions to appear in the logic programs and by using more general propagation functions. Moreover, the semantics is extended to deal with programs with negation under the assumption that the considered programs are stratified. In [11], negative literals are allowed in both the heads and the bodies of the rules, in addition, model theoretic semantics has been given for annotated programs that are stratified, which have a unique model. A general annotated logic programming framework with non-monotonic was introduced in [23] that captures reasoning with any arbitrary annotated multi-valued logic programming approach with non-monotonic negation. Although general framework, [23] does not allow classical negation in its semantics.

A probabilistic annotated logic programming framework, called Hybrid Probabilistic Programs (HPP), that enables the user to explicitly encode the knowledge about the type of dependencies existing between the probabilistic events being described by the programs is presented in [2]. HPP generalizes the probabilistic annotated logic programming framework of [26] that is extended in [27]. The probabilistic annotated logic programming framework of [27] was extended in [28] to allow non-monotonic negation and a probabilistic stable model semantics is developed for the proposed language. The probabilistic stable model semantics of [28] is computationally expensive, since at every fixpoint iteration an exponential number of linear programs, each having an exponential number of variables, needs to be solved. A generalization of HPP of [2] was proposed in [4] by providing a more general semantical characterization in which HPP fits. However, [4] does not allow non-monotonic negation in defining its semantics. In addition, it relies on a complex translation process which is exponential in the size of HPP. A generalization and new semantics for HPP have been defined in [29, 30, 35–37]. The new semantics, intuitively, capture the probabilistic reasoning according to how likely are the various events to occur. It was shown that the new HPP framework subsumes Lakshmanan and Sadri's [15] probabilistic implication-based framework as well as it is a natural extension of classical logic programming. In addition, these probabilistic logic programming approaches [29, 30, 35–37] have proven intuitively applicable in representing and reasoning about a variety of fundamental probabilistic reasoning tasks including probabilistic planning [31], probabilistic planning with imperfect sensing actions [34], stochastic satisfiability [32], and reinforcement learning [33].

3 Fuzzy Sets

In this section we review the basic notions of fuzzy sets as presented in [44]. Let U be a set of objects. A fuzzy set, F, in U is defined by the grade membership

function $\mu_F : U \rightarrow [0,1]$, where for each element $x \in U$, μ_F assigns to x a value $\mu_F(x)$ in $[0,1]$. The support for F denotes the set of all objects x in U for which the grade membership of x in F is a non-zero value. Formally, $support(F) = \{x \in U \mid \mu_F(x) > 0\}$. The intersection (conjunction) of two fuzzy sets F and F' in U, denoted by $F \wedge_f F'$ is a fuzzy set G in U where the grade membership function of G is $\mu_G(x) = \min(\mu_F(x), \mu_{F'}(x))$ for all $x \in U$. However, the union (disjunction) of two fuzzy sets F and F' in U, denoted by $F \vee_f F'$ is a fuzzy set G in U where the grade membership function of G is $\mu_G(x) = \max(\mu_F(x), \mu_{F'}(x))$ for all $x \in U$. The complement (negation) of a fuzzy set F in U is a fuzzy set in U denoted by $\overline{F}$ where the grade membership function of $\overline{F}$ is $\mu_{\overline{F}}(x) = 1 - \mu_F(x)$ for all $x \in U$. A fuzzy set F in U is said to be contained in another fuzzy set G in U if and only if $\mu_F(x) \leq \mu_G(x)$ for all $x \in U$. Notice that we use the notations $\wedge_f$ and $\vee_f$ to denote fuzzy conjunction and fuzzy disjunction respectively to distinguish them from $\wedge$ and $\vee$ for propositional conjunction and disjunction respectively. Furthermore, other function characterizations for the fuzzy conjunction and fuzzy disjunction operators can be used. However, we will stick with the min and max function characterizations for the fuzzy conjunction and fuzzy disjunction as originally proposed in [44].

4 Extended Fuzzy Logic Programs

In this section, we present the syntax of extended fuzzy logic programs, which are annotated logic programs with both classical negation and non-monotonic negation whose underlying semantics is the fuzzy set theory.

Let L be an arbitrary first-order language with finitely many predicate symbols, constants, and infinitely many variables. The Herbrand base of L is denoted by B_L. A literal is either an atom a in B_L or the negation of a ($\neg a$), where $\neg$ is the classical negation and *not* is the non-monotonic negation or the negation as failure. We denote the set of all literals in L by Lit. More formally, $Lit = \{a | a \in B_L\} \cup \{\neg a | a \in B_L\}$. The grade membership are assigned to literals in L as values from $[0,1]$. Let $\alpha_1, \alpha_2 \in [0,1]$, the set $[0,1]$ and the relation $\leq$ form a complete lattice. In particular, the join ($\oplus$) operation is defined as $\alpha_1 \oplus \alpha_2 = \max(\alpha_1, \alpha_2)$ and the meet ($\otimes$) is defined as $\alpha_1 \otimes \alpha_2 = \min(\alpha_1, \alpha_2)$ w.r.t. $\leq$. An *annotation*, α, is either a constant in $[0,1]$, a variable (*annotation variable*) ranging over $[0,1]$, or $f(\alpha_1, \ldots, \alpha_n)$ (called *annotation function*) where f is a representation of a computable total function $f : ([0,1])^n \rightarrow [0,1]$ and $\alpha_1, \ldots, \alpha_n$ are annotations. A fuzzy literal is an expression of the form $l : \mu$, where l is a literal in Lit and μ is an annotation.

Definition 1 (Rules). *An extended fuzzy rule is an expression of the form*

$$l : \mu \leftarrow l_1 : \mu_1, \ldots, l_n : \mu_n, not\ l_{n+1} : \mu_{n+1}, \ldots, not\ l_m : \mu_{n+m}$$

where $l, l_i (1 \leq i \leq m+n)$ *are literals and* μ, μ_i $(1 \leq i \leq m+n)$ *are annotations.*

A fuzzy rule is an extended fuzzy rule such that $m = 0$*—i.e., there are no non-monotonic negation in the rule.*

The intuitive meaning of an extended fuzzy rule, in Definition 1, is that, if for each $l_i : \mu_i$ $(1 \leq i \leq n)$, it is *known* that the grade membership of l_i is at least μ_i and for each *not* $l_j : \mu_j$ $(n+1 \leq j \leq n+m)$, it is not *known* that the grade membership of l_j is at least μ_j, then the grade membership of l is μ.

Definition 2 (Programs). *An extend fuzzy logic program is a finite set of extended fuzzy rules. A fuzzy logic program is an extended fuzzy logic program where all the rules are fuzzy rules (extended fuzzy rules without non-monotonic negation).*

An extended fuzzy logic program is ground if no variables appear in any of its rules. The following is a typical extended fuzzy logic program.

Example 1. Consider the following fuzzy planning problem from [38] and adapted from [13]. An arm of a robot is grasping a block from a table, where the *pickup* action the robot performs has effects that are imprecisely defined. The arm is able to tightly hold a block (hb) with a grade membership 0.9 after executing the *pickup* action in the state in which the gripper is dry (gd), and the arm cannot tightly hold the block ($\neg hb$), after executing the *pickup* action in the same state, with grade membership value 0.1. When the *pickup* action is executed in the state while the gripper is not dry ($\neg\, gd$) causes the block to be tightly held (hb) with grade membership equal to 0.4 and tightly not held ($\neg\, hb$) with grade membership 0.6. We assume initially the grade membership of gripper dry (gd) is 0.8 and the grade membership of gripper is not dry ($\neg gd$) is 0.2. Hence, the distribution of the initial states of the world is given by the grade membership function $\mu_{\mathcal{S}}$ such that $\mu_{\mathcal{S}}(s_1) = 0.8$ and $\mu_{\mathcal{S}}(s_2) = 0.2$, where $s_1 = \{gd, \neg hb\}$ and $s_2 = \{\neg gd, \neg hb\}$, with the understanding that the grade membership of the other states of the world is 0. This fuzzy planning problem can be represented as an extended fuzzy logic program as follows which contains the following extended fuzzy rules.

$$\begin{array}{ll} action(pickup^1) : 1 \leftarrow & action(pickup^2) : 1 \leftarrow \\ action(pickup^3) : 1 \leftarrow & action(pickup^4) : 1 \leftarrow \end{array}$$

where *pickup* is an action. Action generation rules are represented by the following rules that generate action occurrences one at a time, where AC_i and AC_j are variables representing actions.

$$\begin{array}{l} occ(AC, T) : 1 \quad \leftarrow action(AC) : 1, not\ abocc(AC, T) : 1 \\ abocc(AC_i, T) : 1 \leftarrow action(AC_i) : 1, action(AC_j) : 1, occ(AC_j, T) : 1, AC_i \neq AC_j \end{array}$$

Properties of the world are described by the atoms gd (gripper dry) and hb (holding block). The set of possible initial states are encoded by the rules:

$$\begin{array}{l} \neg hb(0) : 1 \quad \leftarrow \\ gd(0) : 0.8 \quad \leftarrow not\ \neg gd(0) : 0.2 \\ \neg gd(0) : 0.2 \leftarrow not\ gd(0) : 0.8 \end{array}$$

Let V, V_1, and V_2 be annotation variables act as place holders, then the following rules encode that a literal and its negation cannot hold at the same time.

$$inconsistent : 1 \leftarrow not\ inconsistent : 1, gd(T) : V_1, \neg gd(T) : V_2$$
$$inconsistent : 1 \leftarrow not\ inconsistent : 1, hb(T) : V_1, \neg hb(T) : V_2$$

Constraint to prevents executing an action in a state in which its preconditions does not hold are encoded as:

$$inconsistent : 1 \leftarrow not\ inconsistent : 1, occ(pickup^1, T) : 1, \neg gd(T) : V$$
$$inconsistent : 1 \leftarrow not\ inconsistent : 1, occ(pickup^2, T) : 1, \neg gd(T) : V$$
$$inconsistent : 1 \leftarrow not\ inconsistent : 1, occ(pickup^3, T) : 1, gd(T) : V$$
$$inconsistent : 1 \leftarrow not\ inconsistent : 1, occ(pickup^4, T) : 1, gd(T) : V$$

Frame axioms are encoded as the rules

$$gd(T+1) : V_1 \leftarrow gd(T) : V_1, not\ \neg gd(T+1) : V_2$$
$$\neg gd(T+1) : V_1 \leftarrow \neg gd(T) : V_1, not\ gd(T+1) : V_2$$
$$hb(T+1) : V_1 \leftarrow hb(T) : V_1, not\ \neg hb(T+1) : V_2$$
$$\neg hb(T+1) : V_1 \leftarrow \neg hb(T) : V_1, not\ hb(T+1) : V_2$$

Effects of the *pickup* action are encoded by the rules

$$hb(T+1) : 0.9 \leftarrow occ(pickup^1, T) : 1, gd(T) : V$$
$$\neg hb(T+1) : 0.1 \leftarrow occ(pickup^2, T) : 1, gd(T) : V$$
$$hb(T+1) : 0.4 \leftarrow occ(pickup^3, T) : 1, \neg gd(T) : V$$
$$\neg hb(T+1) : 0.6 \leftarrow occ(pickup^4, T) : 1, \neg gd(T) : V$$

4.1 Satisfaction and Models

In this subsection, we define the declarative semantics and the fixpoint semantics of extended fuzzy logic programs and fuzzy logic programs, along with their notions of interpretations, models, and satisfaction.

Definition 3. *A fuzzy interpretation, I, is a fuzzy set in the set of all literals Lit where the grade membership function of I is a mapping $\mu_I : Lit \rightarrow [0, 1]$. We say that a fuzzy interpretation I is a partial fuzzy interpretation iff the grade membership function of I is a partial mapping from Lit to $[0, 1]$.*

For simplicity, we refer to the fuzzy interpretation as the mapping $I : Lit \rightarrow [0, 1]$, where the grade membership of a literal l in the fuzzy interpretation I is $I(l)$. If the grade membership of a literal, l, in the fuzzy interpretation, I, is $I(l)$, then the grade membership of the negation of l ($\neg l$) in I is $I(\neg l) = 1 - I(l)$. As a literal and its negation are allowed in a fuzzy interpretation, more conditions are required to ensure the consistency of fuzzy interpretations. This can be characterized by the following definition.

Definition 4. *A total or partial fuzzy interpretation, I, is inconsistent if there exists $l, \neg l \in Lit$ $(l, \neg l \in dom(I))$ such that $I(\neg l) \neq 1 - I(l)$.*

Definition 5. *Let S be a subset of literals from Lit. We say that S is a set of consistent literals if there is no pair of complementary literals l and $\neg l$ belonging to S.*

Definition 6. *A consistent fuzzy interpretation I is either not inconsistent or maps a consistent set of literals S to $[0, 1]$.*

A fuzzy interpretation is consistent if for every $l, \neg l \in dom(I)$, $I(\neg l) = 1 - I(l)$ or it maps a consistent set of literals into $[0, 1]$. Let I_1 and I_2 be two (partial or total) fuzzy interpretations in Lit and $dom(I_1), dom(I_2)$ be the domains of I_1 and I_2 respectively. For a fuzzy interpretation I, $dom(I) \subseteq Lit$ if I is total fuzzy interpretation and $dom(I) \subsetneqq Lit$ if I is partial fuzzy interpretation. Then, we say that $I_1 \leq I_2$ iff $dom(I_1) \subseteq dom(I_2)$ and $\forall l \in dom(I_1)$ we have $I_1(l) \leq I_2(l)$. The set of all fuzzy interpretations in Lit (denoted by $\mathcal{F}$) and the relation $\leq$ form a complete lattice. The meet $\otimes$ and the join $\oplus$ operations on $\mathcal{F}$ are defined as follows.

Definition 7. *Let I_1 and I_2 be two partial fuzzy interpretations. The meet $\otimes$ and join $\oplus$ operation corresponding to the partial order $\leq$ are defined respectively as:*

- $(I_1 \otimes I_2)(l) = I_1(l) \otimes I_2(l) = \min(I_1(l), I_2(l))$ *for all l defined in both I_1 and I_2, otherwise, undefined.*
- $(I_1 \oplus I_2)(l)$ *is equal to*
  - $I_1(l) \oplus I_2(l) = \max(I_1(l), I_2(l))$ *for all l defined in both I_1 and I_2.*
  - $(I_1 \oplus I_2)(l) = I_1(l)$ *for all l defined in I_1 but not defined in I_2*
  - $(I_1 \oplus I_2)(l) = I_2(l)$ *for all l defined in I_2 but not defined in I_1*
  - *otherwise, undefined.*

Definition 8 (Fuzzy Satisfaction). *Let P be a ground extended fuzzy logic program, I be a fuzzy interpretation, and r be*

$$l : \mu \leftarrow l_1 : \mu_1, \ldots, l_n : \mu_n, not\ l'_1 : \beta_1, \ldots, not\ l'_m : \beta_m \in P.$$

Then

- *I satisfies $l_i : \mu_i$ (denoted by $I \models l_i : \mu_i$) iff $\mu_i \leq I(l_i)$ and $l_i \in dom(I)$.*
- *I satisfies not $l'_j : \beta_j$ (denoted by $I \models not\ l'_j : \beta_j$) iff $\beta_j \nleq I(l'_j)$ and $l'_j \in dom(I)$ or $l'_j \notin dom(I)$.*
- *I satisfies $Body \equiv l_1 : \mu_1, \ldots, l_n : \mu_n, not\ l'_1 : \beta_1, \ldots, not\ l'_m : \beta_m$ (denoted by $I \models Body$) iff $\forall (1 \leq i \leq n), I \models l_i : \mu_i$ and $\forall (1 \leq j \leq m), I \models not\ l'_j : \beta_j$.*
- *I satisfies $l : \mu \leftarrow Body$ iff $I \models l : \mu$ or I does not satisfy $Body$.*
- *I satisfies P iff I satisfies every extended fuzzy rule in P and for every literal $l \in dom(I)$, we have $\max\{\mu | l : \mu \leftarrow Body \in P$ and $I \models Body\} \leq I(l)$.*

Definition 9 (Models). *Let P be an extended fuzzy logic program. A fuzzy model of P is a total or partial fuzzy interpretation of P that satisfies P.*

A fuzzy model I is said to be a minimal fuzzy model of an extended fuzzy logic program, P, if there is no fuzzy model I' of P such that $I' < I$ w.r.t. the order $\leq$. Extended fuzzy logic programs without non-monotonic negation, which are called *fuzzy logic programs*, has exactly one minimal fuzzy model. This is characterized by the following results.

Proposition 1. *Let P be a ground fuzzy logic program and I_1, I_2 be fuzzy models of P. Then $I_1 \otimes I_2$ is also a fuzzy model of P.*

Proposition 2. *Let P be a fuzzy logic program. Then, $I_P = \otimes \{I | I$ is a fuzzy model of $P\}$ is the least fuzzy model of P.*

Example 2. Consider the following fuzzy logic program P:

$$\begin{array}{ll} r : 0.9 & \leftarrow p : 0.1, q : 0.8 \\ \neg r : 0.2 & \leftarrow p : 0.1, \neg q : 0.05 \\ s : 0.18 & \leftarrow r : 0.35 \\ \neg s : 0.55 & \leftarrow p : 0.15, \neg q : 0.02, \neg r : 0.1 \\ \neg q : 0.3 & \leftarrow \\ p : 0.2 & \leftarrow \end{array}$$

It is easy to verify that the least fuzzy model of P is I, where $I(p) = 0.2, I(\neg q) = 0.3, I(\neg r) = 0.2, I(\neg s) = 0.55$.

A fuzzy logic program (extended fuzzy logic program without non-monotonic negation), P, is inconsistent if the least fuzzy model of P is inconsistent. If P is inconsistent, then we say that the mapping $Lit \rightarrow \{1\}$, is the least fuzzy model of P. This implies that every literal with the grade membership 1 follows from P. We adopt this view from the semantics of classical logic programs with classical negation [7].

Associated with each fuzzy logic program P, is an operator, T_P, called the *fixpoint operator*, which maps fuzzy interpretations to fuzzy interpretations.

Definition 10. *Let P be a ground fuzzy logic program, $\mathcal{I}$ be the set of all fuzzy interpretations of P, and I be a fuzzy interpretation in $\mathcal{I}$. The fixpoint operator T_P is a mapping $T_P : \mathcal{I} \rightarrow \mathcal{I}$ which is defined as follows. For each $l \in Lit$,*

1. *$T_P(I)(l) = \max\{\!\{\mu | l : \mu \leftarrow l_1 : \mu_1, \ldots, l_n : \mu_n \in P$ and $I \models (l_1 : \mu_1, \ldots, l_n : \mu_n)\}\!\}$.*
2. *Otherwise undefined, i.e., there is no fuzzy rule in P such that l appears in its head and its body is satisfied by I.*

If there are no fuzzy rules in P whose heads contain l such that their bodies are satisfied by a fuzzy interpretation I, then no grade membership value is assigned to l. This implies that the grade membership of l is *unknown* with respect to I. The following definition provides the construction of the least fuzzy model of a fuzzy logic program as a repeated iteration of the fixpoint operator T_P.

Definition 11. *Let P be a ground fuzzy logic program. Then*

- $T_P \uparrow 0 = \emptyset$ *where* $\emptyset$ *is the empty set.*
- $T_P \uparrow \alpha = T_P(T_P \uparrow (\alpha - 1))$ *where* α *is a successor ordinal.*
- $T_P \uparrow \lambda = \oplus\{T_P \uparrow \alpha | \alpha < \lambda\}$ *where* λ *is a limit ordinal.*

Lemma 1. *The T_P operator is monotonic.*

The properties of the T_P operator guarantee the existence of a least fixpoint.

Proposition 3. *Let P be a fuzzy logic program and I be a fuzzy interpretation. Then I is a fuzzy model of P iff $T_P(I) \leq I$.*

Theorem 1. *Let P be a fuzzy logic program and I_P be the least fuzzy model of P. Then, $I_P = lfp(T_P)$.*

Example 3. Consider again the fuzzy logic program, P, presented in Example 2. It is easy to verify that $lfp(T_P)$ assigns 0.2 to p, 0.3 to $\neg q$, 0.2 to $\neg r$, and 0.55 to $\neg s$.

Now we show that the model-theoretic and fixpoint semantics of fuzzy logic programs generalize their classical counterparts for classical logic programs with classical negation [7]. A classical logic program P' can be represented as a fuzzy logic program P where each rule

$$l \leftarrow l_1, \ldots, l_n \in P'$$

can be encoded, in P, as a fuzzy rule of the form

$$l : 1 \leftarrow l_1 : 1, \ldots, l_n : 1$$

where $l, l_1, \ldots, l_n$ are literals and 1 represents the truth value *true*.

Proposition 4. *Let P' be a classical logic program and P be the fuzzy logic program encoding of P'. Then, I is the least fuzzy model of P iff I' is the classical least model of P' where $I(l) = 1$ iff $l \in I'$.*

5 Fuzzy Answer Set Semantics

In this section we introduce the notion of *fuzzy answer sets* for extended fuzzy logic programs, which generalizes the notion of answer sets of classical extended logic programs [7]. The fuzzy answer set semantics of extended fuzzy logic programs is defined by first guessing a fuzzy model I for an extended fuzzy logic program P, then the notion of the fuzzy reduct of P with respect to I is defined, which is a fuzzy logic program (an extended fuzzy logic program without non-monotonic negation). Then, the fuzzy model I of P is verified as a fuzzy answer set of P if I is recognized as the least fuzzy model of the fuzzy reduct of P w.r.t. I. Every fuzzy logic program has a unique least fuzzy model.

Definition 12 (Fuzzy Reduct). *Let P be a ground extended fuzzy logic program and I be a fuzzy interpretation. The fuzzy reduct P^I of P w.r.t. I is:*

$$P^I = \left\{ l : \mu \leftarrow l_1 : \mu_1, \ldots, l_n : \mu_n \;\middle|\; \begin{array}{l} l : \mu \leftarrow l_1 : \mu_1, \ldots, l_n : \mu_n, \\ not\ l'_1 : \beta_1, \ldots, not\ l'_m : \beta_m \in P\ and \\ \forall (1 \leq j \leq m),\ \beta_j \not\leq I(l'_j)\ or\ l'_j \notin dom(I). \end{array} \right\}$$

The fuzzy reduct, P^I, of an extended fuzzy logic program, P, w.r.t. the fuzzy interpretation, I, is a fuzzy logic program. For any *not* $l'_j : \beta_j$ in the body of an extended fuzzy rule $r \in P$ such that $\beta_j \not\leq I(l'_j)$ implies that it is not known that the grade membership of l'_j is at least β_j given the available knowledge, and *not* $l'_j : \beta_j$ is removed from the body of r. Moreover, if $l'_j \notin dom(I)$, i.e., l'_j is undefined in I, then it is entirely *not known* that the grade membership of l'_j is at least β_j. In this case, *not* $l'_j : \beta_j$ is also removed from the body of r.

Definition 13. *A fuzzy interpretation I of an extended fuzzy logic program P is a fuzzy answer set of P if I is the least fuzzy model of fuzzy reduct P^I of P w.r.t. I.*

Intuitively, the fuzzy answer sets of an extended fuzzy logic program are the agent's possible sets of beliefs with associated beliefs degrees. It is worth noting that extended fuzzy logic programs without classical negation, called normal fuzzy logic programs [42], which are extended fuzzy logic programs without negative literals neither in head nor in the body of their rules, have fuzzy answer sets whose domains consisting of only atoms. Moreover, the definition of fuzzy answer sets for normal fuzzy logic programs coincides with the definition of stable fuzzy models defined in [42] for the same class of programs. This implies that the fuzzy answer sets of a normal fuzzy logic program are equivalent to its stable fuzzy models. This means that the application of fuzzy answer set semantics to normal fuzzy logic programs is reduced to the stable fuzzy model semantics for normal fuzzy logic programs.

However, there are two main differences between the two semantics. A stable fuzzy model for a normal fuzzy logic program is a total fuzzy model, however, the fuzzy answer set of a normal fuzzy logic program could be a partial fuzzy model. Furthermore, for each atom, a, with grade membership value 0 in a stable fuzzy model I of a normal fuzzy logic program P, we have a undefined in the fuzzy answer set I' of P that is equivalent to I. I.e., a is false in the stable fuzzy model I of P, but a is undefined and hence unknown in the fuzzy answer set I' equivalent to I.

Proposition 5. *Let P be a normal fuzzy logic program. Then I is a fuzzy answer set for P iff I' is a stable fuzzy model of P, where $I(a) = I'(a)$ for $I'(a) \neq 0$ and $I(a)$ is undefined for $I'(a) = 0$.*

Proposition 5 shows that there is a simple reduction from extended fuzzy logic programs with fuzzy answer set semantics to normal fuzzy logic programs with stable fuzzy model semantics [42]. The importance of that is, under the consistency condition, computational methods developed for normal fuzzy logic programs can be applied to extended fuzzy logic programs.

Example 4. Considering only one time step, i.e. $T = 1$, the extended fuzzy logic program of Example 1 has four fuzzy answer sets as follows. For the ease of presentation, we present these fuzzy answer sets as sets of annotated literals as:

$$I_1 = \{gd(0) : 0.8, \neg hb(0) : 1, occ(pickup^1, 0) : 1, gd(1) : 0.8, hb(1) : 0.9, \ldots\}$$
$$I_2 = \{gd(0) : 0.8, \neg hb(0) : 1, occ(pickup^2, 0) : 1, gd(1) : 0.9, \neg hb(1) : 0.1, \ldots\}$$
$$I_3 = \{\neg gd(0) : 0.2, \neg hb(0) : 1, occ(pickup^3, 0) : 1, \neg gd(1) : 0.2, hb(1) : 0.4, \ldots\}$$
$$I_4 = \{\neg gd(0) : 0.2, \neg hb(0) : 1, occ(pickup^4, 0) : 1, \neg gd(1) : 0.2, \neg hb(1) : 0.6, \ldots\}$$

Theorem 2. *Every fuzzy logic program P has a unique fuzzy answer set I iff I is the least fuzzy model of P.*

In the rest of this section we define the *immediate consequence operator* of extended fuzzy logic programs and study its relationship to the fuzzy answer sets semantics.

Definition 14. *Let P be a ground extended fuzzy logic program and $I \in \mathcal{I}$ be a fuzzy interpretation. The* immediate consequence operator *T'_P is a mapping $T'_P : \mathcal{I} \to \mathcal{I}$ which is defined as:*

1. $T'_P(I)(l) = \max\left\{\left\{ \mu \;\middle|\; \begin{array}{c} l : \mu \leftarrow l_1 : \mu_1, \ldots, l_n : \mu_n, not\ l'_1 : \beta_1, \ldots, not\ l'_m : \beta_m \in P \text{ and} \\ \forall (1 \le i \le n), \mu_i \le I(l_i) \text{ and } \forall (1 \le j \le m), \beta_j \not\le I(l'_j) \text{ or } l'_j \notin dom(I) \end{array} \right\}\right\}$
2. *Undefined otherwise, i.e., there is no extended fuzzy rule in P such that l appears in its head and its body is satisfied by I.*

Obviously, T'_P extends T_P to handle fuzzy rules with non-monotonic negation, hence, $T'_P = T_P$ for any fuzzy logic program P.

Theorem 3. *Let P be an extended fuzzy logic program without non-monotonic negation. Then $T'_P = T_P$.*

The following example shows that T'_P is not monotonic w.r.t. the order $\leq$.

Proposition 6. *T'_P is not monotonic w.r.t. the order $\leq$.*

Example 5. Consider the extended fuzzy logic program P: $\neg p : 0.25 \leftarrow not\ q : 0.7$. Let $I_1 = \emptyset$ be a fuzzy interpretation for P. Furthermore, let I_2 be a fuzzy interpretation for P that assigns 0.8 to q. Hence, $I_1 \leq I_2$. But $T'_P(I_1)$ assigns 0.25 to $\neg p$ and $T'_P(I_2) = \emptyset$. Thus, $T'_P(I_1) \not\leq T'_P(I_2)$

The following results show the relationship between the fuzzy answer set semantics and the T'_P operator of extended fuzzy logic programs.

Lemma 2. *Let P be an extended fuzzy logic program and I be a fuzzy answer set of P. Then $T'_P(I) = I$, i.e., I is a fixpoint of T'_P.*

Theorem 4. *Let P be an extended fuzzy logic program and I be a fuzzy answer set of P. Then I is a minimal fixpoint of T'_P.*

It is not the case that every minimal fixpoint of T'_P is a fuzzy answer set of P. Consider the following extended fuzzy logic program P.

Example 6. Let P be an extended fuzzy logic program that contains

$$p : 0.2 \leftarrow not\ p : 0.2$$
$$p : 0.2 \leftarrow \neg q : 1$$

We can see that the fuzzy interpretation I, where $I(p) = 0.2$ and $I(\neg q) = 1$ is a minimal fixpoint of T'_P. However, the fuzzy reduct, P^I, of P w.r.t. I is $p : 0.2 \leftarrow b : 1$, where $lfp(T_{P^I}) = \emptyset$. Thus, I is not a fuzzy answer set for P.

Now, we show that the fuzzy answer set semantics of extended fuzzy logic programs extends the classical answer set semantics of classical extended logic programs [7]. A classical extended logic program P' can be represented as an extended fuzzy logic program P where each classical extended rule

$$l \leftarrow l_1, \ldots, l_m, not\ l_{m+1}, \ldots, not\ l_n$$

in P' can be encoded as an extended fuzzy rule of the form

$$l : 1 \leftarrow l_1 : 1, \ldots, l_m : 1, not\ l_{m+1} : 1, \ldots, not\ l_n : 1$$

in P, where $l, l_1, \ldots, l_m, l_{m+1}, \ldots, l_n$ are literals and 1 represents the truth value *true*. The following result shows that classical extended logic programs [7] are subsumed by extended fuzzy logic programs.

Proposition 7. *Let P' be a classical extended logic program. Then I' is a classical answer set of P' iff I is a fuzzy answer of an extended fuzzy logic program P equivalent to P', where $I(l) = 1$ iff $l \in I'$ and $I(l)$ is undefined iff $l \notin I'$.*

6 Conclusions

We presented the notion of extended fuzzy logic programs that extend annotated logic programs under the fuzzy set theory [12, 42] to allow both classical negation and non-monotonic negation. We defined the *fuzzy answer set semantics* of extended fuzzy logic programs. We showed that the fuzzy answer set semantics of extended fuzzy logic programs generalize the classical answer set semantics of classical extended logic programs [7]. Furthermore, we showed that stable fuzzy model semantics of normal fuzzy logic programs [42] is subsumed by the fuzzy answer set semantics. The importance of that is computational methods developed for normal fuzzy logic programs can be applied to extended fuzzy logic programs. Moreover, we defined fixpoint semantics for extended fuzzy logic programs with and without non-monotonic negation, and studied their relationship to the fuzzy answer set semantics. We showed that a fuzzy answer set of an extended fuzzy logic program is a minimal fixpoint of the fixpoint operator of

the extended fuzzy logic program. We showed that actions with fuzzy effects can be intuitively represented and reasoned about by extended fuzzy logic programs with fuzzy answer set semantics.

References

1. Apt, K.R., Bol, R.N.: Logic programming and negation:a survey. Journal of logic programming 19(20), 9–71 (1994)
2. Dekhtyar, A., Subrahmanian, V.S.: Hybrid probabilistic program. Journal of Logic Programming 43(3), 187–250 (2000)
3. Damasio, C.V., et al.: Coherent well-founded annotated logic programs. In: LP-NMR. Springer, Heidelberg (1999)
4. Damasio, C.V., Moniz Pereira, L.: Hybrid probabilistic logic programs as residuated logic programs. In: Ojeda-Aciego, M., de Guzmán, I.P., Brewka, G. (eds.) JELIA 2000. LNCS (LNAI), vol. 1919, pp. 57–72. Springer, Heidelberg (2000)
5. Dubois, D., et al.: Towards possibilistic logic programming. In: ICLP. MIT Press, Cambridge (1991)
6. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In: ICSLP. MIT Press, Cambridge (1988)
7. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New Generation Computing 9(3-4), 363–385 (1991)
8. Van Gelder, A.: The alternating fixpoint of logic programs with negation. Journal of Computer and System Sciences 47(1), 185–221 (1993)
9. Van Gelder, A., Ross, K.A., Schlipf, J.S.: The well-founded semantics for general logic programs. Journal of ACM 38(3), 620–650 (1991)
10. Janssen, J., Schockaert, S., Vermeir, D., De Cock, M.: General fuzzy answer set programs. In: International Workshop on Fuzzy Logic and Applications (2009)
11. Kifer, M., Li, A.: On the semantics of rule-based expert systems with uncertainty. In: Intl. Conf. on Database Theory. Springer, Heidelberg (1988)
12. Kifer, M., Subrahmanian, V.S.: Theory of generalized annotated logic programming and its applications. Journal of Logic Programming 12, 335–367 (1992)
13. Kushmerick, N., Hanks, S., Weld, D.: An algorithm for probabilistic planning. Artificial Intelligence 76(1-2), 239–286 (1995)
14. Lakshmanan, V.S.L., Sadri, F.: Modeling uncertainty in deductive databases. In: Conf. on Database Expert Systems and Applications. Springer, Heidelberg (1994)
15. Lakshmanan, V.S.L., Sadri, F.: Probabilistic deductive databases. In: Intl. Logic Programming Symposium. MIT Press, Cambridge (1994)
16. Lakshmanan, V.S.L., Sadri, F.: Uncertain deductive databases: a hybrid approach. Information Systems 22(8), 483–508 (1997)
17. Lakshmanan, V.S.L., Shiri, N.: A Parametric approach to deductive databases with uncertainty. IEEE TKDE 13(4), 554–570 (2001)
18. Loyer, Y., Straccia, U.: The well-founded semantics in normal logic programs with uncertainty. In: FLOPS. Springer, Heidelberg (2002)
19. Loyer, Y., Straccia, U.: The approximate well-founded semantics for logic programs with uncertainty. In: 28th International Symposium on Mathematical Foundations of Computer Science (2003)
20. Lukasiewicz, T.: Fuzzy description logic programs under the answer set semantics for the semantic Web. Fundamenta Informaticae 82(3), 289–310 (2008)

21. Lukasiewicz, T.: Many-valued disjunctive logic programs with probabilistic semantics. In: LPNMR (1999)
22. Madrid, N., Ojeda-Aciego, M.: Towards a fuzzy answer set semantics for residuated logic programs. In: Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (2008)
23. Nerode, A., Remmel, J., Subrahmanian, V.S.: Annotated nonmonotone rule systems. Theoretical Computer Science 171(1-2), 77–109 (1997)
24. Niemela, I., Simons, P.: Efficient implementation of the well-founded and stable model semantics. In: Joint International Conference and Symposium on Logic Programming, pp. 289–303 (1996)
25. Nieuwenborgh, D., Cock, M., Vermeir, D.: An introduction to fuzzy answer set programming. Annals of Mathematics and Artificial Intelligence 50(3-4), 363–388 (2007)
26. Ng, R.T., Subrahmanian, V.S.: Probabilistic logic programming. Information & Computation 101(2) (1992)
27. Ng, R.T., Subrahmanian, V.S.: A semantical framework for supporting subjective and conditional probabilities in deductive databases. ARJ 10(2) (1993)
28. Ng, R.T., Subrahmanian, V.S.: Stable semantics for probabilistic deductive databases. Information & Computation 110(1) (1994)
29. Saad, E.: Incomplete knowlege in hybrid probabilistic logic programs. In: 10th European Conference on Logics in Artificial Intelligence (2006)
30. Saad, E.: A logical approach to qualitative and quantitative reasoning. In: 9th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (2007)
31. Saad, E.: Probabilistic planning in hybrid probabilistic logic programs. In: 1st International Conference on Scalable Uncertainty Management (2007)
32. Saad, E.: On the relationship between hybrid probabilistic logic programs and stochastic satisfiability. In: 2nd 1st International Conference on Scalable Uncertainty Management (2008)
33. Saad, E.: A logical framework to reinforcement learning using hybrid probabilistic logic programs. In: 2nd 1st International Conference on Scalable Uncertainty Management (2008)
34. Saad, E.: Probabilistic planning with imperfect sensing actions using hybrid probabilistic logic programs. In: Proceedings of the Ninth International Workshop on Computational Logic in Multi-Agent Systems (2008)
35. Saad, E., Pontelli, E.: Towards a more practical hybrid probabilistic logic programming framework. In: Practical Aspects of Declarative Languages (2005)
36. Saad, E., Pontelli, E.: Hybrid probabilistic logic programs with non-monotonic negation. In: International Conference of Logic Programming. Springer, Heidelberg (2005)
37. Saad, E., Pontelli, E.: A new approach to hybrid probabilistic logic programs. Annals of Mathematics and Artificial Intelligence Journal 48(3-4), 187–243 (2006)
38. Saad, E., Elmorsy, S., Gabr, M., Hassan, Y.: Reasoning about actions in fuzzy environment. In: The World Congress of the International Fuzzy Systems Association/European society for Fuzzy Logic and Technology, IFSA/EUSFLAT-09 (2009)
39. Shapiro, E.: Logic Programs with Uncertainties: A Tool for implementing expert systems. In: Proc. of IJCAI, pp. 529–532 (1983)

40. Straccia, U., Ojeda-Aciego, M., Damasio, C.V.: On fixed-points of multivalued functions on complete lattices and their application to generalized logic programs. SIAM Journal on Computing 38(5), 1881–1911 (2009)
41. Subrahmanian, V.S.: On the semantics of quantitative logic programs. In: Symp. on Logic Programming, pp. 173–182. IEEE Computer Society, Los Alamitos (1987)
42. Subrahmanian, V.S.: Amalgamating knowledge bases. ACM TDS 19(2), 291–331 (1994)
43. van Emden, M.H.: Quantitative deduction and Its fixpoint theory. Journal of Logic Programming 4(1), 37–53 (1986)
44. Zadeh, L.: Fuzzy Sets. Information and Control 8(3), 338–353 (1965)
45. Zadeh, L.: Outline of a new approach to the analysis of complex systems and decision processes. IEEE Trans. on Systems, Man, and Cybernetics SMC-3, 28–44 (1973)

Finite Satisfiability in Infinite-Valued Łukasiewicz Logic

Steven Schockaert[1], Jeroen Janssen[2], Dirk Vermeir[2], and Martine De Cock[1,3]

[1] Dept. of Applied Mathematics and Computer Science, Ghent University, Belgium
{steven.schockaert,martine.decock}@ugent.be
[2] Dept. of Computer Science, Vrije Universiteit Brussel, Belgium
{jeroen.janssen,dvermeir}@vub.ac.be
[3] Institute of Technology, University of Washington, Tacoma, WA, USA
mdecock@u.washington.edu

Abstract. Although it is well-known that every satisfiable formula in Łukasiewicz' infinite-valued logic $\mathcal{L}_\infty$ can be satisfied in some finite-valued logic, practical methods for finding an appropriate number of truth degrees do currently not exist. As a first step towards efficient reasoning in $\mathcal{L}_\infty$, we propose a method to find a tight upper bound on this number which, in practice, often significantly improves the worst-case upper bound of Aguzzoli et al.

1 Introduction

The boolean satisfiability problem SAT plays a central role in many areas of computer science at large, and artificial intelligence in particular. Consequently, substantial research efforts have been devoted at finding efficient methods, both complete and heuristic, for solving SAT [22]. The success of this line of research has led to efficient solvers for many of the AI formalisms that are polynomially reducible to SAT, including constraint satisfaction [20], planning [6], answer set programming [12], and temporal reasoning [17].

When moving from classical propositional logic to (propositional) multi-valued and fuzzy logics, the central importance of the satisfiability problem seems preserved. Indeed, as exemplified by existing work on fuzzy description logics [19], fuzzy answer set programming [10], or fuzzy spatial reasoning [18], there is a tendency to tackle fuzzy reasoning tasks by reducing them to a fuzzy version of SAT (even if the connection with SAT is not always made explicit). Essentially, a given problem is then translated into fuzzy clauses of the following form:

$$\langle 0.4 \rightarrow (x \otimes y) \geq 0.7 \rangle \vee \langle x \vee z \leq 0.7 \rangle \tag{1}$$

When all connectives correspond to the Łukasiewicz operators, satisfiability checking for fuzzy clauses can be reduced to mixed integer programming (MIP), which has nice theoretical properties; other types of mathematical programming can be used for other fuzzy logic connectives [4]. Unfortunately, the limited scalability of MIP and mathematical programming in general, quickly becomes

L. Godo and A. Pugliese (Eds.): SUM 2009, LNAI 5785, pp. 240–254, 2009.

problematic when we are faced with non-trivial fuzzy SAT instances. In practice, it is therefore more common to assume only a finite number of truth degrees, taken from a set $\{0, \frac{1}{k}, \frac{2}{k}, \ldots, 1\}$ with $k \in \mathbb{N} \setminus \{0\}$, such that other techniques than MIP can be employed. In the finite case, the fuzzy SAT problem can, among others, be solved by reducing it to boolean satisfiability, by treating it as a constraint satisfaction problem, or by using dedicated solvers. Moreover, by keeping the number of truth degrees sufficiently small (e.g. $k = 10$), problem instances of a reasonable size can thus be addressed.

The choice of an appropriate number of truth degrees, however, is more important than is generally acknowledged. In particular, let $\oplus$, $\rightarrow$ and $\neg$ be the disjunction, implication and negation from Łukasiewicz logic (see Section 3); for each l in $\mathbb{N} \setminus \{0\}$, we can construct the following two fuzzy clauses:

$$\langle \underbrace{b \oplus \cdots \oplus b}_{l-1} \rightarrow \neg b \geq 1 \rangle \qquad \langle \neg b \rightarrow \underbrace{b \oplus \cdots \oplus b}_{l-1} \geq 1 \rangle \tag{2}$$

It is not hard to prove that these clauses can only be jointly satisfied if b is interpreted as $\frac{1}{l}$. This means in particular that the number $k+1$ of truth degrees considered must be such that k is a multiple of l. Thus, the practice of choosing an (arbitrary) finite number of truth degrees should be considered as a sound, but incomplete approach to satisfiability checking, in the sense that when a set of fuzzy clauses is found to be unsatisfiable, this might be inherent to the fuzzy clauses, or it might be an artifact of the particular number of truth degrees that was chosen. On the other hand, it was shown in [14] that for every set of formulas Θ that are satisfiable in infinite-valued Łukasiewicz logic ($\mathcal{L}_\infty$), there exists a finite number k such that Θ is satisfiable in $k+1$ valued Łukasiewicz logic ($\mathcal{L}_k$). Unfortunately, in practice, finding this particular value of k seems as hard as solving the satisfiability problem itself. As an alternative, we show in this paper how a reasonable upper bound on the value of k can be obtained in polynomial time.

2 Related Work

Theoretical work on the logic $\mathcal{L}_\infty$ was essentially initiated by McNaughton [13], who proved a fundamental representation theorem, relating formulas in $\mathcal{L}_\infty$ to piecewise linear functions with integer coefficients. Using this theorem, the NP-completeness of satisfiability checking in $\mathcal{L}_\infty$ was established by Mundici in [14]. Strengthening a related result from Mundici, in [2] it is shown that the validity of a formula ϕ in $\mathcal{L}_\infty$ can be decided by checking its validity in $\mathcal{L}_k$, for $k = 2^{\sharp\phi}$ with $\sharp\phi$ the number of variable occurrences in ϕ[1]. It is furthermore shown in [2] that validity can also be checked by checking validity in $\mathcal{L}_k$ for *every* strictly positive integer k not larger than $\left(\frac{\sharp\phi}{n}\right)^n$, where n is the number of different variables.

[1] Some care should be taken when counting variable occurrences, since the number of occurrences may be different in two semantically equivalent formulas.

Furthermore, it has been shown in [1] that this latter upper bound is (asymptotically) optimal. Concerning satisfiability, a difference should be made between strong and weak satisfiability (see Section 3.1). It can easily be seen that bounds for deciding weak satisfiability follow immediately from bounds for deciding validity. For strong satisfiability, a similar result follows from [2] (Theorem 14), i.e. the satisfiability of ϕ in $\mathcal{L}_\infty$ can be decided by checking its satisfiability in $\mathcal{L}_k$ for every $k \leq \left(\frac{\sharp\phi}{n}\right)^n$. In contrast to the case for weak satisfiability, however, it is not sufficient to check strong satisfiability for $k = 2^{\sharp\phi}$; note that this was already illustrated by (2) which cannot be satisfied for any number of truth degrees k of the form $k = 2^i$ when e.g. $l = 3$. Despite the theoretical importance of these results, their immediate practical value remains limited, due to the exponential nature of the provided upper bounds. In this paper, we show how the bound for strong satisfiability can nonetheless be considerably tightened in many practical situations by looking at more features than only the number of variables and variable occurrences. Interestingly, results from [2] have been generalized to other t-norm based logics in [3].

Regarding computational approaches to (strong) satisfiability checking in infinite-valued Łukasiewicz logic, it is easy to see that for each set of formulas Θ there exist a (possibly exponential) number of systems of linear inequalities, such that Θ is satisfiable iff at least one of these systems of inequalities has a solution. Hähnle [8] showed how the need for more than one system of linear inequalities can be eliminated by introducing 0-1 integer variables, leading to one MIP problem instance. An alternative reduction to MIP has been presented in [16], based on a Kripke semantics of $\mathcal{L}_\infty$. Other authors have looked at proof calculi which are more similar to the calculi used for classical propositional logic, typically focusing on generalizations of Horn or Krom clauses for which reasoning is in P. For instance, in [21] a resolution principle is discussed for a clause form based on McNaughton functions. In [15], a purely syntactical clause form is used, which covers, however, only a subfragment of the formulas that can be expressed in Łukasiewicz logic.

In the case of multi-valued logics with a finite number of truth degrees, most work is based on the fact that any formula, regardless of the specific interpretation of the logical connectives, can be translated in a satisfiability-preserving way to a boolean combination of expressions of the form $x \in S$, with x a variable and S a set of truth degrees [7]; we refer to [9] for an overview.

3 Basic Notions

3.1 Fuzzy Clauses

We define a *fuzzy clause* as the disjunction of zero or more *fuzzy literals*, which are in turn defined as either an upper bound or a lower bound on a *fuzzy expression*. Fuzzy expressions are either constants or variables, or the application of a fuzzy logic operator on fuzzy expressions. In this paper, we will only consider the

Łukasiewicz connectives as fuzzy logic operators, i.e. the operators $\otimes$, $\oplus$, $\rightarrow$, $\wedge$, $\vee$ and $\neg$ are interpreted for a and b in $[0,1]$ as

$$a \otimes b = \max(0, a+b-1) \quad a \oplus b = \min(1, a+b) \quad a \rightarrow b = \min(1, 1-a+b)$$
$$a \wedge b = \min(a,b) \qquad a \vee b = \max(a,b) \qquad \neg a = 1-a$$

Note that all these operators can be defined in terms of $\rightarrow$ and 0: $\neg a = a \rightarrow 0$, $a \otimes b = \neg(a \rightarrow \neg b)$, $a \oplus b = \neg(\neg a \otimes \neg b)$, $a \wedge b = a \otimes (a \rightarrow b)$, and $a \vee b = \neg(\neg a \wedge \neg b)$. Constants are taken from a finite set $M_k = \{0, \frac{1}{k}, \ldots, \frac{k-1}{k}, 1\}$ with $k \in \mathbb{N} \setminus \{0\}$, and variables are taken from some finite set V. When there is more than one fuzzy literal in a fuzzy clause, angle brackets $\langle . \rangle$ are used around the fuzzy literals for clarity (see (1)). Furthermore, as a notational convenience, we sometimes use expressions of the form $\phi = \lambda$, with ϕ a fuzzy expression and $\lambda \in [0,1]$, as a shorthand for the conjunction of fuzzy literals $\phi \geq \lambda$ and $\phi \leq \lambda$.

The notion of satisfiability for a set of fuzzy clauses Θ is defined as follows. Let M be a subset of $[0,1]$; then an M-interpretation is any mapping from V to M. We write $[\phi]_{\mathcal{I}}$ for the valuation of a fuzzy expression ϕ under the interpretation $\mathcal{I}$. An M-interpretation $\mathcal{I}$ satisfies a fuzzy literal $\phi \geq \lambda$, resp. $\phi \leq \lambda$, if it holds that $[\phi]_{\mathcal{I}} \geq \lambda$, resp. $[\phi]_{\mathcal{I}} \leq \lambda$; $\mathcal{I}$ satisfies a fuzzy clause if it satisfies at least one of its disjuncts. Next, $\mathcal{I}$ satisfies a set of fuzzy clauses Θ iff $\mathcal{I}$ satisfies each of the fuzzy clauses in Θ. Such an interpretation $\mathcal{I}$ is called an M-model of Θ. Finally, the set Θ is called M-satisfiable iff Θ has at least one M-model. For convenience $[0,1]$-interpretations, $[0,1]$-models, and $[0,1]$-satisfiability will simply be referred to as interpretations, models, and satisfiability when there is no cause for confusion. Similarly, when $M = \{0, \frac{1}{k}, \ldots, 1\}$ we will talk about k-interpretations, k-models and k-satisfiability. It is interesting to note that the satisfiability problem is NP-complete even when every fuzzy clause only contains one fuzzy literal.

In Łukasiewicz logic, a well-formed formula is either a variable, or the application of $\oplus$ or $\neg$ to well-formed formulas. We call a formula ϕ from $\mathcal{L}_\infty$ satisfiable if there is an interpretation $\mathcal{I}$ such that $[\phi]_{\mathcal{I}} = 1$. Alternatively, ϕ is sometimes called satisfiable as soon as $[\phi]_{\mathcal{I}} > 0$ for some $\mathcal{I}$. To avoid confusion, we will sometimes refer to the former variant as strong satisfiability and to the latter as weak satisfiability. Note that strong satisfiability indeed implies weak satisfiability. Clearly, every well-formed formula ϕ from Łukasiewicz logic naturally corresponds to the fuzzy clause $\langle \phi \geq 1 \rangle$, in the sense that the former is (strongly) satisfiable iff the latter is satisfiable. Conversely, it is also possible to convert a fuzzy clause to a well-formed formula from $\mathcal{L}_\infty$ in a satisfiability-preserving way. To see this, first note that $\rightarrow$, $\otimes$, $\wedge$ and $\vee$, as defined above, can all be represented in terms of $\oplus$ and $\neg$. Furthermore, every fuzzy clause can be converted into a fuzzy clause of the form $\phi \geq 1$, where ϕ is a well-formed formula from L_∞, which is satisfiable iff ϕ is strongly satisfiable. For example $\phi \geq \lambda$ is satisfiable iff $\lambda \rightarrow \phi \geq 1$ is satisfiable; $\langle \phi_1 \geq 1 \rangle \vee \langle \phi_2 \geq 1 \rangle$ is satisfiable iff $\max(\phi_1, \phi_2) \geq 1$ is satisfiable; and rational constants can be replaced by variables, using fuzzy clauses similar to (2). Fuzzy clauses seem to be more convenient in practical applications, whereas the

well-formed formulas from Łukasiewicz logic make it easier to study theoretical properties of the logic (axiomatization, completeness, etc.).

Example 1. Consider the following sets Θ_n of fuzzy clauses:

$$\{a_1 \otimes a_1 = 0.5, a_2 \otimes a_2 \rightarrow a_1 \geq 1, a_3 \otimes a_3 \rightarrow a_2 \geq 1, \ldots, a_n \otimes a_n \rightarrow a_{n-1} \geq 1,$$
$$a_1 \rightarrow a_2 \otimes a_2 \geq 1, a_2 \rightarrow a_3 \otimes a_3 \geq 1, \ldots, a_{n-1} \rightarrow a_n \otimes a_n \geq 1\}$$

Then Θ_n is k-satisfiable iff k is a multiple of 2^{n+1}, its unique[2] k-model $\mathcal{I}$ being defined by $\mathcal{I}(a_i) = 1 - (\frac{1}{2})^{i+1}$. The exponential number of truth degrees needed illustrates that we cannot, in general, significantly strengthen the upper bound from [2] by only looking at the number of variables and variable occurrences.

3.2 Disjunctive Linear Relations

A linear relation is an expression of the form $a_1 v_1 + a_2 v_2 + \cdots + a_n v_n \diamond b$ where a_i and b are real numbers, and $\diamond$ is $\geq$ or $\leq$. A disjunctive linear relation (DLR) is an expression of the form $\gamma_1 \vee \gamma_2 \vee \cdots \vee \gamma_s$ with each γ_i a linear relation. We write $vars(\gamma)$ to denote the set of variables occurring in a DLR γ (with a non-zero coefficient), and $vars(\Gamma) = \bigcup_{\gamma \in \Gamma} vars(\gamma)$ for a set of DLRs Γ. An M-solution of Γ is a mapping from $vars(\Gamma)$ to M ($M \subseteq \mathbb{R}$) which verifies at least one disjunct from every DLR in Γ. Let Θ be a set of fuzzy clauses involving only constants from the set M_k. Then we can construct a set Γ of DLRs in polynomial time such that all coefficients that occur in the linear relations are integers and the right-hand sides are of the form $\frac{l}{k}$ for some l in $\mathbb{N}$, and such that every $[0,1]$-solution of Γ corresponds to a model of Θ and vice versa. In particular, writing $\mathcal{L}(\theta)$ for the set of DLRs corresponding to the fuzzy clause θ, we have

$$\begin{aligned}
\mathcal{L}(\neg\alpha \leq \lambda) &= \mathcal{L}(\alpha \geq 1 - \lambda) \\
\mathcal{L}(\neg\alpha \geq \lambda) &= \mathcal{L}(\alpha \leq 1 - \lambda) \\
\mathcal{L}(\alpha_1 \oplus \alpha_2 \leq \lambda) &= \mathcal{L}(\alpha_1 \leq x_1) \cup \mathcal{L}(\alpha_1 \geq x_1) \cup \mathcal{L}(\alpha_2 \leq x_2) \cup \mathcal{L}(\alpha_2 \geq x_2) \\
&\quad \cup \{x_1 + x_2 \leq \lambda \vee \lambda \geq 1\} \\
\mathcal{L}(\alpha_1 \oplus \alpha_2 \geq \lambda) &= \mathcal{L}(\alpha_1 \leq x_1) \cup \mathcal{L}(\alpha_1 \geq x_1) \cup \mathcal{L}(\alpha_2 \leq x_2) \cup \mathcal{L}(\alpha_2 \geq x_2) \\
&\quad \cup \{x_1 + x_2 \geq \lambda, \lambda \leq 1\} \\
\mathcal{L}(\alpha_1 \vee \alpha_2 \leq \lambda) &= \mathcal{L}(\alpha_1 \leq \lambda) \cup \mathcal{L}(\alpha_2 \leq \lambda) \\
\mathcal{L}(\alpha_1 \vee \alpha_2 \geq \lambda) &= \mathcal{L}(\alpha_1 \leq x_1) \cup \mathcal{L}(\alpha_1 \geq x_1) \cup \mathcal{L}(\alpha_2 \leq x_2) \cup \mathcal{L}(\alpha_2 \geq x_2) \\
&\quad \cup \{x_1 \geq \lambda \vee x_2 \geq \lambda\}
\end{aligned}$$

The translations for the other operators are obtained by writing them in terms of $\neg$, $\oplus$ and $\vee$, e.g. $\mathcal{L}(\alpha_1 \rightarrow \alpha_2 \leq \lambda) = \mathcal{L}(\neg\alpha_1 \oplus \alpha_2 \leq \lambda)$. For disjunctive clauses, the transformation is performed by applying the transformation to each of the disjuncts and converting the result to conjunctive-normal form. In the transformations above, x_1 and x_2 refer to fresh variables. When α_1 and α_2 are

[2] We assume that V contains no irrelevant variables, i.e. $V = \{a_1, \ldots, a_n\}$.

variables or constants, we can of course have simpler translations, e.g. $\mathcal{L}(\alpha_1 \oplus \alpha_2 \leq \lambda) = \{\alpha_1 + \alpha_2 \leq \lambda \vee \lambda \geq 1\}$. Other optimizations are possible, resulting in a more concise translation; we will briefly come back to this in Section 4.3. Note that λ can be a variable or a constant of the form $\frac{l}{k}$, i.e. the argument of the mapping $\mathcal{L}$ does not need to be a fuzzy clause. Also note that we are slightly abusing notation, in the sense that e.g. $x_1 + x_2 \geq \lambda$ should be read as $x_1 + x_2 - \lambda \geq 0$ when λ is a variable.

Geometrically, the solution space of Γ corresponds to the union of a finite number of polyhedra, whose dimension is at most $|vars(\Gamma)|$. Now assume that Θ is satisfiable; then there is at least one non-empty polyhedron. All points in these polyhedra, and in particular also the vertices of these polyhedra, correspond to models of Θ. From Cramer's rule, we know that the coordinates $(x_1, \ldots, x_m)$ of such vertices are of the form $\frac{det(A_i)}{det(A)}$ where A is an $m \times m$ matrix whose rows correspond to the coefficients of particular disjuncts of particular DLRs from Γ and columns are associated to the corresponding variables; A_i is the matrix that is obtained by replacing the i^{th} column of A by the corresponding right-hand sides. Every x_i is the interpretation of a given variable in the associated model of Θ; if m is smaller than the number of variables of Γ, the remaining variables are unconstrained. Note that both $k \cdot det(A_i)$ and $det(A)$ are integers, since all the coefficients in Γ are integers and the right-hand sides are of the form $\frac{l}{k}$. This means that Θ is k'-satisfiable for $k' = k \cdot abs(det(A))$. Although finding an appropriate matrix A is in itself NP-hard, this idea can be exploited by deriving upper bounds for the value of $det(A)$. This idea was pursued in [3], where, using the Hadamard inequality, it led to the aforementioned upper bound of $\left(\frac{\sharp\phi}{n}\right)^n$.

3.3 Cycle Basis

We associate a bipartite graph $G_\Gamma = (N, E)$ with Γ as follows. The set N contains one node $\nu(\gamma)$ for every DLR γ in Γ (called a DLR-node) and one node $\nu(v)$ for every variable v occurring in Γ (called a variable-node). Furthermore, there is an edge in E between $\nu(\gamma)$ and $\nu(v)$ iff $v \in vars(\gamma)$.

Each cycle in an undirected graph with n_e edges can be represented as an n_e-dimensional vector $\mathbf{v} = (v_i)$ over F_2, the Galois field of order 2. Every dimension corresponds to a specific edge, and $v_i = 1$ iff the corresponding edge is contained in the cycle. The vector space (over F_2) spanned by all the cycle vectors of G is called the cycle space of G. It can be shown that the dimension of the cycle space is equal to $n_e - n_v + n_c$, where n_e, n_v and n_c are the number of edges, nodes and connected components of G [5]. Finding a cycle basis of a graph is in P, even when some notion of minimality is required [11]

4 Relating Cycles to Truth Degrees

Throughout this section, we let Θ be some set of fuzzy clauses, involving only constants from M_k, and we let Γ be an equivalent set of DLRs with associated

graph $G_\Gamma = (N, E)$. We use $\mathcal{A}$ to denote the set of coefficient matrices corresponding to *possible* vertices of the polyhedra that define the solution space of Γ. More precisely, there is an A in $\mathcal{A}$ for every subset of linear relations occurring as disjuncts in Γ such that the number of variables involved is equal to the size of this subset. Most importantly, for every vertex of the aforementioned polyhedra, there will be a corresponding A in $\mathcal{A}$. From the preceding discussion, we therefore know that if Θ is satisfiable, there will be an A in $\mathcal{A}$ such that Θ is also k'-satisfiable for $k' = k \cdot abs(det(A))$. Our aim is therefore to provide an upper bound for $\max_{A \in \mathcal{A}} abs(det(A))$.

If G_Γ consists of several connected components $G_1, \ldots, G_s$, then there must be a corresponding partitioning $\Gamma = \Gamma_1 \cup \cdots \cup \Gamma_s$ and $\Theta = \Theta_1 \cup \cdots \cup \Theta_s$ such that $vars(\Gamma_i) \cap vars(\Gamma_j) = \emptyset$ for $i \neq j$, and similar for Θ_i and Θ_j. Thus, we have that Θ is satisfiable iff each of the subproblems Θ_i is satisfiable. Without loss of generality, we can therefore assume henceforth that G_Γ is connected.

If G_Γ does not contain any cycles, the following upper bound can be obtained.

Lemma 1. *If G_Γ does not contain any cycles, it holds for any $A = (a_{ij})$ in $\mathcal{A}$ that*

$$abs(det(A)) \leq \max_{\sigma} \prod_{i=1}^{m} abs(a_{i\sigma(i)})$$

where m is the dimension of A, and the maximum is taken over all permutations σ of $(1, \ldots, m)$.

Note that when Γ is obtained using the procedure from Section 3.2, all non-zero coefficients are in fact -1 or 1, in which case we obtain $abs(det(A)) \in \{0, 1\}$. Certain optimizations, however, may lead to different coefficients, in which case we still need to obtain an upper bound for $abs(a_{i\sigma(i)})$ (recall that the matrix A is unknown). A straightforward possibility is to use $\prod_{\gamma \in \Gamma} c(\gamma)$, with $c(\gamma)$ the largest value among the absolute values of the coefficients occurring in γ.

In the general case, where G_Γ contains cycles, we can split the problem in two subproblems, as follows. Let C be a cycle appearing in G_Γ, and let n be a node through which this cycle passes. Furthermore, let $E_1 \cup E_2$ be the set of edges that are incident with n, such that both E_1 and E_2 contain an edge from cycle C. Then let $G_1 = (N, E \setminus E_1)$ and $G_2 = (N, E \setminus E_2)$, and let Γ_1 and Γ_2 be the sets of DLRs corresponding to G_1 and G_2 respectively. Note that a given edge corresponds to the fact that a given variable occurs in a given DLR with a non-zero coefficient. Thus, it is clear that Γ_1 and Γ_2 can be obtained from Γ by removing the occurrences of particular variables, i.e. by replacing their coefficients with 0. Furthermore, let $\mathcal{A}_1$ and $\mathcal{A}_2$ be the sets of coefficient matrices corresponding to Γ_1 and Γ_2. Then it is straightforward to show the following lemma.

Lemma 2. *For any A in $\mathcal{A}$, there is an A_1 in $\mathcal{A}_1$ and an A_2 in $\mathcal{A}_2$ such that*

$$abs(det(A)) \leq abs(det(A_1)) + abs(det(A_2))$$

Note in particular that $\mathcal{A}_1$ and $\mathcal{A}_2$ correspond to subproblems in which the dimension of the cycle basis of the associated graph has decreased by at least one. Both subproblems can be further divided into subproblems with a cycle space of an even smaller dimension. This can be repeated until a set of subproblems is obtained whose associated graphs do no longer contain any cycles. It is easy to see that the number of subproblems can at most be $2^{dim(G_\Gamma)}$, where $dim(G_\Gamma)$ is the dimension of the cycle space of G_Γ. Since, moreover, the absolute values of the coefficients in these subproblems cannot be larger than in Γ, we arrive at the following corollary.

Corollary 1. *Let Θ, Γ and G_Γ be defined as before, and assume that G_Γ is connected. It holds that Θ is satisfiable iff Θ is $k \cdot l$-satisfiable for at least one l not greater than*

$$2^{(\sum_{\gamma \in \Gamma} |vars(\gamma)|) - |\Gamma| - |vars(\Gamma)| + 1} \cdot \prod_{\gamma \in \Gamma} c(\gamma)$$

Note that this bound will often be significantly lower than the bound from [2], because it does not only take into account the number of variables and variable occurrences, but to some extent also the structure of the problem.

4.1 Splitting into Subproblems

In this section, we improve the procedure above by looking at how a given problem can be split into subproblems more effectively. First note that by defining the sets of edges E_1 and E_2 more carefully, it is often possible to reduce the dimension of the cycle space by more than 1. Moreover, if we allow that a given problem may be split into more than two subproblems, it is always possible to define the subproblems such that none of their cycle spaces contains any cycle that passes through n.

Specifically, let n be a node from G_Γ such that at least one cycle passes through n, and let $\mathcal{C}$ be an arbitrary cycle basis of G_Γ. We now define the graph $G^n = (E^n, M^n)$ as follows. Every node in E^n corresponds to an edge in G_Γ that is incident with n and vice versa. Furthermore, there is an edge in M^n between the nodes e_1 and e_2 from E^n iff there is a cycle in $\mathcal{C}$ that contains both the edges corresponding to e_1 and e_2. Once the graph G^n has been constructed, we define s subproblems of Γ, where s is the size of the largest connected component of G^n. In particular, the graphs $G_1, \ldots, G_s$ are obtained from G_Γ such that they contain all edges that are not incident with n, and moreover

1. Each graph G_i does not contain more than one edge from the same connected component of G^n (recall that the nodes of G^n are edges in G_Γ).
2. Every edge that is incident with n in G_Γ occurs in exactly one of the subgraphs G_i.

Then we have the following result.

Lemma 3. *Let $G_\Gamma, G_1, \ldots, G_s$ and n be defined as above. It holds that C is a cycle in G_i ($i \in \{1, \ldots, s\}$) iff C is a cycle in G_Γ and C contains no edges that are incident with node n.*

Note that the graph G^n is constructed by looking at the cycles in $\mathcal{C}$ only, and therefore depends on the particular cycle basis that was chosen. However, as part of the proof of Lemma 3, we can show that the size of the largest connected component of G^n is independent of this choice.

One important problem remains. In principle, we need to apply this process repeatedly on each of the obtained subproblems, which would require an exponential amount of time. However, this can be avoided by considering the fact that the procedure only depends on a given cycle basis of the graph, and not on the graph itself. This means that we do not actually need to construct the subgraphs G_i, only their cycle bases. The feasibility of this approach then follows from the fact that each of the subgraphs G_i has the same cycle space $\mathcal{C}'$, defined as follows. Let $\mathcal{C}_0 = \mathcal{C}$, and let $\{e_1, e_2, \ldots, e_l\}$ be the set of edges that are incident with n in G_Γ, and that appear in some cycle of $\mathcal{C}$. For each e_i, we define a set of vectors (cycles) $\mathcal{C}_i$ which is obtained from $\mathcal{C}_{i-1}$ as follows. Let $\mathbf{a}_1$, $\ldots$, $\mathbf{a}_s$ be the vectors (cycles) from $\mathcal{C}_{i-1}$ that contain the edge e_i. Then

$$\mathcal{C}_i = (\mathcal{C}_{i-1} \setminus \{a_j | j = 1, \ldots, s\}) \cup \{a_j + a_{j+1} | j = 1, \ldots, s-1\}$$

where the vector addition is carried out in F_2. Furthermore, we let $\mathcal{C}' = \mathcal{C}_l$.

Lemma 4. *Let $G_1, \ldots, G_s$ and $\mathcal{C}'$ be defined as above. It holds that $\mathcal{C}'$ is a cycle basis of G_i, for all i in $\{1, \ldots, s\}$.*

Thus, the whole procedure consists of producing a cycle basis for G_Γ, repeatedly choosing a node n, and in each step reducing the cycle basis accordingly. As a result, we end up with a certain number K of subproblems (being the product of the numbers of subproblems obtained in each reduction of the cycle basis), each of which satisfies the conditions of Lemma 1.

4.2 Harmless Cycles

Note that every row of a coefficient matrix $A \in \mathcal{A}$ corresponds to a DLR from Γ and every column corresponds to a variable. We write G_A to refer to the subgraph of G_Γ that corresponds to A, i.e. there is a node $\nu(\gamma)$ in G_A for every row (i.e. DLR) in A and a node $\nu(v)$ for every column (i.e. variable). There is an edge between $\nu(\gamma)$ and $\nu(v)$ iff the coefficient of v in the row corresponding to γ is non-zero. To apply Lemma 1, it is in fact sufficient that G_A does not contain any cycles. The reason that we have insisted that G_Γ does not contain any cycles is that we do not know, a priori, which nodes and edges from G_Γ will be contained in G_A. There are, however, some types of cycles which are in some sense harmless, and do not need to be eliminated as a result. In this section, we explore this idea with the aim of further tightening the upper bound. We

will restrict our attention to the special, but common, case where all non-zero coefficients occurring in Γ, and therefore also in A, are either 1 or -1. Recall that it is always possible to construct Γ such that this requirement is met.

Given a subgraph G' of G_Γ containing an equal number of DLR-nodes and variable-nodes, it is easy to see that there is a unique matrix A such that $G_A = G'$, and such that the non-zero elements of A are determined by the corresponding coefficients in Γ. If G' corresponds to a cycle, the corresponding matrix A contains exactly two non-zero positions on each row. Now let us call a row in such a matrix positive if the two non-zero positions have the same sign, and negative otherwise. A first type of harmless cycles results from the following lemma, which can be shown using Leibniz' rule for determinants.

Lemma 5. *Let A be the matrix corresponding to some cycle C. It holds that*

$$abs(det(A)) = \begin{cases} 2 & \text{if the number of positive rows in } A \text{ is odd} \\ 0 & \text{otherwise} \end{cases}$$

Cycles whose associated matrix has a zero determinant will be called zero cycles. As clarified in the following lemma, such cycles can be ignored when they are not entangled with other cycles.

Lemma 6. *Let G_Γ, A and G_A be defined as before. Furthermore, assume that exactly s cycles $C_1, \ldots, C_s$ occur in G_A, that no two cycles have an edge in common, and that z of these cycles are zero-cycles. Then either $det(A) = 0$ or $abs(det(A)) = 2^l$ for some $l \leq s - z$.*

Thus to find an upper bound on the number of truth degrees, we only need to eliminate cycles from the cycle basis until all the remaining cycles are isolated zero cycles. A second type of harmless cycles is obtained by the following variant of Lemma 6, involving a stronger form of disentanglement.

Lemma 7. *Let G_Γ, A and G_A be defined as before. Furthermore, assume that exactly s cycles $C_1, \ldots, C_s$ occur in G_A, and that no two cycles have a node in common. It holds that*

$$abs(det(A)) = \begin{cases} abs(det(A)) = 2^s & \text{if none of the cycles is a zero cycle} \\ 0 & \text{otherwise} \end{cases}$$

What is crucial in this lemma is that the value of the determinant in each subproblem depends only on the cycle basis and the original matrix A. In particular, assume that after we have split the original problem in K subproblems, all cycles remaining in the cycle basis are disentangled in the strong sense of Lemma 7. Then for some s, the value of $det(A)$ will be equal to

$$det(A) = det(A_1) + \cdots + det(A_K) = 2^s(y_1 + \cdots + y_K)$$

where each y_i is either 1 or -1. The value s depends on how many of the remaining cycles are also present in the subgraph G_A. At most, s can be equal to the number

of cycles remaining in the cycle basis. Now assume that there is some cycle C still in the cycle basis and that this cycle is contained in G_A. Furthermore, assume that (at least) one of the DLRs involved in this cycle is such that in each of its disjuncts, the right-hand side is an even multiple of $\frac{1}{k}$. Then we can divide the right-hand sides, as well as the coefficients of this DLR by two, without changing the solution space. Note that the coefficients occurring in this DLR will then not necessarily be integers anymore. The result will be that the value of $det(A)$ is halved to $2^{s-1}(y_1 + \cdots + y_K)$, which is exactly the value that would have been obtained if the cycle C did not occur. Consequently, when all cycles are disentangled in the sense of Lemma 7 we can ignore those cycles which have the property that at least one participating DLR has only even multiples of $\frac{1}{k}$ as right-hand sides. The optimizations resulting from Lemma 6 and Lemma 7 can also be combined; we omit the details.

4.3 Further Optimizations

The general method we introduced above can be optimized in several aspects. For example, it is important to choose the "right" node n to reduce the cycle basis in each step. A second aspect which is amenable to optimization is the definition of Γ. After a straightforward translation of Θ to a set of DLRs Γ, we can simplify Γ in several ways. For example, it may happen that a given variable x only occurs positively, in the sense that increasing the value of x may never cause a DLR to be violated, in which case we can simply replace x by 1. A similar observation applies to the dual notion of negatively occurring variables.

Moreover, from the theory of linear programming, it follows that we can remove one arbitrary DLR from Γ without compromising the soundness of the overall approach. Indeed, assume for example that this DLR γ is of the form $\alpha \geq \lambda$ (a similar reasoning applies when the DLR contains several disjuncts). While not all solutions of $\Gamma \setminus \{\gamma\}$ correspond to models of Θ anymore, we know that the solution of $\Gamma \setminus \{\gamma\}$ which maximizes the linear expression α corresponds to a vertex of the solution space of $\Gamma \setminus \{\gamma\}$, which will also be a vertex in the solution space of Γ when Θ is satisfiable. As a consequence, the upper bound resulting from $\Gamma \setminus \{\gamma\}$ is still an upper bound for the minimal number of truth degrees needed to decide satisfiability.

5 Preliminary Experimental Results

The aim of our experiments is two-fold. First, we want to verify that the proposed techniques indeed lead to significantly tighter upper bounds for some problems. Second, we want to verify that having a tighter upper bound can indeed lead to complete satisfiability checking procedures that outperform MIP for some problems. We can expect that such results would only hold for certain types of problem instances, i.e. MIP will still outperform the methods we sketch below in many cases. Similarly, as already illustrated in Example 1, it is sometimes not even possible to improve the bound from [2] significantly.

Table 1 reports the upper bounds that were found for a number of randomly generated data sets. In particular, we have generated sets of n fuzzy clauses in which every fuzzy clause consists of exactly one fuzzy literal, containing a fixed number m of variable occurrences. Literals are of the form $\phi \geq \lambda$ or $\phi \leq \lambda$, where λ is a constant which is chosen randomly from M_4 (with a uniform distribution), and ϕ is a fuzzy expression consisting of applications of $\otimes$ and $\neg$ to variables, e.g. $x_1 \otimes \neg(x_2 \otimes x_3)$. Clearly the number of applications of $\otimes$ is equal to $m-1$. The probability that a negation occurs in front of a subexpression was set to 0.5. Clearly, the upper bounds presented in Table 1 are a substantial improvement over the upper bound for strong satisfiability resulting from [2], which is e.g. of the order $4 \cdot 3^{100}$ in the case where $m = 3$ and $n = 100$. The upper bounds are also a substantial improvement over the upper bound derived in Corollary 1, which is of the order $4 \cdot 2^{100}$ for $m = 3$ and $n = 100$; note that the exact bound depends on the translation to DLRs of the fuzzy clauses.

Table 1. Value of l such that satisfiability can be reduced to $4l$-satisfiability. The values shown are *median(min,max)* for 50 randomly generated, satisfiable data sets with m variable occurrences per fuzzy clause, and a total of n different variables and n fuzzy clauses.

n	10	20	30	40	50	60	80	100
$m = 2$	1(1,2)	1(1,2)	1(1,1)	1(1,2)	1(1,2)	1(1,2)	1(1,2)	1(1,2)
$m = 3$	1 (1,8)	1 (1,8)	1 (1,8)	1 (1,256)	2 (1,32)	2 (1,256)	2(1,32)	4 (1,256)
$m = 4$	1(1, 64)	2(1, 64)	8(1, −)	16(1, −)	32(1, −)	24(1, −)	160(2, −)	192(2, −)

Note that Table 1 reports the median values, rather than the average. One reason is that the latter is very sensitive to outliers and therefore less informative; e.g. for $m = 4$ and $n = 100$ the average value is over 72000, whereas the median is 192. The second, related reason is that the actual value of outliers is of no practical importance. When the upper bound is too large, we need to fall back on existing techniques such as MIP. Our general strategy for satisfiability checking in $\mathcal{L}_\infty$ is therefore given by

1. Determine an upper bound l^* for the number of truth degrees.
2. Use a reasoner for finite multi-valued logics, and consider all values $k = 4l$ for $1 \leq l \leq \min(t, l^*)$, until a model is found, where t is a predefined threshold.
3. If no models were found and $l^* > t$, use a reasoner based on MIP.

The intuition behind this approach is that the computation time for the first step is insignificant compared to that of the second step, which is in turn insignificant compared to that of the third step (if t is sufficiently small). In Figure 1, the computation time of this procedure is compared against that of a reasoner which is exclusively based on MIP. Again the results are based on randomly generated sets of fuzzy clauses, in which the number of variables n is equal to the number of clauses, and in which there are three variable occurrences per fuzzy clause. For each value of n, 50 different sets of fuzzy clauses were generated, either satisfiable or unsatisfiable. As the fuzzy clauses are generated independent of each other,

Table 2. Number of sets of fuzzy clauses that are satisfiable/unsatisfiable for different sizes n

n	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
satisfiable	40	40	36	32	32	35	31	28	24	23	23	18	17	26	17	17	17	17	16	16	18	14	7	8
unsatisfiable	10	10	14	18	18	15	19	22	26	27	27	32	33	24	33	33	33	33	34	34	32	36	43	42

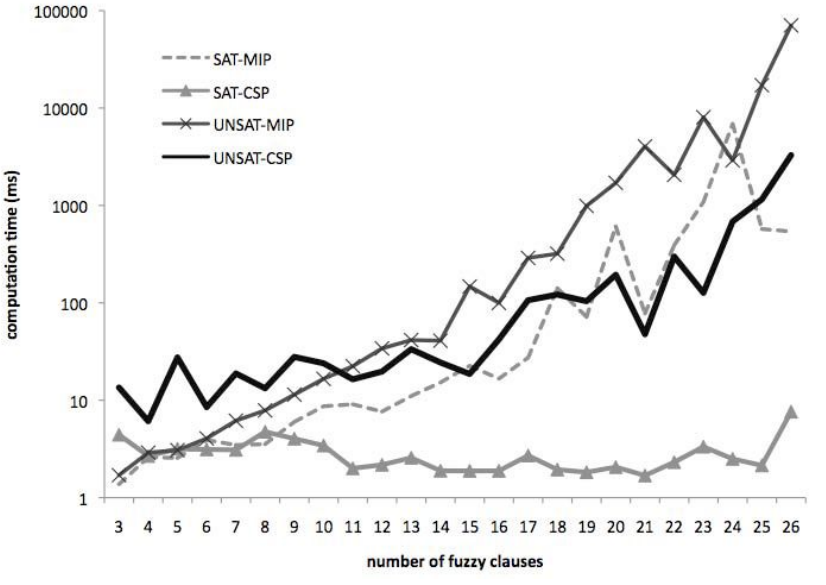


Fig. 1. Comparison of a MIP based reasoner with a CSP based reasoner. Each data point is the average value for 50 randomly generated problem instances.

the larger the number of clauses n, the higher the probability that a given set is unsatisfiable. Table 2 reports the number of sets that are satisfiable/unsatisfiable for each n. To solve MIP instances we used lp_solve[3]. To check the satisfiability in finite-valued logics, we straightforwardly reduce the satisfiability problem to a constraint satisfaction problem (CSP), which is subsequently solved using Choco[4]. The results in Figure 1 were obtained for $t = 10$. In the figure, the execution time for satisfiable instances is shown separately from that of unsatisfiable instances. Interestingly, when first trying to find models in finite-valued logics (SAT-CSP), satisfiable instances are recognized extremely fast on average. Regarding unsatisfiable instances, Figure 1 suggests that the approach based on finite-valued logics (UNSAT-CSP) is at least one order of magnitude faster than the traditional MIP approach (UNSAT-MIP) on average.

[3] `http://lpsolve.sourceforge.net/`
[4] `http://choco-solver.net/`

6 Concluding Remarks

We showed how a reasonable upper bound can be found for the number of truth degrees needed to decide (strong) satisfiability in $\mathcal{L}_\infty$. Our main contribution lies in the fact that, unlike existing approaches, we do not only look at the number of variables and variable occurrences, but also at the structure of the formulas. This is accomplished by relating fuzzy clauses, or equivalently, well-formed formulas in $\mathcal{L}_\infty$, to cycles in an associated bipartite graph. Our results open the door for efficient reasoners in $\mathcal{L}_\infty$, and they furthermore offer valuable insights in the connection between finite and infinite Łukasiewicz logic. Experimental results demonstrate the potential of our approach, although we should be careful in extrapolating results for (simple) random data to real-world problem instances. Future work will therefore focus on a more thorough evaluation, characterizing when the proposed approach is successful and when it is not.

Acknowledgments

Steven Schockaert was funded as a postdoctoral fellow of the Research Foundation – Flanders. Jeroen Janssen was funded by a joint Research Foundation – Flanders project. We are grateful to the anonymous reviewer whose detailed comments helped in clarifying the presentation of our results.

References

1. Aguzzoli, S.: An asymptotically tight bound on countermodels for łukasiewicz logic. International Journal of Approximate Reasoning 43, 76–89 (2006)
2. Aguzzoli, S., Ciabattoni, A.: Finiteness in infinite-valued Łukasiewicz logic. Journal of Logic, Language, and Information 9, 5–29 (2000)
3. Aguzzoli, S., Gerla, B.: Finite-valued reductions of infinite-valued logics. Archive for Mathematical Logic 41, 361–399 (2002)
4. Bobillo, F., Straccia, U.: Fuzzy description logics with general t-norms and datatypes. Fuzzy Sets and Systems (in press) doi:doi:10.1016/j.fss.2009.03.006
5. Bollobás, B.: Modern Graph Theory. Springer, Heidelberg (1998)
6. Ernst, M., Millstein, T., Weld, D.: Automatic SAT-compilation of planning problems. In: Proceedings of the 15th International Joint Conference on Artificial Intelligence, pp. 1169–1176 (1997)
7. Hähnle, R.: Towards an efficient tableau proof procedure for multiple-valued logics. In: Börger, E., Kleine Büning, H., Richter, M., Schönfeld, W. (eds.) Selected Papers from Computer Science Logic, pp. 248–260. Springer, Heidelberg (1991)
8. Hähnle, R.: Many-valued logic and mixed integer programming. Annals of Mathematics and Artificial Intelligence 12, 231–264 (1994)
9. Hähnle, R., Escalada-Imaz, G.: Deduction in multivalued logics: a survey. Mathware & Soft Computing 4(2), 69–97 (1997)
10. Janssen, J., Heymans, S., Vermeir, D., De Cock, M.: Compiling fuzzy answer set programs to fuzzy propositional theories. In: Proceedings of the 24th International Conference on Logic Programming, pp. 362–376 (2008)

11. Kavitha, T., Mehlhorn, K., Michail, D., Paluch, K.: An $o(m^2n)$ algorithm for minimum cycle basis of graphs. Algorithmica 52, 333–349 (2008)
12. Lin, F., Zhao, Y.: ASSAT: computing answer sets of a logic program by SAT solvers. Artificial Intelligence 157(1-2), 115–137 (2004)
13. McNaughton, R.: A theorem about infinite-valued sentential logic. Journal of Symbolic Logic 16, 1–13 (1951)
14. Mundici, D.: Satisfiability in many-valued sentential logic is NP-complete. Theoretical Computer Science 52, 145–153 (1987)
15. Mundici, D., Olivetti, N.: Resolution and model building in the infinite-valued calculus of Łukasiewicz. Theoretical Computer Science 200, 335–366 (1998)
16. Olivetti, N.: Tableaux for Łukasiewicz infinite-valued logic. Studia Logica 73(1), 81–111 (2003)
17. Pham, D., Thornton, J., Sattar, A.: Modelling and solving temporal reasoning as propositional satisfiability. Artificial Intelligence 172(15), 1752–1782 (2008)
18. Schockaert, S., De Cock, M., Kerre, E.: Spatial reasoning in a fuzzy region connection calculus. Artificial Intelligence 173, 258–298 (2009)
19. Straccia, U., Bobillo, F.: Mixed integer programming, general concept inclusions and fuzzy description logics. Mathware & Soft Computing 14(3), 247–259 (2007)
20. Tamura, N., Taga, A., Kitagawa, S., Banbara, M.: Compiling finite linear CSP into SAT. In: Benhamou, F. (ed.) CP 2006. LNCS, vol. 4204, pp. 590–603. Springer, Heidelberg (2006)
21. Wagner, H.: A new resolution calculus for the infinite-valued propositional logic of Łukasiewicz. In: Proceedings of the International Workshop on First order Theorem Proving, pp. 234–243 (1998)
22. Zhang, L., Malik, S.: The quest for efficient boolean satisfiability solvers. In: Brinksma, E., Larsen, K.G. (eds.) CAV 2002. LNCS, vol. 2404, pp. 641–653. Springer, Heidelberg (2002)

Compression of Probabilistic XML Documents

Irma Veldman, Ander de Keijzer, and Maurice van Keulen

University of Twente
P. O. Box 217 Enschede
The Netherlands
{veldmani,keijzer,keulen}@cs.utwente.nl

Abstract. Database techniques to store, query and manipulate data that contains uncertainty receives increasing research interest. Such UDBMSs can be classified according to their underlying data model: relational, XML, or RDF. We focus on uncertain XML DBMS with as representative example the Probabilistic XML model (PXML) of [10,9]. The size of a PXML document is obviously a factor in performance. There are PXML-specific techniques to reduce the size, such as a *push down* mechanism, that produces equivalent but more compact PXML documents. It can only be applied, however, where possibilities are dependent. For normal XML documents there also exist several techniques for compressing a document. Since Probabilistic XML is (a special form of) normal XML, it might benefit from these methods even more. In this paper, we show that existing compression mechanisms can be combined with PXML-specific compression techniques. We also show that best compression rates are obtained with a combination of PXML-specific technique with a rather simple generic DAG-compression technique.

1 Introduction

Probabilistic XML (PXML) is XML that allows the representation of uncertainty in the data [10, 9]. Uncertainty can, for example, arise from the integration of two or more XML documents when conflicts or ambiguities are encountered. Resolving these at integration time often is a severe obstacle, because it requires a huge amount of user effort. Being able to leave unresolved issues as uncertainty in the integrated document removes this obstacle, as they can resolved when they become visible during use, i.e., at query time.

In the field of probabilistic XML some good solutions have been achieved with respect to data representation and efficient querying of the uncertain data, as in [10, 9]. For illustration purposes, we use the same running example as in [10]. The example concerns the integration of two small address books, each containing a record of a person named `John`, whose phone number is `1111` in one address book and `2222` in the other. Integrating these two address books will result in ambiguity and a possible conflict. In the real world, different situations could be possible:

L. Godo and A. Pugliese (Eds.): SUM 2009, LNAI 5785, pp. 255–267, 2009.

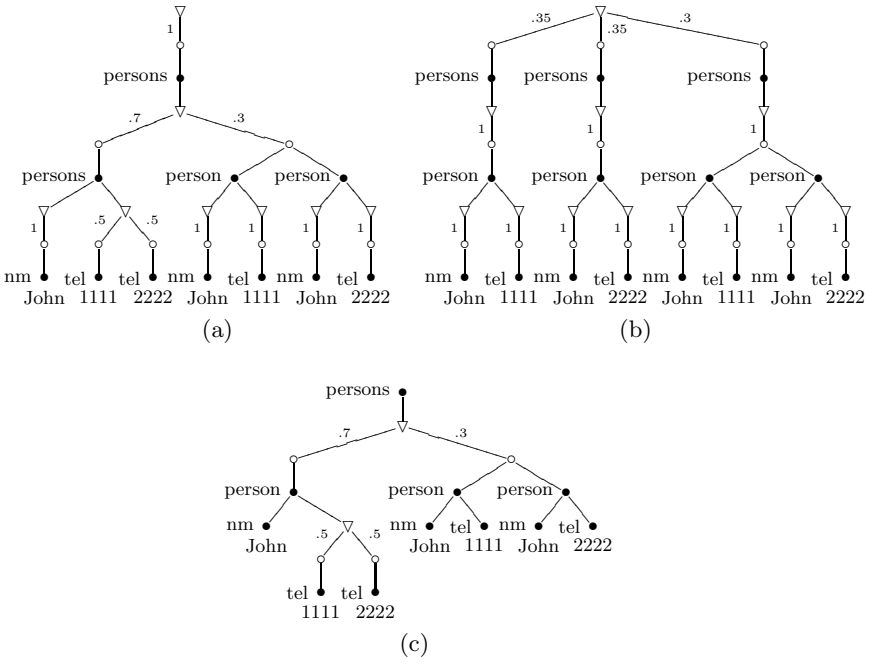


Fig. 1. Example probabilistic XML tree (a), its possible world style (PWS) tree (b), and its reduced form where probability and possibility nodes that give no extra information are omitted (c)

- Both records refer to the same person named `John`, but one of the phone numbers is wrong.
- Both records refer to a different person named `John` and for each person the phone number is correct.

A rule engine could assign probability values to these different situations. During integration these possible situations are represented as uncertainty in the PXML document. After integration, the probabilistic XML document could look like the PXML tree in Figure 1(a).

The $\triangledown$ nodes denote probability nodes. They represent choice points in the tree. Its children are possibility nodes that represent the mutually exclusive possible subtrees. The $\circ$ nodes denote the possibility nodes. They have an associated probability value. This value lies within the range $\langle 0.0, 1.0]$. The actual probability value is determined by a rule engine. The $\bullet$ nodes represent normal XML nodes.

Obviously it is important to be able to query the uncertain data represented by a PXML document. This is what [10] says about the semantics of querying uncertain data:

> "*Uncertainty can be treated as having more than one possible instantiation describing a particular real world object. Choosing one possible instantiation, or* possibility *for short, for each real world object, results in a possible world. Analogous to the notion of parallel universes, all possible worlds co-exist in the database and a query should, therefore, be evaluated in every possible world separately.*"

Our example document represents 3 possible worlds (see Figure 1(b)).

Unfortunately, this *Possible World Style* (PWS) comes with a major drawback, which does not emerge from this example because it is too small. Imagine the integration of address books with over 100 records each. With n ($n < 100$) conflicting records, each with 3 possible real world situations, the PWS of the document could grow a factor 3^n. Efficient querying techniques do not suffer from this drawback, because they work directly on the compact representation [7]. In these compact representations possibilities are pushed down to lower levels in the tree and probability and possibility nodes that do not provide extra information are removed, see Figure 1(c) and section 2.3.

It is evident that there is a strong relationship between the size of the PXML-document and query performance. This paper addresses the issue of how to obtain a PXML-document that is as compact as possible. Since PXML is a special kind of XML, one solution direction is to apply XML compression techniques. Another is to apply PXML-specific compression techniques. Note that the latter produce a PXML-document that is *equivalent*, i.e., encodes the same set of possible worlds, but that may from a normal XML point-of-view be structurally completely different (see for example Figures 1(a), 1(b), and 1(c) which are all equivalent). Also note that most compression techniques are orthogonal to each other, hence can be applied in combination to possibly achieve even better compression than each individually.

In this paper, we evaluate several combinations of PXML-specific and XML-generic compression techniques. Important evalation criteria are compression ratio, compression time, and query execution time on the compressed documents. This paper focuses solely on the former, i.e., we determine experimentally which combination has the highest compression ratio. We use a real-world data set originating from the application area of probabilistic data integration.

2 PXML Compression

There are several ways to categorize compression mechanisms. In the comparative study of Ng et al. [5], the authors chose to categorize the compression techniques into queriable versus unqueriable techniques. Queriable techniques come with the important feature that they can be queried directly on the compressed data, but unfortunately do not perform as well in terms of compression ratio and execution time. Unqueriable techniques need to fully decompress their data before it can be queried again, but they can achieve a much higher compression ratio.

In our study of XML compression techniques we also group the compressors by their (in)ability of supporting queries. Due to space limitations, we only review queriable techniques here. For a complete review, we refer to [11]. PXML documents obtained from probabilistic data integration appear to contain many subtrees that are highly similar, but not completely equal. Therefore, we pay special attention to an advanced compression technique called BPLEX in Section 2.2, because it can compress parameterized subtrees.

2.1 Queriable XML Compression Techniques

XGrind [6] and XPress [4] are queriable compression techniques that adopt a homomorphic transformation, which means that the structure and semantics of the XML document are preserved. This enables the document to be parsed as any other XML document. As with XMill, XGrind uses a dictionary encoding approach for the tag and attribute names. The data values are encoded by Huffman encoding (for the non-enumerated attribute values and the PCDATAs) or binary encoded (for the enumerated attribute values).

XPress [4] uses a reverse arithmetic encoding scheme for the encoding of the skeleton. This method encodes not only the tag name, but also the tree path to this tag. Such a tree path is modeled as a real number interval in the range $[0.0, 1.0\rangle$ that satisfies the suffix containment property. This means that if an element path P is a suffix of an element path Q, the interval that represents P should contain the interval of Q. XPress can automatically determine the type of a data value and hence apply the proper compression for it. Besides, XPress also supports query updates directly on the compressed data.

Another approach for the compression of the skeleton of an XML document is the use of DAGs (Directed Acyclic Graphs). This technique is based on the sharing of common subtrees and is applied in [1]. The compressed document is still queriable and results can be returned in compressed form to serve as an input for another query.

We illustrate DAG-compression with an example from [2]. It is based on the tree *c(c(a,a),c(a,a))*. Figure 2(a) shows the XML code of the example. The corresponding tree can be seen in 2(b). The minimal DAG for this tree is illustrated in Figure 2(c).

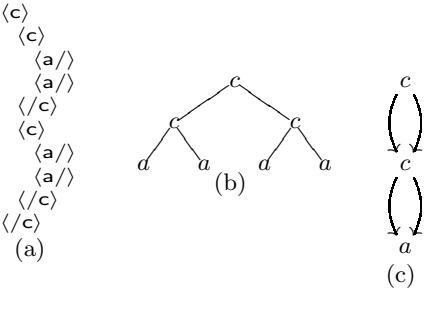


Fig. 2. Example XML fragment (a), its corresponding tree (b) and minimal DAG (c)

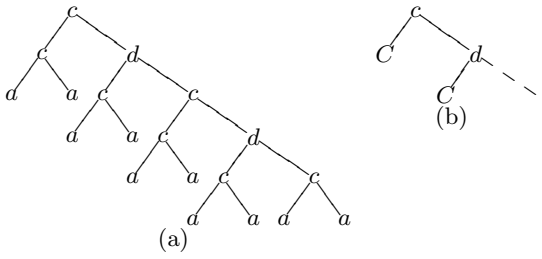


Fig. 3. Example XML tree (a) with the pattern $C \rightarrow c(A,A)$ and $A \rightarrow a$ illustrated in (b) which shows the restrictions of DAG-compression

2.2 BPLEX

As mentioned, we pay some special attention to the compression technique, called BPLEX (bottom-up multiplexing). For a more detailed description of the algorithm, we refer to [2]. It takes the idea of transforming the XML tree into a DAG one step further. It is based on the sharing of common parameterized subgraphs instead of common subtrees. This makes it possible to share parts of a subtree instead of complete subtrees, which increases the sharing opportunities.

XML trees can be expressed as grammars. The minimal unique DAG can also be seen as the minimal regular tree grammar that generates the tree. A generalization of the sharing of subtrees is the sharing of arbitrary patterns, i.e., connected subgraphs of a tree. A sharing graph can be seen as a context-free (cf) tree grammar. For example, the minimal DAG of Figure 2(c) can be described by a minimal regular tree grammar consisting of the following productions: $S \rightarrow c(V,V)$, $V \rightarrow c(W,W)$ and $W \rightarrow a$.

We illustrate the idea of a sharing graph in the next example, also from [2]. We take the tree $c(c(a,a),d(c(a,a),c(c(a,a),d(c(a,a),c(a,a)))))$ which is depicted in Figure 3(a). In this tree, there is a pattern (3(b)) that is repeated. Because different subtrees are hanging underneath, DAG-compression is not able to obtain sharing for this pattern. However, with the introduction of formal parameters, we can share this subgraph. The resulting grammar would have the following productions: $S \rightarrow B(B(C))$, $B(y_1) \rightarrow c(C,d(C,y_1))$, $C \rightarrow c(A,A)$ and $A \rightarrow a$. This context-free grammar is also called a *straight-line* (SL) grammar.

The sharing of the pattern is depicted in Figure 4(b). We see that the most upper c has two special incoming edges. These special incoming edges are recognizable by the $\perp$ symbol at the end of the edge. This means that the subtree is shared.

Walking through the tree, we arrive at this c from the incoming edge marked with a 1. This number at the end of the edge means that whenever a choice has to be made between two or more outgoing edges belonging to a choice, recognizable by the $\perp$ symbol at the start of the edge, you have to choose the one with the same number. In this case this outgoing edge itself is again a special incoming edge, marked with a 2, meaning that the next time you come across a choice

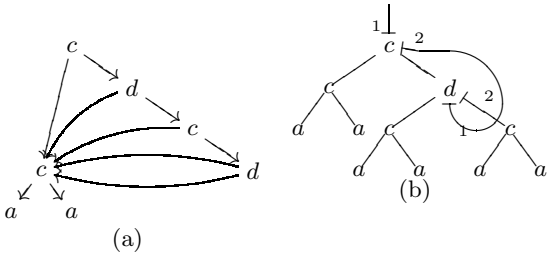


Fig. 4. The DAG created from Figure 3 (a) and the plexed version (b)

point, you have to choose the other special outgoing edge. The other 'normal' outgoing edge from node d is not marked, hence it is not shared. The numbers alongside the edges represent the formal parameters in the grammar.

In Figure 4, we can also see the difference between creating a DAG from the tree in Figure 3 and plexing it. The original tree has 19 nodes. The DAG has already reduced this to 7 nodes. Obviously, on the plexed tree the *c(a,a)*-subtrees can also be shared, even completely. However, we did not do that for simplicity. However, when we would have done that, another 6 nodes disappear and we are left with only 5 nodes, which is less than the DAG variant.

2.3 PXML-specific Compression

A method to compact specifically PXML documents is described in [10]. This technique is different from generic XML compression techniques, because it produces a document that is fundamentally different according to XML semantics. The resulting document is, however, possible worlds equivalent, i.e., it represents the same possible worlds. We illustrate the push-down technique in Figure 5. We call this technique *simplification* in this paper. The *compact representation* of a tree, is the tree after simplification. We refer to [11] for a more detailed description of the algorithm.

3 Experiments

To evaluate the effectiveness of the various combinations of compression techniques on PXML documents, we conducted some experiments. First, we briefly discuss the prototype implementation, and experimental setup and measurements. A more detailed description of the prototype and experiments can be found in [11].

Prototype. The main goal of the prototype is to measure compression ratios of PXML documents. The performance of the prototype in terms of compression and decompression speed will not be part of the comparison.

We adapted the BPLEX algorithm of [2] to work on a DOM structure of the tree [3] instead of the SL grammar. We call this algorithm PLEX.

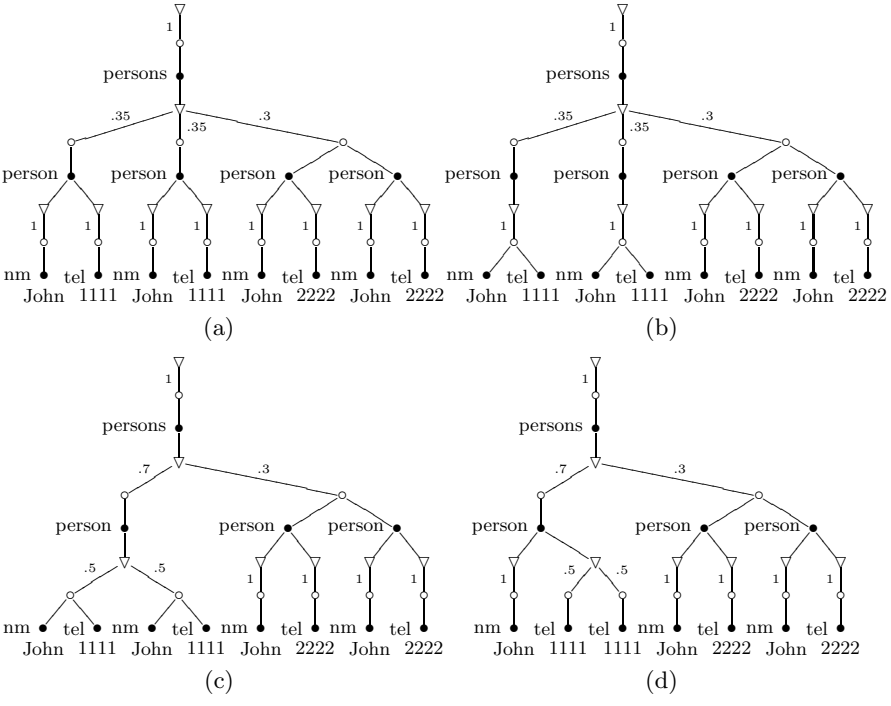


Fig. 5. Four example iterations of the PXML-specific push-down compression

3.1 Measurement

We evaluate the compression techniques and combinations thereof based on *compression ratio*. Common definitions for compression ratio are: 1) the number of bits required to represent a byte, 2) the fraction in terms of bytes of the input document eliminated. Since there are many encoding techniques used to store XML in file systems or XML databases, we believe that a compression ratio measure based on a size in bytes is not suitable. XML databases, for instance, use indices for tag names and text fields, hence the byte size of a document is very system-specific. The compression techniques we focus on compress XML documents by reducing the number of nodes in the XML tree. We therefore measure compression ratios in terms of numbers of nodes.

The number of nodes in a document (n_{total}) is the total number of elements (n_{elements}), text nodes (n_{text}) and attributes ($n_{\text{attributes}}$): $n_{\text{total}} = n_{\text{elements}} + n_{\text{text}} + n_{\text{attributes}}$. The compression ratio r is defined as: $r = 1 - \frac{n^{\text{after}}_{\text{total}}}{n^{\text{before}}_{\text{total}}}$. In this formula, $n^{\text{after}}_{\text{total}}$ is the number of nodes after applying the compression, and $n^{\text{before}}_{\text{total}}$ the number of nodes before compression.

Besides this measurement, we are interested in the amount of *overhead* involved in compressed documents due to necessary additional bookkeeping. Typically, id's and other attributes need to be added to represent for example

references. Overhead nodes n_{overhead} are all such attributes and nodes initially not present in the uncompressed documents. The overhead o is defined as: $o = \frac{n_{\text{overhead}}}{n_{\text{total}}}$.

3.2 Data Sets

We use PXML documents representing the uncertain integration result produced by the probabilistic integration technique of [10]. We use PXML documents obtained from integration under different conditions such as other integration rules and thresholds. The integration scenario used in [8,9] concerns integration of data from a TV guide[1] with data from IMDB[2].

We use two sets of documents. It is beyond the scope of this paper to elaborate on all parameters that are involved in the generation of these documents. It suffices to know that these parameters are directly related to the amount of uncertainty in the integrated document. Since there is no generally accepted measure for 'amount of uncertainty', we use the parameters themselves as indicators for a growing amount of uncertainty. The first set of documents contains integration results where the parameter *actor threshold* has been varied. The threshold determines a minimum similarity when the system considers two `actor` elements to possibly refer to one and the same actor in real life. The second set of documents contains integration results where the parameter *movie margin* has been varied. The margin determines how much more dissimilar a movie title may be than the best matching title for the system to consider the possibility that the movie title may happen to refer to the same movie in real life. Details can be found in [8]. It suffices to know that a lower actor treshold and a higher margin mean more uncertainty. The size of the documents varies from 3177 nodes in the smallest document to 44815 nodes in the biggest document.

3.3 Results

We feed the documents to various combinations of compression techniques. We use the following abbreviations for the different combinations: *SIMP* is the simplification method; *RRPP* the removal of redundant probability and possibility nodes; *PXML* is a combination of these two methods; *SIMP_DAG* stands for the combination of simplification and building a DAG. The rest is self-explanatory.

Figure 6 shows the average compression ratio and overhead over all documents in both sets for each combination of compression techniques.

Note that there is no overhead after applying SIMP, RRPP and PXML. Remarkable is that DAG scores better than PLEX on these documents. SIMP on its own, as well as combinations with DAG and PLEX score better than DAG or PLEX solely, as expected. Same is true for RRPP and PXML. However, the ratio for combinations with RRPP or PXML are almost exactly the same. We explore this later.

[1] http://www.tvguide.com
[2] http://www.imdb.com

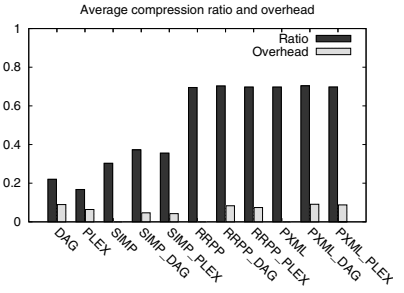


Fig. 6. Average compression ratio and overhead for each combination of compression techniques

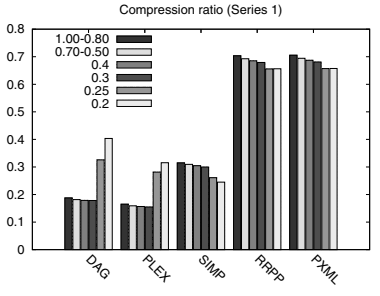


Fig. 7. Influence of amount of uncertainty (actor threshold) on the compression ratio obtained for various compression techniques (first document set)

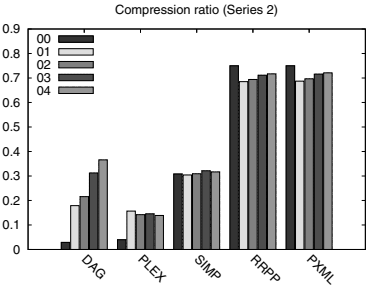


Fig. 8. Influence of amount of uncertainty (movie margin) on the compression ratio obtained for various compression techniques (second document set)

We can also see in Figure 6 that the amount of overhead decreases when we combine DAG or PLEX with the other methods. This makes sense, since the other methods already achieve a certain amount of compression. The amount of nodes to which DAG or PLEX can be applied is then already decreased, hence the smaller overhead.

Let's take a more detailed look on the results. Figure 7 and Figure 8 show the compression ratios for the first, respectively the second document set. Documents associated with thresholds that display highly similar compression ratios have been combined. The bars in the figures from left to right belong to documents with increasing uncertainty.

It is interesting to see that for both the series, DAG, and for the first series, PLEX, benefit from the increasing amount of uncertainty. More uncertainty means more duplicated data and hence more chances for matches. Surprisingly, simplification does not benefit from the uncertainty.

For RRPP it is not really a surprise that it doesn't benefit from uncertainty, because some probability and possibility nodes are not redundant any more. Since PXML is the combination of SIMP and RRPP, this method doesn't benefit

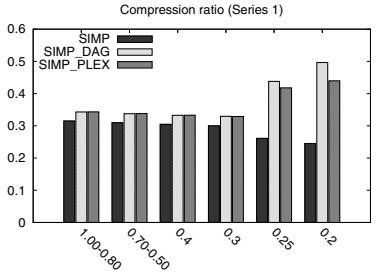


Fig. 9. Sensitivity of combinations with SIMP to the amount of uncertainty (first document set)

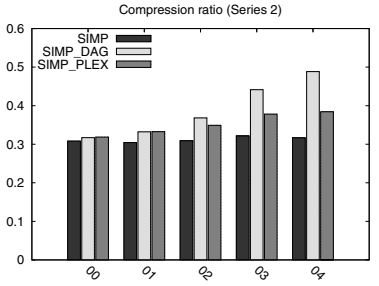


Fig. 10. Sensitivity of combinations with SIMP to the amount of uncertainty (second document set)

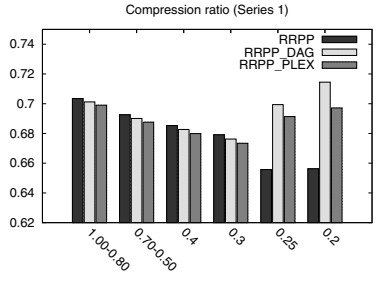


Fig. 11. Sensitivity of combinations with RRPP to the amount of uncertainty (first document set)

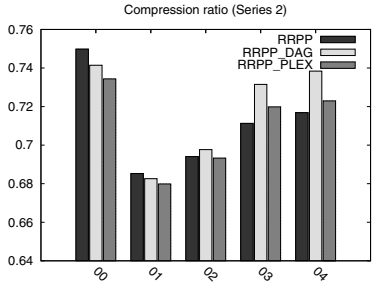


Fig. 12. Sensitivity of combinations with RRPP to the amount of uncertainty (second document set)

either. In the second series however, RRPP does benefit from uncertainty. This might be caused by the new (partially) duplicated subtrees that are added due to the uncertainty. If these subtrees are deep and don't have any useful probability and possibility nodes, then the number of redundant probability and possibility nodes increases, hence compression ratio increases.

In Figure 9 and Figure 10 we see how PXML and DAG, and PXML and PLEX these combinations perform with only simplification. Both diagrams show us that the combinations perform better when uncertainty increases. Especially the combination with DAG performs well. One might think that this is caused by the larger amount of overhead that naturally comes with PLEX, but the results do not confirm this. Another explanation for this, is that matches can be applied straightforward with DAGs, whereas with PLEX one needs to check if previous matches did not change the subtree to which the match refers so that this match is not applicable anymore. For the combination of RRPP with DAG and PXML, we see the same pattern, see Figure 11 and Figure 12.

Table 1. Details for the first and last document of the first series

Document	**1.00**		**0.20**	
n_{total}^{before}	9692		16041	
Mode	PXML	RRPP	PXML	RRPP
n_{total} after SIMP	6638	-	12111	-
n_{total} after RRPP	2822	2840	5496	5514
r (ratio)	0.709	0.707	0.758	0.757
Contribution of SIMP	52%	-	37%	-

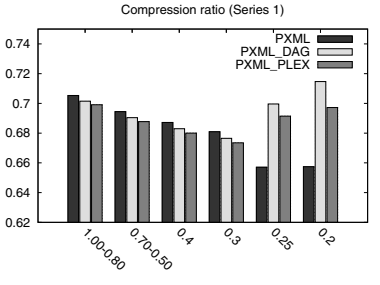


Fig. 13. Sensitivity of combinations with PXML to the amount of uncertainty (first document set)

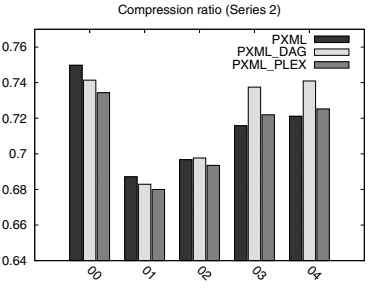


Fig. 14. Sensitivity of combinations with PXML to the amount of uncertainty (second document set)

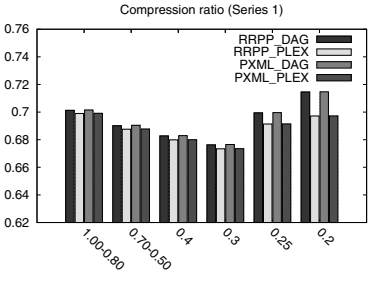


Fig. 15. Sensitivity of combinations with DAG and PLEX to the amount of uncertainty (first document set)

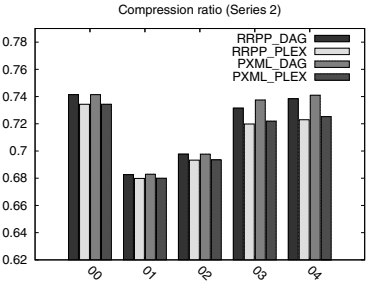


Fig. 16. Sensitivity of combinations with DAG and PLEX to the amount of uncertainty (second document set)

As we have already seen in Figure 6, PXML has almost the same results as RRPP. However, when we look at intermediary results, we see that SIMP significantly contributes to the compression ratio. It is just that, when RRPP is applied after SIMP, one almost gets the same end result as without SIMP,

see Table 1. It is important that we know that SIMP does in fact contribute to the compression, which cannot be distinguished in the diagrams. From Figure 6, we can conclude that on average the methods that involve RRPP perform best. This is not a surprise. RRPP is the method that could and should always be used since it deletes redundant nodes, and no method is restricted by the removal of these nodes.

As the uncertainty increases in a document, a combination of RRPP with DAG outperforms a combination with PLEX, see Figure 15 and 16. Although the differences between the results are small, we believe DAG is better than PLEX. Not only is the algorithm of PLEX more complex than DAG, it also results in documents that are more complex. This means that not only the compression itself could suffer from performance problems, also querying the document would take more time.

4 Conclusions

In this paper we have evaluated the compression ratios of various combinations of compression techniques, both generic XML compression techniques and PXML-specific compression techniques. As representative techniques, we took the DAG and PLEX methods for generic XML compression techniques, and simplification and redundant nodes removal for PXML-specific compression techniques.

We experimentally evaluated the compression ratios and overhead for the different methods and combination of methods. We used documents with an increasing amount of uncertainty. RRPP is the method that should always be used, since it removes useless nodes. The compression ratio can be improved by combining it with DAG or PLEX. Although both methods show good increasing compression ratio with increasing amount of uncertainty DAG is preferred, since the algorithm and the resulting document are less complex than with PLEX.

In terms of size reduction, it is not worth the effort of simplifying the document, since RRPP alone achieves almost the same reduction. However, simplifying the document comes with a more simple structure of the document, that might increase performance when querying the document or even navigating. If we look only at the size of the document, performing SIMP is not worth it. Besides compression ratio we also measured the amount of overhead as a consequence of additional bookkeeping necessary in the DAG and PLEX methods. The results showed that the amount of overhead was reasonable. The results gave no indication that the amount of overhead was problematic.

Although, the compression ratio is substantial, it can be increased even more. Furthermore, for PXML documents to be used in practice, not only performance in terms of compression but also time and space complexity of the compression and decompression algorithms need to be improved to be able to handle large documents. Finally, document size is only one factor in query and update performance. It is an open problem how query algorithms can be adapted to support the compressed formats. Also the influence of these adaptations on query performance may change the suitability of the investigated combinations of compression techniques.

References

1. Buneman, P., Grohe, M., Koch, C.: Path queries on compressed XML. In: Proceedings of the 29th VLDB Conference (2003)
2. Busatto, G., Lohrey, M., Maneth, S.: Efficient memory representation of XML document trees. Information Systems 33(4-5), 456–474 (2008)
3. Document object model, DOM (2009), `http://www.w3.org/DOM`
4. Min, J.-K., Park, M.-J., Chung, C.-W.: A compressor for effective archiving, retrieval, and updating of XML documents. ACM Trans. Internet Technol. 6(3), 223–258 (2006)
5. Ng, W., Lam, W.-Y., Cheng, J.: Comparative analysis of XML compression technologies. World Wide Web 9(1), 5–33 (2006)
6. Tolani, P.M., Haritsma, J.R.: Xgrind: A query-friendly XML compressor. In: IEEE Proceedings of the 18th International Conference on Data Engineering (2002)
7. van Kessel, R.: Querying probabilistic XML. Master's thesis, University of Twente (April 2008)
8. van Keulen, M., de Keijzer, A.: Qualitative effects of knowledge rules in probabilistic data integration. Technical Report TR-CTIT-08-42, Centre for Telematics and Information Technology, University of Twente, Enschede (2008)
9. van Keulen, M., de Keijzer, A.: Qualitative effects of knowledge rules and user feedback in probabilistic data integration. The VLDB Journal (2009); To Appear in Special Issue on Uncertain and Probabilistic Databases
10. van Keulen, M., de Keijzer, A., Alink, W.: A probabilistic XML approach to data integration. In: Proc. ICDE Conf., pp. 459–470 (2005)
11. Veldman, I.E., de Keijzer, A., van Keulen, M.: Compression of probabilistic XML documents. Technical Report TR-CTIT-09-20, CTIT, Enschede (May 2009)

Query Answering in Belief Logic Programming*

Hui Wan and Michael Kifer

State University of New York at Stony Brook
Stony Brook, NY 11794, USA

Abstract. In this paper we introduce a fixpoint semantics for quantitative logic programming, which is able to both combine and correlate evidence from different sources of information. Based on this semantics, we develop efficient algorithms that can answer queries for non-ground programs with the help of an SLD-like procedure. We also analyze the computational complexity of the algorithms and illustrate their uses.

1 Introduction

Quantitative logic programming is a widely popular approach for dealing with uncertainty and inconsistency in knowledge representation. The issue of combining evidence obtained from different sources has been known since the early attempts to develop expert systems for the medical and other domains [20]. However, researchers quickly realized the difficulty of addressing this issue formally and one of the most common ways to sidestep this problem is to arbitrarily assume that all information sources are independent.

Recently, the problem of combining correlated information obtained from distributed sources has been formally addressed by a new theory called *Belief Logic Programming* (BLP) [23]. BLP's semantics takes into account correlation of evidence obtained from different, but overlapping and, possibly, contradicting information sources. BLP's semantics is based on belief combination functions and is inspired by Dempster-Shafer theory of evidence [3, 18]. In our earlier work [23], we related BLP semantics to Dempster-Shafer theory and also showed the connection with certain forms of defeasible reasoning, such as Courteous Logic Programming [5] and, more generally, Logic Programming with Courteous Argumentation Theories (LPDA) [22].

However, the previous work didn't address the issues of efficient reasoning and query answering, especially for non-ground programs. The present paper presents a fixpoint semantics which coincides with the declarative semantics introduced in [23], and an efficient query answering algorithm, which can answer queries on non-ground programs with the help of an SLD-like procedure.

Although this paper is technically self-contained, it is a followup work on [23], which presented a model-theoretic semantics and detailed motivation for BLP. To avoid repetition, we do not rehash here the arguments showing limitations of the

* This work is part of the SILK (Semantic Inference on Large Knowledge) project sponsored by Vulcan, Inc.

L. Godo and A. Pugliese (Eds.): SUM 2009, LNAI 5785, pp. 268–281, 2009.

earlier approaches—those based on probability theory, Fuzzy Logic, Dempster-Shafer theory, and other approaches [1, 2, 4, 6–12, 14–17, 19, 21, 24].

The paper is organized as follows. Section 2 gives a brief overview of the syntax of BLP and necessary notions. Section 3 introduces the fixpoint semantics of BLP. The query answering algorithm and the SLD-like procedure are presented in Section 4. Section 5 concludes the paper.

2 Preliminaries

2.1 Syntax of BLP

A **belief logic program** (or a *blp*, for short) is a set of annotated rules. Each **annotated rule** is of the form

$$[v, w]\ X\ \texttt{:-}\ Body$$

where X is a positive atom and *Body* is a Boolean combination of atoms, i.e., a formula composed out of atoms and negated atoms by conjunction and disjunction. An atom in BLP has the form $p(t_1, ..., t_n)$, where p is a predicate and t_i is a constant or a variable, $1 \leq i \leq n$. We will use capital letters to denote positive atoms, e.g., A, and a bar over such a letter will denote negation, e.g., $\overline{A}$. The annotation $[v, w]$ is called a **belief factor**, where v and w are real numbers such that $0 \leq v \leq w \leq 1$.

The informal meaning of the above rule is that if *Body* is true, then this rule supports X to the degree v and $\overline{X}$ to the degree $1 - w$. The difference, $w - v$, is the *information gap* (or the degree of ignorance) with regard to X.

Note that, in keeping with the theory of evidence, BLP uses what is known as explicit negation (or, strong negation) [13] rather than negation as failure. That is, if nothing is known about A, it only means that there is no evidence that A holds; it does not mean that the negation of A holds.

An annotated rule with empty body is called an **annotated fact**; it is often written as $[v, w]\ X$. In the remainder of this paper we will deal only with annotated rules and facts and refer to them simply as rules and facts.

Definition 1. *Given a blp* $\boldsymbol{P}$*, an atom* X *is said to* ***depend*** *on an atom* Y

- *directly, if* X *is the head of a rule* R *and* Y *occurs in the body of* R*;*
- *indirectly, if* X *is dependent on* Z*, and* Z *depends on* Y*.* □

We require that in a ground blp no atom depends on itself. So, there can be no recursion among ground atoms, but recursion among predicates is possible. An extension of BLP that drops this restriction is future work.

2.2 Combination Functions

Definition 2. *Let* D *be the set of all belief factors,* $\Phi : D \times D \rightarrow D$ *is said to be a* ***belief combination function*** *if* Φ *is associative and commutative.* □

Due to the associativity of Φ, we can extend it from two arguments to nullary case, single argument, and three and more arguments: $\Phi() = [0,1]$, $\Phi([v,w]) = [v,w]$, $\Phi([v_1,w_1],...,[v_k,w_k]) = \Phi\big(\Phi([v_1,w_1],...,[v_{k-1},w_{k-1}]),[v_k,w_k]\big)$. Note that the order of arguments in a belief combination function is immaterial, since such functions are commutative, so we often write such functions as functions on multisets of belief factors, e.g., $\Phi(\{[v_1,w_1],...,[v_k,w_k]\})$.

Different types of beliefs might require different ways to combine them, so predicates in the same blp might be using different combination functions. Here are some popular combination functions:

- *Dempster's combination rule*:
  - $\Phi^{DS}([0,0],[1,1]) = [0,1]$.
  - $\Phi^{DS}([v_1,w_1],[v_2,w_2]) = [v,w]$ if $\{[v_1,w_1],[v_2,w_2]\} \neq \{[0,0],[1,1]\}$, where $v = \frac{v_1 \cdot w_2 + v_2 \cdot w_1 - v_1 \cdot v_2}{K}$, $w = \frac{w_1 \cdot w_2}{K}$, and $K = 1 + v_1 \cdot w_2 + v_2 \cdot w_1 - v_1 - v_2$. In this case, $K \neq 0$ and thus v and w are well-defined.
- *Maximum*: $\Phi^{max}([v_1,w_1],[v_2,w_2]) = [max(v_1,v_2), max(w_1,w_2)]$.
- *Minimum*: $\Phi^{min}([v_1,w_1],[v_2,w_2]) = [min(v_1,v_2), min(w_1,w_2)]$.

2.3 Support and Belief Functions

Given a blp $\boldsymbol{P}$, the definitions of *Herbrand Universe* $U_{\boldsymbol{P}}$ and *Herbrand Base* $B_{\boldsymbol{P}}$ of $\boldsymbol{P}$ are the same as in the classical case. Each atom X has an associated belief combination function, denoted Φ_X.[1] Intuitively, Φ_X is used to help determine the *combined* belief in X accorded by the rules in $\boldsymbol{P}$ that support X.

Definition 3. *A* ***truth valuation*** *over a set of atoms α is a mapping from α to $\{\mathbf{t},\mathbf{f},\mathbf{u}\}$. The set of all the truth valuations over α is denoted as $\mathcal{TV}al(\alpha)$.*

A ***truth valuation*** *I for a blp $\boldsymbol{P}$ is a truth valuation over $B_{\boldsymbol{P}}$. We often write $\mathcal{TV}al(B_{\boldsymbol{P}})$ as $\mathcal{TV}al(\boldsymbol{P})$.* □

Definition 4. *A* ***support function*** *for a set of atoms α is a mapping m_α from $\mathcal{TV}al(\alpha)$ to $[0,1]$ such that $\sum_{I \in \mathcal{TV}al(\alpha)} m_\alpha(I) = 1$.*

The atom-set α is called the ***base*** *of m_α. A* ***support function*** *for a blp $\boldsymbol{P}$ is a mapping m from $\mathcal{TV}al(\boldsymbol{P})$ to $[0,1]$ such that $\sum_{I \in \mathcal{TV}al(\boldsymbol{P})} m(I) = 1$.* □

If α is a set of atoms, we will use $\mathcal{B}ool(\alpha)$ to denote the set of all Boolean formulas constructed out of these atoms (i.e., using $\wedge$, $\vee$, and negation).

Definition 5. *Given a truth valuation I over a set of atoms α and a formula $F \in \mathcal{B}ool(\alpha)$, $I(F)$ is defined as in Lukasiewicz's three-valued logic: $I(A \vee B) = max\big(I(A), I(B)\big)$, $I(A \wedge B) = min\big(I(A), I(B)\big)$, and $I(\overline{A}) = \neg I(A)$, where $\mathbf{f} < \mathbf{u} < \mathbf{t}$ and $\neg\mathbf{t} = \mathbf{f}$, $\neg\mathbf{f} = \mathbf{t}$, $\neg\mathbf{u} = \mathbf{u}$. We say that $I \models F$ if $I(F) = \mathbf{t}$.* □

Definition 6. *A mapping* $\texttt{bel} : \mathcal{B}ool(B_{\boldsymbol{P}}) \longrightarrow [0,1]$ *is said to be a* ***belief function*** *for $\boldsymbol{P}$ if there exists a support function m for $\boldsymbol{P}$, so that for all $F \in \mathcal{B}ool(B_{\boldsymbol{P}})$,* $\texttt{bel}(F) = \sum_{I \in \mathcal{TV}al(\boldsymbol{P}) \text{ such that } I \models F} m(I)$. □

[1] Separate belief combination functions can be associated to different predicates or even ground atoms.

3 Fixpoint Semantics of BLP

Belief functions form the basis for the model-theoretic semantics for blps introduced in [23]. However, such semantics does not provide any effective way of computing the models or answering queries. In this section, we introduce a fixpoint semantics for BLP, which will serve as a basis for the development of a query answering algorithm in subsequent sections. As is customary in logic programming, we define the semantics using *ground* (i.e., variable-free) blps. Later this is lifted to the *non-ground* case.

Definition 7. *Suppose I is a truth valuation over a set of atoms β and $\alpha \subseteq \beta$. A* **restriction** *of I to α, denoted $I\mid_\alpha$, is the truth valuation over α such that $\forall X \in \alpha,\ I\mid_\alpha (X) = I(X)$.* □

Definition 8. *Let I_1 and I_2 be truth valuations over the sets of atoms α and β, respectively, where $\alpha \cap \beta = \varnothing$. The* ***union*** *of I_1 and I_2, denoted $I_1 \uplus I_2$, is the truth valuation over $\alpha \cup \beta$ such that $(I_1 \uplus I_2)\mid_\alpha = I_1$ and $(I_1 \uplus I_2)\mid_\beta = I_2$. It is easy to see that $I_1 \uplus I_2 = I_2 \uplus I_1$.* □

There is a special truth valuation whose domain is the empty set; we write it as $I_\varnothing : \varnothing \longrightarrow \{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$. It is clear that $\mathcal{TV}al(\varnothing) = \{I_\varnothing\}$ and $\forall \alpha \forall I \in \mathcal{TV}al(\alpha)\ I\mid_\varnothing = I_\varnothing$. There is also a special support function whose base is the empty set: $m_\varnothing(I_\varnothing) = 1$. We can see that $m_\varnothing$ is the only support function for $\varnothing$.

We define $\mathcal{SF}(\boldsymbol{P})$ to be the set of all support functions for $\boldsymbol{P}$: $\mathcal{SF}(\boldsymbol{P}) = \{m \mid m$ is a support function for $\alpha \in B_{\boldsymbol{P}}\}$. Associated with each blp $\boldsymbol{P}$, there is an operator $\hat{T}_{\boldsymbol{P}}$, which takes a support function in $\mathcal{SF}(\boldsymbol{P})$ as an argument and returns another support function in $\mathcal{SF}(\boldsymbol{P})$.

Definition 9. *Let $\boldsymbol{P}$ be a blp, $\hat{T}_{\boldsymbol{P}}$ is a mapping from $\mathcal{SF}(\boldsymbol{P})$ to $\mathcal{SF}(\boldsymbol{P})$, defined as follows. For any support function m in $\mathcal{SF}(\boldsymbol{P})$, let α be the base of m, let*

$$dep(\alpha) = \{X \mid X \in B_{\boldsymbol{P}} - \alpha, \text{ and every atom that } X \text{ depends on is in } \alpha\}.$$

Then $\hat{T}_{\boldsymbol{P}}(m)$ is a support function over the set of atoms $\alpha \cup dep(\alpha)$ such that for every $I_1 \in \mathcal{TV}al(\alpha)$ and $I_2 \in \mathcal{TV}al(dep(\alpha))$,

$$\hat{T}_{\boldsymbol{P}}(m)(I_1 \uplus I_2) = m(I_1) \cdot \prod_{X \in dep(\alpha)} Val\Big(\Phi_X\big(BF(X, \boldsymbol{P}, I_1)\big), I_2(X)\Big) \qquad (1)$$

where $BF(X, \boldsymbol{P}, I)$ is the multiset of the belief factors of all the rules in $\boldsymbol{P}$, which have atom X in head and whose bodies are true in I, and

$$Val([v, w], \tau) = \begin{cases} v, & \text{if } \tau = \mathbf{t}; \\ 1 - w, & \text{if } \tau = \mathbf{f}; \\ w - v, & \text{if } \tau = \mathbf{u}. \end{cases}$$

In the special case when the base of m is $B_{\boldsymbol{P}}$, we have $\hat{T}_{\boldsymbol{P}}(m) = m$. □

It can be shown that $\hat{T}_{\boldsymbol{P}}(m)$ is a support function because, for every $I_1 \in \mathcal{TV}al(\alpha)$, it is the case that $\sum_{I_2 \in \mathcal{TV}al(dep(\alpha))} \hat{T}_{\boldsymbol{P}}(m)(I_1 \uplus I_2) = m(I_1)$ and, consequently, $\sum_{I_1 \in \mathcal{TV}al(\alpha),\ I_2 \in \mathcal{TV}al(dep(\alpha))} \hat{T}_{\boldsymbol{P}}(m)(I_1 \uplus I_2) = 1$.

If the equation (1) looks mysterious, it should not be. All it says is that when we construct $\hat{T}_{\boldsymbol{P}}(m)$ by expanding the base of the support function m from α to $\alpha \cup dep(\alpha)$, we split each support amount $m(I_1)$ according to the constituent components $\hat{T}_{\boldsymbol{P}}(m)(I_1 \uplus I_2)$. Each such component is assigned a fraction specified by the product $\prod_{X \in dep(\alpha)}$ in (1). Product is used here because under a *fixed* truth valuation I_1 over α, the rule-sets that fire in I_1 and have *different* heads are treated as independent and each member of the product represents the contribution of each particular head $X \in dep(\alpha)$. In the contribution of a particular head, X, $\Phi_X\big(BF(X, \boldsymbol{P}, I_1)$ is simply the combined belief factor $[v, w]$ of the rules that support X in I_1. Val examines $I_2(X)$ and, depending on whether $I_2(X)$ is true, false, or unknown, interprets the first argument (the combined belief factor $[v, w]$ for X) as the degree of belief in X (i.e., v), or in $\overline{X}$ (i.e., $1 - w$), or as the uncertainty gap for X (i.e., $w - v$).

Theorem 1. *Let $\boldsymbol{P}$ be a blp. Starting with $m_0 = m_\varnothing$, let m_k be $\hat{T}_{\boldsymbol{P}}^{\uparrow k}(m_0)$, $k = 0, 1, \ldots$. There is an integer $0 \leq n \leq |B_{\boldsymbol{P}}|$ such that $m_n = m_{n+1}$. The support function m_n coincides with the support function $\hat{m}_{\boldsymbol{P}}$ defined in purely model-theoretic terms in [23].* □

Theorem 1 shows that there is always a fixpoint for the $\hat{T}_{\boldsymbol{P}}$ operator. And the fixpoint semantics coincides with the model-theoretic semantics defined in [23].

Definition 10. *Let $\boldsymbol{P}$ be a blp and m_ω be the fixpoint of $\hat{T}_{\boldsymbol{P}}$. The **model** of $\boldsymbol{P}$ is the following belief function:*

$$\texttt{model}(F) = \sum_{I \in \mathcal{TV}al(\boldsymbol{P}) \ \textit{such that}\ I \models F} m_\omega(I), \qquad \text{where } F \in \mathcal{B}ool(B_{\boldsymbol{P}}). \qquad \square$$

It turns out that `model` is precisely the "correct" and unique belief function that one should expect: it provides each atom in the Herbrand base with precisely the right amount of support from all the applicable rules. Namely, if S is a suitable set of the rules that support A (see [23] for a precise formulation) then

$$\frac{\texttt{model}\big(A \wedge \bigwedge_{R \in \mathcal{S}} Body(R)\big)}{\texttt{model}\big(\bigwedge_{R \in \mathcal{S}} Body(R)\big)} = v \qquad \frac{\texttt{model}\big(\overline{A} \wedge \bigwedge_{R \in \mathcal{S}} Body(R)\big)}{\texttt{model}\big(\bigwedge_{R \in \mathcal{S}} Body(R)\big)} = 1 - w.$$

where $[v, w] = \Phi_X(BF_{\mathcal{S}})$ and $BF_{\mathcal{S}}$ is the multiset of belief factors of rules in $\mathcal{S}$.

4 Query Answering

In this section we introduce proof DAGs and present an algorithm that uses proof DAGs for query answering. First, we consider only ground blps and queries, then in Section 4.3 we generalize the algorithm to *non-ground* blps and queries.

A **ground query** to a blp $\boldsymbol{P}$ is a statement of the form $? -\ Goal$, where $Goal \in \mathcal{B}ool(B_{\boldsymbol{P}})$. An answer to ground query $? -\ Goal$ is the belief by $\boldsymbol{P}$ in $Goal$.

Given a blp $\boldsymbol{P}$ and a ground query $? -\ Goal$, let $\boldsymbol{P}_g$ be a rule-set composed of all the rules in $\boldsymbol{P}$ plus the additional rule of the form $[1,1]\ g$:- $Goal$, where g is a new proposition. The belief in $Goal$ by $\boldsymbol{P}$ is equivalent to the belief in g by $\boldsymbol{P}_g$. This simple standard trick simplifies the matters by allowing us to assume that all ground queries are simply singleton atoms.

To simplify the description of the main algorithm in this section, we assume that each rule, R, has a unique identifier, denoted ID_R — a new propositional constant.

4.1 Proof DAG

We first introduce *dependency DAGs*, which represent derivation paths through which belief factors are propagated in a blp.

Definition 11. *The* ***dependency DAG****, $\mathcal{H}$, of a ground blp $\boldsymbol{P}$ is a directed acyclic bipartite graph whose nodes are partitioned into a set of* **atom nodes** *(*a-nodes*, for short) and* **rule nodes** *(*r-nodes*, for short). The nodes and edges are defined as follows:*

- *For each atom A in $\boldsymbol{P}$, $\mathcal{H}$ has an a-node labeled A.*
- *For each rule R in $\boldsymbol{P}$, $\mathcal{H}$ has an r-node labeled with* proposition ID_R.
- *For each rule R in $\boldsymbol{P}$, an edge goes from the r-node labeled ID_R to the a-node labeled with the atom in R's head.*
- *For each rule R in $\boldsymbol{P}$ and each atom A that appears in R's body, an edge goes from the a-node labeled A to the r-node labeled ID_R.*
- *Each edge that goes from an r-node, labeled ID_R, to an a-node is labeled with the belief factor associated with the rule R.* □

Proof DAGs are defined next as subgraphs of dependency DAGs, which contain only the atoms and rules involved with particular queries.

Definition 12. *Let $\mathcal{H}$ be the dependency DAG of a ground blp $\boldsymbol{P}$. The* ***proof DAG*** *of $\boldsymbol{P}$ for a ground query $? -\ g$, denoted $\mathcal{H}_g$, is a subgraph of $\mathcal{H}$ such that*

- *Every node in $\mathcal{H}_g$ is connected to the a-node labeled g by a directed path; and*
- *$\mathcal{H}_g$ contains all the nodes that are so connected to the a-node labeled g.* □

In dependency DAGs and proof DAGs, if there is an edge from node A to node B in a DAG, We call A a **child node** of B and B a **parent node** of A.

Example 1. Figure 1 shows the proof DAG of the following blp for the query $? -\ g$.

$rg:$	$[1,1]$	g	:-	b	$\vee$	e				
$r1:$	$[0.2,0.4]$	a	:-	b	$\wedge$	c	$r4:$	$[0.3,0.6]$	b	
$r2:$	$[0.8,0.8]$	e	:-	a	$\wedge$	c	$r5:$	$[1,1]$	c	
$r3:$	$[0.2,0.3]$	e	:-	b	$\wedge$	d	$r6:$	$[0.4,0.6]$	d	□

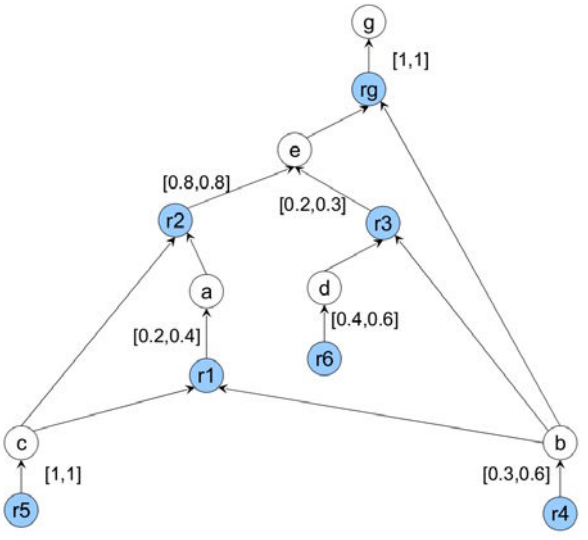


Fig. 1. The proof DAG for Example 1

Definition 13. *Let m_α and m_β be support functions for the sets of atoms α and β, respectively, where $\alpha \subseteq \beta$. We say m_α is a* ***projection*** *of m_β on α if for every $I \in \mathcal{TV}al(\alpha)$, $m_\alpha(I) = \sum_{J \in \mathcal{TV}al(\beta) \text{ where } J|_\alpha = I} m_\beta(J)$. (Recall that $J|_\alpha$ is defined in Definition 7.)* □

It is clear that for any support function m and any subset α of m's base, there exists one and only one projection of m on α. We denote it by $projection(m, \alpha)$.

If α is a set of atoms and $\tau \in \{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$, we will use $\{\alpha \to \tau\}$ to denote the *constant truth valuation* over α, which maps every atom $X \in \alpha$ to τ. If $\alpha = \{A\}$ is a singleton set, we will simply write $\{A \to \tau\}$ instead of $\{\{A\} \to \tau\}$.

Algorithm 1 computes answers to the query $?- g$, i.e., the degree of belief in g. The algorithm is based on the same idea as the fixpoint semantics: for every node X in the post-order traversal, the algorithm first expands m from the base $Base$ to a larger base $Base \cup \{X\}$. The difference is that Algorithm 1 then projects m to a smaller base by throwing out the nodes which are no longer needed for the rest of the computation. Each r-node ID_R represents the statement that the body of rule R is true and is used to facilitate the computation.

Theorem 2 (Correctness of Algorithm 1). *Given a ground blp $\boldsymbol{P}$ and a ground query $?- g$, let* `model` *be the model of $\boldsymbol{P}$ and* `bel`(g) *be the outcome of Algorithm 1. The algorithm is correct in the sense that* `bel`$(g) =$ `model`(g). □

Example 2. Algorithm 1 does not specify any concrete post-order traversal to use. One possible order of traversal for the proof DAG in Figure 1 is: $\langle r5, c, r4, b, r1, a, r2, r6, d, r3, e, rg, g\rangle$. With this traversal, the base of the support function

Algorithm 1. Query answering by traversing proof DAG

Input: A ground blp $\boldsymbol{P}$, a ground query $? - \ g$, and the proof DAG, $\mathcal{H}$, of $\boldsymbol{P}$ and of the query $? - \ g$.

Output: The answer to the query $? - \ g$, i.e., the degree of belief in g.

Let $X_1, ..., X_n$ be a post-order traversal of $\mathcal{H}$.
$m \Leftarrow m_{\varnothing}$; $Visited \Leftarrow \varnothing$; $Base \Leftarrow \varnothing$;
for $i = 1$ to n **do**
  if X is an a-node labeled with an atom A **then**
    Initialize a new support function m' for $Base \cup \{A\}$.
    Let $Id_1, \ldots, Id_k$ be the labels of the child r-nodes of X.
    for all $I \in \mathcal{TV}al(Base)$ such that $m(I) \neq 0$ **do**
      $\mathcal{S} \Leftarrow \varnothing$;
      for $j = 1$ to k **do**
        if $I(Id_j) = \mathbf{t}$ **then**
          Let $[V_j, W_j]$ be the belief factor of the rule identified by Id_j.
          $\mathcal{S} \Leftarrow \mathcal{S} \cup \{[V_j, W_j]\}$;
        end if
      end for
      $[V, W] \Leftarrow \Phi_A(\mathcal{S})$;
      $m'(I \uplus \{A \rightarrow \mathbf{t}\}) \Leftarrow m(I) \cdot V$;
      $m'(I \uplus \{A \rightarrow \mathbf{f}\}) \Leftarrow m(I) \cdot (1 - W)$;
      $m'(I \uplus \{A \rightarrow \mathbf{u}\}) \Leftarrow m(I) \cdot (W - V)$;
    end for
    $m' \Leftarrow projection(m', Base \cup \{A\} - \{Id_1, \ldots, Id_k\})$;
    $m \Leftarrow m'$; $Visited \Leftarrow Visited \cup \{A\}$; $Base \Leftarrow Base \cup \{A\} - \{Id_1, \ldots, Id_k\}$;
  else /* X is an r-node labeled with Id which is the identifier of rule R. */
    Initialize a new support function m' for $Base \cup \{Id\}$.
    Let $A_1, \ldots, A_k$ be the labels of the child a-nodes of X.
    Let F be the body of R.
    for all $I \in \mathcal{TV}al(Base)$ such that $m(I) \neq 0$ **do**
      for all $\tau \in \{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$ **do**
        if $\tau = I(F)$ **then**
          $m'(I \uplus \{Id \rightarrow \tau\}) \Leftarrow m(I)$;
        end if
      end for
    end for
    $\alpha \Leftarrow \varnothing$;
    for $i = 1$ to k **do**
      if every parent node of A_i is in $Visited$ **then**
        $\alpha \Leftarrow \alpha \cup \{A_i\}$;
      end if
    end for
    $m' \Leftarrow projection(m', Base \cup \{Id\} - \alpha)$;
    $m \Leftarrow m'$; $Visited \Leftarrow Visited \cup \{Id\}$; $Base \Leftarrow Base \cup \{Id\} - \alpha$;
  end if
end for
return $\sum_{I \in \mathcal{TV}al(Base), I(g) = \mathbf{t}} m(I)$

m changes during the runtime of Algorithm 1 in the following order: $\{r5\}$, $\{c\}$, $\{c, r4\}$, $\{c, b\}$, $\{c, b, r1\}$, $\{c, b, a\}$, $\{b, r2\}$, $\{b, r2, r6\}$, $\{b, r2, d\}$, $\{b, r2, r3\}$, $\{b, e\}$, $\{rg\}$, $\{g\}$. Suppose $\Phi_e = \Phi^{DS}$, the output of Algorithm 1 is $\texttt{bel}(g) = 0.3$. □

4.2 Complexity

First we derive complexity bounds for the unoptimized fixpoint computation in Section 3. It is easy to see that the space complexity is bounded by the domain size of the final and largest support function, which is $O(3^{|B_P|})$. For the time complexity, consider the operator $\hat{T}_{\boldsymbol{P}}$ from Definition 9. During the execution of the fixpoint semantics, each rule is processed exactly once by the operator $\hat{T}_{\boldsymbol{P}}$, and the computation is dominated by evaluating the rule body under each truth valuation over the base of the current support function. Therefore, the time complexity of processing a rule R is bounded by $O(|Body(R)| \cdot 3^{|B_P|})$, where $|Body(R)|$ denotes the size of the body of R, and the overall complexity of the fixpoint semantics is $O(\sum_{R \in \boldsymbol{P}} |Body(R)| \cdot 3^{|B_P|})$.

The complexity of Algorithm 1 is much lower. First of all, the space complexity is bounded by the domain size of the largest support function. Let M be the maximum among the sizes of support function bases constructed during the run of the algorithm. Then space complexity is $O(3^M)$. The time needed to construct the proof DAG $\mathcal{H}$ for a query is the same as the time needed to find all the proof paths in classical deductive databases (with certainty factors removed): it is polynomial in the size of the data. Then the algorithm performs a post-order traversal of $\mathcal{H}$. At each step, when visiting a node, X, it tries to expand the support function $m_{\mathcal{S}}$ from the base $\mathcal{S}$ to $\mathcal{S} \cup \{X\}$. By the same argument as in the previous paragraph, the time complexity for processing each rule R is bounded by $O(|Body(R)| \cdot 3^M)$. Therefore, since each rule is processed by exactly one step of the algorithm, the overall time complexity is bound by $O(\sum_{R \in \mathcal{T}} |Body(R)| \cdot 3^M)$, where $\mathcal{T}$ is the set of rules that appear in $\mathcal{H}$.

It is clear from the query answering algorithm that, compared to the stock fixpoint computation, it requires less space in a typical run by "garbage-collecting" atoms in the base of the support function. It is less clear where the time savings come from. The two time complexity bounds estimates above indicate that the exponent has dominant effect. For instance, in Example 2, the fixpoint computation yields the exponent $|B_{\boldsymbol{P}}| = 6$. In contrast, the exponent for the query answering algorithm is $M = 3$.

It is interesting to note that different traversals of the proof DAG might generate support function bases whose maximum sizes vary greatly. An important issue is, therefore, finding heuristics that produce smaller bases.

4.3 Non-ground Cases

Now we turn to non-ground programs and non-ground queries.

A **non-ground query** to a blp $\boldsymbol{P}$ is a statement of the form $? - \mathit{Goal}$, where *Goal* is a Boolean combination of non-ground atoms. A ground instance *goal* of *Goal* is called a **positive instance** of *Goal* if *goal* is entailed by $\boldsymbol{P}$ with a

positive belief. An **answer to non-ground query** $? - \ Goal$ is any positive instance *goal* of *Goal*, together with the belief in *goal*. Using the same trick as for ground queries, we can make the assumption that all non-ground queries are simply singleton non-ground atoms.

The main problem in answering queries to non-ground blps is how to construct proof DAGs without grounding the whole program and constructing the whole dependency DAG. It turns out that a procedure in the style of SLD-resolution can solve this problem and even construct a *pruned* proof DAG which does not contain atom that is not supported at all or rules that never fires.

First, recall that SLD-resolutions can only be applied to programs without body-disjunctions, while blps might have such disjunctions. Fortunately, every blp can be transformed into an equivalent blp without body-disjunctions.

Definition 14. *For any blp* $\boldsymbol{P}$*, its* ***non-disjunctive equivalent****,* $nodisjunct(\boldsymbol{P})$*, is obtained from* $\boldsymbol{P}$ *by replacing every rule* R *of the form*

$$[v, w]\ H \ \texttt{:-}\ Body_1 \vee \cdots \vee Body_k.$$

where $k > 1$ *and* $Body_i$ $(1 \leq i \leq k)$ *are conjunctions of literals, with a set of rules of the form*

$$\begin{array}{ll} [v, w]\ H & \texttt{:-}\ H_R. \\ [1, 1]\ H_R & \texttt{:-}\ Body_i. \quad 1 \leq i \leq k \end{array}$$

where H_R *is a new predicate that does not appear anywhere else in* $nodisjunct(\boldsymbol{P})$ *and* H_R *has the same arguments (variables) as* H *does.* □

Theorem 3. *Let* $\boldsymbol{P}$ *be a blp,* $\texttt{bel}_1$ *be the model of* $\boldsymbol{P}$*, and* $\texttt{bel}_2$ *be the model of* $nodisjunct(\boldsymbol{P})$*. If all combination functions in* $\boldsymbol{P}$ *have the property* $\Phi([1, 1], [1, 1]) = [1, 1]$ *then* $\texttt{bel}_1(F) = \texttt{bel}_2(F)$ *for all* $F \in \mathcal{B}ool(B_{\boldsymbol{P}})$*.* □

In other words, the models of $\boldsymbol{P}$ and $nodisjunct(\boldsymbol{P})$ coincide where it matters. *From now on we assume that blps do not have disjunctions in rule bodies.*

Definition 15. *The* ***pruned proof DAG*** *of a ground blp* $\boldsymbol{P}$ *for a ground query* $? - \ g$ *is obtained from the proof DAG of* $\boldsymbol{P}$ *for* $? - \ g$ *by the following steps:*

- *Mark all leaf a-nodes as "failed" and all leaf r-nodes as "non-failed".*
- *Perform a post-order traversal to mark every node as follows:*
  - *If an a-node has at least one child r-node which is marked "non-failed," mark the a-node as "non-failed;" otherwise, mark the a-node as "failed."*
  - *If an r-node has at least one child a-node which is marked "failed," mark the r-node as "failed;" otherwise, mark the r-node as "non-failed."*
- *Delete all the nodes which are marked as "failed."*
- *Delete all the nodes which are not connected to the a-node* g*.* □

It is clear that pruned proof DAG contains all the necessary information to answer $? - \ g$, and executing Algorithm 1 on pruned proof DAGs eliminates unnecessary computation compared to running it without pruning the DAGs.

Now we define SLD-trees on *non-ground* programs and queries, which we will use in Algorithm 2 to construct the pruned proof DAGs.

Definition 16. *An **SLD-tree** for a blp $\boldsymbol{P}$ and a query $? - G$ (both $\boldsymbol{P}$ and G can be non-ground) is constructed by the following steps:*

- *Add a node labeled with $? - G$ as root.*
- *Repeat the following until no nodes can be added:*
  Let o be a leaf node labeled with $? - G_1 \wedge ... \wedge G_n$, for every rule R in $\boldsymbol{P}$ of the form $[v, w]$ A :- $B_1 \wedge ... \wedge B_m$ such that G_1 and A or G_1 and $\overline{A}$ unify with the most general unifier (mgu) θ,
  - *Add a node o' labeled with $? - (B_1 \wedge ... \wedge B_m \wedge G_2 \wedge ... \wedge G_n)\theta$;*
  - *Add an edge from o to o' and label that edge with a triple $\langle \theta, ID_R, Vars\theta \rangle$, where ID_R is the identifier of R and $Vars$ is the list of variables in R.*

  In a special case when o above is labeled with $? - G_1$ (has only one atom) and R above is a fact that unifies with G_1 or $\overline{G_1}$, o' becomes a leaf node labeled with $? - \{\}$. We call it a success node.
- *Delete the nodes and edges that are not connected to any success node.* □

Using the SLD-tree for $? - G$, Algorithm 2 constructs the set of pruned proof DAGs for the grounding of $\boldsymbol{P}$ and each query $? - g$, where g is a positive instance of G. The query $? - G$ can then be answered by applying Algorithm 1 on the output of Algorithm 2.

Algorithm 2. Construct pruned proof DAGs from SLD-tree

Input: An SLD-tree, $\mathcal{T}$, of $\boldsymbol{P}$ and of the query $? - G$.
Output: *Set*, the set of pruned proof DAGs for the grounding of $\boldsymbol{P}$ and each query $? - g$, where g is a positive instance of G.

1: $\mathcal{HS} \Leftarrow \varnothing$; $\Theta \Leftarrow \varnothing$;
2: **for all** path, *path*, in $\mathcal{T}$ from root to a leaf node labeled $? - \{\}$ **do**
3: Let θ be the composition of all the mgu's labeled along *path*, $\Theta \Leftarrow \Theta \cup \{\theta\}$;
4: **for all** edge e along *path* **do**
5: Let ID_R be the rule identifier labeled on e and R the corresponding rule.
6: Let $Vars$ be the variable list labeled on e.
7: $Insert(\mathcal{HS}$, an r-node labeled with $ID_R(Vars\theta))$;
/* *Insert* means insert a node or an edge if it is not already in the graph */
8: $Insert(\mathcal{HS}$, an a-node labeled with $head(R)\theta)$;
9: $Insert(\mathcal{HS}, edge(ID_R(Vars\theta), head(R)\theta))$;
10: label the new edge with the belief factor of R.
11: **for all** atom A in the body of R **do**
12: $Insert(\mathcal{HS}$, an a-node labeled with $A\theta)$;
13: $Insert(\mathcal{HS}, edge(A\theta, ID_R(Vars\theta)))$;
14: **end for**
15: **end for**
16: **end for**
17: $Set \Leftarrow \{\mathcal{H} \mid \exists \theta \in \Theta,\ \mathcal{H}$ is a subgraph of $\mathcal{HS}$ which contains all the nodes that are connected to a node labeled with $g\theta\}$
18: **return** *Set*

Example 3. Consider a system that assesses the risk for people getting a certain disease. The relevant inference rules and facts can be expressed as follows:

$@r1\ [0.2,1]\ \ disease(X)\ \texttt{:-}\ little\_sports(X).$
$@r2\ [0.1,1]\ \ disease(X)\ \texttt{:-}\ favorite(X,Y), unhealthy(Y).$

$@r3$	$[1,1]$	$little\_sports(p1).$			
$@r4$	$[1,1]$	$favorite(p1,a).$	$@r5$	$[1,1]$	$favorite(p1,b).$
$@r6$	$[1,1]$	$favorite(p1,c).$	$@r7$	$[1,1]$	$favorite(p2,a).$
$@r8$	$[1,1]$	$favorite(p2,d).$	$@r9$	$[1,1]$	$favorite(p2,e).$
$@r10$	$[1,1]$	$favorite(p3,b).$	$@r11$	$[1,1]$	$favorite(p3,f).$
$@r12$	$[1,1]$	$unhealthy(a).$	$@r13$	$[1,1]$	$unhealthy(d).$

Figure 2 shows an SLD-tree for this program and the query $? -\ disease(X)$. The pruned proof DAGs are shown in Figure 3. Suppose $\Phi_{disease} = \Phi^{DS}$. Then the result is $\{\langle disease(p1), \texttt{belief} : 0.28\rangle, \langle disease(p2), \texttt{belief} : 0.19\rangle\}$. □

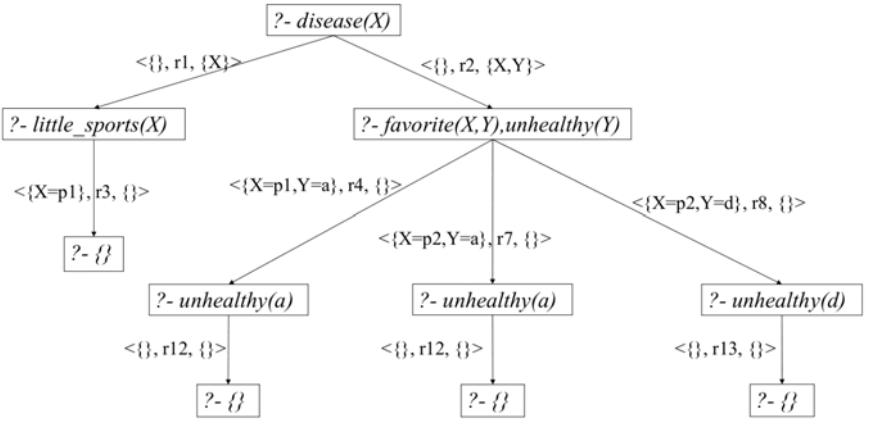


Fig. 2. SLD-tree for Example 3

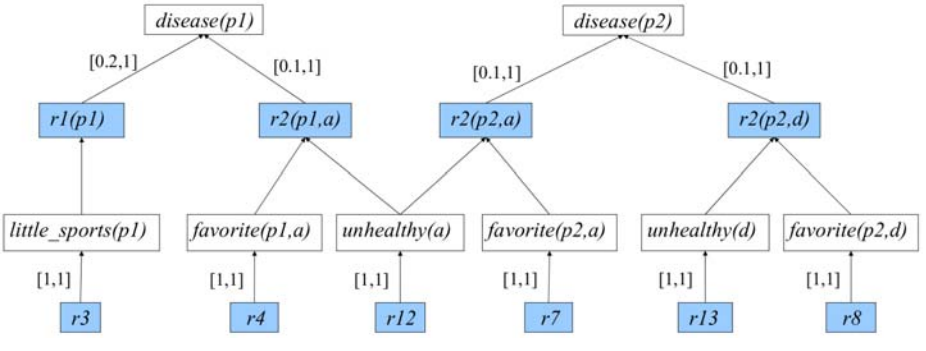


Fig. 3. Pruned proof DAGs for Example 3

5 Conclusions

We presented a fixpoint semantics for Belief Logic Programming (BLP) and optimized algorithms for answering general BLP queries. The current version of the BLP semantics does not permit recursion at the ground level, which is left as future work. Another interesting future direction is combining BLP with annotated logic programming [7].

References

1. Baldwin, J.F.: Evidential support logic programming. Fuzzy Sets and Systems 24(1), 1–26 (1987)
2. Dekhtyar, A., Subrahmanian, V.S.: Hybrid probabilistic programs. J. of Logic Programming 43, 391–405 (1997)
3. Dempster, A.P.: Upper and lower probabilities induced by a multi-valued mapping. Ann. Mathematical Statistics 38 (1967)
4. Dubois, D., Lang, J., Prade, H.: Possibilistic logic. In: Handbook of logic in artificial intelligence and logic programming, nonmonotonic reasoning and uncertain reasoning, vol. 3, pp. 439–513. Oxford University Press, Inc., Oxford (1994)
5. Grosof, B.N.: A courteous compiler from generalized courteous logic programs to ordinary logic programs. Technical Report Supplementary Update Follow-On to RC 21472, IBM (July 1999)
6. Kifer, M., Li, A.: On the semantics of rule-based expert systems with uncertainty. In: Gyssens, M., Van Gucht, D., Paredaens, J. (eds.) ICDT 1988. LNCS, vol. 326, pp. 102–117. Springer, Heidelberg (1988)
7. Kifer, M., Subrahmanian, V.S.: Theory of generalized annotated logic programming and its applications. J. of Logic Programming 12(3,4), 335–367 (1992)
8. Lakshmanan, L.V.S., Shiri, N.: A parametric approach to deductive databases with uncertainty. IEEE T. Knowledge and Data Engineering 13(4), 554–570 (2001)
9. Lukasiewicz, T.: Probabilistic logic programming with conditional constraints. ACM Trans. on Computational Logic 2(3), 289–339 (2001)
10. Muggleton, S.: Learning stochastic logic programs. Electron. Trans. Artif. Intell. 4(B), 141–153 (2000)
11. Ng, R.T.: Reasoning with uncertainty in deductive databases and logic programs. Intl. J. of Uncertainty,Fuzziness and Knowledge-Based Sys. 5(3), 261–316 (1997)
12. Ng, R.T., Subrahmanian, V.S.: A semantical framework for supporting subjective probabilities in deductive databases. J. of Automated Reasoning 10(2), 191–235 (1993)
13. Pearce, D., Wagner, G.: Logic programming with strong negation. In: Proceedings of the international workshop on Extensions of logic programming, pp. 311–326. Springer-Verlag New York, Inc., New York (1991)
14. Poole, D.: The independent choice logic and beyond. In: Probabilistic Inductive Logic Programming, pp. 222–243 (2008)
15. De Raedt. L, Kersting, K.: Probabilistic inductive logic programming. In: Probabilistic inductive logic programming, pp. 1–27 (2008)
16. Saad, E., Pontelli, E.: Towards a more practical hybrid probabilistic logic programming framework. In: Hermenegildo, M.V., Cabeza, D. (eds.) PADL 2004. LNCS, vol. 3350, pp. 67–82. Springer, Heidelberg (2005)

17. Sato, T., Kameya, Y.: Parameter learning of logic programs for symbolic-statistical modeling. J. of Artificial Intelligence Research 15, 391–454 (2001)
18. Shafer, G.: A Mathematical Theory of Evidence. Princeton University Press, Princeton (1976)
19. Shenoy, P.P., Shafer, G.: Axioms for probability and belief-function propagation. In: Uncertainty in Artificial Intelligence, pp. 169–198. North-Holland, Amsterdam (1990)
20. Shortliffe, E.H.: Computer Based Medical Consultations: MYCIN. American Elsevier, Amsterdam (1976)
21. Subrahmanian, V.S.: On the semantics of quantitative logic programs. In: Symposium on Logic Programming (SLP), pp. 173–182 (1987)
22. Wan, H., Grosof, B.N., Kifer, M., Fodor, P., Liang, S.: Logic programming with defaults and argumentation theories. In: Intl. Conf. on Logic Programming (2009)
23. Wan, H., Kifer, M.: Belief logic programming: Uncertainty reasoning with correlation of evidence. In: Intl. Conf. on Logic Programming and Nonmonotonic Reasoning, LPNMR (2009)
24. Zadeh, L.A.: Fuzzy sets. Information Control 8, 338–353 (1965)

Towards Effective Elicitation of NIN-AND Tree Causal Models

Yang Xiang, Yu Li, and Zoe Jingyu Zhu

University of Guelph, Canada

Abstract. To specify a Bayes net (BN), a conditional probability table (CPT), often of an effect conditioned on its n causes, needs assessed for each node. It generally has the complexity exponential on n. Noisy-OR reduces the complexity to linear, but can only represent reinforcing causal interactions. The non-impeding noisy-AND (NIN-AND) tree is the first causal model that explicitly expresses reinforcement, undermining, and their mixture. It has linear complexity, but requires elicitation of a tree topology for types of causal interactions. We study their topology space and develop two novel techniques for more effective elicitation.

1 Introduction

To specify a BN, a CPT needs to be assessed for each non-root node. It is often advantageous to construct BNs along the causal direction, in which case a CPT is the distribution of an effect conditioned on its n causes. In general, assessment of a CPT has the complexity exponential on n.

Noisy-OR [7] is the most well known technique that reduces this complexity to linear. A number of extensions have also been proposed such as [4,3,5]. However, noisy-OR, noisy-AND [3], as well as related techniques, can only represent causal interactions that are reinforcing [9]. The NIN-AND tree [9] extends noisy-OR and provides the first causal model that explicitly expresses reinforcing and undermining causal interactions, as well as their mixture. It requires elicitation of a linear number of probability parameters and, in addition, a tree topology which specifies the types of causal interactions among causes.

The elicitation relies on expert to describe the tree topology. When the number of causes is more than four or five, accurate description may be challenging. We study the topology space of NIN-AND tree models and develop novel techniques for more effective elicitation. One allows expert to select a topology from an enumeration. Another allows expert to specify only types of pairwise interactions among causes, from which a unique tree topology is identified.

2 Background

This section is mostly based on [9]. An *uncertain cause* is a cause that can produce an effect but does not always do so. Denote a set of binary cause variables as $X = \{c_1, ..., c_n\}$ and their effect variable (binary) as e. For each c_i, denote

L. Godo and A. Pugliese (Eds.): SUM 2009, LNAI 5785, pp. 282–296, 2009.

$c_i = true$ by c_i^+ and $c_i = false$ by c_i^-. Similarly, denote $e = true$ by e^+ and $e = false$ by e^-.

A *causal event* refers to an event that a cause c_i caused an effect e to occur successfully when all other causes of e are absent. Denote this causal event by $e^+ \leftarrow c_i^+$ and its probability by $P(e^+ \leftarrow c_i^+)$. The causal failure event, where e is false when c_i is true and all other causes of e are false, is denoted as $e^+ \not\leftarrow c_i^+$. Denote the causal event that a set $X = \{c_1, ..., c_n\}$ of causes caused e by $e^+ \leftarrow c_1^+, ..., c_n^+$ or $e^+ \leftarrow \underline{x}^+$. Denote the set of *all causes* of e by C. The CPT $P(e|C)$ relates to probabilities of causal events as follows: If $C = \{c_1, c_2, c_3\}$, then $P(e^+|c_1^+, c_2^-, c_3^+) = P(e^+ \leftarrow c_1^+, c_3^+)$. Note that C is assumed to include a leaky variable (if any) to capture causes that we do not wish to represent explicitly, and hence $P(e^+|c_1^-, c_2^-, c_3^-) = 0$.

Causes reinforce each other if collectively they are at least as effective in causing the effect as some acting by themselves. If collectively they are less effective, then they undermine each other. Note that if $C = \{c_1, c_2\}$ and c_1 and c_2 undermine each other, then all the following hold:

$$P(e^+|c_1^-, c_2^-) = 0, \quad P(e^+|c_1^+, c_2^-) > 0, \quad P(e^+|c_1^-, c_2^+) > 0,$$

$$P(e^+|c_1^+, c_2^+) < min(P(e^+|c_1^+, c_2^-), P(e^+|c_1^-, c_2^+)).$$

The following Def.1 defines the two types of causal interactions generally. Note that reinforcement and undermining occur between individual variables as well as sets of variables. For instance, variables within each of two sets can be reinforcing, while the two sets can undermine each other. Hence, each W_i in Def. 1 is not necessarily a singleton.

Def. 1. *Let $R = \{W_1, W_2, ...\}$ be a partition of a set X of causes, $R' \subset R$ be any proper subset of R, and $Y = \cup_{W_i \in R'} W_i$. Sets of causes in R reinforce each other, iff*

$$\forall R' \; P(e^+ \leftarrow \underline{y}^+) \leq P(e^+ \leftarrow \underline{x}^+).$$

Sets of causes in R undermine each other, iff

$$\forall R' \; P(e^+ \leftarrow \underline{y}^+) > P(e^+ \leftarrow \underline{x}^+).$$

Disjoint sets of causes $W_1, ..., W_m$ satisfy *failure conjunction* iff

$$(e^+ \not\leftarrow \underline{w}_1^+, ..., \underline{w}_m^+) = (e^+ \not\leftarrow \underline{w}_1^+) \wedge ... \wedge (e^+ \not\leftarrow \underline{w}_m^+).$$

That is, collective failure is attributed to individual failures. They also satisfy *failure independence* iff

$$P((e^+ \not\leftarrow \underline{w}_1^+) \wedge ... \wedge (e^+ \not\leftarrow \underline{w}_m^+)) = P(e^+ \not\leftarrow \underline{w}_1^+) \; ... \; P(e^+ \not\leftarrow \underline{w}_m^+).$$

Disjoint sets of causes $W_1, ..., W_m$ satisfy *success conjunction* iff

$$e^+ \leftarrow \underline{w}_1^+, ..., \underline{w}_m^+ = (e^+ \leftarrow \underline{w}_1^+) \wedge ... \wedge (e^+ \leftarrow \underline{w}_m^+).$$

That is, collective success requires individual effectiveness. They also satisfy *success independence* iff

$$P((e^+ \leftarrow \underline{w}_1^+) \wedge ... \wedge (e^+ \leftarrow \underline{w}_m^+)) = P(e^+ \leftarrow \underline{w}_1^+) \ ... \ P(e^+ \leftarrow \underline{w}_m^+).$$

It has been shown that causes are reinforcing when they satisfy failure conjunction and independence, and they are undermining when they satisfy success conjunction and independence. Hence, undermining can be modeled by a direct

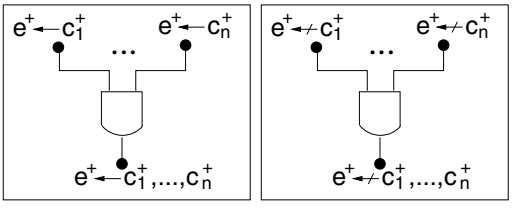


Fig. 1. Direct (left) and dual (right) NIN-AND gates

NIN-AND gate (Fig. 1, left), and reinforcement by a dual NIN-AND gate (right).

As per Def. 1, a set of causes can be reinforcing (undermining), but the set is undermining (reinforcing) with another set. Such causal interaction can be modeled by a NIN-AND tree. As shown in Fig. 2 (a), causes c_1 through c_3 are undermining, and they are collectively reinforcing c_4. The following defines NIN-AND tree models in general:

Def. 2. *An* `NIN-AND tree` *is a directed tree for effect* e *and a set* $X = \{c_1, ..., c_n\}$ *of occurring causes.*

1. *There are two types of nodes. An* `event` *node (a black oval) has an in-degree* ≤ 1 *and an out-degree* ≤ 1. *A* `gate` *node (a NIN-AND gate) has an in-degree* ≥ 2 *and an out-degree* 1.
2. *There are two types of links, each connecting an event and a gate along input-to-output direction of gates. A* `forward` *link (a line) is implicitly directed. A* `negation` *link (with a white oval at one end) is explicitly directed.*
3. *Each terminal node is an event labeled by a causal event* $e^+ \leftarrow \underline{y}^+$ *or* $e^+ \not\leftarrow \underline{y}^+$. *There is a single* `leaf` *(no child) with* $\underline{y}^+ = \underline{x}^+$, *and the gate it connects to is the* `leaf gate`. *For each* `root` *(no parent; indexed by* i*),* $\underline{y}_i^+ \subset \underline{x}^+$, $\underline{y}_j^+ \cap \underline{y}_k^+ = \emptyset$ *for* $j \neq k$, *and* $\bigcup_i \underline{y}_i^+ = \underline{x}^+$.
4. *Inputs to a gate* g *are in one of two cases:*
   (a) *Each is either connected by a forward link to a node labeled* $e^+ \leftarrow \underline{y}^+$, *or by a negation link to a node labeled* $e^+ \not\leftarrow \underline{y}^+$. *The output of* g *is connected by a forward link to a node labeled* $e^+ \leftarrow \cup_i \underline{y}_i^+$.
   (b) *Each is either connected by a forward link to a node labeled* $e^+ \not\leftarrow \underline{y}^+$, *or by a negation link to a node labeled* $e^+ \leftarrow \underline{y}^+$. *The output of* g *is connected by a forward link to a node labeled* $e^+ \not\leftarrow \cup_i \underline{y}_i^+$.

An NIN-AND tree model for effect e and its causes C can be obtained by eliciting a tree topology with $|C|$ roots plus $|C|$ single-cause probabilities $P(e^+ \leftarrow c_i^+)$. The CPT $P(e|C)$ can then be derived using the model. For each $P(e^+|c_i, ..., c_j)$, first modify the model to remove roots corresponding to inactive causes, i.e., $c_i = c_i^-$, and related gates if necessary. Then apply algorithm GetCausalEventProb below to the modified tree. It recursively processes from the leaf to roots. As soon as probabilities of input events to a gate is obtained, probability of its output event is computed.

Algorithm 1. *GetCausalEventProb(T)*

Input: A NIN-AND tree T of leaf v and leaf gate g, with root probabilities specified.

for each node w directly inputting to g, do
  if $P(w)$ is not specified,
    denote the sub-NIN-AND-tree with w as the leaf by T_w;
    $P(w) = GetCausalEventProb(T_w)$;
  if (w, g) is a forward link, $P'(w) = P(w)$;
  else $P'(w) = 1 - P(w)$;
return $P(v) = \prod_w P'(w)$;

By default, each root event in a NIN-AND tree is a single-cause event, and all causal interactions satisfy failure (or success) conjunction and independence. If a subset of causes do not satisfy these assumptions, suitable multi-cause probabilities $P(e^+ \leftarrow \underline{x}^+)$, where $X \subset C$, can be directly elicited and incorporated into the NIN-AND tree model. Hence, by trading efficiency, any non-deterministic CPT can be encoded through NIN-AND trees. The default is assumed in this paper.

3 Minimal NIN-AND Tree Topology Space

NIN-AND tree models allow a CPT of generally exponential complexity to be obtained by eliciting a tree topology and a linear number of probabilities of single-cause events. Reference [9] relies on human expert to describe the tree topology. One alternative is to show expert all possible tree topologies so that one can be selected. We study the space of NIN-AND trees below so that tree topologies can be enumerated.

First of all, what qualifies as an individual in the space? For instance, it would be undesirable that two distinct topologies in the space correspond to the same CPT. Consider the two NIN-AND trees in Fig. 2. Although the topologies appear different, given an identical set of single-cause probabilities, they yield the same probability $P(e^+ \not\leftarrow c_1^+, c_2^+, c_3^+, c_4^+)$. Hence, it is desirable that only one of them is deemed legal in the topology space.

We establish below associativity of NIN-AND gates, which allows identification of equivalent topologies such as the above. Proposition 1 shows that an NIN-AND tree of multiple dual NIN-AND gates is equivalent to a single gate.

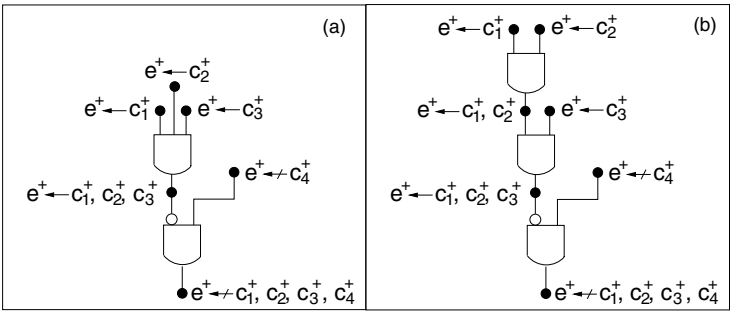


Fig. 2. NIN-AND trees depicting the same causal model

Proposition 1. *Let T be an NIN-AND tree for $n \geq 3$ causes with $m \geq 1$ dual NIN-AND gates, and $P(e^+ \leftarrow c_1^+, ..., c_n^+)$ be the probability of the causal event obtained from T. Let T' be a single dual NIN-AND gate for the same causes and $P'(e^+ \leftarrow c_1^+, ..., c_n^+)$ the probability from T'. Then $P(e^+ \leftarrow c_1^+, ..., c_n^+) = P'(e^+ \leftarrow c_1^+, ..., c_n^+)$.*

Proof: We prove by induction on the number of gates in T. For $m = 1$, T and T' are identical. Applying GetCausalEventProb to T, we obtain

$$P(e^+ \not\leftarrow c_1^+, ..., c_n^+) = P'(e^+ \not\leftarrow c_1^+, ..., c_n^+) = \prod_{i=1}^{n} P(e^+ \not\leftarrow c_i^+).$$

Assume that the proposition holds for $m = 1, 2, ..., k$ where $k \geq 1$. Below we consider $m = k + 1$.

Let g denote the leaf gate of T. Since $k \geq 1$ and $m \geq 2$, there exists a gate t that outputs to g. Let v be the output event of t. Let S denote the subtree seated at v and for a subset X of causes. In other words, v is the leaf in S. S is a valid NIN-AND tree with no more than k gates. By inductive assumption, applying GetCausalEventProb to S, we obtain

$$P(e^+ \not\leftarrow \underline{x}^+) = \prod_{c_i \in X} P(e^+ \not\leftarrow c_i^+).$$

The above argument holds for each gate t. When GetCausalEventProb is applied to the leaf node in T, the probability of the corresponding event is the product of the probability of each input event to g. Hence,

$$P(e^+ \not\leftarrow c_1^+, ..., c_n^+) = \prod_{i=1}^{n} P(e^+ \not\leftarrow c_i^+)$$

when $m = k + 1$. □

Proposition 2 shows that an NIN-AND tree of multiple direct NIN-AND gates is equivalent to a single direct NIN-AND gate. It can be proven similarly as for Proposition 1.

Proposition 2. *Let T be an NIN-AND tree for $n \geq 3$ causes with $m \geq 1$ direct NIN-AND gates, and $P(e^+ \leftarrow c_1^+, ..., c_n^+)$ be the probability of the causal event obtained from T. Let T' be an NIN-AND tree for the same causes with a single direct NIN-AND gate, and $P'(e^+ \leftarrow c_1^+, ..., c_n^+)$ be the probability from T'. Then $P(e^+ \leftarrow c_1^+, ..., c_n^+) = P'(e^+ \leftarrow c_1^+, ..., c_n^+)$.*

Base on the associativity of NIN-AND gates, we select the single NIN-AND gate to represent all equivalent NIN-AND trees of the same input events, and to be the only legal individual in the topology space. Applying this to NIN-AND gates embedded in an NIN-AND tree, we have the following classification of tree topologies.

Def. 3. *Let T be an NIN-AND tree. If T contains a gate t that outputs to another gate g of the same type (direct or dual), delete t and connect its inputs to g. If such deletion is possible, then T is* `superfluous`*. Apply such deletions until no longer possible. The resultant NIN-AND tree is* `minimal`*.*

As per Def. 3, we require individuals in the topology space to be minimal. That is, we require a *minimal* topology space. Hence, the NIN-AND tree in Fig. 2 (a) is legal in the space, and that in (b) is not. This leads to Corollary 1.

Corollary 1. *Let T be a minimal NIN-AND tree. Then whenever a NIN-AND gate g outputs to another NIN-AND gate t, g and t are of different type (direct or dual).*

From Corollary 1, a minimal NIN-AND tree has the following structure: If the leaf gate g is a direct gate, then all gates outputting to g are dual, and their inputs are all from direct gates. That is, from the leaf towards root nodes, gates alternate in types. This alternation implies that, in the minimal space, for every legal NIN-AND tree T with a direct leaf gate, there exists a legal NIN-AND tree T' obtained by replacing each gate in T with its opposite type. This is summarized in the following:

Proposition 3. *Let Ψ be the collection of minimal NIN-AND trees for n causes with direct leaf gates. Let Ψ' be the collection of minimal NIN-AND trees for n causes with dual leaf gates. Then an one-to-one mapping exists between Ψ and Ψ', defined by replacing each gate with the opposite type.*

We refer to Ψ as the minimal topology space for n causes with direct leaf gates, and Ψ' as the minimal topology space with dual leaf gates. As per Proposition 3, properties from one of them are applicable to the other. Below, we focus on tree enumeration in the minimal space with direct leaf gates.

From Corollary 1, given the type of leaf gate, types of all gates in a minimal NIN-AND tree are unique, as well as the nature of all event nodes (causal failure or success). We thus omit labels for event nodes. Note that we choose minimal trees to be *unlabeled* as the space is more compact. How to enumerate root-labeled minimal trees given the unlabeled enumeration is studied in [10].

4 Enumerating NIN-AND Trees by Local Insertion

There is only one minimal NIN-AND tree for 2 causes. For 3 causes, there are two minimal NIN-AND trees with direct leaf gate (see Fig. 3 (a) and (b)). For a larger number of causes, Theorem 1 suggests operations for automatic tree topology generation, which are illustrated in Fig. 3 (c) and (d).

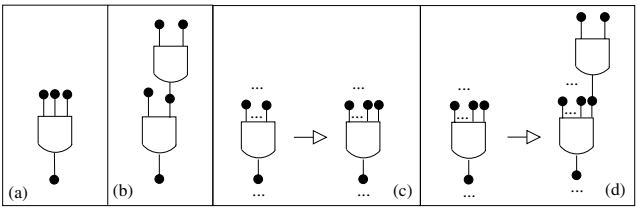


Fig. 3. (a) and (b): Minimal NIN-AND trees for 3 causes. (c) Op1. (d) Op2.

Theorem 1. *Every minimal NIN-AND tree for $n \geq 3$ causes can be constructed by starting from an NIN-AND gate for* 2 *causes and applying a sequence of operations made of the following two:*

Op1. *Add a root event as an input to a gate.*
Op2. *Insert a new gate between a root event and an existing gate and add a new root event as the second input of the new gate.*

Proof by induction (sketch): For $n = 3$, there are exactly two minimal NIN-AND trees. The trees in Fig. 3 (a) and (b) can be constructed by applying Op1 and Op2 to the NIN-AND gate for 2 causes.

Assume that the theorem holds for $n = k \geq 3$ and consider $n = k + 1$. Let T' be a minimal tree for n causes, x be a root event in T' and is connected to a gate g. Let T be a tree obtained by removing x from T'. Analyzing the following cases and applying the inductive assumption to T, it can be shown that, in each case, T' can be constructed by a sequence made of Op1 and Op2.

- Gate g has three or more input events.
- Gate g has two input events x and y.
  - Gate g is the leaf gate.
  - Gate g is a non-leaf gate.
    - Event y is a root event.
    - Event y is a non-root event. □

Theorem 1 suggests the following procedure to generate all minimal NIN-AND trees for n causes: Start with the NIN-AND gate for 2 causes. Apply Op1 and Op2 to the gate to generate the two NIN-AND trees for 3 causes. Repeat the process to generate all minimal trees for 4 causes based on minimal trees for 3 causes, then those for 5 causes based on minimal trees for 4 causes, and so on,

until those for n causes based on minimal trees for $n-1$ causes. At round $k+1$, all minimal trees for k causes have been constructed from the round k. With each tree T for k causes, for each gate g in T, apply Op1 once to generate a tree for $k+1$ causes, and apply Op2 once to generate another tree if g has any root input event.

5 Removing Duplication Trees

The above procedure can generate all minimal NIN-AND trees for n causes. However, when executed, no matter what data structure is used, nodes in each tree is implicitly labeled. This causes generation of labeled trees corresponding to the same unlabeled NIN-AND tree. For instance, from the minimal tree in

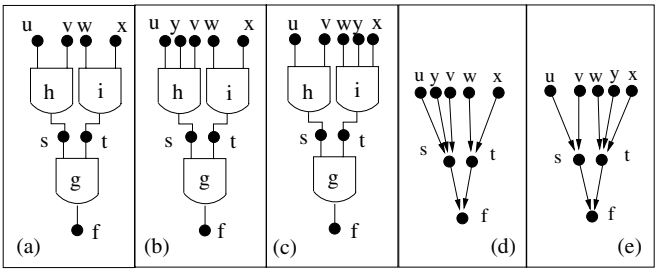


Fig. 4. Trees in (b) and (c) are generated from (a). (d) Reduced tree from (b). (e) Reduced tree from (c).

Fig. 4 (a), when Op1 is applied to gate h, the labeled tree in (b) is generated. When the operation is applied to gate i, the tree in (c) is generated. In both (b) and (c), the new node is labeled y. However, in (b) it is connected to a gate adjacent to u, v and s. But in (c), it is connected to a gate adjacent to w, x and t. It is not obvious that trees in (b) and (c) correspond to the some unlabeled tree. To turn a list of generated trees into an enumeration, all labeled trees corresponding to the same unlabeled tree must be removed, except one.

Algorithm 2 below decides if a newly generated labeled NIN-AND tree T' corresponds to the unlabeled tree represented by an existing labeled tree T. It uses a more compact representation where each gate node is merged with the unique non-root event node connected to it. This converts a minimal NIN-AND tree to a *reduced* tree, defined below and illustrated in Fig. 4 (d) and (e).

Def. 4. *Let T be a minimal NIN-AND tree. A reduced tree relative to T is a directed graph obtained by (1) merging each gate node with the unique non-root event node that it is connected to, and (2) making all links explicitly directed in the same directions as in T.*

Now the task is to recognize two reduced trees such as those in Fig. 4 (d) and (e) to be isomorphic. Two reduced trees are *isomorphic* if there is an one-to-one

mapping of their nodes which (1) maps the leaf in one to the leaf in the other, and (2) preserves in-degree and out-degree. Algorithm 2 performs the task, where $|T|$ denotes the number of nodes in T.

Algorithm 2. *IsIsomorphicTree(T, T')*

Input: Two reduced trees T and T', with leaf nodes v and v', and their parent sets π and π', respectively.

```
1   if |T'| ≠ |T|, return false;
2   if |T'| = 1, return true;
3   if |π'| ≠ |π|, return false;
4   Sub' = the set of subtrees each with the leaf in π';
5   Sub = the set of subtrees each with the leaf in π;
6   for each subtree S' ∈ Sub',
7     for each unmarked subtree S ∈ Sub,
8       if IsIsomorphicTree(S, S') returns true,
9         mark S' and S;
10        break inner for loop;
11    if S' is unmarked, return false;
12  return true;
```

Theorem 2 establishes that IsIsomorphicTree recognizes isomorphic reduced trees correctly.

Theorem 2. *$IsIsomorphicTree(T, T')$ returns true iff reduced trees T and T' are isomorphic.*

Proof: We prove by strong induction on $n = |T|$. Denote $n' = |T'|$. For $n = 1$, the only case where T' and T are isomorphic is $n' = 1$. IsIsomorphicTree returns true in line 2. If T' and T are not isomorphic, we must have $n' \neq 1$. IsIsomorphicTree returns false in line 1.

Since T and T' are reduced NIN-AND trees, $n > 1$ implies $n \geq 3$ (similarly $n' > 1$ implies $n' \geq 3$). We therefore also consider the base case $n = 3$, where T has the leaf and two roots. If T and T' are not isomorphic, we must have $n' \neq 3$. IsIsomorphicTree returns false in line 1. T and T' are isomorphic whenever $n' = 3$. IsIsomorphicTree returns true in line 2.

Assume that the theorem holds for $n = 1, 3, 4, ..., k$ where $k \geq 3$. We consider the case $n = k+1$. If T and T' are not isomorphic, either $n' \neq n$, or the degree of v' differs from that of v, or subtrees seated at parents of v' cannot be one-to-one mapped to subtrees seated at parents of v such that each pair is isomorphic. In the first two cases, IsIsomorphicTree returns false in lines1 and 3. In the third case, for at least one subtree S', no subtree S exists such that IsIsomorphicTree returns true by inductive assumption (the number of nodes in S is at most $k-1$). Hence, IsIsomorphicTree returns false in line 11.

On the other hand, if T and T' are identical, IsIsomorphicTree will enter the outer *for* loop. For each subtree S', there exists a subtree S such that IsIsomorphicTree returns true by inductive assumption (the number of nodes in

S is at most $k-1$). Hence, the outer *for* loop will complete and IsIsomorphicTree will return true in line 12. □

The following algorithm enumerates minimal NIN-AND trees of $n \geq 3$ causes in terms of reduced trees.

Algorithm 3. *EnumerateReducedTree(n)*

```
1   initialize set ST_2 with a single reduced tree for 2 causes;
2   for i = 3 to n,
3     initialize ST_i to empty set;
4     for each reduced tree T in ST_{i-1},
5       for each non-root node x of T,
6         duplicate T as T';
7         add a root parent to x in T';
8         if for each S ∈ ST_i, IsIsomorphicTree(S,T') returns false, add T' to ST_i;
9         if x has a root parent p;
10          duplicate T as T'';
11          insert a new node z between x and p in T'';
12          add a root parent to z in T'';
13          if for each S ∈ ST_i, IsIsomorphicTree(S,T'') returns false,
              add T'' to ST_i;
14  return ST_n;
```

The algorithm enumerates reduced trees of i causes based on the enumeration of $i-1$ causes, collected in set ST_{i-1}. For each reduced tree T in ST_{i-1}, each non-root node x is processed, which corresponds to a gate of the original NIN-AND tree. For each x, Op1 and Op2 (if applicable) are applied. If the new tree is not isomorphic to one in set ST_i, it is added. Each reduced tree in ST_n can be easily converted to a minimal NIN-AND tree for n causes.

Due to Theorem 1 and one-to-one mapping between a reduced tree and the corresponding minimal NIN-AND tree, minimal NIN-AND trees corresponding to each ST_i are exhaustive. Due to Theorem 2, they are also mutually exclusive. Hence, we have the following theorem.

Theorem 3. *ST_n produced by EnumerateReducedTree enumerates minimal NIN-AND trees for n causes.*

Using EnumerateReducedTree, we enumerated minimal NIN-AND trees for n causes with n between 2 and 10. The table below shows the execution result.

n	2	3	4	5	6	7	8	9	10
$\lvert St_n \rvert$	1	2	5	12	33	90	261	766	2312

Our enumeration enables an NIN-AND tree for n causes to be elicited by displaying all alternative trees to human expert so that one can be selected. As it is often easier to select a target object from a list than to describe the object from vacuum, this technique is expected to improve the accuracy and efficiency in elicitation.

6 Pairwise Causal Interactions

Although selection from a list is usually less demanding mentally than description from vacuum, when the number of causes is more than six or seven, identifying the target NIN-AND tree accurately from the enumeration may still be challenging. For instance, there are 261 minimal NIN-AND trees for 8 causes, On the other hand, eliciting whether a pair of causes is reinforcing or undermining is much less demanding (binary selection). We therefore explore the possibility to uniquely identify a minimal NIN-AND tree based on elicitation of $O(n^2)$ pairwise causal interactions.

To do so, we need to understand how a minimal NIN-AND tree determines a set of pairwise causal interactions. However, this is impossible because a minimal NIN-AND tree is unlabeled while a set of pairwise causal interactions must be specified over specific pairs of causes. On the other hand, any minimal NIN-AND tree with its root nodes labeled determines uniquely a set of pairwise causal interactions, as shown by the following proposition. We refer to such a tree as a minimal, *root-labeled* NIN-AND tree. Note that in a root-labeled tree, root nodes are labeled but non-root nodes are unlabeled. Note also the default assumption that every root node is a single-cause event.

Proposition 4. *Let T be a minimal NIN-AND tree for a set X of causes and rl be a labeling of root nodes. Then T and rl define a function pci from pairs of distinct causes $\{c_i, c_j\} \subset X$, where $i \neq j$, to the set $\{rif, udm\}$, where rif stands for reinforcing and udm stands for undermining.*

For instance, $pci(c_i, c_j) = udm$ means that causal interaction between c_i and c_j is undermining. Proof of Proposition 4 depends on Proposition 5 below, which in turn relies on the concept of the *closest common gate* (ccg):

Def. 5. *Let x and y be two root nodes in a minimal NIN-AND tree T. Let $path_x$ be the directed path from x to the leaf of T and $path_y$ be that from y. Then, the first node in $path_x$ common to a node in $path_y$ is the ccg of x and y.*

In Fig. 4 (c), the ccg of x and y is i. Note that the first node common in $path_x$ and $path_y$ is always a gate. The following proposition shows how the interaction between a pair of causes is encoded in an NIN-AND tree.

Proposition 5. *Let T be a minimal, root-labeled NIN-AND tree. Let x and y be a pair of root nodes, c_i and c_j be their corresponding causes, and g be their ccg. Then the causal interaction between c_i and c_j is of the type of g.*

Proof: Given root-labeled T, assume that all causes become inactive except c_i and c_j. Now except g, all other gates have no more than one active causal input event, and hence can be removed. The resultant minimal NIN-AND tree has a single gate g with root input events x and y. □

From Proposition 5, it is clear that the topology of T and a root labeling uniquely determine the type of causal interaction between each pair of causes (corresponding to a pair of roots in T). Hence, Proposition 4 holds.

7 Are NIN-AND Trees PCI Differentiable?

Given that each minimal NIN-AND tree, plus a root labeling, uniquely determines a set of pairwise causal interactions, can a minimal NIN-AND tree be identified from a set of pairwise causal interactions? In other words, are pairwise causal interactions sufficient to differentiate minimal NIN-AND trees?

More specifically, let $X = \{c_1, ..., c_n\}$ be a set of n causes for an effect, and ST_n be the set of all minimal NIN-AND trees for n causes. Let *pci* be a function from pairs of distinct causes $\{c_i, c_j\} \subset X$ $(i \neq j)$ to the set $\{rif, udm\}$, determined by $T \in ST_n$ and a root labeling. The question then is whether it is possible to *uniquely* identify T given *pci* only. For instance, the NIN-AND tree in Fig. 2 (a) has $pci(c_1, c_2) = pci(c_1, c_3) = pci(c_2, c_3) = udm$, $pci(c_1, c_4) = pci(c_2, c_4) = pci(c_3, c_4) = rif$. Can the tree model be uniquely identified from the function? We refer to this as the *identification* question.

Note that since T and a root labeling together define a causal interaction function *pci*, to answer the above question in general, the search space is not ST_n but the space of all root-labeled minimal NIN-AND trees, whose complexity is $O(n!\ |ST_n|)$.

We answer the identification question by exhaustively testing whether there exists a pair of NIN-AND trees T and T' and there exists a root labeling for each of them, such that the two root-labeled trees satisfy the same set of pairwise causal interactions corresponding to some *pci* function. Although this method does not scale for very large n, we point out that a NIN-AND tree model is used to acquire a single CPT in a BN and hence very large n is not expected given the conditional independence expressed by the BN. To make the testing computation effective, we developed the following test conditions:

Given n causes, there are $n(n-1)/2$ pairs. A *pci* function maps each pair to one of rif and udm, thus defining two sets of pairs which we refer to as Rif and Udm. That is, a pair $\{c_i, c_j\} \in Udm$, iff $pci(c_i, c_j) = udm$. Once one of Rif and Udm is defined, the other is uniquely determined.

Proposition 6 below shows that if two minimal NIN-AND trees, under arbitrary root-labeling, produce two Rif sets of different cardinalities, then the two trees are differentiable from pairwise casual interactions.

Proposition 6. *Let T and T' be two minimal NIN-AND trees of n causes, rl and rl' be some root labeling of T and T', and pci and pci$'$ be the corresponding pairwise causal interaction functions, respectively. Let Rif and Rif' be the sets of reinforcing pairs, defined by pci and pci$'$, respectively.*

If $|Rif| \neq |Rif'|$, there exist no root labeling for T and T', such that the two root-labeled trees satisfy the same pairwise causal interaction function.

Proof: When $|Rif| \neq |Rif'|$, the number of reinforcing cause pairs defined by T and rl differs from that defined by T' and rl'. Since the number of reinforcing cause pairs is independent of root labeling, no matter what alternative root labeling are used, the inequality remains. This implies that $Rif \neq Rif'$ no matter what root labeling is used. Hence, the proposition holds. □

Proposition 6 suggests an inexpensive test (to be referred as Test 1) that rules out the tree pair from possibly contributing to the negative answer of identification question.

If two minimal NIN-AND trees, under some root-labeling, produce Rif sets of identical cardinalities, it is uncertain how the tree pair contributes to the answer of identification question. The following proposition suggests a further test.

Proposition 7. *Let T, T', rl, rl', pci and pci' be defined as in Proposition 6. For each cause c, let $k(c)$ be the number of other causes that are pairwise reinforcing with c according to pci, and κ be the sorted list of $k(c)$'s. Let κ' be the corresponding sorted list defined by pci'.*

If $\kappa \neq \kappa'$, there exist no root labeling for T and T', such that the two root-labeled trees satisfy the same pairwise causal interaction function.

For the NIN-AND tree in Fig. 2 (a), $\kappa = (1, 1, 1, 3)$ because $k(c_1) = k(c_2) = k(c_3) = 1$ and $k(c_4) = 3$.

Proof: If $\kappa \neq \kappa'$, then there exists k $(0 \leq k < n(n-1)/2)$ such that m elements of κ have value k, m' elements of κ' have value k, and $m \neq m'$. That is, according to T under root-labeling rl, each of m causes reinforces with another k causes. But according to T' under root-labeling rl', each of m' causes reinforces with another k causes.

The number of other causes which a given cause reinforces with is independent of root labeling. Hence, no matter what alternative root labeling are used, according to T the number of causes reinforcing with k other causes remains m, and according to T' the number remains m'. This implies that $\kappa \neq \kappa'$ no matter what root labeling is used. Hence, the proposition holds. □

Proposition 7 suggests another inexpensive test (to be referred as Test 2) that rules out a tree pair from possibly contributing to the negative answer of identification question. Algorithm 4 utilizes Test 1 and Test 2 in answering identification question.

Lines 1 through 4 enumerate minimal NIN-AND trees and compute the pci function for each tree under some root labeling. Lines 6 and 7 perform Test 1, and lines 8 and 9 perform Test 2. Lines 10 through 12 (Test 3) tries each of the $n! - 1$ alternative root labeling on T'. If one produces Rif' identical to Rif for T, then T and T' cannot be differentiated by pairwise causal interactions. The algorithm will return false.

Algorithm 4. *IsPciIdentificable(n)*

```
1   run EnumerateReducedTree(n) to produce ST_n;
2   convert reduced trees in ST_n to minimal NIN-AND trees;
3   for each minimal NIN-AND tree T,
4     compute its pci function under some root labeling;
5   for each pair of trees T and T' with pci and pci',
6     compute Rif and Rif' from pci and pci';
7     if |Rif| ≠ |Rif'|, go to line 5 for next pair;
```

8 compute κ and κ' from pci and pci';
9 if $\kappa \neq \kappa'$, go to line 5 for next pair;
10 for each alternative root labeling rl' of T',
11 recompute Rif';
12 if Rif = Rif', return false;
13 return true;

If for every pair of T and T', either Test 1 fails ($|Rif| \neq |Rif'|$), or Test 2 fails ($\kappa \neq \kappa'|$), or Test 3 fails ($Rif \neq Rif'$ for all root labeling), the algorithm returns true. This means that every minimal NIN-AND tree for n causes can be identified based solely on a set of pairwise causal interactions. This leads to the following theorem, whose proof is straightforward given the above analysis.

Theorem 4. *Let T be a minimal NIN-AND tree for n causes and Rif is the set of pairwise causal interactions determined by T under some root labeling. Then given Rif only, T can be identified from all minimal NIN-AND trees for n causes iff algorithm IsPciIdentificable(n) returns true.*

For example, given $Rif = \{\{c_1, c_4\}, \{c_2, c_4\}, \{c_3, c_4\}\}$, the minimal NIN-AND tree in Fig. 2 (a) is uniquely identified. Justified by Theorem 4, we implemented IsPciIdentificable(n). Executions for $n = 3, ..., 10$ all returned true. Tests 1 and 2 suggested by Propositions 6 and 7 trim computation significantly. For $n = 10$, there are 2312 minimal NIN-AND trees, and hence 2671516 pairs. Only 122588 pairs ($< 5\%$) passed the least expensive Test 1, where Test 2 is needed. Out of these pairs, only 467 pairs ($< 0.4\%$) passed Test 2, where the most expensive Test 3 has to be run. This amounts to the processing of 1694649600 root-labeled trees in Test 3, which is about 20% of the total 8389785600 root-labeled trees.

8 Remarks

Assessment of CPTs is often a bottleneck in practical applications of BNs when frequency data are not available and elicitation from expert is necessary. This work follows the effort by many, e.g., [7,4,3,5,9], to make this step in probabilistic reasoning more efficient. The main contributions are the following:

From associativity of NIN-AND gates, we characterized minimal NIN-AND tree topology space and partitioned it into the subspace with direct leaf NIN-AND gates and the subspace with dual leaf gates. This partition allows subsequent investigation to be focused on one subspace while the results are applicable to the other. We developed a method to enumerate NIN-AND trees based on local insertion. This result provides an alternative method for NIN-AND tree elicitation and allows expert to select from enumeration. We demonstrated that NIN-AND trees for up to 10 causes (about the necessary upper bound for CPTs in BNs) can be differentiated based on pairwise causal interactions. This result provides an even more powerful technique for eliciting NIN-AND trees by eliciting from expert only pairwise causal interactions.

In developing the two elicitation techniques, enumeration of NIN-AND trees by number of causes (Section 4) is needed. As information flows from single-cause

events to the multi-cause event in an NIN-AND tree, following the convention in causal graphical models, we directed reduced NIN-AND trees with single-cause events as roots and with a single leaf. We therefore needed to enumerate unlabeled trees of a single leaf by number of roots. Many methods of tree enumeration in the mathematics literature, e.g., [1,8,6], do not address this problem. In [2], under the context of phylogenetic trees, counting of *rooted multifurcating tree shapes by tips* is presented. Reversing directions of links, these tree shapes are equivalent to what we enumerate. However, Felsenstein's counting is based on an ordered partition of tips, and our method is based on local insertion. It focuses on *counting* without generation and ours emphasizes generation. In [10], we extend [2] into a method as an alternative to the method presented here.

Acknowledgements

We acknowledge the financial support through the Discovery Grant, NSERC, Canada. We thank anonymous reviewers for their helpful comments.

References

1. Cayley, A.: A theorem on trees. Quarterly J. Mathematics, 376–378 (1889)
2. Felsenstein, J.: Inferring Phylogenies. Sinauer Associates, Sunderland (2004)
3. Galan, S.F., Diez, F.J.: Modeling dynamic causal interaction with Bayesian networks: temporal noisy gates. In: Proc. 2nd Inter. Workshop on Causal Networks, pp. 1–5 (2000)
4. Heckerman, D., Breese, J.S.: Causal independence for probabilistic assessment and inference using Bayesian networks. IEEE Trans. on System, Man and Cybernetics 26(6), 826–831 (1996)
5. Lemmer, J.F., Gossink, D.E.: Recursive noisy OR - a rule for estimating complex probabilistic interactions. IEEE Trans. on System, Man and Cybernetics, Part B 34(6), 2252–2261 (2004)
6. Moon, J.W.: Counting Labeled Trees. William Clowes and Sons, London (1970)
7. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann, San Francisco (1988)
8. Riordan, J.: The enumeration of trees by height and diameter. IBM J., 473–478 (November 1960)
9. Xiang, Y., Jia, N.: Modeling causal reinforcement and undermining for efficient cpt elicitation. IEEE Trans. Knowledge and Data Engineering 19(12), 1708–1718 (2007)
10. Xiang, Y., Zhu, J., Li, Y.: Enumerating unlabeled and root labeled trees for causal model acquisition. To appear in Proc. Canadian AI 2009., Springer, Heidelberg (2009)

An Evidence-Theoretic *k*-Nearest Neighbor Rule for Multi-label Classification

Zoulficar Younes, Fahed Abdallah, and Thierry Denœux

UMR CNRS 6599 Heudiasyc,
Université de Technologie de Compiègne, France
{firstname.lastname}@hds.utc.fr

Abstract. In multi-label learning, each instance in the training set is associated with a set of labels, and the task is to output a label set for each unseen instance. This paper describes a new method for multi-label classification based on the Dempster-Shafer theory of belief functions to classify an unseen instance on the basis of its k nearest neighbors. The proposed method generalizes an existing single-label evidence-theoretic learning method to the multi-label case. In multi-label case, the frame of discernment is not the set of all possible classes, but it is the powerset of this set. That requires an extension of evidence theory to manipulate multi-labelled data. Using evidence theory makes us able to handle ambiguity and imperfect knowledge regarding the label sets of training patterns. Experiments on benchmark datasets show the efficiency of the proposed approach as compared to other existing methods.

1 Introduction

Traditional *single-label classification* assigns an object to exactly one class, from a set of Q disjoint classes. In contrast, *Multi-label classification* is the task of assigning an instance to one or multiple classes simultaneously. In other words, the target classes are not exclusive: an object may belong to an unrestricted set of classes instead of exactly one. This task makes multi-label classifiers more difficult to train than traditional single-label classifiers. Recently, multi-label classification methods have been increasingly required by modern applications where it is quite natural that some instances belong to several classes at the same time. In text categorization, each document may belong to multiple topics, such as arts and humanities [8].

In natural scene classification, each image may belong to several image types at the same time, such as sea and sunset [1]. In classification of music into emotions, music may evoke more than one emotion at the same time, such as relaxing and sad [7].

Few algorithms have been proposed for multi-label learning. A first family of algorithms transforms the multi-label classification problem into a set of binary classification problems; each binary classifier is then trained to separate one class from the others [1]. A second family consists in extending common

L. Godo and A. Pugliese (Eds.): SUM 2009, LNAI 5785, pp. 297–308, 2009.

learning algorithms and making them able to manipulate multi-label data directly. In [14] and [15], a *Bayesian* approach based on multi-label extension of the k-nearest neighbor (k-NN) rule is presented. In the literature, there also exist multi-label extensions of neural networks [2], support vector machine [6], and boosting learning algorithms [9].

In this paper, we present a new method for multi-label classification based on the Dempster-Shafer theory of belief functions to classify an unseen instance on the basis of its k nearest neighbors.

The Dempster-Shafer (D-S) theory [10] is a formal framework for representing and reasoning with uncertain and imprecise information. Different approaches for pattern classification in the framework of evidence theory have been presented in the literature [4] [5]. In [3], A k-NN classification rule based on D-S theory is presented. Each neighbor of an instance to be classified is considered as an item of evidence supporting certain hypotheses regarding the class membership of that instance. The degree of support is defined as a function of the distance between the two samples. The evidence of the k nearest neighbors is then pooled by means of Dempster's rule of combination.

The proposed method generalizes the k-NN classification rule based on the D-S theory to the multi-label case. This generalization requires an extension of the D-S theory in order to handle multi-labelled data. In mono-labelled data case, the uncertainty is represented by evidence on multiple hypotheses where each hypothesis is a label to be assigned or not to an unseen instance. In contrast, when the data is multi-labelled, each hypothesis represents a set of labels and the uncertainty is then expressed by evidence on sets of label sets. The proposed algorithm is called $EML-kNN$ for Evidential Multi-Label k-Nearest Neighbor.

The remainder of the paper is organized as follows. Section 2 recalls the basics of the D-S theory and the principle of the single-label evidence-theoretic k-NN rule [3]. Section 3 introduces the extension of the D-S theory to the multi-label case and describes the proposed algorithm for multi-label learning that consists in applying the D-S multi-label extended theory using the k-NN rule. Section 4 presents experiments on two real datasets and shows the effectiveness of the proposed algorithm as compared to a recent high-performance method for multi-label learning based on k-NN rule, referred to as $ML-kNN$ [15]. Finally Section 5 summarizes this work and makes concluding remarks.

2 Single-label Classification

2.1 Basics of Dempster-Shafer Theory

In D-S theory, a *frame of discernment* Ω is defined as the set of all hypotheses in a certain domain, e.g., in classification Ω is the set of all possible classes. A *basic belief assignment* (BBA) is a function m that defines a mapping from the power set of Ω to the interval $[0, 1]$ verifying:

$$m : 2^{\Omega} \longrightarrow [0, 1] \tag{1}$$

$$\sum_{A \in 2^{\Omega}} m(A) \;=\; 1. \tag{2}$$

Given a certain piece of evidence, the value of the BBA for a given set A expresses a measure of belief that one is willing to commit exactly to A. The quantity $m(A)$ pertains only to the set A and makes no additional claims about any subsets of A. If $m(A) > 0$, then the subset A is called a *focal element* of m.

The BBA m and its associated focal elements define a *body of evidence*, from which a belief function Bel and a plausibility function Pl mapped from 2^{Ω} to $[0, 1]$ can be deduced. For a set A, $Bel(A)$, called *belief* in A or *credibility* of A, represents a measure of the total belief committed to the set $A \subseteq \Omega$. $Bel(A)$ is defined as the sum of all the BBAs of the non-empty subsets of A.

$$Bel(A) = \sum_{\emptyset \neq B \subseteq A} m(B) \tag{3}$$

$Pl(A)$, called *plausibility* of A, represents the amount of belief that could potentially be placed in A, if further information became available [3]. $Pl(A)$ is defined as the sum of all the BBAs of the sets that intersect A.

$$Pl(A) = \sum_{B \cap A \neq \emptyset} m(B) \tag{4}$$

From the definitions of belief and plausibility functions, it follows that:

$$Pl(A) = Bel(\Omega) \ - \ Bel(\bar{A}) \tag{5}$$

where $\bar{A}$ is the complement of A.

Given the belief function Bel, it is possible to derive the corresponding BBA as follows:

$$m(\emptyset) = 1 - Bel(\Omega), \tag{6}$$

$$m(A) = \sum_{B \subseteq A} (-1)^{|A \backslash B|} \ Bel(B), \ A \neq \emptyset \tag{7}$$

where $|A \backslash B|$ is the cardinality of the complement of B in A.

As a consequence of (5), (6) and (7), given any one of the three functions m, Bel and Pl it is possible to recover the other two.

The unnormalized *Dempster's rule of combination* [10] [11] is an operation for pooling evidence from a variety of sources. This rule aggregates two independent bodies of evidence defined within the same frame of discernment into one body of evidence. Let m_1 and m_2 be two BBAs. Let m_{12} be the new BBA obtained by combining m_1 and m_2 using the unnormalized Dempster's rule of combination. m_{12} is the *orthogonal sum* of m_1 and m_2 denoted as $m_{12} = m_1 \textcircled{\cap} m_2$. The aggregation is calculated in the following manner:

$$m_{12}(A) = \sum_{B \cap C = A} m_1(B) m_2(C), \ A \subseteq \Omega. \tag{8}$$

This rule is commutative and associative, and admits the *vacous* BBA ($m(\Omega) = 1$) as neutral element.

2.2 Evidence-Theoretic *k*-NN Rule

Let $\mathcal{X} = R^P$ denote the domain of instances and let $\mathcal{Y} = \{1, 2, \ldots, Q\}$ be the finite set of classes, also called labels or categories. The available information is assumed to consist in a training set $\mathcal{T} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_M, y_M)\}$ of M single-labelled samples, where $\mathbf{x}_i \in \mathcal{X}$ and the corresponding class label y_i takes value in $\mathcal{Y}$, for each $i = 1, \ldots, M$.

Let $\mathbf{x}$ be a new instance to be classified on the basis of its nearest neighbors in $\mathcal{T}$. Let $\mathcal{N}_{\mathbf{x}} = \{(\mathbf{x}_i, y_i) | i = 1 \ldots, k\}$ be the set of the k-nearest neighbors of $\mathbf{x}$ in $\mathcal{T}$ based on a certain distance function $d(.,.)$, e.g, the Euclidean distance. Each pair $(\mathbf{x}_i, y_i)$ in $\mathcal{N}_{\mathbf{x}}$ constitutes a distinct item of evidence regarding the class membership of $\mathbf{x}$. If $\mathbf{x}$ is *close* to $\mathbf{x}_i$ according to the distance function d, then one will be inclined to believe that both instances belong to the same class, while when $d(\mathbf{x}, \mathbf{x}_i)$ increases, this belief decreases and that yields to a situation of almost complete ignorance concerning the class of $\mathbf{x}$. Consequently, each pair $(\mathbf{x}_i, y_i)$ in $\mathcal{N}_{\mathbf{x}}$ induces a basic belief assignment m_i over $\mathcal{Y}$ defined by:

$$m_i(\{y_i\}) = \alpha\phi(d_i) \tag{9}$$

$$m_i(\mathcal{Y}) = 1 - \alpha\phi(d_i) \tag{10}$$

$$m_i(A) = 0, \ \forall A \in 2^{\mathcal{Y}} \backslash \{\mathcal{Y}, \{y_i\}\} \tag{11}$$

where $d_i = d(\mathbf{x}, \mathbf{x}_i)$, α is a parameter such that $0 < \alpha < 1$ and ϕ is a decreasing function verifying $\phi(0) = 1$ and $\lim_{d \to \infty} \phi(d) = 0$. In [3], the author suggests to choose the function ϕ as:

$$\phi(d) = \exp(-\gamma d^{\beta}) \tag{12}$$

where $\gamma > 0$ and $\beta \in \{1, 2, \ldots\}$. As explained in [3], parameter β has been found to have very little influence on the performance of the method, and can be arbitrarily fixed to a small value (1 or 2). The most influential parameter on the performance of the classifier is γ. In [3], a distinct parameter γ_q was associated for each class $q \in \mathcal{Y}$. When considering the item of evidence $(\mathbf{x}_i, y_i)$ for the class membership of $\mathbf{x}$, if $y_i = q$, using (12), $\phi(d_i)$ in (9) and (10) was replaced by $\gamma_q d_i^{\beta}$. The values of α and γ_q, $q = 1, \ldots, Q$ were fixed via heuristics [3].

As a result of considering each training instance in $\mathcal{N}_{\mathbf{x}}$ as an item of evidence, we obtain k BBAs that can be pooled by means of the unnormalized Dempster's rule of combination yielding to the aggregated BBA m synthesizing one's final belief regarding the class membership of $\mathbf{x}$:

$$m = m_1 \textcircled{\cap} \ldots \textcircled{\cap} m_k. \tag{13}$$

For making decisions, functions Bel and Pl can be derived from m using (3) and (4) respectively, and the test instance $\mathbf{x}$ is assigned to the class q that corresponds to the maximum credibility or the maximum plausibility.

3 Multi-label Classification

3.1 Multi-label Extension of Dempster-Shafer Theory

In Sect. 2.1, we have recalled the basics of D-S theory used to handle uncertainty in problems where *only one single hypothesis* is true. Moreover, there exist problems where *more than one hypothesis* is true at the same time, e.g., the multi-label classification task. To handle such problems, we need to extend the classical D-S framework. The frame of discernment of the multi-label extended D-S theory is not the set Ω of all possible single hypotheses but its power set $\Theta = 2^{\Omega}$. A basic belief assignment is now defined as a mapping from the power set of Θ to the interval $[0, 1]$. Instead of considering the whole power set of Θ, we will focus on the subset $\mathcal{C}(\Omega)$ of 2^{Θ} defined as:

$$\mathcal{C}(\Omega) = \{\varphi(A, B) |\ A \cap B = \emptyset\} \cup \{\emptyset_{\Theta}\} \tag{14}$$

where $\emptyset_{\Theta}$ represents the conflict in the frame 2^{Θ}, and for all $A,\ B \subseteq \Omega$ with $A \cap B = \emptyset$, $\varphi(A, B)$ is the set of all subsets of Ω that include A and have no intersection with B:

$$\varphi(A, B) = \{C \subseteq \Omega |\ C \supseteq A \ and \ C \cap B = \emptyset\}. \tag{15}$$

The size of the subset $\mathcal{C}(\Omega)$ of 2^{Θ} is equal to $3^{|\Omega|} + 1$, it is thus much smaller than the size of 2^{Θ} ($|2^{\Theta}| = 2^{2^{|\Omega|}}$), while being rich enough to express evidence in many realistic situations. That reduces the complexity of such problems.

The chosen subset $\mathcal{C}(\Omega)$ of 2^{Θ} is closed under intersection, i.e., for all $\varphi(A, B)$, $\varphi(A', B') \in \mathcal{C}(\Omega)$, $\varphi(A, B) \cap \varphi(A', B') \in \mathcal{C}(\Omega)$. Based on the definition of $\varphi(A, B)$, one can deduce that:

$$\varphi(\emptyset, \emptyset) = \Theta, \tag{16}$$

$$\forall A \subseteq \Omega,\ \varphi(A, \bar{A}) = \{A\}, \tag{17}$$

$$\forall A \subseteq \Omega,\ A \neq \emptyset,\ \varphi(A, A) = \emptyset_{\Theta}. \tag{18}$$

By convention, we will note $\emptyset_{\Theta}$ by $\varphi(\Omega, \Omega)$ in the rest of the paper.

Example 1. Let $\Omega = \{a, b\}$ be a frame of discernment. The corresponding subset $\mathcal{C}(\Omega)$ of 2^{Θ}, where Θ is the power set of Ω, is:

$$\begin{aligned}\mathcal{C}(\Omega) = \{&\varphi(\emptyset, \emptyset), \varphi(\emptyset, \{a\}), \varphi(\emptyset, \{b\}), \varphi(\emptyset, \Omega), \varphi(\{a\}, \emptyset),\\ &\varphi(\{b\}, \emptyset), \varphi(\Omega, \emptyset), \varphi(\{a\}, \{b\}), \varphi(\{b\}, \{a\}), \varphi(\Omega, \Omega)\}.\end{aligned}$$

For instance, $\varphi(\{a\}, \emptyset) = \{\{a\}, \Omega\}$ and $\varphi(\{a\}, \{b\}) = \{\{a\}\}$.

For any $\varphi(A, B),\ \varphi(A', B') \in \mathcal{C}(\Omega)$ the intersection operator over $\mathcal{C}(\Omega)$ is defined as follow:

$$\varphi(A, B) \cap \varphi(A', B') = \begin{cases} \varphi(A \cup A', B \cup B') & if\ A \cap B' = \emptyset \ and \ A' \cap B = \emptyset \\ \varphi(\Omega, \Omega) & otherwise, \end{cases} \tag{19}$$

and the inclusion operator over $\mathcal{C}(\Omega)$ is defined as:

$$\varphi(A,B) \subseteq \varphi(A',B') \iff A \supseteq A' \text{ and } B \supseteq B'. \tag{20}$$

The description of a BBA m on $\mathcal{C}(\Omega)$ can be represented with the following two equations:

$$m : \mathcal{C}(\Omega) \longrightarrow [0,1] \tag{21}$$

$$\sum_{\varphi(A,B)\in\mathcal{C}(\Omega)} m(\varphi(A,B)) = 1. \tag{22}$$

In the following, the notation $m(\varphi(A,B))$ will be simplified to $m(A,B)$. For any $\varphi(A,B) \in \mathcal{C}(\Omega)$, the belief and plausibility functions are defined as:

$$Bel(A,B) = \sum_{\varphi(\Omega,\Omega)\neq\varphi(A',B')\subseteq\varphi(A,B)} m(A',B'), \tag{23}$$

and

$$Pl(A,B) = \sum_{\varphi(A',B')\cap\varphi(A,B)\neq\varphi(\Omega,\Omega)} m(A',B'). \tag{24}$$

Given two independent bodies of evidence over the same frame of discernment like $\mathcal{C}(\Omega)$, the aggregated BBA, denoted by m_{12}, obtained by combining the BBAs m_1 and m_2 of the two bodies of evidence using the unnormalized Dempster's rule is calculated in the following manner:

$$m_{12}(A,B) = \sum_{\varphi(A',B')\cap\varphi(A'',B'')=\varphi(A,B)} m_1(A',B')m_2(A'',B''). \tag{25}$$

This rule is commutative and associative, and has the vacuous BBA ($m(\emptyset,\emptyset) = 1$) as neutral element.

3.2 Evidential Multi-label *k*-NN

Problem. As in Sect. 2.2, let $\mathcal{X} = R^P$ denote the domain of instances and let $\mathcal{Y} = \{1,2,\ldots,Q\}$ be the finite set of labels. The multi-label classification problem can be formulated as follows. Given a set $\mathcal{S} = \{(\mathbf{x}_1,Y_1),\ldots,(\mathbf{x}_M,Y_M)\}$ of M training examples drawn from $\mathcal{X}\times 2^{\mathcal{Y}}$, and identically distributed, where $\mathbf{x}_i \in \mathcal{X}$ and $Y_i \subseteq \mathcal{Y}$, the goal of the learning system is to output a multi-label classifier $\mathcal{H} : \mathcal{X} \to 2^{\mathcal{Y}}$ that optimizes some pre-defined criteria.

The method. Let $\mathbf{x}$ be an unseen instance that we search to estimate its set of labels Y on the basis of its k nearest neighbors in $\mathcal{S}$ represented by $\mathcal{N}_{\mathbf{x}}$ using the multi-label extension of the D-S theory introduced in Sect. 3.1. The frame of discernment of the multi-label classification problem is the powerset of $\mathcal{Y}$.

Each pair $(\mathbf{x}_i, Y_i)$ in $\mathcal{N}_\mathbf{x}$ constitutes a distinct item of evidence regarding the label set of $\mathbf{x}$. Regarding the label set Y_i, we can conclude either that Y must include all the labels in Y_i, or that Y must contain at least one of the labels that belong to Y_i, or that Y does not contain any label not belonging to Y_i. Let $\varphi(A_i, B_i)$ be the set of label sets that corresponds to the item of evidence $(\mathbf{x}_i, Y_i)$, where A_i, $B_i \subseteq \mathcal{Y}$. We recall that the set $\varphi(A_i, B_i)$ contains all the label sets that include A_i and having no intersection with B_i. There exist different ways to express our beliefs about the labels to be assigned to the instance $\mathbf{x}$ based on the item of evidence $(\mathbf{x}_i, Y_i)$. This leads to different versions of our proposed method $EML-kNN$:

- Version 1 (V1): the mass is attributed to the set Y_i, thus $\varphi(A_i, B_i) = \varphi(Y_i, \bar{Y}_i)$.
- Version 2 (V2): the mass is attributed to the set Y_i and all its supersets, thus $\varphi(A_i, B_i) = \varphi(Y_i, \emptyset)$.
- Version 3 (V3): the mass is attributed to the set Y_i and all its subsets, thus $\varphi(A_i, B_i) = \varphi(\emptyset, \bar{Y}_i)$.

The BBA m_i over $\mathcal{C}(\mathcal{Y})$ induced by the item of evidence $(\mathbf{x}_i, Y_i)$ regarding the label set of $\mathbf{x}$ can then be defined as:

$$m_i(A_i, B_i) = \alpha\phi(d_i) \tag{26}$$

$$m_i(\emptyset, \emptyset) = 1 - \alpha\phi(d_i) \tag{27}$$

where $d_i = d(\mathbf{x}, \mathbf{x}_i)$, ϕ is the decreasing function introduced in Sect. 2.2 (see (12)).

After considering each item of evidence in $\mathcal{N}_\mathbf{x}$, we obtain the BBAs m_i, $i = 1, \ldots, k$ that can be combined 2 by 2 using the multi-label extension of the unnormalized Dempster's rule of combination presented in Sect. 3.1 (see (25)) to form the resulting BBA m.

Let $\hat{Y}$ denote the estimated label set of the instance $\mathbf{x}$ to differentiate it from the ground truth label set Y of $\mathbf{x}$. One of the methods to determine $\hat{Y}$ that we have adopted in this paper is to assign $\mathbf{x}$ to the set $C \subseteq \mathcal{Y}$ that corresponds to the maximum plausibility. Thus, the estimated label set of $\mathbf{x}$ is:

$$\hat{Y} = \max_{C \subseteq \mathcal{Y}} Pl(C, \bar{C}). \tag{28}$$

The plausibility function Pl derived from the aggregated BBA m is determined using (24).

4 Experiments

4.1 Datasets

Two datasets are used for experiments: the emotion and the scene datasets.

Emotion Dataset. This dataset contains 593 songs, each represented by a 72-dimensional feature vector (8 rhythmic features and 64 timbre features) [7]. The emotional labels are: *amazed-surprised, happy-pleased, relaxing-calm, quiet-still, sad-lonely and angry-fearful.*

Scene Dataset. This dataset contains 2000 natural scene images. Each image is associated with some of the six different semantic scenes: *sea, sunset, trees, desert and mountains*. For each image, spatial color moments are used as features. Images are divided into 49 blocks using a 7×7 grid. The mean and variance of each band are computed corresponding to a low-resolution image and to computationally inexpensive texture features, respectively [1]. Each image is then transformed into a $49 \times 3 \times 2 = 294$-dimensional feature vector.

Each dataset was split into a training set and a test set. Table 1 summarizes the characteristics of the datasets used in the experiments. The label cardinality of a dataset is the average number of labels of the instances, while the label density is the average number of labels of the instances divided by the total number of labels [12].

4.2 Evaluation Metrics

Let $\mathcal{D} = \{(\mathbf{x}_1, Y_1), \ldots, (\mathbf{x}_N, Y_N)\}$ be a multi-label evaluation dataset containing N labelled examples. Let $\widehat{Y}_i = \mathcal{H}(\mathbf{x}_i)$ be the predicted label set for the pattern $\mathbf{x}_i$, while Y_i is the ground truth label set for $\mathbf{x}_i$.

A first metric called *Accuracy* gives an average degree of similarity between the predicted and the ground truth label sets of all test examples:

$$Accuracy(\mathcal{H}, \mathcal{D}) = \frac{1}{N} \sum_{i=1}^{N} \frac{|Y_i \cap \widehat{Y}_i|}{|Y_i \cup \widehat{Y}_i|}. \tag{29}$$

Two other metrics called *Precision* and *Recall* are also used in the literature to evaluate a multi-label learning system. The former computes the proportion of correct positive predictions while the latter calculates the proportion of true labels that have been predicted as positives:

$$Precision(\mathcal{H}, \mathcal{D}) = \frac{1}{N} \sum_{i=1}^{N} \frac{|Y_i \cap \widehat{Y}_i|}{|\widehat{Y}_i|}, \tag{30}$$

$$Recall(\mathcal{H}, \mathcal{D}) = \frac{1}{N} \sum_{i=1}^{N} \frac{|Y_i \cap \widehat{Y}_i|}{|Y_i|}. \tag{31}$$

These metrics have been cited in [12].

Another evaluation criterion is the $F1$ measure that is defined as the harmonic mean of the *Precision* and *Recall* metrics [13]:

$$F1 = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}}. \tag{32}$$

The values of these evaluation criteria are in the interval $[0, 1]$. Larger values of these metrics correspond to higher classification quality.

Table 1. Characteristics of datasets

Dataset	Number of instances	Feature vector dimension	Number of labels	Training instances	Test instances	Label cardinality	Label density	maximum size of a label set
emotion	593	72	6	391	202	1.868	0.311	3
scene	2407	294	6	1211	1196	1.074	0.179	3

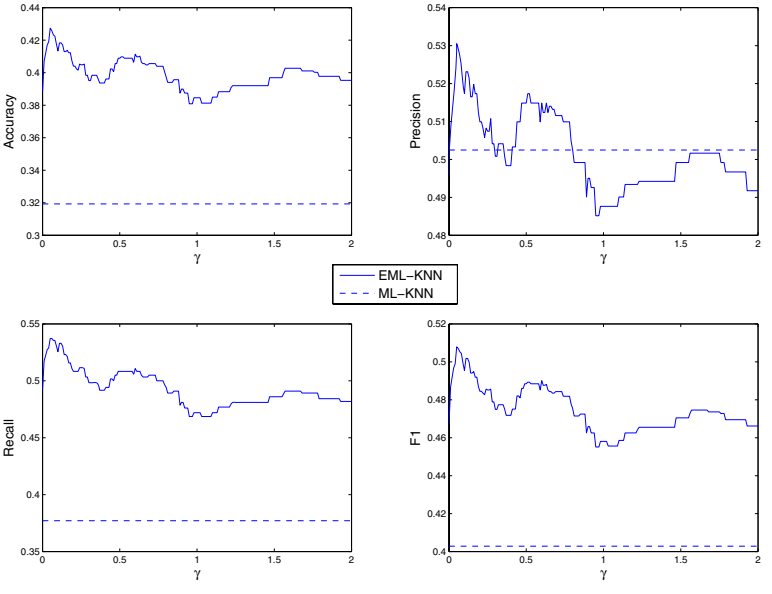


Fig. 1. *Accuracy*, *Precision*, *Recall* and $F1$ measures for $EML - kNN$ (V1) and $ML - kNN$ algorithms as a function of γ on the emotion dataset, for $k = 10$

4.3 Results and Discussions

The proposed algorithm was compared to a *Bayesian* method for multi-label classification based on the k-NN rule named $ML - kNN$ [15].

The model parameters for $EML - kNN$ are : The number of neighbors k, and the parameters for the induced BBAs, α, β and γ. $ML - KNN$ has only one parameter that needs to be optimized, which is k. As in [3], α was fixed to 0.95 and β to 1. For all experiments, $EML-kNN$ and $ML-kNN$ were trained on the training data and evaluated on the test data of each of the two datasets.

To take an idea about the influence of the parameter γ on the performance of the proposed algorithm, we evaluated version 1 of our method on the emotion dataset where k was fixed to 10 and γ was varied from 0 to 2 with 0.01 steps. Figure 1 shows the results. For the different values of γ, our algorithm performs better than $ML-KNN$ for all criteria except *Precision*. Based on (26) and (27), we can notice that for small values of γ, we favor the allocation of mass to the

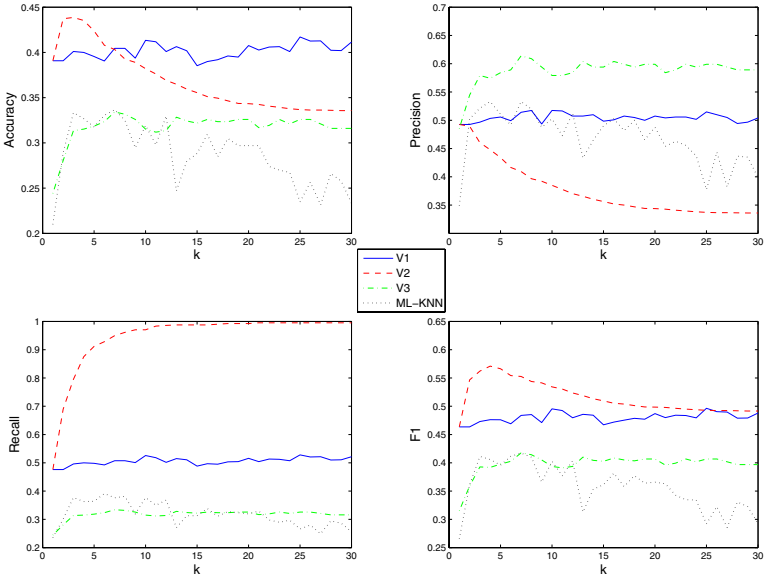


Fig. 2. *Accuracy*, *Precision*, *Recall* and $F1$ measures for the three versions of $EML-kNN$ and $ML-kNN$ algorithms as a function of k on the emotion dataset, for $\gamma = 0.1$

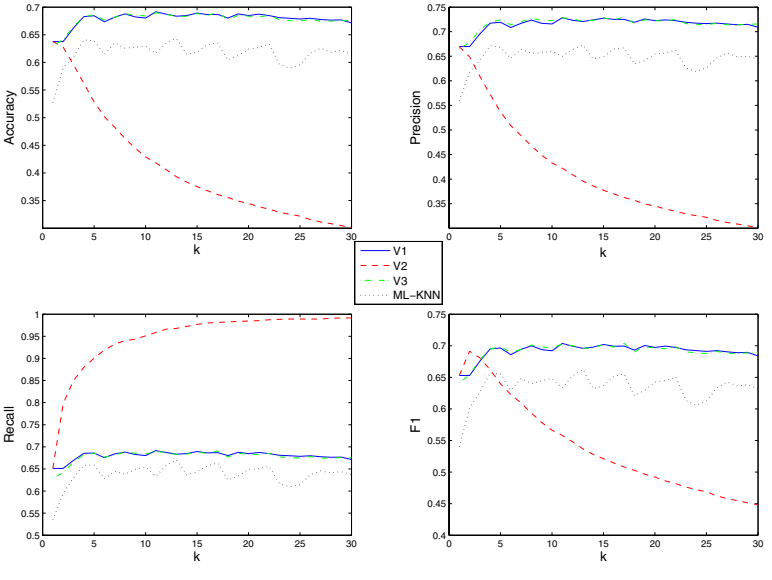


Fig. 3. *Accuracy*, *Precision*, *Recall* and $F1$ measures for the three versions of $EML-kNN$ and $ML-kNN$ algorithms as a function of k on the scene dataset, for $\gamma = 0.1$

set of label sets $\varphi(A_i, B_i)$ that corresponds to the item of evidence $(\mathbf{x}_i, Y_i)$. In contrast, for larger values of γ, a larger fraction of the mass is assigned to the ignorance set $\varphi(\emptyset, \emptyset)$.

In a second step, γ was fixed to 0.1 and k was varied from 1 to 30. Figures 2 and 3 show the performance of the three versions of $EML - kNN$ (denoted by V1, V2 and V3) and the $ML - kNN$ algorithms on the emotion and scene datasets, respectively. Algorithm V1 yields the better performance on the emotion dataset based on all criteria. On the scene dataset, algorithms V1 and V3 yield similar results and both outperform $ML - KNN$ for the different values of k and for all evaluation measures. Algorithm V2 yields poor results for all values of k and for all metrics except *Recall*. We recall that for version 2 of $EML - kNN$, given an item of evidence $(\mathbf{x}_i, Y_i)$, the belief is allocated to the set Y_i and all its supersets. For higher values of k, given an unseen instance $\mathbf{x}$, the most plausible label set after pooling the BBAs induced by the k nearest neighbors will be the set of all labels, i.e., the predicted label set will be $\hat{Y} = \mathcal{Y}$. That explains the fact that the *Recall* measure tends to 1 while the *Precision* measure decreases when the value of k increases.

5 Conclusion

In this paper, an evidence-theoretic k-NN rule for multi-label classification has been presented. Using the evidence theory makes us able to handle the ambiguity and making decisions with multiple possible label sets for an unseen instance without having to resort to assumptions about these sets. The proposed method generalizes the single-label evidence-theoretic k-NN rule to the multi-label case. An unseen instance is classified on the basis of its k nearest neighbors. Each neighbor of an instance to be classified is considered as an item of evidence supporting some hypotheses regarding the set of labels of this instance. A first approach consists in supporting the hypothesis that the label set of the unseen instance is identical to the label set of the ith neighbor considered as an item of evidence. A second one consists in supporting the label set of the ith neighbor and all its supersets. The hypotheses supported by a third approach are the label set of the ith neighbor and all its subsets. The experiments on two real datasets demonstrate the effectiveness of the proposed method as compared to state-of-the-art method also based on the k-NN principle. Especially, the first and the third approaches gave better performance than the second one.

Another contribution of this paper is the presentation of an extension of the D-S theory to manipulate multi-labelled data. In the multi-label case, the frame of discernment defined as the set of all hypotheses in a certain domain is not the set of all possible classes but the powerset of this set. Thus, each hypothesis represents a set of labels and the uncertainty is then expressed by evidence on sets of label sets.

References

1. Boutell, M.R., Luo, J., Shen, X., Brown, C.M.: Learning multi-label scene classification. Pattern Recognition 37(9), 1757–1771 (2004)
2. Crammer, K., Singer, Y.: A Family of Additive Online Algorithms for Category Ranking. In: Proc. of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 151–158 (2002)
3. Denœux, T.: A k-nearest neighbor classification rule based on Dempster-Shafer theory. IEEE Trans. on Systems, Man and Cybernetics 25(5), 804–813 (1995)
4. Denœux, T., Smets, P.: Classification using Belief Functions: the Relationship between the Case-based and Model-based Approaches. IEEE Trans. on Systems, Man and Cybernetics B 36(6), 1395–1406 (2006)
5. Denœux, T., Zouhal, L.M.: Handling possibilistic labels in pattern classification using evidential reasoning. Fuzzy Sets and Systems 122(3), 47–92 (2001)
6. Elisseeff, A., Weston, J.: Kernel methods for multi-labelled classification and categorical regression problems. Advances in Neural Information Processing Systems 14, 681–687 (2002)
7. Konstantinos, T., Tsoumakas, G., Kalliris, G., Vlahavas, I.: Multi-label classification of music into emotions. In: Proc. of the 9th International Conference on Music Information Retrieval (ISMIR 2008), Philadelphia, PA, USA (2008)
8. McCallum, A.: Multi-Label Text Classification with a Mixture Model Trained by EM. In: Working Notes of the AAAI 1999 Workshop on Text Learning (1999)
9. Schapire, R.E., Singer, Y.: BoosTexter: A Boosting-based System for Text Categorization. Machine Learning 39(2-3), 135–168 (2000)
10. Shafer, G.: A mathematical theory of evidence. Princeton University Press, Princeton (1976)
11. Smets, P.: The combination of evidence in the Transferable Belief Model. IEEE Trans. on Pattern Analysis and Machine Intelligence 12(5), 447–458 (1990)
12. Tsoumakas, G., Katakis, I.: Multi-Label Classification: An Overview. International Journal of Data Warehousing and Mining 3(3), 1–13 (2007)
13. Yang, Y.: An evaluation of statistical approaches to text categorization. Journal of Information Retrieval 1, 78–88 (1999)
14. Younes, Z., Abdallah, F., Denœux, T.: Multi-label classification algorithm derived from k-nearest neighbor rule with label dependencies. In: Proc. of the 16th European Signal Processing Conference, Lausanne, Switzerland, August 25–29 (2008)
15. Zhang, M.-L., Zhou, Z.-H.: ML-KNN:A lazy learning approach to multi-label learning. Pattern Recognition 40(7), 2038–3048 (2007)

Author Index