Leora Morgenstern Petros Stefaneas
François Lévy Adam Wyner
Adrian Paschke (Eds.)

Theory, Practice, and Applications of Rules on the Web

LNCS 8035

7th International Symposium, RuleML 2013
Seattle, WA, USA, July 2013
Proceedings

Springer

Lecture Notes in Computer Science 8035

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Leora Morgenstern Petros Stefaneas
François Lévy Adam Wyner
Adrian Paschke (Eds.)

Theory, Practice, and Applications of Rules on the Web

7th International Symposium, RuleML 2013
Seattle, WA, USA, July 11-13, 2013
Proceedings

Springer

Volume Editors

Leora Morgenstern
SAIC / Leidos, Arlington, VA, USA
E-mail: leora.morgenstern@saic.com

Petros Stefaneas
National Technical University of Athens, Attiki, Greece
E-mail: petros@math.ntua.gr

François Lévy
Université Paris 13, Villetaneuse, France
E-mail: francois.levy@lipn.univ-paris13.fr

Adam Wyner
University of Aberdeen, Scotland, UK
E-mail: adam@wyner.info

Adrian Paschke
Freie Universität Berlin, Germany
E-mail: paschke@inf.fu-berlin.de

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-39616-8 e-ISBN 978-3-642-39617-5
DOI 10.1007/978-3-642-39617-5
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2013942664

CR Subject Classification (1998): I.2.4, H.3.5, I.2.6, D.2, I.2.11, H.4.1, F.3.2, D.1.6, J.1

LNCS Sublibrary: SL 2 – Programming and Software Engineering

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

Printed on acid-free paper

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

Preface

RuleML 2013, the 7th International Web Rule Symposium: Theory, Practice, and Applications of Rules on the Web, served, as have all preceding RuleML meetings, as the premier place for theoreticians and practitioners from a wide range of areas of rule technologies to meet and exchange ideas. The aim of RuleML 2013 was to build bridges between academia and industry in the field of rules and semantic technology. By bringing together rule-system providers, participants in rule standardization efforts, open source communities, practitioners, and researchers, the RuleML symposium series stimulates cooperation and interoperability between business and research. This annual symposium is the flagship event of the Rule Markup and Modeling (RuleML) Initiative. RuleML 2013 (July 11–13, 2013) was collocated with AAAI 2013, the 27th Conference on Artificial Intelligence, in Seattle, Washington.

The RuleML Initiative (http://ruleml.org) is a non-profit umbrella organization of several technical groups organized by representatives from academia, industry, and national governments who are working on rule technology and its applications. Its aim is to advance the study, research, and application of rules in heterogeneous distributed environments such as the Web. RuleML maintains effective links with other major international societies and acts as intermediary between various specialized rule vendors, applications, industrial, and academic research groups; it also maintains effective links with other major international societies and standardization efforts, including W3C, OMG, OASIS, and ISO.

The RuleML Symposium series began as an annual series of workshops in 2002, and has been run as an annual international symposium since 2007.

The core technical program for Rule ML 2013 consisted of a main technical conference Track, a special Human Language Technology Track focusing on business and legal regulations, a Rule Challenge Track, and a Doctoral Consortium.

The main technical track included 12 papers on a variety of aspects of research on rules, including rule-based approaches for spatial reasoning, answer-set programming methods for solving the stable marriage problem, using rules for complex event processing for gamification applications, and using defeasible rules to formalize theories of belief, desire, intention, and obligation in order to choose optimal goals. It also featured several keynote and invited talks from leaders in our community: a keynote talk by Michael Grüninger on integrating ontologies within repositories, a keynote talk by Lise Getoor on a framework for integrating probabilistic and relational reasoning, and an invited talk by Benjamin Grosof, describing his work on rapid text-based authoring of higher-order defeasible rules. In addition, this track included two tutorials, one on LegalRuleML, a rule interchange language for legal domains standardized by OASIS, and one on Brahms, a set of software tools, based on a theory of work practice and situ-

ated cognition, for developing and simulating multi-agent models of human and machine behavior.

The Human Language Technology Track focused on methods for addressing the knowledge acquisition bottleneck that arises when trying to convert the vast amount of regulatory text on the Web, nearly all of which is written in natural language, to the formal expression of such rules. Topics represented in this track included the use of controlled languages, semi-formal languages that aim to facilitate the expression of regulations by human experts in languages that are both close to natural language and have relatively straightforward mappings to formal representations; techniques for extracting semantic information from legislative text; and mapping English onto fuzzy logic. This track included seven talks and one tutorial on SBVR structured English.

The 7th International Rule Challenge Track highlighted the practical application of rule-based systems. It served as a forum for presenting new ways of using rule-based systems and reporting practical experiences about implementing these systems. The focus was on benchmarks/evaluations, demos, case studies, use cases, experience reports, best practice solutions (such as design patterns, reference architectures, and models), rule-based implementations/tools/applications, demonstrations of engineering methods, implementations of rule standards (such as RuleML, W3C RIF, ISO Common Logic, SBVR, PRR, and API4KB), industrial standards for representing and exchanging rules and related information (such as XBRL, MISMO, and Accord), and industrial problem statements. The Rule Challenge Track also included an invited tutorial on using rules to handle XML in government contexts such as NIEM, the United States National Information Exchange Model.

This was the third year that RuleML held a Doctoral Consortium. Aimed at attracting and promoting PhD research in this area, the Doctoral Consortium offers students close contact and mentoring opportunities with leading experts in the field, as well as the opportunity to present and discuss their ideas in a dynamic and friendly setting.

This volume includes all papers from the main technical track, the top three papers from the Human Language Technology Track, and abstracts of the keynote and invited talks and tutorials. In two cases, for Benjamin Grosof's invited paper and for the LegalRuleML tutorial, extended abstracts are provided. Human Language Technology Track papers that are not published in this volume, as well as Rule Challenge papers and Doctoral Consortium papers, are published as a CEUR Workshop Proceedings, at http://ceur-ws.org/. The five editors of this volume comprise the two Program Chairs of the main technical track, Leora Morgenstern and Petros Stefaneas, the two Program Chairs of the Human Language Technology Track, François Lévy and Adam Wyner, and the Conference Chair, Adrian Paschke. Note that the chairs of the Human Language Technology track were not involved in the selection for this volume of any papers that they authored.

We are pleased to announce the winner of the RuleML 2013 Best Paper Award, "Computing the Stratified Semantics of Logic Programs over Big Data

Through Mass Parallelization" by Ilias Tachmazidis and Grigoris Antoniou. This paper examines how logic programming techniques, particularly stratification, can facilitate mass parallelization in order to handle vast quantities of data.

Special thanks are due to the excellent Program Committee for their hard work in reviewing the submitted papers. Their criticism and very useful comments and suggestions were instrumental in the high quality of the papers. We thank the symposium authors for submitting solid contributions to research, responding to the comments of the reviewers, and abiding by our production schedule. We are grateful to the keynote and invited speakers for contributing thought-provoking talks and providing research leadership for the RuleML community, and to the tutorial presenters for organizing and transmitting their deep and broad knowledge of their areas of expertise.

We thank OASIS LegalXML for its financial support of RuleML 2013 and acknowledge our various industrial, academic, and professional society partners for technical and organizational support. We also thank the EasyChair conference management system for facilitating the paper submission and reviewing process and both EasyChair and our publisher, Springer, for their support in the preparation of this volume and the publication of the proceedings.

Finally, we gratefully acknowledge the assistance and support of Frank Olken, who contributed so much to the organization of this symposium and helped ensure its success.

May 2013

Leora Morgenstern
Petros Stefaneas
François Lévy
Adam Wyner
Adrian Paschke

Organization

Program Committee

Darko Anicic	FZI Forschungszentrum Informatik, Germany
Martin Atzmueller	University of Kassel, Germany
Costin Badica	University of Craiova, Romania
Ebrahim Bagheri	Ryerson University, Canada
Nick Bassiliades	Aristotle University of Thessaloniki, Greece
Bernhard Bauer	University of Augsburg, Germany
Antonis Bikakis	University College London, UK
Pedro Bizarro	University of Coimbra, Portugal
Luiz Olavo Bonino Da Silva Santos	University of Twente, The Netherlands
Johan Bos	University of Groningen, The Netherlands
Lars Braubach	University of Hamburg, Germany
Christoph Bussler	Voxeo Labs, USA
Federico Chesani	University of Bologna, Italy
Horatiu Cirstea	Loria, France
Jack G. Conrad	Thomson Reuters, Switzerland
Claudia D'Amato	University of Bari, Italy
Célia Da Costa Pereira	Université de Nice Sophia, Antipolis, France
Christian De Sainte Marie	IBM, France
Juergen Dix	Clausthal University of Technology, Germany
Schahram Dustdar	TU Wien, Austria
Jenny Eriksson Lundström	Uppsala University, Sweden
Vadim Ermolayev	Zaporozhye National University, Ukraine
Luis Ferreira Pires	University of Twente, The Netherlands
Michael Fink	Vienna University of Technology, Austria
Paul Fodor	Stony Brook University, USA
Enrico Francesconi	ITTIG-CNR, Italy
Fred Freitas	CIn-UFPE, Brazil
Norbert E. Fuchs	University of Zurich, Switzerland
Aldo Gangemi	University of Paris 13, France
Dragan Gasevic	Athabasca University, Canada
Adrian Giurca	BTU Cottbus, Germany
Guido Governatori	NICTA, Australia
Matthias Grabmair	University of Pittsburgh, USA
Christophe Gravier	Université Jean Monnet, France
Giancarlo Guizzardi	Federal University of Espirito Santo (UFES), Brazil
Ioannis Hatzilygeroudis	University of Patras, Greece

Stijn Heymans	SRI International, USA
Yuh-Jong Hu	National Chengchi University, Taiwan
Minsu Jang	Electronics & Telecommunications Research Institute, South Korea
Tobias Kuhn	Yale University, USA
Wolfgang Laun	Thales Rail Signalling Solutions GesmbH, Austria
Francois Levy	University of Paris 13, France
Francesca Alessandra Lisi	Università degli Studi di Bari "Aldo Moro", Italy
Emiliano Lorini	IRIT, France
Yue Ma	TU Dresden, Germany
Michael Maher	University of New South Wales, Australia
Angelo Montanari	University of Udine, Italy
Leora Morgenstern	SAIC, USA
Grzegorz J. Nalepa	AGH University of Science and Technology, Poland
Adeline Nazarenko	University of Paris 13, France
Monica Palmirani	University of Bologna, Italy
Jose Ignacio Panach Navarrete	Universitat de València, Spain
Jeffrey Parsons	Memorial University of Newfoundland, Canada
Adrian Paschke	Freie Universität Berlin, Germany
Wim Peters	University of Sheffield, UK
Giovanni Sartor	European University Institute, Italy
Rolf Schwitter	Macquarie University, Australia
Guy Sharon	IBM, Israel
Davide Sottara	DEIS, University of Bologna, Italy
Petros Stefaneas	NTUA, Greece
Umberto Straccia	ISTI-CNR, Italy
Terrance Swift	CENTRIA, Universidade Nova de Lisboa, Portugal
Daniela Tiscornia	CNR-ITTIG, Italy
Wamberto Vasconcelos	University of Aberdeen, UK
Giulia Venturi	ILC-CNR, Italy
George Vouros	University of Piraeus, Greece
Renata Wassermann	University of Sao Paulo, Brazil
Radboud Winkels	University of Amsterdam, The Netherlands
Adam Wyner	University of Aberdeen, UK
Amal Zouaq	Royal Military College of Canada

Additional Reviewers

Barros, Rafael
Kaczor, Krzysztof
Ribeiro, Ryan
Sciavicco, Guido
Venhuizen, Noortje

RuleML 2013 Organization

Organizing Committee

General Chair

Adrian Paschke	Freie Universität Berlin, Germany

Program Chairs

Leora Morgenstern	SAIC, USA
Petros Stefaneas	National Technical University of Athens, Greece

Local Organization Chair

Benjamin Grosof	Benjamin Grosof and Associates, LLC, USA

Track Chairs, Special Track on Human Language Technology

François Lévy	University of Paris North, France
Adam Wyner	University of Aberdeen, UK

Rule Challenge Chairs

Darko Anicic	FZI, Germany
Paul Fodor	SUNY at Stony Brook, USA
Dumitru Roman	SINTEF / University of Oslo, Norway
Adam Wyner	University of Aberdeen, UK

Doctoral Consortium Chairs

Monica Palmirani	Università di Bologna C.I.R.S.F.I.D., Italy
Davide Sottara	Arizona State University, USA

Publicity Chairs

Robert Golan	DBMind, USA
Dumitru Roman	Sintef, Norway
Xing Wang	Northeastern University Shenyang, China

Rule Responder Chairs

Adam Wyner	University of Aberdeen, UK
Zhili Zhao	Freie Universität Berlin, Germany

Web Chair

Alexandru Todor Freie Universität Berlin, Germany

Sponsor

Media Partner

Table of Contents

Invited Talks

Probabilistic Soft Logic: A Scalable Approach for Markov Random Fields over Continuous-Valued Variables (Abstract of Keynote Talk) ... 1
Lise Getoor

Rapid Text-Based Authoring of Defeasible Higher-Order Logic Formulas, via Textual Logic and Rulelog (Summary of Invited Talk) 2
Benjamin N. Grosof

Ontology Repositories Make a World of Difference (Abstract of Keynote Talk) 12
Michael Grüninger

Tutorials

LegalRuleML: From Metamodel to Use Cases (A Tutorial) 13
Tara Athan, Harold Boley, Guido Governatori, Monica Palmirani, Adrian Paschke, and Adam Wyner

Formalization of Natural Language Regulations through SBVR Structured English (Tutorial) 19
François Lévy and Adeline Nazarenko

Multi-agent Activity Modeling with the Brahms Environment (Abstract of Tutorial) 34
Maarten Sierhuis

Rules and Policy Based Handling of XML in Government Contexts Including NIEM (Abstract of Tutorial) 36
David Webber

Technical Papers, Main Track

Reasoning over 2D and 3D Directional Relations in OWL: A Rule-Based Approach 37
Sotiris Batsakis

Grailog 1.0: Graph-Logic Visualization of Ontologies and Rules 52
Harold Boley

Modeling Stable Matching Problems with Answer Set Programming 68
Sofie De Clercq, Steven Schockaert, Martine De Cock, and Ann Nowé

A Fuzzy, Utility-Based Approach for Proactive Policy-Based Management 84
Christoph Frenzel, Henning Sanneck, and Bernhard Bauer

Picking Up the Best Goal: An Analytical Study in Defeasible Logic 99
Guido Governatori, Francesco Olivieri, Antonino Rotolo, Simone Scannapieco, and Matteo Cristani

Computing Temporal Defeasible Logic 114
Guido Governatori and Antonino Rotolo

Efficient Persistency Management in Complex Event Processing: A Hybrid Approach for Gamification Systems 129
Philipp Herzig, Bernhard Wolf, Svenja Brunstein, and Alexander Schill

Ontology Patterns for Complex Activity Modelling 144
Georgios Meditskos, Stamatia Dasiopoulou, Vasiliki Efstathiou, and Ioannis Kompatsiaris

A Rule-Based Contextual Reasoning Platform for Ambient Intelligence Environments 158
Assaad Moawad, Antonis Bikakis, Patrice Caire, Grégory Nain, and Yves Le Traon

Extending an Object-Oriented RETE Network with Fine-Grained Reactivity to Property Modifications 173
Mark Proctor, Mario Fusco, and Davide Sottara

Computing the Stratified Semantics of Logic Programs over Big Data through Mass Parallelization 188
Ilias Tachmazidis and Grigoris Antoniou

Distributed ECA Rules for Data Management Policies 203
Hao Xu, Arcot Rajasekar, Reagan W. Moore, and Mike Wan

Technical Papers, Human Language Technology Track: Translating between Human-Created Regulations and Formal Rules

Semantic Relation Extraction from Legislative Text Using Generalized Syntactic Dependencies and Support Vector Machines 218
Guido Boella, Luigi Di Caro, and Livio Robaldo

Interpreting Spatiotemporal Expressions from English to Fuzzy Logic . . . 226
William R. Murray, Philip Harrison, and Tomas Singliar

Combining Acquisition and Debugging of Business Rule Models 234
Adeline Nazarenko and François Lévy

Author Index 249

Probabilistic Soft Logic: A Scalable Approach for Markov Random Fields over Continuous-Valued Variables

(Abstract of Keynote Talk)

Lise Getoor

Department of Computer Science
University of Maryland at College Park
College Park, MD 20740

Many problems in AI require dealing with both relational structure and uncertainty. As a consequence, there is a growing need for tools that facilitate the development of complex probabilistic models with relational structure. These tools should combine high-level modeling languages with general purpose algorithms for inference in the resulting probabilistic models or probabilistic programs. A variety of such frameworks has been developed recently, based on ideas from graphical models, relational logic, or programming languages. In this talk, I will give an overview of our recent work on probabilistic soft logic (PSL), a framework for collective, probabilistic reasoning in relational domains. PSL models have been developed in a variety of domains, including collective classification, entity resolution, ontology alignment, opinion diffusion, trust in social networks, and modeling group dynamics.

A key distinguishing feature of PSL is its use of continuous-valued random variables. These can either be interpreted as soft truth values in the interval [0; 1] or as similarities. It uses first order logic rules to capture the dependency structure of the domain, based on which it builds a joint probabilistic model over all random variables. A PSL program defines a form of Markov random field over continuous-valued random variables which is computationally tractable. Inference in PSL corresponds to a convex optimization problem, which can be solved significantly more efficiently than the corresponding discrete optimization. Our recent results show that by using state-of-the-art optimization methods and distributed implementations, we can solve problems over millions of variables in minutes rather than days.

L. Morgenstern et al. (Eds.): RuleML 2013, LNCS 8035, p. 1, 2013.

Rapid Text-Based Authoring of Defeasible Higher-Order Logic Formulas, via Textual Logic and Rulelog
(Summary of Invited Talk)

Benjamin N. Grosof

Benjamin Grosof & Associates, LLC, USA

Abstract. We present textual logic (TL), a novel approach that enables rapid semi-automatic acquisition of rich logical knowledge from text. The resulting axioms are expressed as defeasible higher-order logic formulas in Rulelog, a novel extended form of declarative logic programs. A key element of TL is textual terminology, a phrasal style of knowledge in which words/word-senses are used directly as logical constants. Another key element of TL is a method for rapid interactive disambiguation as part of logic-based text interpretation. Existential quantifiers are frequently required, and we describe Rulelog's approach to making existential knowledge be defeasible. We describe results from a pilot experiment that represented the knowledge from several thousand English sentences in the domain of college-level cell biology, for purposes of question-answering.

1 Introduction and Requirements Analysis

1.1 Reducing the Cost of Authoring Rich Logical Knowledge

A key goal in the field of expressive knowledge representation and reasoning (KRR) is to reduce the cost of authoring rich logical knowledge.

Rulelog is an expressive knowledge representation logic that is an extended form of declarative logic programs (LP). Rulelog transforms into normal (unextended) LP. Previous work on Rulelog, implemented in XSB [12,10], Flora-2 [3], SILK [9], and Cyc [2], has developed novel methods that help improve scale-able evolution and combination of such KB's, and thus the cost of overall knowledge acquisition (KA). These methods enable: defeasibility, based on argumentation theories (AT's) [11], i.e., *AT-defeasibility*; higher-order syntax, based on hilog [1] and other meta-knowledge enabled by rule id's, i.e., *hidlog*; bounded rationality *restraint*; [6]; interactive authoring, based on a rapid edit-test-inspect loop and incremental truth maintenance; knowledge debugging, based on a graphical integrated development environment with justification graphs and reasoning trace analysis; and knowledge interchange, based on strong semantics and knowledge translations. Rulelog's full set of major features was first implemented in SILK [9]. A W3C RIF dialect based on Rulelog is in draft [9], in cooperation also with RuleML [8].

In this work on Rulelog, we present another, more radical step that we have developed in order to further reduce such cost: a method that enables text-based

L. Morgenstern et al. (Eds.): RuleML 2013, LNCS 8035, pp. 2–11, 2013.

authoring, based on a novel approach called *textual logic (TL)*. We also present a novel expressive feature of Rulelog: *omniformity*, which permits defeasible existentials, and is used to support TL.

1.2 In Quest of a Dream

"In dreams lie responsibilities" — Delmore Schwartz.

> "Classic knowledge-based AI [artificial intelligence] approaches to QA [question-answering] try to logically prove an answer is correct from a logical encoding of the question and all the domain knowledge required to answer it. Such approaches are stymied by two problems: the prohibitive time and manual effort required to acquire massive volumes of knowledge and formally encode it as logical formulas accessible to computer algorithms; and the difficulty of understanding natural language questions well enough to exploit such formal encodings if available. Techniques for dealing with huge amounts of natural language text, such as Information Retrieval, suffer from nearly the opposite problem in that they can always find documents or passages containing some keywords in common with the query but lack the precision, depth, and understanding necessary to deliver correct answers with accurate confidences." — IBM Watson FAQ.

What if was "cheap" to acquire massive volumes of knowledge formally encoded as logical formulas?

What if it was "easy" to understand natural language questions well enough to exploit such formal encodings?

A central dream for semantic technology is to make knowledge (K) and reasoning be deeper and cheaper — to overcome the famous "knowledge acquisition bottleneck" of AI. That would enable creation of widely-authored, very large knowledge bases (KB's) that automatically answer sophisticated questions (Q's) and proactively supply info, about science, business, and government, with (collectively) broad and deep comprehensiveness.

These KB's would *harness humanity's accumulated storehouse of knowledge*, and be *learned from communication and instruction*, as well as from observation. Such harnessing and learning is *far more powerful than learning from individual experience*. Yet machine learning has not primarily focused on it, to date.

Achieving this dream could create huge amounts of social value. In the remainder of this presentation, we discuss technical requirements in the context of this dream.

1.3 Logical Expressiveness

Logical knowledge is desirable for several reasons. First, *accuracy*: it can provide high-precision of answers. Second, *transparency*: it can provide detailed justifications/explanations. Third, *sharability*: particularly when semantic and

semantic-web-friendly, it facilitates reusability and merging of larger KB's, via knowledge interchange.

Expressive richness of logical knowledge is desirable because it enables more kinds of knowledge and reasoning, e.g., scientific, to be represented and automated. Richness is required to represent the logical substance of many text statements, which involves: negation, modifiers, quantifiers (for-all, exists), implication, and conjunction — all flexibly composed. Also, some text statements are about other statements. Thus text requires expressiveness equivalent to that of: first-order-logic (FOL) formula syntax, plus some meta expressiveness, especially higher-order syntax.

Another requirement for expressiveness is that the logic must be *defeasible*, so that it can gracefully handle exceptions and change. Defeasibility is needed: to represent the empirical character of knowledge; to aid the evolution and combination of KB's, i.e., to be *socially scalable*; and to represent causal processes and "what-if's" (hypotheticals, e.g., counterfactual). In other words, defeasibility is needed to represent *change in knowledge and change in the world*. Yet, despite this expressive richness, inferencing in the logic must be computationally scalable, and thus at least *tractable* (worst-case polynomial-time). SPARQL and SQL databases are tractable, for example.[1]

Defeasibility of knowledge that is existentially quantified, in particular, is required expressively to support TL. Existentials appear in many naturally arising text sentences that must be represented as defeasible, e.g., about biology. For example, in the relatively simple sentence "Each eukaryotic cell has a visible nucleus.", the "a" before "nucleus" is an existential. Yet this statement has exceptions. Red blood cells are eukaryotic, yet lack nuclei. Eukaryotic cells lack visible nuclei during anaphase, a step within the process of cell division.

1.4 Text-Based Authoring

Text-based authoring is desirable for several reasons. Natural language (NL) — not logic — is the language of "business users", for KA and also for QA. NL is required for broad accessibility by the knowledgeable community of (potential) contributors, in science and many similar areas. In particular, NL is much more broadly accessible and familiar to subject matter experts (SMEs), as opposed to knowledge engineers (KEs) trained in logic. Examples of SME's include scientists, business process owners, executives, lawyers, doctors, educators, engineers, analysts, civil servants, merchants, soldiers, chefs, and members of many other occupations. NL is required also for ordinary end users, e.g., students, citizens, shoppers, salespersons, clerks, and patients — i.e., for the community of (potential) "consumers" of the knowledge in science and many similar areas. Even KE's usually find much easier to articulate and understand text than logic. Most of the world's knowledge is currently described in text, so working from text sources is crucial. Economic scalability of KA thus requires authoring to be text-based, rather than directly in strict logical syntax which requires KE skill.

[1] i.e., for querying, when the number of distinct logical variables per query is bounded; this is often described in terms of data complexity being tractable.

Economic scalability requires not only that the authoring be accessible in this regard, but also that it take a relatively low amount of effort per sentence. This implies, first, that the encoding *text should not be onerously restricted*. Second, there must be methods for *rapid disambiguation with logical and semantic precision*.

Previous approaches to text-based KA of rich logical K have suffered from major difficulties. One category of approaches permit the input text to be fairly unrestricted NL. Next, we consider that category. Natural language processing (NLP) technology has not yet been able (in general, i.e., reliably) to "go all the way" in fully automatic text interpretation: much of the semantics is not captured. Substantial further disambiguation is needed, for most sentences. Also, the NLP field has been messy and unstandardized in regard to components and forms of info. It has been weak architecturally in regard to flexibly composable, reusable components. Thus fully automatic NLP has typically produced logical K that is inaccurate (quite noisy), and/or shallow/partial (thus inoperational for desirably effective QA). Fully automatic NLP has also tended to produce K that is opaque, i.e., difficult to understand; often, one sees K that is in terms of the parser's innards. For KE author to finish the text interpretation, they tend to need skills in not only logic, but also NLP as well, i.e., to be a "super" KE. In consequence overall, it has been quite costly to do text-based KA of rich logical K, and it has hardly been used practically, especially compared to the dream.

A second category of approaches compromises by only allowing "controlled" NL, i.e., restricts the vocabulary, grammar, and/or phrases. This typically requires much upfront phraseological work to define exactly what's allowed. It also requires the author to become familiar with the particular phraseology and its restrictions. It still requires the author to have KE skills if the generated logic is rich. In consequence overall, it has been still quite costly and not used practically to nearly the extent envisioned in the dream.

2 Textual Logic

TL overall is a logic-based approach to both text interpretation and text generation, for both KA and question answering (QA). In TL: text is mapped to logic; logic is mapped to text; and these mappings themselves are based on logic.

The spirit of TL has a "Gettysburg" principle: "logic for the text, of the text, by the text".[2] "*For* the text" means for the sake of text, in that knowledge and questions are input in text form, and answers and explanations are output in text form. "*Of* the text" means that the content of the text is represented in logic, and that the mappings in and out of text are represented in logic. "*By* the text" means that once there is, sufficiently, logic for the text and of the text, then logical knowledge can be specified by text and viewed as text.

[2] "Gettysburg" here refers to Abraham Lincoln's Gettysburg address (1863) in which he said that the cause for which the Union soldiers died (in the great Civil War battle of Gettysburg) was that "government of the people, by the people, and for the people, shall not perish from the earth.

A novel aspect of TL is *textual terminology* — a phrasal style of knowledge. Words, and more generally word senses, are employed directly as logical constants. Each constant is a hilog functor (i.e., a function or predicate). A textual phrase corresponds (one-to-one) to a logical term; there is a natural style of composition.

Another novel aspect of TL is that it leverages defeasibility. "The thing about NL is that there's a gazillion special cases."[3].

During TL text interpretation, authors (1.) *articulate* sentences in text, then (2.) logically *disambiguate* those sentences, and (3.) *generate* logical axioms as output. These three steps are, in general, *interactive*, i.e., semi-automatic. Multiple authors may collaborate in these steps, for each sentence, including to divide the labor, edit, and review/comment/rate.

Next we describe particulars for our TL work to date, which we call *TL phase 1 (TL1)*, on text interpretation. The text is in English. The Linguist^{TM} tool from Automata, Inc. was employed, together with SILK. Linguist^{TM} leverages the ERG lexical ontology and associated PET parser, which are open source. ERG has a large vocabulary and broad coverage. Disambiguation (step (2.)) is highly interactive, via a novel GUI-based approach that enables users to disambiguate relatively rapidly. Disambiguation has several sub-steps in which an author specifies additional information, as needed, about: (a.) the parse; (b.) quantifier types and scopes; (c.) co-references; and (d.) word senses. Word sense, so far, is limited in need/use/implementation. Logic generation (step (3.)) is fully automatic, and outputs Rulelog axioms. One *main* axiom is generated for each disambiguated (text) sentence. In addition, *support* axioms are generated that represent auxiliary information, e.g., about paraphrases and types. The support axioms are used together with the main axioms for purposes of inferencing. Other annotation axioms are generated, also, as part of comprehensively capturing the info specified during disambiguation. Articulation (step (1.)) is fully manual. The (text) *encoding* sentence it produces must meet two restrictions, but those restrictions are not onerous for purposes of KA. The first restriction on text is that the sentence be *stand-alone*, i.e., (nominal) co-reference is within a sentence, not between sentences. The second restriction is that the text be *straightforward*, i.e., it should minimize ellipsis (missing words), rhetoric, and metaphor. In the articulation step, sentences may be drawn from a *source* text and then reformulated. E.g., in our pilot TL KA experiment, sentences were drawn from a first-year college-level biology textbook chapter on membranes. Some textbook sentences were used verbatim, i.e., not changed during articulation. However, other textbook sentences were not stand-alone straightforward text, or were too long to be most productively disambiguated, thus were reformulated during articulation to clarify or break them up into multiple (encoding) sentences.

For example, a source sentence (with id ss72) is "Some transport proteins, called channel proteins, function by having a hydrophilic channel that certain molecules or atomic ions use as a tunnel through the membrane (see Figure 7.10a, left).".

[3] Peter E. Clark, private communication.

The KE articulates a foreground encoding sentence (with id es2298) based on ss72: "Channel proteins have a hydrophilic channel.". Another KE disambiguates es2298, producing a Rulelog axiom with formula

```
forall(?x5)^(channel(protein)(?x5) ==>
    exist(?x8)^(have(?x5,?x8) and hydrophilic(channel)(?x8));
```

(here, shown in SILK's human-consumption syntax). This axiom is defeasible and includes meta-facts about its prioritization tag, author, creation-time, etc. (not shown above).

TL1 also includes two text-oriented extensions within: the KRR/KA system for Rulelog itself (both of these were implemented in SILK). The first is to employ simple text generation that is specified by KB's (i.e., rules). The generated text is displayed in the UI for KA and related tasks, notably for viewing justifications [5]. The second is (a subset of) *textic*, a novel technique for fine-grain mixing of text-style syntax with logic-style syntax. Textic is an extension of the concrete (human-consumption) syntax of Rulelog's KR language, that improves readability and writeability. In textic, a word is typically treated as a functor having arity 1, and a space is typically interpreted as a left parenthesis. The textic expressive syntactic feature is defined by a deterministic (reduction) transformation, similar in spirit to several other Rulelog features such as hilog, defeasibility, head conjunction, and body disjunction.

3 Omniform Rules

Rulelog is a logical extension of declarative logic programs (LP) that has a unique set of logical features including hidlog, AT-defeasibility, omniformity, and restraint. Rulelog transforms into normal LP.

Next, we describe omniformity, which enables defeasible knowledge to be existential.

An omniform rule (*omni* for short) permits a rule head and body each to be any formula in first-order logic (FOL) syntax. I.e., each can be a formula composed freely from the usual first-order logic (FOL) connectives (disjunction as well as conjunction and (strong) negation) and quantifiers (existential as well as universal). In addition, the head and body formulas each may employ hilog, and thus be higher-order (HOL), rather than first-order, in their syntax. (Note that hilog does impose a few non-onerous restrictions as compared to full higher-order syntax, e.g., the head formula may not be simply a single variable.) Furthermore, the body may employ negation-as-failure (naf) but (as usual in LP) only outside the scope of strong negation (neg). For convenience, the usual FOL implication and equivalence connectives are also permitted.

The semantics of the omniformity feature is defined via a transformation that reduces any omniform rule to a set of one or more rules that are *conjunctive*. A conjunctive rule is one whose head is a single literal, and whose body is a conjunction of literals. (A literal is an atom preceded possibly by neg and/or naf; as usual in LP, naf must not appear within the scope of neg.) This omni transformation OT generalizes three transformations previously employed in SILK:

(head) omnidirectionality [4], Lloyd-Topor [7], and head conjunction splitting. Splitting here means a rule $(H1\ and\ ...\ and\ Hn)\ :-\ B;$ is transformed into a set of n rules: $\{Hi\ :-\ B\ ;\ |\ i = 1, ..., n\}$.

In *OT*, *neg* is first driven to be innermost. Then the head is put into *tight normal form (TNF)*, by pushing: *exist* inward past *or*; $forall$ inward past *and*; $forall$ inward past *or* when the disjunct does not mention the $forall$'s quantified variable; and *exist* inward past *and* when the conjunct does not mention that *exist*'s quantified variable. Then, recursing top-down on the expression tree: existentials are skolemized; disjunctions are directionalized, cf. omni-directionality, but generalized to non-literal expressions; and conjunctions are split. *OT* transforms the body in a manner similar to Lloyd-Topor. TNF addresses a subtlety that directionalizing should be done "before" skolemizing. TNF differs, in general, from Skolem normal form (used in FOL resolution theorem proving).

We have also developed a new family of argumentation theories, called ATCO, that improves the behavior of (AT-)defeasibility in combination with omniformity, particularly with existentials, as compared to previous AT's.

4 Experiment: Case Study

We conducted a TL1 KA experiment during January-March 2013, that resulted in a case study in the rapid acquisition of rich logical knowledge from one chapter (on cell membranes) of a popular college-level biology textbook, with implications for biomedical education and research. A distributed team of collaborators — knowledge engineers (KE's) — started from effectively unconstrained natural language text and disambiguated various aspects of English sentences, semi-automatically translating text into defeasible higher-order logic formulas expressed in Rulelog, an extended form of declarative logic programs and a draft W3C RIF dialect, implemented in SILK. The distributed team's workflow authored and curated the knowledge base from the text into several thousand Rulelog axioms targeting question answering by a Digital Aristotle as part of Vulcan Inc.s Project Halo.

In this TL1 KA experiment, about 2,500 English encoding sentences were axiomatized. These included hundreds of questions.

A number of questions, some of them sophisticated, answered successfully using Rulelog inferencing (in SILK) on the axioms. However, due to resource limitations of the study, only relatively limited tests of question-answering (QA) were conducted. The focus of the experiment was on KA productivity, primarily, and KA coverage, secondarily.

Encoding sentence length averaged 10 words and ranged up to 25 words. One main defeasible axiom in Rulelog (SILK syntax) resulted from each sentence. On average, each such main axiom transformed into over 5 rules in normal (unextended) LP.

It took less than 10 minutes (of KE labor) on average per sentence to: author, disambiguate, formalize, review, and revise a sentence.

One should expect in future that more intensive QA testing, with attendant debugging of knowledge, would tend to increase the amount of KE labor effort on average per sentence.

On the other hand, one should expect in future that KA tooling and process improvements would tend to decrease the amount of KE labor effort on average per sentence.

Some book source sentences were also encoding sentences, i.e., were disambiguated verbatim from the book. More frequently, one book source sentence was articulated into two or three encoding sentences before disambiguation, in order to make them clearer and easier to disambiguate.

Collaboration resulted in an average of over 2 authors/editors/reviewers per sentence. Collaborative review and revision of the sentences, their disambiguation, and formalization, approximately doubled the average time per sentence.

The resulting axioms were typically more sophisticated than what skilled KE's typically produce when directly authoring into logical syntax.

The number of candidate parses (generated from the lexical ontology (ERG), essentially) per sentence averaged over 30, and commonly ranged into the hundreds. Disambiguation of the parse alone typically required a fraction of a minute. Typically the correct parse was not the parse ranked best by statistical natural language processing.

Expressive coverage was very good, due to Rulelog's expressiveness: all sentences encountered were representable. Terminological coverage was also very good, due to the textual terminology aspect of the TL approach: little handcrafted logical ontology was required. Several hundred mostly domain-specific lexical entries were added to the ERG. There were some small (less than a few percent) shortfalls from implementation issues, in terminological coverage and in reasoning (e.g., about numerics) related to expressive coverage.

5 Discussion

In the experiment, the KE labor cost for TL1 KA was very roughly USD $3–4/word (actual word, not simply 5 characters). That implies a cost of very roughly USD $500–1500/page (at roughly 175–350 words/page). That is in the same ballpark as the cost of the labor to author the *text itself*, for many formal text documents, e.g., college science textbooks and some kinds of business documents. "Same ballpark" here means same order of magnitude.

The approach of Textual Logic plus Rulelog has major advantages for KA.

- Interactive disambiguation: relatively rapidly produces rich K with logical and semantic precision, starting from effectively unconstrained text.
- Textual terminology: greatly reduces the need for KE labor to specify logical ontology explicitly and to become familiar with it, since logical ontology instead emerges naturally and automatically from the texts phrasings. Textual terminology, and textic, also provide a bridge to work in text mining and "textual entailment".

- Rulelog as rich target logic: can handle exceptions and change and, moreover, is computationally tractable[4], due to its defeasibility and restraint features, respectively.
- Rulelog supports knowledge interchange (translation and integration) with: both LP and FOL; all the major semantic tech/web standards (RDF(S), SPARQL, OWL, RIF, CL, SBVR); Prolog, SQL, and production rules. (Although for many of these, with restrictions.)

The approach appears to be significant progress on the famous "KA bottleneck" of AI. It provides "better, faster, cheaper" logical knowledge. That logical knowledge is usable on a variety of KRR platforms.

It's early days still in developing and pursuing this approach, so lots of future work remains to do. One direction is tooling, e.g., to leverage inductive learning to aid disambiguation. Another direction is more KA experiments, e.g.: to push on QA; and to scale up.

A third direction is to try out the approach in various applications. In terms of system architecture, this usage context will often call for specialized UI and service interfaces from apps to TL. Rulelog KRR can make use of databases and other service resources in the apps-relevant environment.

6 Summary

Rich logical knowledge is desirable for its accuracy, transparency, coverage depth, and reusability. Economically scalable KA and QA of rich logical knowledge require methods for: (1.) rapid disambiguation in text-based authoring; and (2.) defeasibility plus tractability in the logical KRR. Textual Logic plus Rulelog is a step forward in both regards. Future directions include more on tooling, more ambitious experiments, and exploring applications.

Acknowledgements. This work was partly supported by Vulcan, Inc., as part of the Halo Advanced Research (HalAR) project, which the author led and which centered around the development of Rulelog and SILK. This presentation describes work done by the author jointly with Paul V. Haley (Automata, Inc., USA), Carl Andersen (Raytheon BBN Technologies, USA), Brett Benyo (Raytheon BBN Technologies, USA), Michael Kifer (Stony Brook University, USA), and Paul Fodor (Stony Brook University, USA). Thanks to the entire HalAR team for helpful discussions and supporting implementation. Thanks also to the overall Project Halo team at Vulcan, Inc.

References

1. Chen, W., Kifer, M., Warren, D.: HiLog: A foundation for higher-order logic programming. Journal of Logic Programming 15(3), 187–230 (1993)
2. Cyc: Cyc (2013) (project begun in approx. 1984), `http://www.cyc.com`

[4] When radial restraint is utilized.

3. Flora-2: Flora-2 (2013) (project begun in approx. 2000), http://flora.sourceforge.net
4. Grosof, B., Andersen, C., Dean, M., Kifer, M.: Omni-directional Hyper Logic Programs in SILK and RIF. In: Proc. RuleML 2010, the 4th Intl. Web Rule Symp (Demonstration and Poster) (2010)
5. Grosof, B., Burstein, M., Dean, M., Andersen, C., Benyo, B., Ferguson, W., Inclezan, D., Shapiro, R.: A SILK Graphical UI for Defeasible Reasoning, with a Biology Causal Process Example. In: Proc. RuleML 2010, the 4th Intl. Web Rule Symp. (Demonstration and Poster) (2010)
6. Grosof, B., Swift, T.: Radial Restraint: A Semantically Clean Approach to Bounded Rationality for Logic Programs. In: Proc. AAAI 2013, the 27th AAAI Conf. on Artificial Intelligence (July 2013)
7. Lloyd, J.W.: Foundations of Logic Programming. Springer, Berlin (1984)
8. RuleML: Rule Markup and Modeling Initiative (2013) (project begun in approx. 2000), http://www.ruleml.org
9. SILK: SILK: Semantic Inferencing on Large Knowledge (2013) (project begun in 2008), http://silk.semwebcentral.org
10. Swift, T., Warren, D.S.: XSB: Extending Prolog with Tabled Logic Programming. TPLP 12, 157–187 (2012)
11. Wan, H., Grosof, B., Kifer, M., Fodor, P., Liang, S.: Logic Programming with Defaults and Argumentation Theories. In: Hill, P.M., Warren, D.S. (eds.) ICLP 2009. LNCS, vol. 5649, pp. 432–448. Springer, Heidelberg (2009)
12. XSB: XSB (2013) (project begun in approx. 1993), http://xsb.sourceforge.net

Ontology Repositories Make a World of Difference
(Abstract of Keynote Talk)

Michael Grüninger

Semantic Technologies Laboratory
Department of Mechanical & Industrial Engineering
University of Toronto
Toronto, Ontario M5S 3G8
Canada

Ontology repositories have been proposed as part of the infrastructure required to support interoperability of ontology-based software systems through the reusability and shareability of ontologies. More recently, this situation has become even more complicated by the axiomatization of ontologies in different logics and representation languages, such as OWL, SWRL, and Common Logic. This talk will explore recent work on the integration of ontologies within repositories, and pose challenges for the design, evaluation, and application of ontologies axiomatized using rules.

L. Morgenstern et al. (Eds.): RuleML 2013, LNCS 8035, p. 12, 2013.

LegalRuleML: From Metamodel to Use Cases
(A Tutorial)

Tara Athan[1], Harold Boley[2], Guido Governatori[3],
Monica Palmirani[4], Adrian Paschke[5], and Adam Wyner[6]

[1] Athan Services, USA
taraathan@gmail.com
[2] Faculty of Computer Science
University of New Brunswick, Canada
harold.boley@ruleml.org
[3] NICTA, Australia
guido.governatori@nicta.com.au
[4] CIRSFID
University of Bologna, Italy
monica.palmirani@unibo.it
[5] Corporate Semantic Web
Freie Universitaet Berlin, Germany
paschke@inf.fu-berlin.de
[6] Department of Computing Science
University of Aberdeen, UK
adam@wyner.info

1 Motivation and Background

Several XML-based standards have been proposed for describing rules (RuleML, RIF, SWRL, SBVR, etc.), or specific dialects (RuleML family [1,2]). In 2009, the Legal Knowledge Interchange Format (LKIF [4]) was proposed to extend rule languages to account for the specifics of the legal domain and to manage legal resources. To further develop the representation of the law in XML-based standards, the OASIS LegalRuleML TC held its first technical meeting on 19 January 2012 [9]. The objective of the TC is to extend the RuleML family with features specific to the formalisation of norms, guidelines, policies, and legal reasoning [3].

The work of the LegalRuleML Technical Committee has been focusing on four specific needs:

1. To close the gap between natural language text description and semantic norm modelling, in order to realise an integrated and self-contained representation of legal resources that can be made available on the Web as XML representations [8].
2. To integrate technologies such as NLP and Information Extraction (IE) with Semantic Web technologies such as graph representation and web-based ontologies and rules.
3. To provide an expressive XML standard for modelling normative rules that is able to satisfy the requirements of the legal domain. This enables use of a legal reasoning level on top of the ontological layer in the W3C Semantic Web stack. This approach

L. Morgenstern et al. (Eds.): RuleML 2013, LNCS 8035, pp. 13–18, 2013.

seeks also to fill the gap between regulative norms, guidelines, and business rules in order to capture and model the processes in them and to make them usable for the workflow and business layer [6,7].
4. To support the Linked Open Data approach to modelling with respect to the semantics of raw legal data (acts, contracts, court files, judgements, etc.) and also the rules together with their functionality and usage. Without rules, legal concepts constitute just a taxonomy [10].

The LegalRuleML TC work has been addressing these four main goals and has provided means to semantically model norms, guidelines, judgements, and contracts. The outcome is intended to define a standard (expressed with XML Schema Definition Language (XSD) and Relax NG) that is able to represent the peculiarities of the legal normative rules easily and meaningfully.

2 Methodology of the Tutorial

This tutorial presents the principles of the LegalRuleML applied to the legal domain and discuss why, how, and when LegalRuleML is well-suited for modelling norms. To provide a framework of reference, we present a comprehensive list of requirements for devising rule interchange languages that capture the peculiarities of legal rule modelling in support of legal reasoning. The tutorial comprises syntactic, semantic, and pragmatic foundations, a LegalRuleML primer, a comparison with related other approaches, as well as use case examples from the legal domain.

3 LegalRuleML Model

A key tenet of LegalRuleML is that the concepts and features of the language should provide a conceptually faithful representation of legal textual provisions and the norms they encode. To this end the language captures the following functionalities and peculiarities of the legal domain.

- qualification of norms: legal documents can contains different types of norms (constitutive, technical, prescriptive, etc.). Some norms are intended to define the terms used in the document, others to produce normative effects, and others to describe legal procedures.
- defeasibility of rules; norms are often written in a way that they admit exceptions. Defeasiblity allows for a natural representation of exceptions and permits terms to be defined in an open textured fashion.
- deontic operators: the function of prescriptive rules is to describe the normative effects that they produce (e.g., obligations, permissions, prohibitions, . . .), the parties related to them, and the conditions under which such effects are produced.
- temporal management of the rules and temporal expressions within the rules: norms are affected over the time in their validity and efficacy. LegalRuleML is able to define temporal instants and intervals that can be used to build complex legal events and situations (e.g. date of publication, interval of suspension, interval of efficacy

but not applicability). These temporal parameters are called external temporal characteristics of the norm, and thus permit the representation of the temporal information of the rules. The temporal characteristics can be associated with different rules or with a part of the rule.
- jurisdiction of norms: norms emanated from different authorities, different locations, and different times. Relative to such differences, norms can produce different effects. To properly model such contextual dependence, LegalRuleML associates rules with the jurisdictions where the rules hold.
- isomorphism between rules and natural language normative provisions: norms have a lifecyle - they are created, enter into force, can be modified, and can be repealed. Where the language of the provisions changes, so too must the corresponding formal expression in LegalRuleML (isomorphism). LegalRuleML provides for this isomorphism by maintaining a link between the units of natural language textual provisions and the sets of rules.
- authorial tracking of the rules: rules constitute an interpretation of the textual provisions and so of the norms. Accordingly, it is important to trace who is author of the interpretation to establish a level of trust in a ruleset or to identify the context in which a ruleset is suitable to be used.

4 Metamodel of the Rule Properties

The LegalRuleML syntax is modelled using a metamodel founded on RDF triples. This permits us to serialize LegalRuleML XML into RDF assertions for Semantic Web interoperability and Linked Open Data integration. The tutorial presents this aspect of the design. Rules have properties expressed in separate blocks in a generic way. We have provided a mechanism for defining an identifier of type xsd:ID for property values, which is used as the fragment identifier in an IRI that may be efficiently referenced through a relative IRI or CURIE. This ”internal” IRI may act as an alias for exernal IRIs, entities identified in external non-IRI identifier systems, and entities having no external identifiers:

• <References> and <LegalSources> for referencing the textual provisions that are modelled by the rules:

```
<lrml:LegalSource key="ref9"
    sameAs="http://www.law.cornell.edu/wiki/lexcraft/
    section_identifiers_lii"
/>
```

• <TimeInstants> and <TemporalCharateristics> for capturing the external temporal dimensions of the rules are represented. For example, here we show the period for entering into force and the period of efficacy:

```
<lrml:TimeInstants>
    <ruleml:Time key="t1">
      <ruleml:Data xsi:type="xs:dateTime">
```

```
          2012-07-21T00:00:00Z
        </ruleml:Data>
      </ruleml:Time>
</lrml:TimeInstants>
<lrml:TemporalCharacteristics key="tblock1">
      <lrml:TemporalCharacteristic key="nev1">
        <lrml:forRuleStatus iri="lrmlv:Efficacious"/>
        <lrml:hasStatusDevelopment iri="lrmlv:Starts"/>
        <lrml:atTimeInstant keyref="#t1"/>
      </lrml:TemporalCharacteristic>
</lrml:TemporalCharacteristics>
```

• <Agent> and <Authority> are two classes for defining the agent and the authority of the rules in order to represent the provenance of the rules:

```
  <lrml:Agents>
    <lrml:Agent key="aut1"
      sameAs="unibo:person.owl#m.palmirani"/>
  </lrml:Agents>
  <lrml:Authorities>
    <lrml:Authority key="congress"
        sameAs="unibo:organization.owl#congress">
      <lrml:type iri="lrmlv:Legislature"/>
    </lrml:Authority>
  </lrml:Authorities>
```

• The <Context> block associates property values to rules (in the example to rule1) and also adds other important metadata such as jurisdiction, role, and strength (defeasible, defeater, strict):

```
<lrml:Context key="ruleInfo1" hasCreationDate="#t8">
    <lrml:appliesTemporalCharacteristics keyref="#tblock1"/>
    <lrml:appliesStrength iri="lrmlv:Defeasible"/>
    <lrml:appliesRole>
        <lrml:Role iri="lrmlv:Author">
          <lrml:filledBy keyref="#aut1"/>
        </lrml:Role>
    </lrml:appliesRole>
    <lrml:appliesAuthority keyref="#congress"/>
    <lrml:appliesJurisdiction keyref="jurisdictions:us"/>
    <lrml:toStatement keyref="#rule1"/>
</lrml:Context>
```

This mechanism is flexible; it permits us to represent relationships with arity higher than two (e.g. multiple authors, multiple textual sources), which guarantees multiple interpretations over time of the same rule without redundancy. However, it is always possible to transform them into an RDF triples serialization.

5 LegalRuleML Skeleton

LegalRuleML may be composed using the following main blocks skeleton:

- declaration of the internal identifiers for property values in the top of the XML.
- association of the property values to the rules using <lrml:Association>.
- rules, both constitutive and prescriptive, are modelled in one <lrml:Statements> block.
- facts are modelled inside of a second <lrml:Statements> block.

```
<lrml:LegalRuleML>
  <lrml:References>
    <lrml:Reference/>
  </lrml:References>

  <lrml:Context key="ruleInfo1-v2">
     <lrml:Association>
       <lrml:appliesSource
             keyref="#sec2.1-list1-itm31-par1-v2"/>
       <lrml:toTarget keyref="#rulebase-v2"/>
       </lrml:Association>
   </lrml:Context>

   <lrml:Statements key="rulebase-v2">
     <lrml:ConstitutiveStatement key="rule1a-v2">
       <ruleml:if> ...</ruleml:if>
       <ruleml:then>... </ruleml:then>
      </lrml:ConstitutiveStatement>
   </lrml:Statements>

   <lrml:Statements key="facts-v1">
     <lrml:FactualStatement key="fact1">
       <ruleml:Atom key=":atom11">
         <ruleml:Rel iri="#rel5"/>
         <ruleml:Ind iri="#JohnDoe"/>
       </ruleml:Atom>
     </lrml:FactualStatement>
   </lrml:Statements>
</lrml:LegalRuleML>
```

6 Tutorial Use Cases

The use of LegalRuleML is illustrated with some concrete application scenarios:

- in the eHealth domain, LegalRuleML can be used to model privacy issues and security policies for managing document access according to the profile and the

authorizations of the operator. By using LegalRuleML, it is possible to filter sensitive data, according to the law/regulation, and to create particular views of the same health record or document based on the role of the querying agent.
- in the open data domain, LegalRuleML could model the creative commons licences of datasets to permit an automatic IPR compatibility check among different datasets, in particular to evaluate if different datasets could be combined for producing a commercial application.
- in patent law, LegalRuleML can model the judgments and the regulations in order to support the industrial decisions.

References

1. Boley, H., Tabet, S., Wagner, G.: Design rationale for RuleML: A markup language for Semantic Web rules. In: Cruz, I.F., Decker, S., Euzenat, J., McGuinness, D.L. (eds.) Proc. SWWS 2001, The First Semantic Web Working Symposium, pp. 381–401 (2001)
2. Boley, H., Paschke, A., Shafiq, O.: RuleML 1.0: The Overarching Specification of Web Rules. In: Dean, M., Hall, J., Rotolo, A., Tabet, S. (eds.) RuleML 2010. LNCS, vol. 6403, pp. 162–178. Springer, Heidelberg (2010)
3. Gordon, T.F., Governatori, G., Rotolo, A.: Rules and Norms: Requirements for Rule Interchange Languages in the Legal Domain. In: Governatori, G., Hall, J., Paschke, A. (eds.) RuleML 2009. LNCS, vol. 5858, pp. 282–296. Springer, Heidelberg (2009)
4. Gordon, T.F.: Constructing Legal Arguments with Rules in the Legal Knowledge Interchange Format (LKIF). In: Casanovas, P., Sartor, G., Casellas, N., Rubino, R. (eds.) Computable Models of the Law. LNCS (LNAI), vol. 4884, pp. 162–184. Springer, Heidelberg (2008)
5. Athan, T., Boley, H., Governatori, G., Palmirani, M., Paschke, A., Wyner, A.: OASIS LegalRuleML. In: Verheij, B. (ed.) Proceedings of 14th International Conference on Artificial Intelligence and Law (ICAIL 2013). ACM (2013)
6. Governatori, G., Rotolo, A.: Changing legal systems: Legal abrogations and annulments in defeasible logic. The Logic Journal of IGPL (2010)
7. Grosof, B.: Representing e-commerce rules via situated courteous logic programs in RuleML. Electronic Commerce Research and Applications 3(1), 2–20 (2004)
8. Palmirani, M., Contissa, G., Rubino, R.: Fill the Gap in the Legal Knowledge Modelling. In: Governatori, G., Hall, J., Paschke, A. (eds.) RuleML 2009. LNCS, vol. 5858, pp. 305–314. Springer, Heidelberg (2009)
9. Palmirani, M., Governatori, G., Rotolo, A., Tabet, S., Boley, H., Paschke, A.: LegalRuleML: XML-Based Rules and Norms. In: Olken, F., Palmirani, M., Sottara, D. (eds.) RuleML - America 2011. LNCS, vol. 7018, pp. 298–312. Springer, Heidelberg (2011)
10. Sartor, G.: Legal reasoning: A cognitive approach to the law. In: Pattaro, E., Rottleuthner, H., Shiner, R., Peczenik, A., Sartor, G. (eds.) A Treatise of Legal Philosophy and General Jurisprudence, vol. 5. Springer (2005)

Formalization of Natural Language Regulations through SBVR Structured English*
(Tutorial)

François Lévy and Adeline Nazarenko

Université Paris 13, Sorbonne Paris Cité, LIPN, CNRS, (UMR 7030)
F-93430, Villetaneuse, France

Abstract. This paper presents an original use of SBVR to help building a set of business rules out of regulatory documents. The formalization is analyzed as a three-step process, in which SBVR-SE stands in an intermediate position between the Natural Language on the one hand and the formal language on the other hand. The rules are extracted, clarified and simplified at the general regulatory level (expert task) before being refined according to the business application (engineer task). A methodology for these first two steps is described, with different operations composing each step. It is illustrated with examples from the literature and from the ONTORULE use cases.

1 Introduction

Formalizing natural language (NL) regulations is becoming an important challenge for rule systems.

Much more data are available in an electronic form that, let us say, ten years ago. Most governments have set up electronic legal repositories to make their national laws and regulations available (*e.g.* THOMAS in the USA[1], Legifrance in France[2], the Bundestag site in Germany[3]). This is also the case of international institutions (see the European Community[4] or the UNO[5] websites). Smaller organizations such as regions, cities, Länder also produce and publish their own regulations. On the private organizations side, most companies have both public regulations for their customers, and internal ones for their employees.

These rules are involved in applications that mainly concern the opening of rights and the conformance of processes. It is crucial that the formal rules

* We thank to our partners in the ONTORULE project for the fruitful discussions, especially John Hall (Model Systems) for introducing us to the SBVR world, Christian de Sainte Marie for Decision Logic, and Audi for the collaboration on their use case. We are also grateful to American Airline who is the owner of one of our working corpora.

[1] `http://thomas.loc.gov/home/abt_thom.html`
[2] `http://www.legifrance.gouv.fr/`
[3] `http://www.bundestag.de/`
[4] `http://eur-lex.europa.eu`
[5] `http://www.un.org/`

L. Morgenstern et al. (Eds.): RuleML 2013, LNCS 8035, pp. 19–33, 2013.

embedded in those applications be consistent with the published regulations. Hence the need for extracting and building the formal rules from the NL ones.

This paper analyses this formalization process. It shows that it can be decomposed into three main steps, in which SBVR Structured English (SBVR-SE) plays an intermediate role between the natural and formal languages. Section 2 presents this challenging process that has been only partially addressed in the state of the art and Section 3 shows that it can be decomposed into three steps. Sections 4 and 5 describe the main operations that are involved in the first two of these formalization steps.

2 Formalizing NL Regulations: A Challenging Task

2.1 The Translation Task

Rule acquisition is a critical bottleneck for the development of business rules management systems (BRMS) as for most knowledge based systems. The problem is three fold for knowledge engineers. They have to identify all the constraints and rule information that are relevant for the domain of the rule application to develop. They have to specify how these constraints and rule must be handled: some are directly implemented in the decision system; some belong to general policy that apply to the system users but that are not formally expressed in the rule base; some are plain recommendations whereas other are strict constraints; they may concern the structure of the organization to model as well as its procedures and the behavior of actors. Once the relevant information is identified and its enforcement value defined, the knowledge engineers still have to encode this information in a way that is machine-understandable.

This acquisition process is a complex task to handle. Fortunately, in many cases – especially in conformance applications –, the relevant rule information is already encoded in policy documents (*e.g.* contracts, legal regulations, user guides). In those cases, the elicitation work can be based on existing and validated sources instead of experts' introspection and interviews.

However, extracting and translating NL regulations into formal rules remains an open issue [4,13,9]. The translation of text fragments written in NL into formal rules is difficult to automate, due to the reduced expressivity of formal languages and to the complexity of NL discourse, which is redundant and verbose, often elliptic and implicit, frequently ambiguous. Even the translation into SPARQL of LN queries, which are much simpler than texts, is acknowledged as a complex problem [24]. [8] considers a direct translation of legal texts based on a parsing step, but it actually relies on a manual translation of abstract syntax trees in a specific logical language.

2.2 The Problem of Uncommunicability

NL and formal rule languages stand on the opposite extremities of the formalization continuum, which raises a problem of uncommunicability between the

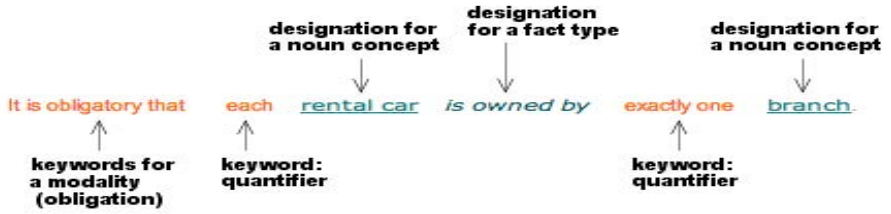


Fig. 1. Example of an SBVR-SE business rule (www.brcommunity.com)

actors. The lack of domain knowledge and the complexity of NL explain the difficulty of writing the formal rules from scratch for the *engineer*. Working with texts requires a certain familiarity with the domain and a thorough understanding of source regulations. The *expert* describes the rules according to his/her knowledge of the organization for which the rule system is being developed. The *engineer* has a different point of view, more focused on the system and its operation as such. The *expert* finds it difficult to read the rule base when it has been written in formal language. This raises many problems, such as regarding the authoring of the rules in conformance to the source documentation and their updating.

Issues related to these deadlocks have been raised [4,13,9] and proposals have been made to improve the actors collaboration. One of them consists in exploiting a new language that acts as an intermediate between NL and formal languages. Controlled languages (CL) have been proposed to play this role. They are more precise, unambiguous, easier to understand – but also, less expressive – than natural language.

2.3 The Complexity of Natural Language

Controlled languages are considered as an interesting substitute for NL, assuming that they are readable for users and closer to formal languages than NL. In the Business Rules domain, CLs are used in Oracle Policy Modeling Suite[6], in IBM SPARCLE policy workbench [5]. RuleSpeak [20] is a dedicated in use controlled language, while Atempto Controlled English[7] is more generalist. SBVR (Semantics of Business Vocabulary and Rules) can be seen as a synthesis of several efforts and as a standard independent of any specific natural language. It has been accepted by the OMG (Object Management Group)[8]. We refer to the English version of SBVR CL, namely SBVR Structured English (SBVR-SE). SBVR relies on formulas (Figure 1) combining linguistic basic templates with logical, modal or quantification operators.

[6] www.oracle.com/technology/products/applications/policy-automation
[7] http://attempto.ifi.uzh.ch/site/docs/
[8] http://www.omg.org

However, the NL to CL translation is itself a complex task. Recently, [2] has proposed NL2SBVR[9], a tool to automatically translate NL into SBVR-SE, but the reported experiments show that the complexity of the translation depends on the complexity of the source sentences. Simple rule statements composed of at most two clauses are translated with a 80% success rate but the translation fails for complex ones.

Another way to handle the complexity of NL texts therefore consists in simplifying them. Text simplification has long been studied. It is usually considered to ease translation [22], to help human understanding (esp. in case of understanding disorders [15]), to prepare text summarization [10,7], and in the context of foreign language learning (see [6] for a general presentation). The notion of simplicity actually depends on the target application. For instance, if a succession of small sentences if preferable for foreign language learners, automatic translation can process long sentences as soon as their internal structure is canonical.

Taking advantage of these works on controlled language and text simplification, we propose to decompose the rule formalization into different steps and to rely on SBVR-SE as an intermediate language that ease the transition from natural to formal languages.

2.4 Organizing the Formalization Continuum

Several authors have considered degrees in the formalization process. [3] argues that there is a "continuum" between knowledge expressed in NL and formal one. Considering controlled languages as intermediary stages in this continuum, [23] points to two different views of controlled languages constituting steps in the formalization: the naturalist one insists on simplicity, the formalist one on eliminating ambiguity.

More focussed on the semantics of business rules, *Decision Logic* or *Decision Modeling* emphasizes the distinction between rules as specification of what should be, and the operational view where rules state what to do[10]. Operational rules take or help to take decisions, i.e. make rational choices between several outcomes [21] expressed in the language by some decision words (*e.g.* "Estimate", "Determine", "Assess", "Calculate", "Accept" [12]) and which follow the specifications. Operational rules are related to SBVR through the vocabulary, and to a Process Model, since branching points in this model need a decision [14].

Decision Logic has to see with a precise inventory of all the conditions which can influence a decision, and with how a decision is reflected in the system. At the moment, there is no clear and agreed formal view of decision in the BR domain, but OMG is currently elaborating a Decision Modeling Notation (DMN) which is a first step in that direction.

[9] `http://www.cs.bham.ac.uk/ isb855/nl2ocl/projects.html`

[10] Both terms have close but different meanings out of the business rules domain. "Decision Modeling" is used in decision theory for a.k.o. simulation, where the goal is to predict (long term) effects of a given decision. "Decision Logic" also deals how decisions are made by individuals or groups in unforeseen conditions (*e.g.* in emergency cases [16])

3 Divide for Conquer

3.1 A Three-Step Process

The Division of Labor. We focus here on the modelization and formalization of business rules. Some previous works [1] have shown that this tasks involves two typical actors. Both acts as knowledge engineers but one is a domain expert (the *expert*) and the other one is more system oriented (the *engineer*):

- The expert knows the domain of the target application, the organization in which it is expected to take place and the available sources of information. He/she has to identify what is the relevant information to take into account, in the form of a draft rule base.
- The engineer focuses on the rule system, that may not model the whole organization but only a part of it. He/she knows how the system works and he/she can implement rules in an operational language.

Formalization Steps. Based on this analysis, we propose to decompose the formalization of rules into three separate steps.

1. The first step produces CL out of NL. The transformation is made by an expert who knows and understands the source texts. The task consists in writing a new CL text that selects the rule information of the source text that is relevant for the organization to model. The resulting text can be considered as the specification of the rule base content.
2. The second step improves the specification text in CL. The transformation is made by the engineer in interaction with the expert. The engineer interprets the rules. He asks the expert to check and complete his/her understanding of the rule base. They may further filter out the text produced in the first step if it contains some rules that are not directly applicable by the rule system.
3. The last step aims at translating the specification text into formal language. It requires a good knowledge of the target application and system.

3.2 SBVR, between Natural and Formal Languages

SBVR [18] is an OMG standard. It is not a language *per se* but a metamodel designed for describing the business knowledge of complex organizations such as enterprises, in a formal and detailed way. It enables to

- assist experts in the specification and definition of the semantic model (semantic vocabulary and rules) of the domain;
- describe a business model in a precise, clear and unambiguous way;
- specify various linguistic structures that cover a larger number of languages, even if it is mainly used for English;
- structure[11] an organization business knowledge and business vocabularies;
- make the business rules accessible both to the experts and engineers, thus improving their inter-communication.

[11] `http://www.omg.org/news/meetings/tc/mn/special-events/br/`

The definition of business vocabulary is not dealt with here, our single focus is on using SBVR as an intermediary language. Business rules are written with the help of a controlled language, SBVR-SE, which makes use of a typed business vocabulary (Fig. 1): business terms or concept names, named entities, phrases which express fact types and facts linking concepts[12], as well as a set of (English) grammatical words (*e.g.* '**the**', '**that**', '**another**', '**a given**') used as operators. SBVR offers several types of operators. The most often used are:

- Logical operators, such as negation (it is **not** the case that p), conjunction (p **and** q), disjunction (p **or** q), implication(**if** p **then** q, q **if** p), equivalence (p **if and only if** q).
- Modal operators, that modify the modal value of a fact from [contingently] true to obligatory, possible or necessary. These operators are often inserted in front of rules: "**It is obligatory that**" and "**It is necessary that**", resp. for operative and structural rules.
- Quantifiers: **a, an, each** (universal quantification), **at least one** (existential quantification), **at most one**, **exactly one**.

In the formalization process, our expectation with respect to SBVR is twofold. In the first step, it is used to faithfully translate source texts while reducing the complexity of natural language. The output is a list of autonomous rule (see Fig. 2) that can be further formalized independently of each other. In a second step, it is used to reformulate the business rules so as to take the application into account and to ease their formalization in the last step.

For instance, the textual rule 1 of the EU-Rent example can be first normalized (Rule 2[13]) and then turned into a decision rule (Rule 3). The last step would translate this differently according to the rule language: production rules and logic programming, for instance, have different technicalities to destroy and create objects.

1. *A rental has exactly one renter.*
2. *It is prohibited that the renter of a rental is changed.*
3. *If a renter asks to change the rental to a different renter, cancel the rental and create a new one.*

The first use of SBVR-SE concerns the expert task (from NL to CL) whereas the second one consists in a CL to CL transformation, which refines the description according to the end application. This last transformation usually requires a negotiation between the expert and the engineer. We will see in Section 5 that it mainly consists in eliminating the modal operators, as suggested in:

> The implementation impact of the alethic necessity tag is that any attempted change that would cause the model of the business domain

[12] We do not take for granted, as SBVR-SE does, that facts and fact types are specifically referred to by verb phrases.

[13] Of course, other readings can be proposed but the normalization process involves some interpretation choices.

to violate the constraint must be dealt with in a way that ensures the constraint is still satisfied (*e.g.*, reject the change, or take some compensatory action). [18, p102]

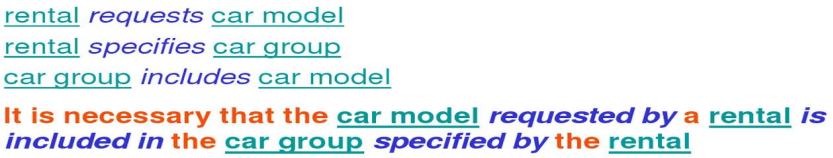


Fig. 2. Example of an autonomous SBVR rule

4 From Natural Language to SBVR Rules

The acquisition methodology relies on the progressive transformation of the source text into a set of self-sufficient rules written in SBVR-SE controlled language. This process relies on four main operations, which are often interlinked: the lexical normalization of the source text, the extraction of the relevant text fragments, the syntactic normalization of these fragments, some semantic transformation for restoring contextual and implicit information. The result of that process is a set of rules written in valid SBVR-SE.

4.1 Lexical Normalization and Annotation of the Source Text

The lexical normalization is often performed on the whole source text. It is a critical step for the whole transformation process. An annotation is a meta-data or label attached to a word, phrase, sentence or section of the source document. A normalized vocabulary entry is associated to the annotated segment. In our framework, first annotated segments are elements of the conceptual vocabulary or keywords.

The *conceptual vocabulary* contains all the terms that have a specific meaning in the domain of the source regulation. For the acquisition of rules, we suppose that a domain ontology already exists, that fixes the conceptual elements to use in the rules [17]. The conceptual vocabulary is composed of the set of terms referring to the ontological concepts, individuals and roles. It can be encoded as a SKOS thesaurus referring to a an OWL ontology [19] or as SKOS annotations in the OWL ontology. The semantic annotation is the process that takes a text and a lexicalized ontology as input and outputs the source text enriched with annotations. It consists in localizing in the text all the mentions of the ontological elements, whatever the form that these mentions may take, and annotating them with a reference to the ontological element they refer to. Lexical normalization consists in normalizing the lexicon of the source text since the various variants of a term are all expected to be annotated with the same canonical form.

The *keyword list* contains all the "linguistic symbols [or grammatical words] used to construct statements – the words that can be combined with other designations [from the conceptual vocabulary] to form statements and definitions" [11, p.238]. Keywords are easier to recognize in the documents since they belong to closed class of words and their form is generally more stable.

The initial annotation process therefore produces a text in which the mentions of the ontological elements and the SBVR-SE keywords are annotated. They are colored according to SBVR rules: orange for keywords, green, blue and red respectively for the words and phrases that refer to concepts, roles and individuals. The underlying analogy is that concepts corresponds to "noun concepts", roles to "verb concepts" and individuals to "individual concepts" in the SBVR terminology, although we do not assume that there us a strict parallelism between the part-of-speech categories (nouns *vs.* verbs) and the conceptual ones (concepts *vs.* roles) as SBVR-SE does.

Since the annotation process does not cover the whole text, the resulting annotated text usually mixes black segments and some colored words and phrases. It looks like an SBVR "informal representation", since "not every word is annotated ('tagged') in accordance with a notation that can be mapped to SBVR" [11, p.152]. It is nevertheless a precious input for the extraction, normalization and transformation process that outputs a rule base specification.

4.2 Extraction of Rule Fragments from the Source Text

Once the text has been automatically annotated, the knowledge engineer has to identify in the source document the fragments (sentences or sequence of sentences) that convey rule information and to mark them as candidate rules that probably need to be reformulated but are nevertheless relevant for the target rule application.

Identifying these rule fragments in the source text is a complex task and the initial annotation eases this work. The knowledge engineer can focus on the passages that are most marked with annotation and extraction patterns can be designed on the basis of remarkable configurations of keywords.

The selection of a fragment consists in the creation of a candidate rule, which is identical at the beginning to the source fragment but which will be further transformed.

4.3 Lexical and Syntactic Normalization of the Rule Fragments

The variability and polysemy of natural language makes it necessary to give a canonical form to the initial statements. This normalization process concerns both the lexicon and the syntax.

One part of the lexical normalization is carried out through the semantic annotation process, since different words and phrases are annotated in the same way if they are alternate labels of the same ontological entity. For instance, in some contexts *member*, *members* and *participants* can be considered as synonyms and may all refer to the same concept.

However, it often happens that some relevant mentions are missed by the semantic annotator: the knowledge engineer has to identify new occurrences of relevant ontological entities. Another problem comes from the polysemic words which may not be properly handled by an automatic semantic annotator. In some context, *city hall* does not refer to a building but to an organization or even to the people involved in that organization. Advanced annotators which take the context into account can take care of these disambiguation cases but they are less performant than others.

Syntactic constructs also have to be simplified and normalized. A lot of syntactic constructs are not supported by a controlled language like SBVR-SE because they are ambiguous or difficult to understand. Some clauses must be reordered within the sentences, enumerations must be split, complex sentences must be simplified by erasing the irrelevant clauses, coordinations have to be decomposed. A single candidate rule may give way to two or more separate and simpler candidate rules. Some rules must also be transformed to stick to the canonical syntactic structure of the target language that is supposed to be unambiguous and easy to understand.

For instance, the following sentences can be decomposed into several ones, which makes the initial statement easier to understand and to exploit.

1. ***Neither*** *accrued mileage,* ***nor*** *award tickets,* ***nor*** *upgrades are transferable by the member upon death*[14].
   Accrued mileage is not transferable by the member upon death. Award tickets are not transferable by the member upon death. Upgrades are not transferable by the member upon death.
2. *The membership year,* ***which*** *is the period in* ***which*** *your elite benefits are available, runs from March 1 through the last day of February of the following year.*
   The ***membership year*** *is a period.* ***Member****'s elite benefits are available in the* ***membership year****. The membership year runs from March 1 through the last day of February of the following year.*

4.4 Semantic Transformation

Transformations at the semantic level are also often required to restore some elements of context or implicit piece of information that condition the interpretation of the rule. The semantic transformation does not preserve the apparent meaning of the source fragment but aims at decontextualize it, providing context independent formulations, fixing some possible ambiguities, deleting irrelevant stylistic fragments.

For instance, in the UNO regulation n°16 dealing with car manufacturers quality tests, it often happens that a generic term like *test* is used in a sentence where it is clear from the context that it has a specific meaning and actually stands for *e.g. micro-slip test*, which refers to a specific type of tests. When the

[14] This example has been extracted form the American Airlines terms and conditions.

rule statement is isolated from its context, the generic term must be replaced by the more specific one so that the initial meaning is preserved.

More complex transformations involve the whole sentence. In the following case, the knowledge engineer separated the extracted fragment into two different SBVR rule statements and then made explicit the modality "hidden" in the use of a future tense.

1. *No mileage credit will be awarded for canceled flights* ***or*** *if you are accommodated on another airline.*
2. *If a member is accommodated on another airline, then* ***it is obligatory that*** *no mileage credit is attributed. If a flight is cancelled, then* ***It is obligatory that*** *no mileage credit is attributed.*

5 From Normalized Rules to Decision Rules

At the end of the normalization and simplification process, the output is SBVR-conformant text. It is a set of business rule statements which are admittedly formal in the sense of the SBVR standard:

> Business rule expressions are classified as formal only if they are expressed purely in terms of fact types in the pre-declared schema for the business domain, as well as certain logical/mathematical operators, quantifiers, etc. Formal statements of rules may be transformed into logical formulations. [18, p.85]

However, the initial goal – having rule directly translatable into rules of an automated decision system – is still not reached. The problem comes from the difference between the regulatory level ontology and the application specific ontology. A semantic gap lies between rules infering deontic modalities and rules infering concrete actions. We consider two points. First, new entities are needed, dedicated to represent the conditions of the decision. Second, a rule modal statement has to be translated into decision terms.

Three examples from different domains are used to illustrate these points.

5.1 Introducing New Specialized Entities

Very often, deciding if the conditions are fulfilled needs to introduce new specialized entities. This is best illustrated through examples.

Example 1. The first example, bellow, is related to the UNO regulation n°16. Concerning the breaking load test after cold-conditioning (bl-cc test for short), the regulation states the fragment 1. At the normalization level, it appears that 'When' has a temporal value rather than a conditional one. It indicates a step of the process (actually, the fourth step – previous ones being omitted for the sake of brievity). 'And' has also a temporal value. It introduces a next step. Naming the steps is a facility for decomposing the rules (see version 2, duration considerations

are provisionally left aside). Taking into account the delays also needs to give a full status of entities. This leads to add two notions, *load-duration* and *removal-measure-delay* (see version 3). More hidden is the fact that the greater the load duration, the more severe is the obligation, so the delay is a *minimum*.

1. *When the strap has been kept under load for 30 minutes in the same low-temperature chamber, the mass shall be removed and the breaking load shall be measured within 5 minutes after removal of the strap from the low-temperature chamber.*
2. *Step 4 of bl-cc test is: the strap is kept under load in the low-temperature chamber.*
   Step 5 of bl-cc test is: the mass is removed.
   Step 6 of bl-cc test is: the breaking load is measured.
3. *The load-duration is the time between the start of step 4 and of step 5.*
   It is obligatory that the load duration be greater than 30mn.
   The removal-measure-delay is the time between the removal of the strap from the cold-chamber after step 5 and the end of step 6.
   It is obligatory that the removal-measure-delay be less than 5 mn.

Example 2. The second example comes from the EU-Rent case of [18]. EU-Rent is a car renting company. The extracted fragment (version 1) is related to branches but nothing is specified for the pick-up and return of cars from and to branches. The knowledge engineer has introduced a specific rule, dealing with the effective and specified drop-off locations (version 2). The obligation is discussed in the following section but we can see here how the ontology is refined to allow writing this version. The fact type "a car is returned to a branch" is broken down into several fact types with the help of added domain vocabulary. New vocabulary (in bold face) and fact types are gathered in fragment 3.

1. *A Local area contains a number of Branches for Rental Car pick-up and return. A rental booking specifies [. . .] the EU-Rent branch from which the rental is to start. Optionally, the reservation may specify a one-way rental (in which the car is returned to a branch different from the pick-up branch)*
2. *It is obligatory that the rental incurs a location penalty charge if the drop-off location of a rental is not the EU-Rent site that is base for the return branch of the rental.*
3. *A return branch has a* ***base****. The* ***base*** *of the return branch is an* ***EU-Rent site****. A rental has a* ***drop-off location****. The* ***drop-off location*** *is the* ***base*** *of the return branch.*

5.2 Exhibiting Decision Variables

When transforming normalized rules into decision rules, the main point consists in getting rid of the modal operators, while preserving as most as possible the meaning of the source fragment in its context. A second operation therefore consists in making explicit some variables related to the decision to take, which often remains implicit.

Example 1. The first example is from Haley's blog[15], which describes the formalization of the rules to qualify for the earned income credit (EIC). The source text is a guide provided by the administration to the applicant. The first extracted rule is the version 1 of the following example (slightly simplified for the sake of brievity). The normalization (version 2) requires to restore a premise from the context (the person is applying for EIC), clarify cardinalities (if you have two children, you do not have one) and normalize the obligation. Haley points that "must" is misleading here since the proposition under its scope is not an obligation for the applicant, rather a condition of success. His re-statement of the rule is merged in version 3. "Being qualified for the EIC" is a new concept. It does not describe the specific data related to the applicant, as do the income and the number of children. It states that the data are not conformant wrt. the intended model. We call these variables *decision variables*.

1. *If you have one qualifying child, your Adjusted Gross Income (AGI) must be less than $29,666.*
2. *If a person applies for the EIC and this person has exactly one qualifying child, then it is obligatory that this person's AGI is less than $29,666.*
3. *If you apply for the EIC and you have exactly one qualifying child and your AGI is more than $29,666, then you do not qualify for EIC.*

Example 2. Let us return to the breaking load test after cold-conditioning introduced in section 5.1. If the obligations stated in 1 are not fulfilled, no valid conclusion can be drawn from the test. This is different from a failure, which is the case when a valid breaking load is obtained with a value is under a given threshold. It is of course not a success either. A decision variable (validity of the bl-cc test) is introduced to account for that in version 2.

1. *It is obligatory that the load duration be greater than 30mn.*
   It is obligatory that the removal-measure-delay be less than 5 mn.
2. *If the load duration is less than 30mn, the bl-cc test is invalid.*
   If the removal-measure-delay is more than 5 mn, the bl-cc test is invalid.

Example 3. The EU-Rent case has a main obligation specified by the rental booking (statement 1). It is again accounted with the help of a decision variable (version 2).

1. *It is obligatory that the pick-up location be the start branch of the rental.*
   It is obligatory that the drop-off location be the return branch of the rental.
2. *If the drop-off location of the rental is not the base of the return branch of this rental, then this rental is non-conformant-for-return.*

The technique proposed here clearly separates two questions. The decision-variables are used to reflect in the model that an obligation has not been fulfilled.

[15] `http://haleyai.com/wordpress/2008/03/28/harvesting-business-rules-from-the-irs/`

In this case, they are introduced in the conclusion of rules, which conditions are obtained from the first SBVR modeling step, from specialized modeling entities and the (negated) first order content of the obligation. This raises a correlated question regarding what to do when the obligation is actually violated. Section 5.3 addresses this point.

5.3 Specifying Actions

Deciding what to do when a decision point has been reached involves more application specific knowledge than the previous ones, because regulations remain relatively application independent and do not examine things beyond the obligation. Possible answers can be divided into three groups :

- Abandon, stop the process, do nothing;
- Retry the same process, after modifying one of the conditions;
- Use a remedial subprocess as a continuation after the decision point. The remedial process is often not mentioned by the regulation, and the application specialist generally has a major role in its description.

The first group is illustrated by the EIC case, at least when the application is to help the user to fill a statement of income and a EIC form. The resulting rule could be 1. Of course, other actions are possible. Formally, the EIC case could also yield to 2 and 3.

1. *If you do not qualify for the EIC, then don't fill the EIC form.*
2. *If you do not qualify for the EIC, then reduce your AGI [to $29,666].*
3. *If you do not qualify for the EIC, then apply for one more child.*

The second group is illustrated by the UNO Regulation n°16 case. The resulting rule can be

> *If the bl-cc test is invalid, then the test must be started afresh with a new strap.*

The last group is illustrated by the EU-Rent case. The following rule is suggested by the knowledge engineer. The process is neither abandoned, nor redone. The drop-off is accepted as is, but the charge is increased.

> *If a rental is non-conformant-for-return, this rental incurs a location penalty charge.*

The above examples show that what remains to be done, once the rules are well-formed SBVR-statement, is of a different nature from the normalization steps. This is the reason why we argue that the knowledge engineer who normalizes the source text and formalizes the resulting candidate rule should be different actors. The first step is driven by the source text and a general idea of the organization to model. The second one directly depends on how the candidate rules must be operationalized, and which part of this operational semantics is relevant for the rule system – for deciding, checking or warning. It makes sense that these interpretations be made before implementation and remain accessible to business people through SBVR.

6 Conclusion

In order to formalize NL regulations into production rules, we have argued for a three steps process. We have focused here on the first two, which both rely on a controlled language but reflect different tasks devoted to different actors. The clarification of the regulation is under the responsibility of a general-level expert. Its refinement also involves an engineer who knows how the rule system works. We have argued that SBVR is convenient as a controlled language. It allows expressing the new formulations while remaining understandable by business people who want to remain in charge of the rule authoring. Then we have described a methodology for acquiring rules from regulatory texts and transforming them into production rules, providing different operations for each step. The methodology is illustrated with examples from the literature and from the Ontorule use cases. It shows the importance of SBVR and more generally controlled languages as flexible intermediate languages at the border between natural and formal languages.

References

1. Bajec, M., Krisper, M.: Issues and challenges in business rule-based information systems development. In: ECIS (2005)
2. Bajwa, I.S., Lee, M.G., Bordbar, B.: Sbvr business rules generation from natural language specification. In: AAAI Spring Symposium 2011 Artificial Intelligence 4 Business Agility, pp. 541–545, AAAI, San Francisco (2011), www.aaai.org/ocs/index.php/SSS/SSS11/paper/download/2378/2918
3. Baumeister, J., Reutelshoefer, J., Puppe, F.: Engineering intelligent systems on the knowledge formalization continuum. International Journal of Applied Mathematics and Computer Science (AMCS) 21(1) (2011)
4. BRG: Defining business rules – what are they really? The Business Rules Group : formerly, known as the GUIDE Business Rules Project - Final Report revision 1.3 (July, 2000)
5. Brodie, C., Karat, C.M., Karat, J.: An empirical study of natural language parsing of privacy policy rules using the sparcle policy workbench. In: SOUPS 2006 (2006)
6. Chandrasekar, R., Doran, C., Srinivas, B.: Motivations and methods for text simplification. In: Proceedings of the Sixteenth International Conference on Computational Linguistics (COLING 1996), pp. 1041–1044 (1996)
7. Chandrasekar, R., Srinivas, B.: Automatic induction of rules for text simplification (1997)
8. Dinesh, N., Joshi, A., Lee, I., Sokolsky, O.: Reasoning about conditions and exceptions to laws in regulatory conformance checking. In: van der Meyden, R., van der Torre, L. (eds.) DEON 2008. LNCS (LNAI), vol. 5076, pp. 110–124. Springer, Heidelberg (2008), http://repository.upenn.edu/cis_papers/371
9. Dubauskaite, R., Vasilecas, O.: An open issues in business rules based information system development. In: Innovative Infotechnologies for Science, Business and Education, vol. 1 (2009)
10. Gasperin, C., Specia, L., Pereira, T.F., Aluisio, S.M.: Learning when to simplify sentences for natural text simplification. In: ENIA 2009 (VII Encontro Nacional de Inteligência Artificial) (2009)

11. The Object Management Group: Semantics of business vocabulary and business rules. Tech. rep., The Object Management Group (2008), http://www.omg.com/
12. von Halle, B., Goldberg, L.: The Decision Model: A Business Logic Framework Linking Business and Technology. Taylor & Francis, LLC, Auerbach (2009)
13. Halle, B., Goldberg, L., Zackman, J.: Business Rule Revolution: Running Business the Right Way. Happy About (2006), http://books.google.com/books?id=I3mvAAAACAAJ
14. Linehan, M.H., de Sainte Marie, C.: The relationship of decision model and notation (dmn) to sbvr and bpmn. Business Rules Journal 12(6) (June 2011), http://www.BRCommunity.com/a2011/b597.html
15. Max, A.: Simplification interactive pour la production de textes adaptés aux personnes souffrant de troubles de la compréhension. In: Proceedings of TALN, Poster Session (2005), http://www.limsi.fr/Individu/amax/recherche/articles/Max-TALN05.pdf
16. Mendonça, D., Wallace, W.A.: Development of a decision logic to support group improvisation: An application to emergency response. In: 35th Hawaii International Conference on System Sciences (2002)
17. Nazarenko, A., Guissé, A., Lévy, F., Omrane, N., Szulman, S.: Integrating Written Policies in Business Rule Management Systems. In: Bassiliades, N., Governatori, G., Paschke, A. (eds.) RuleML 2011 - Europe. LNCS, vol. 6826, pp. 99–113. Springer, Heidelberg (2011)
18. OMG: Sbvr (2008), http://www.omg.org/spec/SBVR/Current
19. Omrane, N., Nazarenko, A., Rosina, P., Szulman, S., Westphal, C.: Lexicalized ontology for a business rules management platform: An automotive use case. In: Olken, F., Palmirani, M., Sottara, D. (eds.) RuleML - America 2011. LNCS, vol. 7018, pp. 179–192. Springer, Heidelberg (2011)
20. Ross, R.G.: Principles of the Business Rule Approach, ch. 8-12. Addison-Wesley, Boston (2003)
21. Ross, R.G.: Decision analysis using decision tables and business rules. Tech. rep., Business Rule Solutions (2010)
22. Siddharthan, A., Caius, G.: Syntactic simplification and text cohesion (2003)
23. Spreeuwenberg, S.: Rule authoring is a creative process. Business Rules Journal 10(9) (September 2009), http://www.BRCommunity.com/a2009/b497.html
24. Unger, C., Bühmann, L., Lehmann, J., Ngomo, A.C.N., Gerber, D., Cimiano, P.: Sparql template based question answering. In: 21st International World Wide Web Conference, WWW 2012 (April 2012), http://data.semanticweb.org/conference/www/2012/paper/1290

Multi-agent Activity Modeling with the Brahms Environment
(Abstract of Tutorial)

Maarten Sierhuis

Nissan Research Center
Silicon Valley

There is increasing interest in developing "day in the life" models and simulations of people's behavior, the interaction between groups of people and systems, as well as movement and interaction within the environment. Cognitive modeling tools (e.g. SOAR, ACT-R) focus on detailed modeling of individual cognitive tasks at the sub-second level. In contrast, Brahms enables multi-agent activity modeling, focusing on higher-abstraction behaviors at the second and longer timeframe. Activity modeling enables modeling the behaviors of individuals and groups (located and situated), how and where communication and synchronization happens, and how people and machines work together to accomplish goals. This tutorial will provide an overview of the Brahms multi- agent activity modeling language by considering a simple day in the life scenario, including hands-on experience with Brahms.

Brahms includes an activity-oriented Belief-Desire-Intention (BDI) language, a compiler and virtual machine for executing Brahms models, as well as an Eclipse plug-in and a post-execution viewer of agent execution, communication and interaction. Brahms enables the creation of multi-agent models that include aspects of reasoning found in cognitive models, task execution, plus the impact of interaction and geography, such as agent movement and physical changes in the environment. Brahms is currently used to automate the work of a flight controller in NASAs International Space Stations Mission Control Center (ISS MCC). This system, called OCAMS, has been in production in the ISS MCC, 24x7, since July of 2008, and is based on a Brahms model of the work practices of the flight controllers. OCAMS is a distributed Multi-Agent System.

Prerequisite knowledge: A useful background to have is some experience in rule-based languages, agent architectures, especially belief-desire-intention architectures and discrete-event simulation.

Bio

Dr. Maarten Sierhuis is founder and Chief Technology Officer of Ejenta and Director of Nissan Research Center-Silicon Valley. Before this, he was the director of the Knowledge, Language and Interaction area at Xerox Palo Alto Research Center (PARC), and a Senior Scientist for over twelve years at NASA Ames Research Center. He is co- inventor of the Brahms multi-agent language. He is also a fellow with the Interactive Intelligence group at Delft University of

L. Morgenstern et al. (Eds.): RuleML 2013, LNCS 8035, pp. 34–35, 2013.

Technology in the Netherlands, where he taught graduate courses on multi-agent and organizational modeling. He has a Ph.D. in AI and Cognitive Science from the University of Amsterdam and an engineering degree in Informatics from the University in The Hague, The Netherlands. He has presented many invited lectures and tutorials on agent languages, agent-based simulation, and multi-agent systems, and has published widely in these areas.

References

1. Clancey, W.J.: Simulating Activities: Relating Motives, Deliberation, and Attentive Coordination. Cognitive Systems Research 3(3), 471–499 (2002)
2. Clancey, W.J., Sachs, P., Sierhuis, M., van Hoof, R.: Brahms: Simulating practice for work systems design. International Journal on Human-Computer Studies 49, 831–865 (1998)
3. Sierhuis, M.: Modeling and Simulating Work Practice; Brahms: A multiagent modeling and simulation language for work system analysis and design. Ph.D. thesis, University of Amsterdam, SIKS Dissertation Series No. 2001-10, Amsterdam, The Netherlands (2001)
4. Sierhuis, M., Clancey, W.J., van Hoof, R.J.J.: Brahms: A multiagent modeling environment for simulating work processes and practices. International Journal of Simulation and Process Modelling 3(3), 134–152 (2007)
5. Sierhuis, M.: "It's not just goals all the way down" – "It's activities all the way down". In: O'Hare, G.M.P., Ricci, A., O'Grady, M.J., Dikenelli, O. (eds.) ESAW 2006. LNCS (LNAI), vol. 4457, pp. 1–24. Springer, Heidelberg (2007)
6. Sierhuis, M., Clancey, W.J., van Hoof, R.J.J.: Brahms: An Agent-Oriented Language for Work Practice Simulation and Multi-Agent Systems Development. In: Bordini, R.H., Dastani, M., Dix, J., El Fallah-Seghrouchni, A. (eds.) Multi-Agent Programming, 2nd edn. Springer (2009)
7. Sierhuis, M., Clancey, W.J., Seah, C.: Organization and Work System Design and Engineering. In: Yilmaz, L., Oren, T. (eds.) Agent Directed Simulation. Wiley (2009)
8. Sierhuis, M., Jonker, C., van Riemsdijk, B., Hindriks, K., ELFallah-segfouchni, A.: Towards Organization Aware Agent-based Simulation. International Journal of Intelligent Control and Systems 14(1), 62–76 (2009)

Rules and Policy Based Handling of XML in Government Contexts Including NIEM
(Abstract of Tutorial)

David Webber

Information Architect at Oracle,
Senior Member of the ACM
drrwebber@acm.org

Managing information privacy and access policies is a critical need and technical challenge. Desired solutions should be both ubiquitous and syntax neutral, yet at the same time incorporate a simple and lightweight approach that meets legal policy requirements through the application of clear, consistent, and obvious assertions.

Today we have low-level tools that developers know how to use for implementation, and we have legal documents created by lawyers, both of which may address privacy and access concerns. However, there is a chasm between these two extremes.

The solution we are introducing will:

- Enable business information analysts to apply and manage policy profiles;
- Provide a clear separation between content and policy artifacts;
- Allow reuse of policies across content instances;
- Provide a clear declarative-assertions-based method, founded on policy approaches developed by the business rules technologies community;
- Leverage open software standards and tools.

This talk reports on joint work with Daniela Florescu, Oracle.

L. Morgenstern et al. (Eds.): RuleML 2013, LNCS 8035, p. 36, 2013.

Reasoning over 2D and 3D Directional Relations in OWL: A Rule-Based Approach

Sotiris Batsakis

Department of Electronic and Computer Engineering
Technical University of Crete (TUC)
Chania, Greece
batsakis@intelligence.tuc.gr

Abstract. Representation of spatial information for the Semantic Web often involves qualitative defined information (i.e., information described using natural language terms such as "North"), since precise arithmetic descriptions using coordinates and angles are not always available. A basic aspect of spatial information is directional relations, thus embedding directional spatial relations into ontologies along with their semantics and reasoning rules is an important practical issue. This work proposes a new representation for directional spatial information in ontologies by means of OWL properties and reasoning rules in SWRL embedded into the ontology. The proposed representation is based on the decomposition of cone shaped directional relations (CSD-9) offering a more compact representation and improved reasoning performance over existing approaches. A 3D representation is proposed as well and both 2D and 3D representations and reasoning are evaluated.

1 Introduction

Ontologies are formal definitions of concepts their properties and their relations. They form the basis of knowledge representation required for materializing the Semantic Web vision. Semantic Web technologies are used for automating tasks handled manually by users, tasks such as organizing a trip. Understanding the meaning of Web information requires formal definitions of concepts and their properties, using the Semantic Web Ontology definition language OWL. OWL provides the means for defining concepts, their properties and their relations and allowing for reasoning over the definitions and the assertions of specific individuals using reasoners such as Pellet. Furthermore, reasoning rules can be embedded into the ontology using the SWRL rule language.

Spatial information is an important aspect of represented objects in many application areas. Spatial information in turn can be defined using quantitative (e.g. using coordinates) and qualitative terms (i.e., using natural language expressions such as "East"). Qualitative spatial terms have specific semantics which can be embedded into the ontology using reasoning rules. In previous work [1] such a representation is proposed for both bi-dimensional (2D) spatial and temporal information in OWL.

Current work deals with the case of directional spatial information and proposes a new representation for such information which is more compact then the representation used in [1]. Specifically, instead of asserting one directional relation between two

L. Morgenstern et al. (Eds.): RuleML 2013, LNCS 8035, pp. 37–51, 2013.

points, such as "North-West", two relations are asserted (e.g., "North" and "West"). The first relation represents the relative placement of points along the North-South axis and the second along the East-West axis. Both relations correspond to cone shaped regions in the plane and their definitions and semantics are introduced in the current work. Reasoning is applied on each set of relations separately, achieving a decomposition of cone-shaped directional relations. Using the proposed representation both the number of required relations and the corresponding reasoning rules are significantly reduced offering increased reasoning performance. Specifically the required number of OWL axioms and SWRL rules for 2D representation have been reduced to 106, compared to 964 in [1].

The compactness of representation and the increased reasoning performance allows for extension of the proposed representation for three-dimensional (3D) space. To the best of author's knowledge this work is the first that proposes the optimized representation based on the decomposition of directional relations, and also the first that deals with 3D representation of directional relations in OWL ontologies.

Current work is organized as follows: related work in the field of spatial knowledge representation is discussed in Section 2. The proposed representation is presented at Section 3 and the corresponding reasoning mechanism at Section 4. The extension to three-dimensional space is presented at Section 5 followed by evaluation in Section 6 and conclusions and issues for future work in Section 7.

2 Background and Related Work

Definition of ontologies for the Semantic Web is achieved using the Web Ontology Language OWL[1]. The current W3C standard is the OWL 2[2] language, offering increased expressiveness while retaining decidability of basic reasoning tasks. Reasoning tasks are applied both on the concept and property definitions into the ontology (TBox) and the assertions of individual objects and their relations (ABox). Reasoners include among others Pellet[3], Fact++[4], RacerPro[5], KAON2[6] and Hermit[7]. Reasoning rules can be embedded into the ontology using SWRL[8]. To guarantee decidability, the rules are restricted to *DL-safe rules* [4] that apply only on named individuals in the ontology ABox. *Horn Clauses* (i.e., a disjunction of classes with at most one positive literal), can be expressed using SWRL, since Horn clauses can be written as implications (i.e., $\neg A \vee \neg B ... \vee C$ can be written as $A \wedge B \wedge ... \Rightarrow C$). The efficiency of reasoning over Horn clauses using forward chaining algorithms is a reason for choosing this form of rules. The antecedent (body) of the rule is a conjunction of clauses. Notice that, neither disjunction nor negation of clauses is supported in the body of rules. Also, the

[1] http://www.w3.org/TR/owl-ref/
[2] http://www.w3.org/TR/owl2-overview/
[3] http://clarkparsia.com/pellet/
[4] http://owl.man.ac.uk/factplusplus/
[5] http://www.racer-systems.com/
[6] http://kaon2.semanticweb.org/
[7] http://hermit-reasoner.com/
[8] http://www.w3.org/Submission/SWRL/

consequence (head) of a rule is one positive clause. Neither negation nor disjunction of clauses can appear as a consequence of a rule. These restrictions improve reasoning performance but complicate qualitative spatial reasoning, since disjunctions of clauses typically appear in the head of a spatial reasoning rule.

Qualitative spatial reasoning (i.e., inferring implied relations and detecting inconsistencies in a set of asserted relations) typically corresponds to Constraint Satisfaction problems which are NP, but tractable sets (i.e., solvable by polynomial algorithms) are known to exist [3]. Formal spatial representations have been studied extensively within the the Semantic Web community. Relations between spatial entities in ontologies can be topological, directional or distance relations. Furthermore, spatial relations are distinguished into qualitative (i.e., relations described using lexical terms such as "South") and quantitative (i.e., relations described using numerical values such as "45 degrees North")..

A representation of topological relations using OWL class axioms has been proposed in [6], but an alternative representation using object properties offered increased performance [5]. Embedding spatial reasoning into the ontology by means of SWRL rules applied on spatial object properties forms the basis of the SOWL model proposed at [1]. Based on the representation proposed at [1] the dedicated Pellet-Spatial reasoner [5] has been extended for directional relations in the CHOROS system [7] (Pellet-Spatial supports only topological relations). CHOROS achieved improved performance over the SOWL model but the spatial reasoner in not embedded into the ontology, thus requiring specific software which must be properly adjusted whenever modifications into the ontology occur. Furthermore, it does not offer support for 3D representation. SOWL on the other hand offers greater flexibility since it can be used and modified freely using only standard Semantic Web tools such as the Protégé editor and the Pellet reasoner[9]. In this work an improved representation of directional spatial relations based on decomposition of relations on each axis is proposed, analogously to the approach proposed for temporal interval relations in [2].

3 Spatial Representation

Directional relations in this work are represented as object properties between OWL objects representing points. For example if $Point1$ if $North\ Of\ Point2$ user asserts the binary relation $Point1\ North\ Point2$, or equivalently *North(Point1, Point2)*. This approach is similar to the approach used in [1] for directional relations as part of the SOWL model. In [1] between two points 9 different directional relations (*CSD-9* relations) can be defined, namely *North (N), NorthEast (NE), East (E), SouthEast (SE), South (S), SouthWest (SW), West (W), NorthWest (NW) and Identity*, corresponding to cone shaped regions (and the identity relation for identical points) in the two-dimensional (2D) space presented in Figure 1. This set of relations, known as CSD-9, is a special case of the modified star calculus presented in [8], when the lines separating the cone-shaped areas belong to only one of these areas. In this case reasoning over basic relations is decided by path consistency and it is tractable [8]. Also additional

[9] SWRL spatial reasoning rules and CHOROS are available on the Web at: `http://www.intelligence.tuc.gr/prototypes.php`

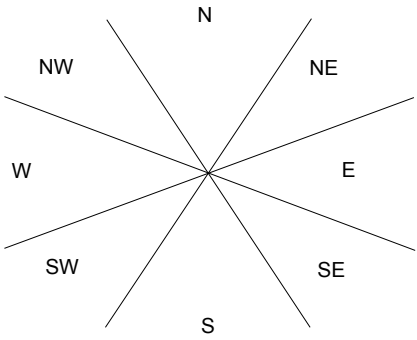


Fig. 1. Cone Shaped Directional Relations (CSD-9)

relations representing disjunctions of the above 9 relations are introduced in [1], since these additional relations are required for implementing reasoning rules similar to the rules proposed in Section 4. This leads to a complicated representation requiring 33 relations and 964 SWRL rules and OWL axioms [1].

Reducing the complexity of representation is necessary in order to improve performance and to allow for efficient 3D representation and reasoning. A representation based on projections on each axis and reasoning over the pairs of relations on these one-dimensional spaces, instead of cone shaped regions in bi-dimensional space, has been proposed as well in [1]. Note that this projection based representation has different semantics than the cone-shaped representation, thus it can not be consider as an alternative to it. For example, using the projection based approach, if a point is located far east relatively to another point and slightly north of it, following the projection based approach relations $East$ and $North$ will hold at the horizontal and the vertical axis respectively, thus and the $NorthEast$ relation. Following the cone-shaped approach only the relation $East$ holds which is conceptually right according to the way humans usually refer to directional relations.

In this work we follow the cone-shaped approach but relations are decomposed into two sets of relations, one for the East-West axis (horizontal) and one for the North-South axis (vertical) is case of 2D representation. Relations on each set are jointly exhaustive and pairwise disjoint but for each pair of points two relations, one from each set can hold. For example point A can be $North$ and $East$ of point B corresponding to the *North-East* CSD-9 cone-shaped relation.

The basic relations on each set are: *North, South, Equal-Vertical and Identical-Vertical* for the first set as presented in Figure 2 and *East, West, Equal-Horizontal and Identical-Horizontal* for the second set presented in Figure 3. Lines separating the cone-shaped regions belong to only one of the adjacent regions. By convention they belong to the $North$ and $South$ relations in the set of Figure 2 and to the $East$ and $West$ areas in in case of relations of Figure 3. Also relations *Identical-Horizontal* and *Identical-Vertical* are sub-properties of the $Identical$ property and also equivalent properties. Furthermore the implementation of the reasoning mechanism from Section 4 requires the definition of additional properties representing the disjunction of basic ones. These relations are the *Equal-North* (representing the fact that a point is equal vertically or

north of another) and Equal-South (representing the fact that a point is equal vertically or south of another) in the first set. In the second set the additional relations are *Equal-East* and *Equal-West* representing disjunction of equality with relations $East$ and $West$ respectively. Notice that, in total 8 basic and 4 additional relations are required for representation and reasoning in this work, compared to 9 basic and 33 total relations for directly implementing 2D cone-shaped CSD-9 relations.

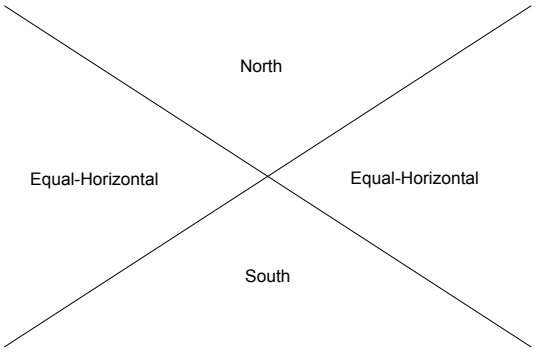


Fig. 2. North-South Relations

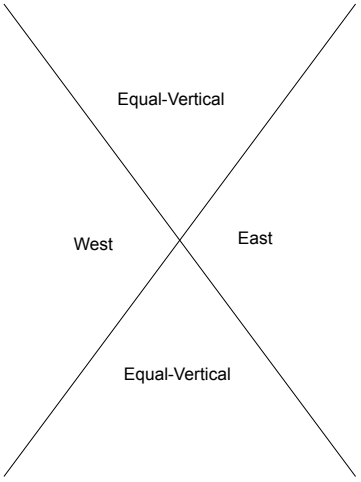


Fig. 3. East-West Relations

Additional OWL axioms required for the proposed representation; basic relations on each set are pairwise disjoint e.g., $North$ is disjoint with $South$. Also $North$ is $inverse$ of $South$ and $East$ is $inverse$ of $West$. Relations *Identical-Horizontal* and *Identical-Vertical* are symmetric. Relations *Equal-North*, Equal-South are the inverse of

each other, and the same holds for relations *Equal-East* and *Equal-West*. Summarizing, the proposed representation is conceptually equivalent to the cone-shaped representation of [1]. By decomposing the relations into two different sets the required number of relations is reduced to 8 basic relations and 4 additional ones. Between each pair of points, (in case of regions the points represent their centroid) two basic relations can hold. Specifically, decomposition of CSD-9 relations into proposed relations is defined as follows:

$$
\begin{aligned}
N_{CSD9}(x,y) &\equiv North(x,y) \wedge Equal\text{–}Vertical(x,y)\\
NE_{CSD9}(x,y) &\equiv North(x,y) \wedge East(x,y)\\
E_{CSD9}(x,y) &\equiv Equal\text{–}Horizontal(x,y) \wedge East(x,y)\\
SE_{CSD9}(x,y) &\equiv South(x,y) \wedge East(x,y)\\
S_{CSD9}(x,y) &\equiv South(x,y) \wedge Equal\text{–}Vertical(x,y)\\
SW_{CSD9}(x,y) &\equiv South(x,y) \wedge West(x,y)\\
W_{CSD9}(x,y) &\equiv Equal\text{–}Horizontal(x,y) \wedge West(x,y)\\
NW_{CSD9}(x,y) &\equiv North(x,y) \wedge West(x,y)\\
Identity_{CSD9}(x,y) &\equiv Identical\text{–}Horizontal(x,y) \wedge Identical\text{–}Vertical(x,y)
\end{aligned}
$$

4 Spatial Reasoning

Reasoning is realized by introducing a set of SWRL[10] rules operating on spatial relations. Reasoners that support DL-safe rules such as Pellet[11] can be used for inference and consistency checking over directional relations. Defining compositions of relations is a basic part of the spatial reasoning mechanism. Table 1 represents the result of the composition of two directional relations of Figure 2 (relations $North$, $South$, *Equal-Horizontal* and *Identical-Horizontal*, are denoted by "N","S","EqH", "IdH" respectively).

Table 1. Composition Table for North-South Directional Relations

Relations	N	S	EqH	IdH
N	N	N, S, EqH, IdH	N, EqH	N
S	N, S, EqH, IdH	S	S, EqH	S
EqH	N, EqH	S, EqH	N, S, EqH, IdH	EqH
IdH	N	S	EqH	IdH

Table 2 represents the result of the composition of two directional relation pairs of Figure 3 (relations $East$, $West$, *Equal-Vertical* and *Identical-Vertical*, are denoted by "E","W","EqV", "IdV" respectively).

[10] `http://www.w3.org/Submission/SWRL/`
[11] `http://clarkparsia.com/pellet/`

Table 2. Composition Table for East-West Directional Relations

Relations	E	W	EqV	IdV
E	E	E, W, EqV, IdV	E, EqV	E
W	E, W, EqV, IdV	W	W, EqV	W
EqV	E, EqV	W, EqV	E, W, EqV, IdV	EqV
IdV	E	W	EqV	IdV

Composition Table can be interpreted as follows: if relation R_1 holds between $point2$ and $point1$ and relation R_2 holds between $point3$ and $point2$, then the entry of the Table 1 corresponding to line R_1 and column R_2 denotes the possible relation(s) holding between $point3$ and $point1$. For example if $point2$ is $North$ of $point1$ and $point3$ is *Equal-Horizontal* to $point1$ then $point3$ is *North OR Equal-Horizontal* to $point1$. Entries in the above composition tables are determined using the following observation: composition of two relations corresponds to the addition of two vectors representing the relative placement of $point2$ to $point1$ and $point3$ to $point2$ forming angles θ_1 and θ_2 respectively with the horizontal axis. The resulting vector represents the relative placement of $point3$ to $point1$, i.e., the composition of two vectors, as illustrated in Figure 4. When adding the two vectors the resulting vector forms an angle θ with the horizontal axis such that $\theta_1 \leq \theta \leq \theta_2$. Angle θ defines the directional relation between $point1$ and $point3$. Using this observation it can be concluded for example that composing relations $North$ and *Equal-Horizontal* yields the disjunction of these two relations as a result.

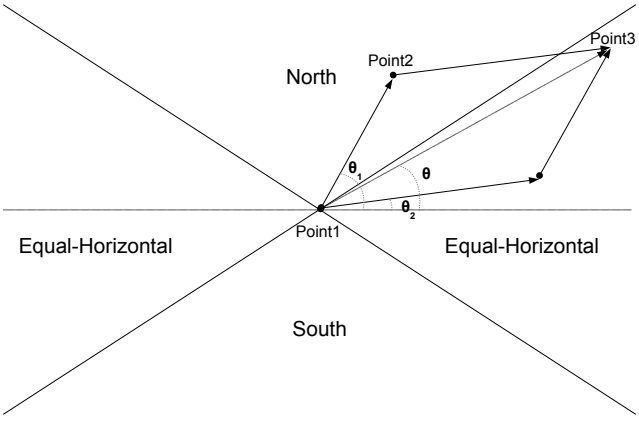


Fig. 4. Composition Example

A series of compositions of relations may yield relations which are inconsistent with existing ones (e.g., the above example will yield a contradiction if *point3 south of point1* has been also asserted into the ontology). Consistency checking is achieved by ensuring path consistency by applying formula:

$$\forall x, y, k \; R_s(x, y) \leftarrow R_i(x, y) \cap (R_j(x, k) \circ R_k(k, y))$$

representing intersection of compositions of relations with existing relations (symbol $\cap$ denotes intersection, symbol $\circ$ denotes composition and R_i, R_j, R_k, R_s denote directional relations). The formula is applied until a fixed point is reached (i.e., the application of the rules above does not yield new inferences) or until the empty set is reached, implying that the ontology is inconsistent. Implementing path consistency formula requires rules for both compositions and intersections of pairs of relations.

Compositions of relations R_1, R_2 yielding a unique relation R_3 as a result are expressed in SWRL using rules of the form:

$$R_1(x, y) \wedge R_2(y, z) \rightarrow R_3(x, z)$$

The following is an example of such a composition rule:

$$North(x, y) \wedge North(y, z) \rightarrow North(x, z)$$

Rules yielding a set of possible relations cannot be represented directly in SWRL since, disjunctions of atomic formulas are not permitted as a rule head. Instead, disjunctions of relations are represented using new relations whose compositions must also be defined and asserted into the knowledge base. For example, the composition of relations $North$ and *Equal-Horizontal* (EqH) yields the disjunction of two possible relations (*North* and *Equal-Horizontal*) as a result:

$$North(x, y) \wedge EqH(y, z) \rightarrow (North \vee EqH)(x, z)$$

If the relation N_EqH represents the disjunction of relations *North* and *EqH*, then the composition of $North$ and EqH can be represented using SWRL as follows:

$$North(x, y) \wedge EqH(y, z) \rightarrow N\_EqH(x, z)$$

A set of rules defining the result of intersecting relations holding between two points must also be defined in order to implement path consistency. These rules are of the form:

$$R_1(x, y) \wedge R_2(x, y) \rightarrow R_3(x, y)$$

where R_3 can be the empty relation. For example, the intersection of relations $North$ and $South$ yields the empty relation, and an inconsistency is detected:

$$North(x, y) \wedge South(x, y) \rightarrow \perp$$

Intersection of relations $North$ and N_EqH (representing the disjunction of $North$ and *Equal-Horizontal* yields relation $North$ as a result:

$$North(x, y) \wedge N\_EqH(x, y) \rightarrow North(x, y)$$

Thus, path consistency is implemented by defining compositions and intersections of relations using SWRL rules and OWL axioms for inverse relations as presented in Section 3.

Another important issue for implementing path consistency is the identification of the additional relations, such as the above mentioned N_EqH relation, that represent

disjunctions. Specifically the *minimal* set of relations required for defining compositions and intersections of all relations that can be yielded when applying path consistency on the basic relations of Figure 2 is identified. The identification of the additional relations is required for the construction of the corresponding SWRL rules.

In this work the closure method [3] of Table 3 is applied for computing the minimal relation sets containing the set of basic relations: starting with a set of relations, intersections and compositions of relations are applied iteratively until no new relations are yielded forming a set closed under composition, intersection and inverse. Since compositions and intersections are constant-time operations (i.e., a bounded number of table lookup operations at the corresponding composition tables is required) the running time of closure method is linear to the total number of relations of the identified set.

Applying the closure method over the set of basic North-South relations yields a set containing 7 relations. These are the four basic relations of Figure 2 and the relations *NorthEqualHorizontal* (denoted by *N_EqH*), representing the disjunction of relations $North$ and *EqualHorizontal*, *SouthEqualHorizontal* (denoted by *S_EqH*), representing the disjunction of relations $South$ and *EqualHorizontal*, and *N_S_EqH_IdH* or All denoting the disjunction of all relations. Applying the closure method over the set of basic South-West relations also yields a set containing 7 relations. These are the four basic relations of Figure 3 and the relations *EastEqualVertical* (denoted by *E_EqV*), representing the disjunction of relations $East$ and *EqualVertical*, *WestEqualVertical* (denoted by *W_EqV*), representing the disjunction of relations $West$ and *EqualVertical*, and *E_W_EqV_IdV* or All denoting the disjunction of all relations.

Table 3. Closure method

```
Input: Set S of tractable relations
Table C of compositions
WHILE S size changes
        BEGIN
            Compute C:Set of compositions of relations in S
            S=S ∪ C
            Compute I:set of intersections of relations in S
            S= S ∪ I
        END
RETURN S
```

A reduction to required relations and rules can be achieved by observing that the disjunction of all basic relations when composed with other relations yields the same relation, while intersections yield the other relation. Specifically, given that All represents the disjunction of all basic relations and, R_x is a relation in the supported set then the following holds for every R_x:

$$All(x, y) \wedge R_x(x, y) \rightarrow R_x(x, y)$$
$$All(x, y) \wedge R_x(y, z) \rightarrow All(x, z)$$
$$R_x(x, y) \wedge All(y, z) \rightarrow All(x, z)$$

Since relation All always holds between two points, because it is the disjunction of all possible relations, all rules involving this relation, both compositions and intersections, do not add new relations into the ontology and they can be safely removed. Also, all rules yielding the relation All as a result of the composition of two supported relations R_{x1}, R_{x2}:

$$R_{x1}(x,y) \wedge R_{x2}(y,z) \rightarrow All(x,z)$$

can be removed as well. Thus, since intersections yield existing relations and the fact that the disjunction over all basic relations must hold between two points, all rules involving the disjunction of all basic relations and consequently all rules yielding this relation can be safely removed from the knowledge base. After applying this optimization the required number of axioms for implementing path consistency over the set of directional relations of Figure 2 or Figure 3 is reduced to 52, while the combined implementation for relations of both Figure 2 and Figure 3 requires 106 axioms and rules, compared to the 964 axioms and rules required for reasoning over the cone-shaped directional relations of Figure 1 [1].

Reasoning over CSD-9 relations can be reduced to reasoning over the proposed 2D relations. This can be proved by decomposing CSD-9 relations into pairs of corresponding 2D relations, composing the resulting relations and checking if the resulting relations correspond to reasoning over CSD-9 relations using the composition table defined in [1,8]. All possible CSD-9 compositions are checked in order to establish the equivalence of the representations. Due to space limitations only a composition example will be provided, but all possible 81 compositions of CSD-9 basic relations can be redefined equivalently. For example the composition of CSD-9 relations N and NE yields the disjunction of relations N and NE as a result [1]. Specifically:

$$N_{CSD9}(x,y) \wedge NE_{CSD9}(y,z) \rightarrow N_{CSD9}(x,z) \vee NE_{CSD9}(x,z)$$

Using the proposed representation the composition of the above relations yields the same result; The corresponding 2D representation as defined in section 3 yields the compositions of relations *North (N)* and *North (N)* of Figure 2 and *Equal-Vertical (EqV), East(E)* of Figure 3. Composing these relations using compositions of Table 1 and Table 2 yields the same relation as the direct composition of the CSD-9 relations. Specifically:

$N_{CSD9}(x,y) \wedge NE_{CSD9}(y,z) \equiv (N(x,y) \wedge EqV(x,y)) \wedge (N(y,z) \wedge E(y,z))$
$\equiv (N(x,y) \wedge N(y,z)) \wedge (EqV(x,y) \wedge E(y,z)) \Rightarrow N(x,z) \wedge (EqV(x,z) \vee E(x,z))$
$\equiv ((N(x,z) \wedge EqV(x,z)) \vee (N(x,z) \wedge E(x,z)) \equiv N_{CSD9}(x,z) \vee NE_{CSD9}(x,z)$

Thus, composing the CSD-9 North and NorthEast relations using the corresponding 2D representation of Section 3 is equivalent to the composition defined in [1,8]. This equivalence also holds for intersections and inverses, thus the two representations are equivalent. An example of inverse operator, applied on the CSD-9 $North$ relation and yielding the desired CSD-9 $South$ relation using the equivalent 2D representation is the following:

$N_{CSD9}(x,y) \equiv N(x,y) \wedge EqV(x,y) \equiv S(y,x) \wedge EqV(y,x) \equiv S_{CSD9}(y,x)$

5 Three-Dimensional Representation and Reasoning

Representing points in three dimensional space is achieved by adding a third relation between two points (in addition to relations of Figure 3 and Figure 2). The basic relations on this additional set presented in Figure 5 are: *Up, Down, Equal-Height and Identical-Height*. Relation *Identical-Height* and relations *Identical-Horizontal* and *Identical-Vertical* are sub-properties of the $Identical$ property and also equivalent properties. These relations correspond to cone-shaped regions on a plane that two points belong, a plane that id perpendicular to the plane that 2D relations of section 3 are defined. The implementation of the reasoning mechanism (as in the 2D case of Section 4) requires the definition of additional properties representing the disjunction of basic ones. These relations which are detected using the closure method are the *Equal-Up* (representing the fact that a point has equal height or is higher than another point) and Equal-Down (representing the fact that a point has equal height or is lower than another point). Combined with existing relations for the 2D representation a total of 8 basic and 6 additional relations are required for representation and reasoning for 3D space.

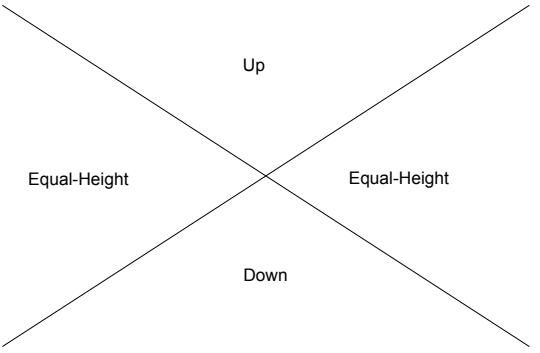


Fig. 5. Up-Down Relations

Table 4 represents the result of the composition of two directional relation pairs of Figure 3 (relations Up, $Down$, *Equal-Height* and *Identical-Height*, are denoted by "U","D","$EqHe$", "$IdHe$" respectively).

Table 4. Composition Table for Up-Down Directional Relations

Relations	U	D	$EqHe$	$IdHe$
U	U	$U, D, EqHe, IdHe$	$U, EqHe$	U
D	$U, D, EqHe, IdHe$	D	$D, EqHe$	D
$EqHe$	$U, EqHe$	$D, EqHe$	$U, D, EqHe, IdHe$	$EqHe$
$IdHe$	U	D	$EqHe$	$IdHe$

Reasoning rules in SWRL implementing path consistency for 3D directional relations, have been defined as well. These rules are almost identical to the rules presented

in Section 4, but they apply on properties of Figure 5. Also relation Up is the inverse of $Down$ and *Identical-Height* and *Equal-Height* are symmetric. New relations, rules and OWL axioms are combined with existing 2D representation requiring a total of 158 axioms and rules for 3D representation and reasoning. These are considerably fewer than the 964 axioms and rules required for 2D representation using the relations of Figure 1. Three-dimensional representation using directly an extension of CSD-9 relations, instead of their decomposition as proposed in the current work, will require thousands of rules and axioms and it will be impractical.

6 Evaluation

In the following the proposed representation and reasoning mechanism is evaluated both theoretically and experimentally.

6.1 Theoretical Evaluation

The required expressiveness of the proposed representation is within the limits of OWL 2 expressiveness. Reasoning is achieved by employing DL-safe rules expressed in SWRL that apply on named individuals in the ontology ABox, thus retaining decidability. Furthermore, since the proposed representation is equivalent to the CSD-9 representation, reasoning using the polynomial time path consistency algorithm is sound and complete, as in the case of CSD-9 relations.

Specifically, any point can be related with every other point with two basic directional relations (one of each set presented in Figures 2 and 3), because relations of each set are mutually exclusive, between n points, at most $2n(n-1)$ relations can be asserted (in case of 3D representation one additional relation belonging to the set presented in Figure 5 can be also be asserted leading to an upper limit of $3n(n-1)$). Furthermore, path consistency has $O(n^5)$ time worst case complexity (with n being the number of points). In the most general case where disjunctive relations are supported in addition to the basic ones, any point can be related with every other point by at most k relations, where k is the size of the set of supported relations (containing four additional relations for 2D and six for 3D besides the basic ones). Therefore, for n points, using $O(k^2)$ rules, at most $O(kn^2)$ relations can be asserted into the knowledge base.

Applying the *closure method* over the proposed directional relations the total number of relations required for 2D representation is 14 (or 12 if the disjunction of all relations for each set are eliminated as proposed in Section 4) compared to 33 for the representation presented at [1]. The required number of axioms is 106 compared to 964 at [1]. In case of 3D representation reasoning the required number of relations is 21 (or 18 after applying the optimizations proposed in Section 4) and the number of required axioms and rules is 158.

The $O(n^5)$ upper limit for path consistency running time referred to above is obtained as follows: At most $O(n^2)$ relations can be added in the knowledge base. At each such addition step, the reasoner selects 3 variables among n points which corresponds to $O(n^3)$ possible different choices. Clearly, this upper bound is pessimistic, since the overall number of steps may be lower than $O(n^2)$ because an inconsistency

detection may terminate the reasoning process early, or the asserted relations may yield a small number of inferences. Also, forward chaining rule execution engines employ several optimizations (e.g., the Rete algorithm employed at the SWRL implementation of Pellet), thus the selection of appropriate variables usually involves fewer than $O(n^3)$ trials. Nevertheless, since the end user may use any reasoner supporting SWRL, a worst case selection of variables can be assumed in order to obtain an upper bound for complexity. Nevertheless retaining control over the order of variable selection and application of rules yields an $O(n^3)$ upper bound for path consistency [5].

6.2 Experimental Evaluation

Measuring the efficiency of the proposed reasoner requires a spatial ontology, thus a data-set of 200 to 1000 points generated randomly was used for the experimental evaluation. Reasoning response times of the spatial reasoning rules are measured as the average over 5 runs. Pellet 2.2.0 running as a plug-in of Protégé 4.2 was the reasoner used in the experiments. All experiments run on a PC, with Intel Core 2 Duo CPU at 3.00 GHz, 4 GB RAM, and Windows 8.

Measurements illustrate that the proposed representation offers faster reasoning performance. Measurements over 2D points (using the decomposition to North-South and East-West relations) of Section 3 and 3D points of Section 5 are presented in Table 5. They are compared to measurements over the CSD-9 representation of [1]. Since the number of basic relations of CSD-9 relations is 9 (Figure 1) and because all possible disjunctions appearing in the supported set must also be supported, the CSD-9 representation is particularly involved. On the other hand, the CSD-9 representation requires only one relation between points while the proposed 2D representation requires two relations between points. Thus, the definition of n directional relations between n points, requires n assertions in case of CSD-9 relations, $2n$ in case of the proposed 2D representation and $3n$ in case of the proposed 3D representation. The reasoner will have to handle 964, 106 and 158 rules and axioms for the CSD-9 and the proposed 2D and 3D representations respectively.

In the following experiment, we measure the performance of reasoning in the cases of both the proposed 2D and 3D representations and the CSD-9 based representation, and their performance is discussed. In all cases, n random points and n random directional relations between them were asserted (using $2n$ and $3n$ assertions in the proposed 2D and 3D representations respectively), and reasoning times using Pellet are measured. Measurements of the time required by each approach for producing all inferred relations from a data set of random points are reported in Table 5. Each entry in the table is the average over 5 runs of the reasoner corresponding to 5 random instantiations of the ontology.

The evaluation indicated that the proposed 2D representation clearly outperforms the CSD-9 based approach, although the number of asserted relations is twice that of the CSD-9 approach. The proposed representation requires fewer rules and axioms (106) applied on a largest set of relations ($2n$) compared to the CSD-9 approach which requires n relation assertions and 964 OWL axioms and SWRL rules. The increased performance allows for a practical 3D representation and reasoning mechanism with performance equal to that of CSD-9 bi-dimensional representation.

Table 5. Average reasoning time for directional relations as a function of the number of points

Number of Points	Reasoning Time (ms)		
	2D	3D	CSD-9
200	299.8	268.6	386.0
400	405.8	685.8	782.4
600	865.2	1099.0	1066.4
800	1053.6	1407.2	1396.4
1000	1526.2	2261.4	2380.2

Summarizing, reasoning over the proposed 2D representation is approximately 30% faster over the CSD-9 based representation, due to the small number of axioms involved. This allows for an efficient and compact 3D cone-shaped directional representation and reasoning mechanism as well. This is the first such representation for 3D directional relations for the Semantic Web.

7 Conclusions and Future Work

In this work a representation framework for handling directional spatial information in ontologies is introduced. The proposed framework handles both, 2D and 3D information using an inference procedure based on path consistency. The proposed representation based on decomposition of CSD-9 relations offers increased performance over existing approaches [1]. Both the proposed and the existing representations are presented and evaluated.

The proposed representation is fully compliant with existing Semantic Web standards and specifications which increases its applicability. Being compatible with W3C specifications the proposed framework can be used in conjunction with existing editors, reasoners and querying tools such as Protégé and Pellet without requiring specialized additional software. Therefore, information can be easily distributed, shared and modified. Directions of future work include the development of real world applications based on the proposed mechanism. Such applications will combine temporal and topological spatial representations with the proposed directional representation and reasoning mechanism.

Acknowledgement. Research leading to these results has received funding from the European Community's Seventh Framework Program (FP7/2007-2013) under grant agreement No 296170 (Project PortDial).

References

1. Batsakis, S., Petrakis, E.G.M.: SOWL: A Framework for Handling Spatio-Temporal Information in OWL 2.0. In: Bassiliades, N., Governatori, G., Paschke, A. (eds.) RuleML 2011 - Europe. LNCS, vol. 6826, pp. 242–249. Springer, Heidelberg (2011)

2. Batsakis, S., Stravoskoufos, K., Petrakis, E.G.M.: Temporal Reasoning for Supporting Temporal Queries in OWL 2.0. In: König, A., Dengel, A., Hinkelmann, K., Kise, K., Howlett, R.J., Jain, L.C. (eds.) KES 2011, Part I. LNCS, vol. 6881, pp. 558–567. Springer, Heidelberg (2011)
3. Renz, J., Nebel, B.: Qualitative Spatial Reasoning using Constraint Calculi. In: Handbook of Spatial Logics, pp. 161–215. Springer, Netherlands (2007)
4. Motik, B., Horrocks, I., Rosati, R., Sattler, U.: Can OWL and Logic Programming Live Together Happily Ever After? In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 501–514. Springer, Heidelberg (2006)
5. Stocker, M., Sirin, E.: PelletSpatial: A Hybrid RCC-8 and RDF/OWL Reasoning and Query Engine. In: OWLED 2009. CEUR Workshop Proceedings, vol. 529, pp. 2–31 (2009)
6. Katz, Y., Grau, B.: Representing Qualitative Spatial Information in OWL-DL. In: Proc. of Int. Workshop: OWL Experiences and Directions, Galway, Ireland (2005)
7. Christodoulou, G., Petrakis, E.G.M., Batsakis, S.: Qualitative Spatial Reasoning using Topological and Directional Information in OWL. In: Proc. of 24th International Conference on Tools with Artificial Intelligence (ICTAI 2012), (November 7-9, 2012)
8. Renz, J., Mitra, D.: Qualitative Direction Calculi with Arbitrary Granularity. In: Zhang, C., Guesgen, H.W., Yeap, W.-K. (eds.) PRICAI 2004. LNCS (LNAI), vol. 3157, pp. 65–74. Springer, Heidelberg (2004)

Grailog 1.0: Graph-Logic Visualization of Ontologies and Rules

Harold Boley

National Research Council, Security and Disruptive Technologies
University of New Brunswick, Faculty of Computer Science
Fredericton, NB, Canada

Abstract. Directed labeled graphs (DLGs) provide a good starting point for visual data & knowledge representation but cannot straightforwardly represent non-binary relationships, nested structures, and relation descriptions. These advanced features require encoded constructs with auxiliary nodes and relationships, which also need to be kept separate from straightforward constructs. Therefore, various extensions of DLGs have been proposed for data & knowledge representation, including n-ary relationships as directed labeled hyperarcs, graph partitionings (possibly interfaced as complex nodes), and (hyper)arc labels used as nodes of other (hyper)arcs. Ontologies and rules have used extended logics for knowledge representation such as description logic, object/frame logic, higher-order logic, and modal logic. The paper demonstrates how data & knowledge representation with graphs and logics can be reconciled, inspiring flexible name specification. It proceeds from simple to extended graphs for logics needed in AI and the Semantic Web. Along with its visual introduction, each graph construct is mapped to its corresponding symbolic logic construct. These graph-logic extensions constitute a systematics defined by orthogonal axes, which has led to the Grailog 1.0 language aligned with the Web-rule industry standard RuleML 1.0.

1 Introduction

The visualization of *data* is useful in many areas, and needed for big data. Visualization, applicable to the entire extraction pipeline, supports knowledge discovery through data analytics.

This paper is about the graph visualization of data & knowledge for removing the entry barrier to logic. It proposes to proceed from 1-dimensional *symbol-logic* knowledge representation to 2-dimensional *graph-logic* representation in a systematic 2D syntax. The developed Graph inscribed logic (Grailog) is motivated by **uniform design**, e.g. across: data and knowledge; ontologies and rules; unary (e.g., class-like), binary, and n-ary relations; as well as atomic and complex relations (e.g., classes). This will support humans in the loop across knowledge elicitation, specification, validation, as well as reasoning. The graph representation is amenable to graph transformation, ('associative') indexing and parallel processing for the efficient implementation of specifications.

L. Morgenstern et al. (Eds.): RuleML 2013, LNCS 8035, pp. 52–67, 2013.

Grailog 1.0 is a standard for graph-logic knowledge built as a systematic combination of visual graph constructs mapped to corresponding symbolic logic constructs. Since its features are orthogonal (hence freely composable along the mutually independent axes), it is easy to learn, e.g. for (business) analytics. Grailog constitutes a generalized-graph framework providing a uniform expressive 2D syntax for major (Semantic Web) logics. Users should be allowed to: pick a Grailog subset as the target for each knowledge elicitation project; specify and semantically validate the knowledge in this subset; get it translated to an equivalent RuleML sublanguage for XML validation, transformation, and exchange; and pose queries to a RuleML engine, via a Grailog interface to RuleML, or through an alternate implementation of Grailog.

Grailog was designed according to the following principles:

- Graphs should ease human knowledge management by uniformly visualizing 1D notations, from (controlled) natural language to symbolic logic, via *orthogonal* 2D constructs supporting (Frege's principle of) *compositionality*
- They should be *natural extensions* (e.g., n-ary) of Directed Labeled Graphs (DLGs), often used to represent simple semantic nets, i.e. of atomic ground formulas in function-free dyadic predicate logic; e.g.: binary Datalog ground facts, RDF triples, Open Graph (https://developers.facebook.com/docs/opengraph), and Knowledge Graph (http://www.google.com/insidesearch/features/search/knowledge.html)
- They should allow *stepwise DLG refinements* for expressive logics: description logic constructions, F-logic frames, PSOA RuleML [Bol11] terms, etc.

To realize these principles, and building on the earlier Directed Recursive Labelnode Hypergraphs (DRLHs [Bol92]), Grailog 1.0's standard 2D visualization syntax is based on the orthogonal combination of three graph generalizations:

Directed hypergraphs: For n-ary relationships, relation-labeled directed (binary) arcs should be generalized to (n-ary) *directed hyperarcs*, e.g. representing relational-database tuples (contrast: *undirected hyperarcs* [Ber73])

Recursive (hierarchical) graphs: For nested terms and formulas, modal logics, and modularization, 'flat' graphs should be generalized to allow other graphs as *complex nodes* to any level of nesting 'depth'

Labelnode graphs: For allowing higher-order logics describing both instances and relations (predicates), arc *labels* should also become usable as *nodes*

Grailog's above fundamental generalizations will be explained in Sections 2, 3, and 4. Sections 5, 6, and 7 will augment these for classes, description logic, and Horn logic. The 'psoa' characteristics of slot and tuple hyperarcs centered on Object IDentifier (OID) nodes as well as psoa rules will follow in Section 8. All of these Grailog features will be mapped to symbolic logics including a 1D presentation syntax as well as Hornlog RuleML [BPS10] and PSOA RuleML.

2 Directed Hyperarcs

In Grailog, a graph can have *directed n-ary hyperarcs* each diagrammed as an arrow starting with a tail at the 1^{st} node and ending with a head at the n^{th} node,

like directed binary arcs (n=2) do, while cutting, in an ordered manner, through any intermediate 2^{nd}, ..., (n-1)st nodes. These hyperarcs can be labeled like binary arcs. The resulting hypergraph diagrams unambiguously distinguish the hyperarcs of *arrow crossings* by following the trajectory of each arrow as done for binary arcs, thus avoiding the need for a half-circle symbol for non-connecting (hyper)arcs. Figs. 1-5 illustrate this by proceeding, as a normalization sequence, from hyperarc crossings to node copies. All figures visually represents the symbolic (controlled) English sentences "John shows Latin to Kate. Mary teaches Latin to Paul." They use layout variants of a directed hypergraph consisting of two 3-ary hyperarcs, one for each sentence.

The connected **nodes** `John`, `Latin`, `Kate`, etc. are *unique names* denoting different entities (implicit disequality). Section 3 will complement these with **node labels** for *non-unique names* which may be (explicitly) equated.

Fig. 1 enlarges the only intermediate (2^{nd}) node, `Latin`, to facilitate its shared, 'crossing' use by the `Show`- and `Teach`-labeled hyperarc arrows, which also helps in (naive) DLG approximation and correlates with *degree centrality*.

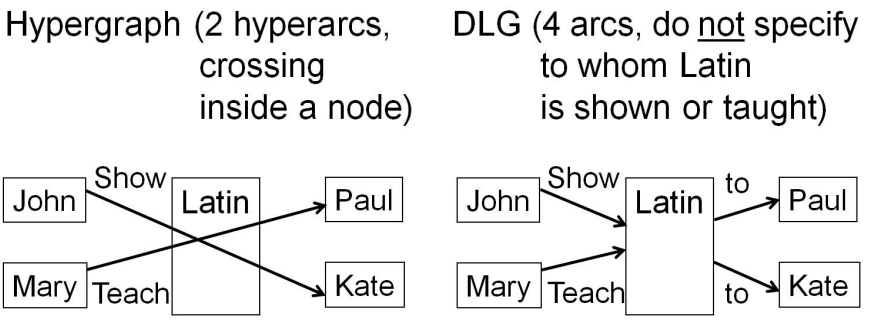


Fig. 1. Distinguishing showing and teaching via hyperarcs crossing inside Latin node

Fig. 2 moves the hyperarc crossing out of the `Latin` node, making the hypergraph look more (DLG-)familiar. It keeps the DLG approximation for contrast.

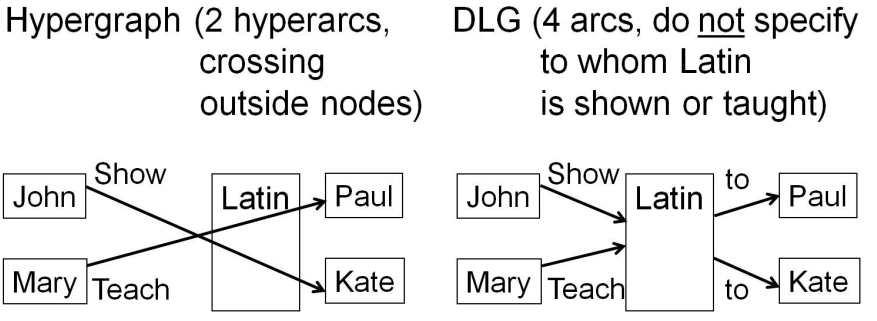


Fig. 2. Distinguishing showing and teaching via hyperarcs crossing outside Latin node

Fig. 3 disentangles the hyperarcs, illustrating that, graph-theoretically, the hyperarc for, e.g., the ternary Show(John,Latin,Kate) can be seen as the path composition of the arcs for the two binary Show(John,Latin) and to(Latin,Kate). The DLG, diagrammed with swapped `Kate` and `Paul` nodes, would be able to capture this if, e.g., different colors were used for the arcs on each path.

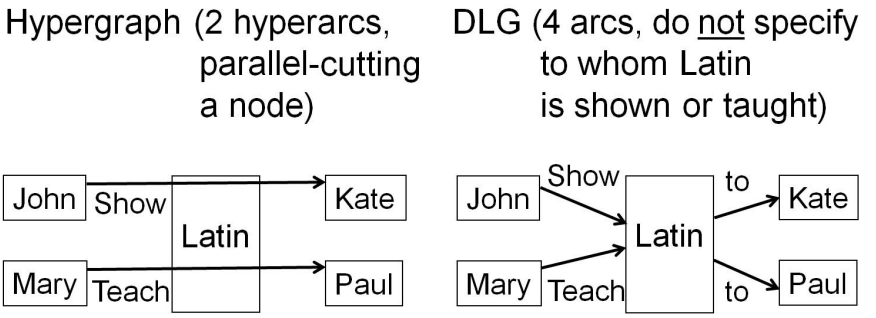


Fig. 3. Distinguishing showing and teaching via disentangled hyperarcs

Fig. 4 gives the correct (but verbose) DLG representation, highlighting the advantages of directed n-ary hyperarcs for n>2: avoiding reification of n-ary relationships and "structural links" [Woo07]. But Grailog also allows DLGs as a special case, can reify labels as nodes (cf. Section 4), and visualizes OID-centered frames with slots usable instead of positional relation arguments (cf. Section 8).

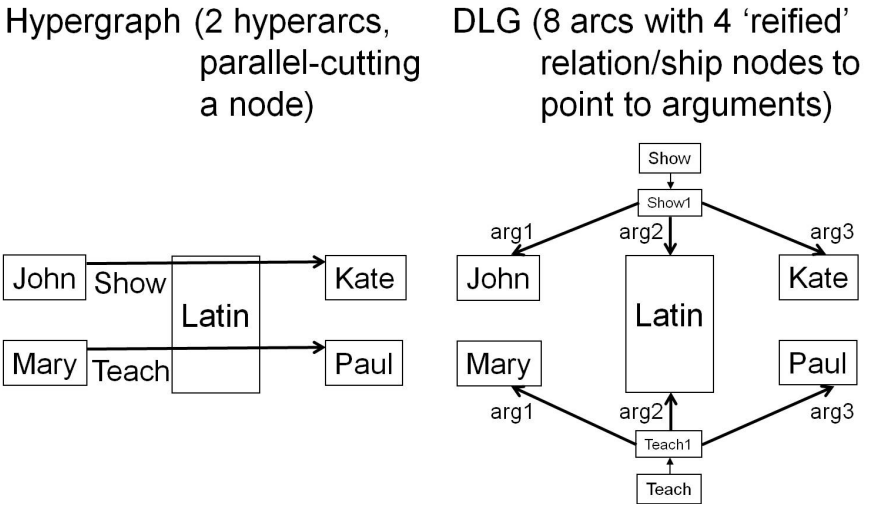


Fig. 4. Distinguishing showing and teaching also with DLG arcs

Fig. 5 splits the `Latin` node to facilitate the mapping of the hypergraph to a symbolic logic representation with two *atomic formulas* (relationships) using the 3-ary predicates `Show` and `Teach`. Both `Latin` occurrences remain one node even after copying: Occurrences having the same unique name can be merged.

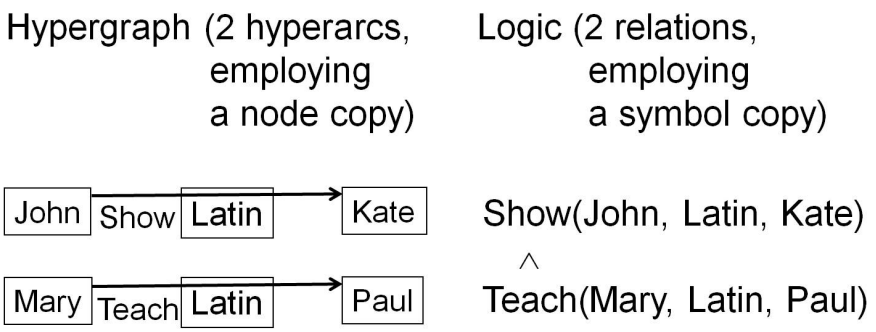


Fig. 5. Copying Latin node for correspondence to symbolic logic

In the RuleML family [BPS10] of logics, the conjunction of the `Show` and `Teach` atomic formulas for assertions is reflected by a `<Rulebase>` containing two `<Atom>`s. An alternate, special-purpose Order-Labeled Tree visualization and the RuleML/XML serialization for such atomic ground facts are introduced in the RuleML Primer (http://ruleml.org/papers/Primer).

3 Complex Nodes

In Grailog, a graph besides *elementary nodes* can have *complex nodes* each drawn as an enclosing box containing another graph, which can again have complex nodes etc., down to any level of nesting. Complex nodes constitute the natural construct for knowledge modularization and for (epistemic) modal logic (cf. http://www.cs.unb.ca/~boley/talks/RuleMLGrailog.pdf). The (conjunctive, disjunctive, ...) contents of complex nodes is amenable to (And-/Or-/...)parallel processing. There are different kinds of enclosing boxes which can be combined along three orthogonal axes. For example, the explicit logical "∧" symbol in Fig. 5 of Section 2 can be reflected in Grailog by a box with a conjunctive line style.

The **first axis** distinguishes logical connectives using line styles *solid vs. dashed* and, again orthogonally, *linear vs. wavy*. Figs. 6-8 illustrate this connective axis with conjunction, negation (of one or more conjuncts), and disjunction. Here, the *node labels* `WB`, `GE`, `JS`, and `VW` (in the symbolic logic prefixed with a vertical bar) are *non-unique names* for the individuals Warren Buffett, General Electric, Joe Smallstock, and Volkswagen, respectively; e.g., Joe Smallstock's node, labeled `JS`, could carry a second label, `JoeS`. Such labels graph-theoretically and logically complement the *nodes* as *unique names* of Section 2. Refining the global **(Non-)Unique Name Assumption**, Grailog thus combines occurrence-specific **Unique Name Specification (UNS)** and

Non-unique Name Specification (NNS) to **flexible Name Specification (xNS**, with x = U or N).

Fig. 6 introduces the general case and an `Invest` example for a (solid+linear) *positive conjunction*. While conjunction is implicit on the top-level of Grailog graphs, enclosing boxes must be made explicit for embedded conjunctions. A node like `US$ 3·10^9` exemplifies UNS for string-like data, e.g. RDF literals [Bol01].

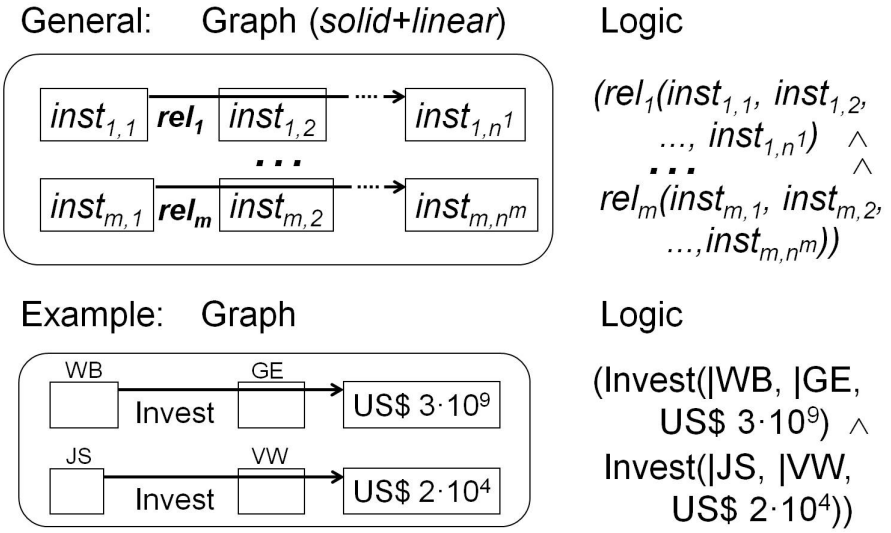


Fig. 6. General conjunction complex node with investment example

Fig. 7 shows a `Nand`, i.e. a (dashed+linear) *negative conjunction*, which can be regarded as a shorthand for the nesting of a (dashed+linear) negation over the complex node of a (solid+linear) conjunction.

Fig. 8 gives a (solid+wavy) *positive disjunction*. This could also be used inside a negation to achieve a `Nor`, i.e. a (dashed+wavy) *negative disjunction*.

The **second axis** concerns the corners of boxes, which can be *pointed vs. snipped vs. rounded* to, respectively, quote/copy vs. reify/instantiate vs. evaluate contents. The *rectoval* complex nodes in Figs. 12 and 13 are pointed (inward) to indicate that their contents is to be quoted (unchanged). The *snipangle* variable nodes in Figs. 14 and 18 are snipped to indicate they should be instantiated (dereferenced). The *roundangle* complex nodes in Figs. 6-8 are rounded to indicate evaluation of their contents, as needed for queries (rather than assertions).

The **third axis** varies the rectangle-derived shapes of boxes, which can be composed from sides that are *straight vs. concave vs. convex* for, respectively, neutral vs. functional vs. relational contents. The nodes in Figs. 6-8 use (function/relation-)neutral straight sides. Functional nodes use two or four concave sides (cf. http://www.cs.unb.ca/~boley/talks/RuleMLGrailog.pdf). Relation applications use two convex sides, relations as in Figs. 12 and 13 use four.

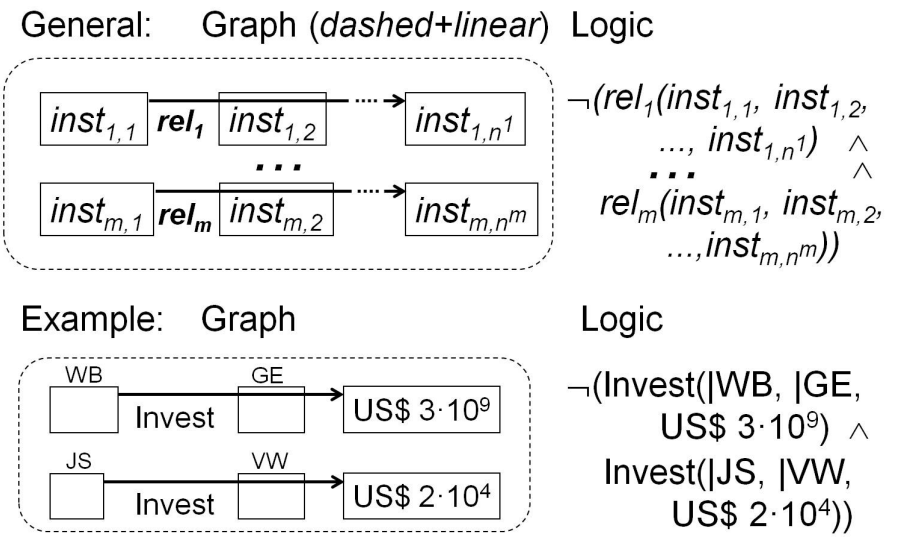


Fig. 7. General `Nand` complex node with investment example

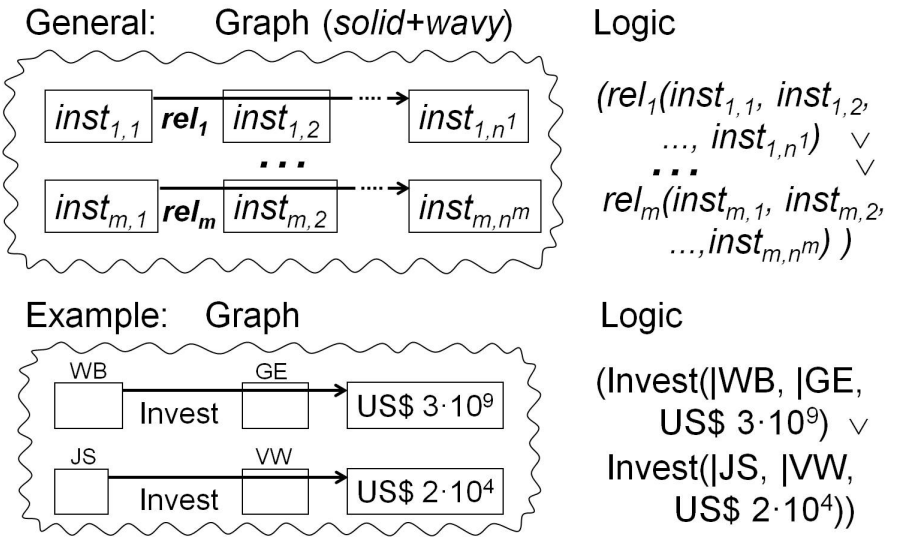


Fig. 8. General disjunction complex node with investment example

Fig. 9 summarizes the Grailog systematics for the second and third axes of corners and shapes, where all matrix cells are filled due to orthogonality.

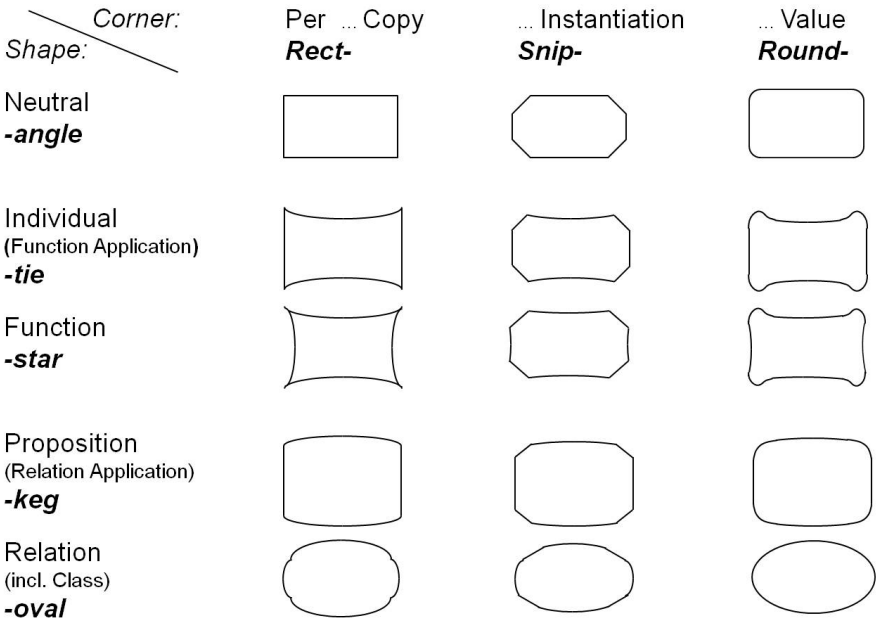


Fig. 9. Node systematics for box corners and shapes from Rectangle to Roundoval

4 Labelnodes

In Grailog, a graph can have *labelnodes* which act as node-reified (hyper)arc labels when diagrammed as a special 0^{th} node starting a (hyper)arc arrow in which this label-as-node is followed by the n regular nodes. A labeled (hyper)arc as in Section 2 can thus be normalized by moving the predicate name from the arrow-labeling position to the extended arrow's 0^{th} label-as-node position.

Fig. 10 introduces the general normalization of predicate *labels on hyperarcs* to *labelnodes starting hyperarcs* and exemplifies this with the `Invest` hyperarc.

Once a first-order predicate is diagrammed as a labelnode, second-order predicates can be applied to it. For example, the second-order predicate `Inverse` is applicable to the labelnodes `Invest` and `Accept`, `Inverse(Invest,Accept)`, so that `Accept(|GE,|WB,US$` $3{\cdot}10^9$`)` is inferable in such a higher-order logic.

5 Classes

Types or *classes* (sometimes also called 'concepts' or 'categories') are crucial for modeling because the typing of an instance or its *membership* in a class

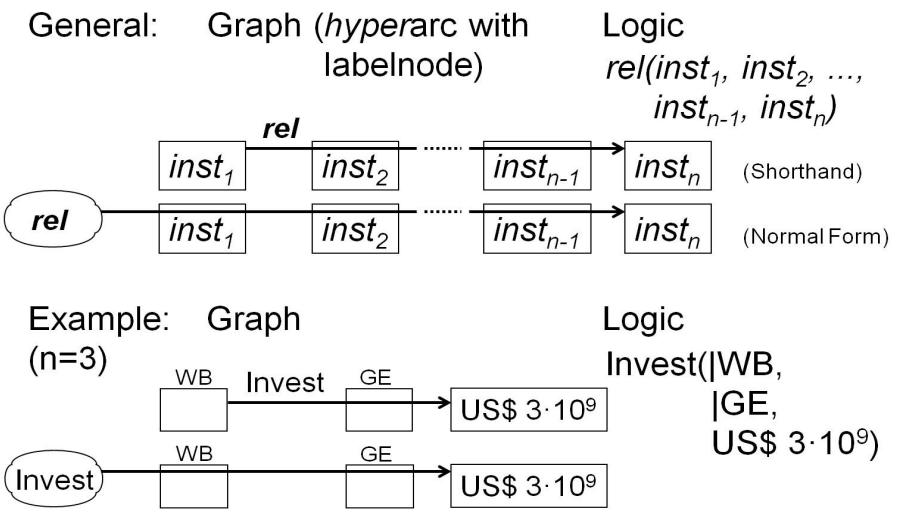


Fig. 10. General labelnode normalization with investment example

makes the instance amenable to classification also by higher levels of a *taxonomy* (i.e., an *ontology* consisting only of a class/generalization tree/hierarchy or DAG/heterarchy). Logically, such classes can be seen as predicates of arity n=1. Hence the labelnode normal form of hyperarcs in Section 4 (with n=1) is a natural means for stating membership, where the label-as-node is the class and the single argument node is the instance.

Fig. 11 introduces general typing and gives a `Billionaire` membership example; the unary hyperarc can also be viewed as an arc with label `HasInstance`.

6 Description Logic Constructions

Description logic permits the (intensional) *construction* of classes from other classes, without need for naming constructed classes. Constructions, complex nodes in Grailog, may use Boolean constructors or class-property restrictions.

Fig. 12 introduces *class intersection* and shows the option of applying the constructed class to an instance (exactly like for atomic classes in Section 5, because of compositionality). It also gives an example of such a membership in a class constructed by 3-way intersection. While most description logics make the Non-unique Name Assumption, Grailog uses xNS, here UNS (cf. Section 3).

Class union and complement are analogous, where the three Boolean class constructors vary in line styles exactly like the logical connectives in Section 3.

Fig. 13 introduces *existential class-property restriction* and shows its optional application to an instance that has the same property, `binrel`. It also gives an example of such a membership. The upper part is a kind of schema, where the **T**op class becomes specialized to have a (multi-valued) attribute/property, `Substance`, with *at least one* value typed by class `Physical`.

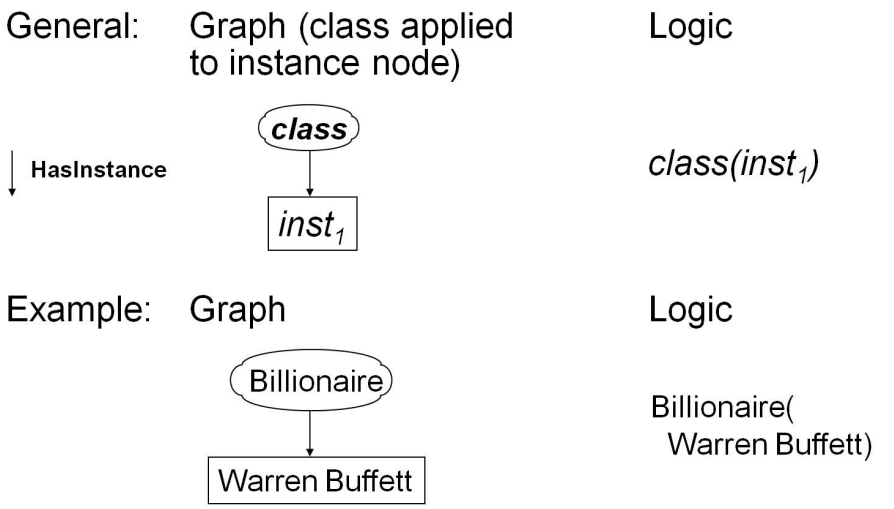


Fig. 11. General typing with billionaire example

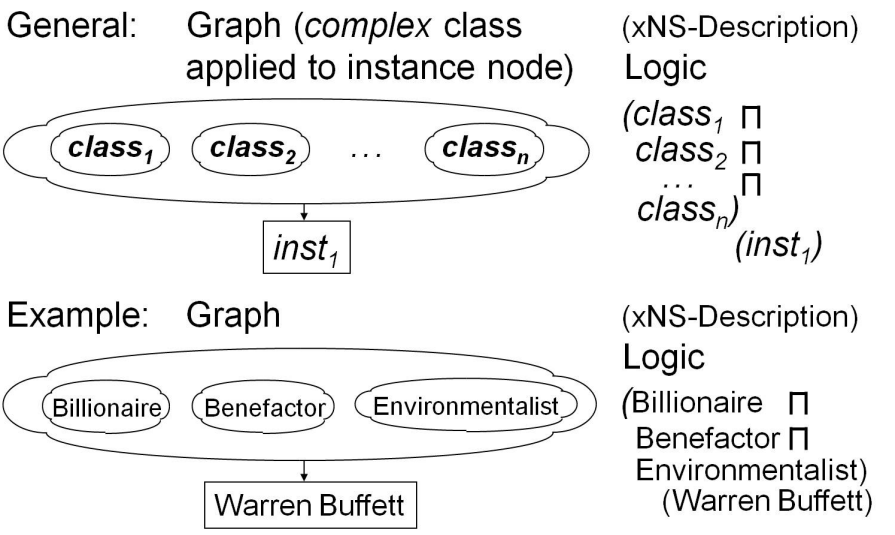


Fig. 12. General intersection with billionaire++ example used for typing

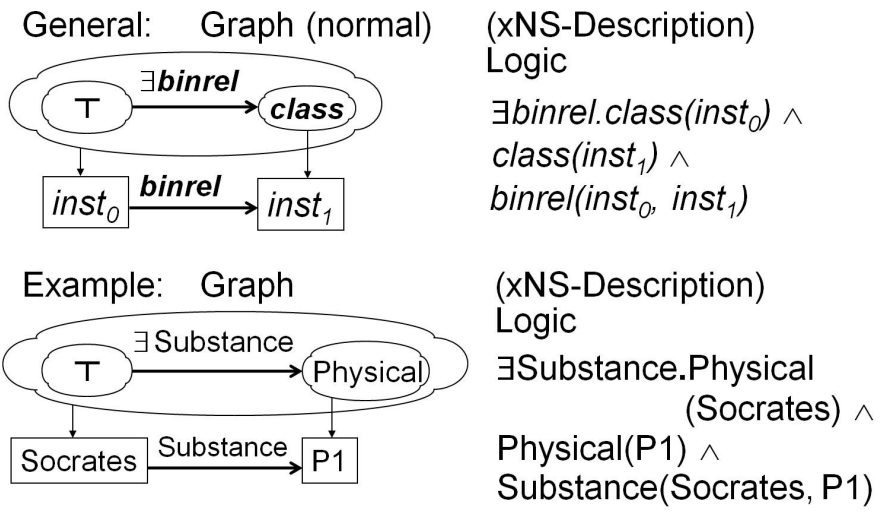


Fig. 13. General existential restriction with substance example used for typing

Universal class-property restriction is the dual of the existential one.

This Description Grailog for ontologies can benefit from, and contribute to, Protégé visualization plug-ins for OWL ontologies such as Jambalaya/OntoGraf and OWLViz. A preliminary Grailog case study in Cognitive Science was done, via a course (https://www.ict.tuwien.ac.at/lva/Boley_LFCS/index.html), using earlier versions of Grailog ontologies (related: [HSTC11]) and rules (cf. Section 7), to be updated for Grailog 1.0 and extended by frames (cf. Section 8).

7 Horn Logic Clauses

Horn logic permits the assertion of *definite clauses* (facts and rules). Section 2 has shown how an atomic formula (applying a predicate to n arguments) can be asserted as a *fact*. A *rule* derives an atomic head formula from an atomic or conjunctive body of such formulas. Horn Grailog specializes to function-free (Datalog) Grailog. For an earlier hyperarc visualization of Datalog see [Bol01].

Fig. 14 introduces 2-conjunct Datalog rules and gives an `Invest`/`Trust` example, where variables are distinguished via hatching. It says: "If `WB` invests in some entity `Y` any amount `A` and `JS` trusts `Y`, then `JS` invests in `Y` the amount `US$` $5{\cdot}10^3$." Generalization to m-premise rules is done with the m-conjunct box shown in Fig. 6 of Section 3. While Horn logic programming (e.g., Prolog) usually makes the global Unique Name Assumption, Grailog's xNS here becomes NNS for `WB` plus `JS` and UNS for `US$` $5{\cdot}10^3$ (cf. Section 3).

Fig. 15 shows a RuleML/XML serialization of the example, proposing an attribute `unique` with value `"no"` for NNS, and `"yes"` for UNS as the default.

Datalog can be easily extended to Horn logic by allowing function applications (cf. Fig. 9 of Section 3), i.e. complex terms, as arguments to predicates.

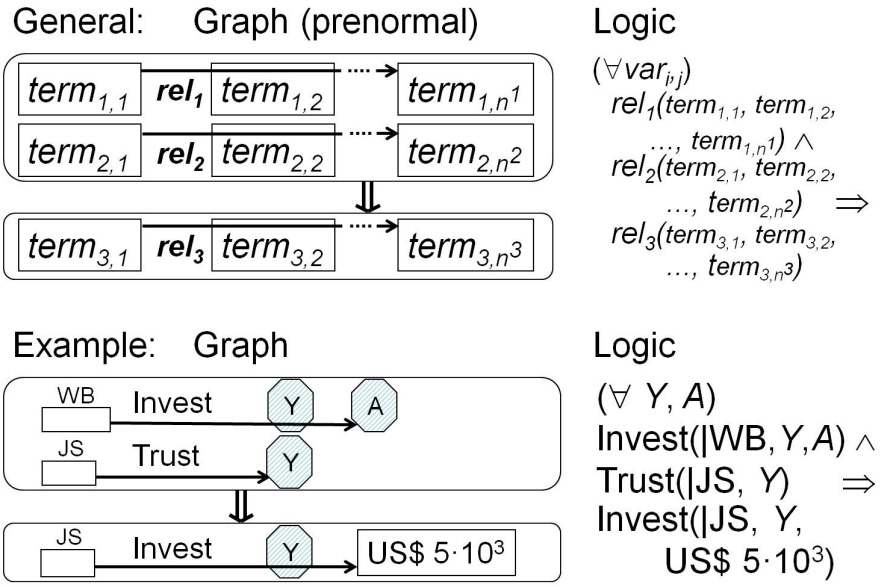


Fig. 14. 2-premise Datalog rule with investment-imitation example

Example: RuleML/XML

```
<Implies closure="universal">
  <And>
    <Atom>
      <Rel>Invest</Rel>
      <Ind unique="no">WB</Ind>
      <Var>Y</Var>
      <Var>A</Var>
    </Atom>
    <Atom>
      <Rel>Trust</Rel>
      <Ind unique="no">JS</Ind>
      <Var>Y</Var>
    </Atom>
  </And>
  <Atom>
    <Rel>Invest</Rel>
    <Ind unique="no">JS</Ind>
    <Var>Y</Var>
    <Data>US$ 5·10³</Data>   <!– superscript "³" to be parsed as Unicode U+00B3 -->
  </Atom>
</Implies>
```

Logic

$(\forall\ Y, A)$
$(Invest(|WB, Y, A) \wedge$
$Trust(|JS,\ Y) \Rightarrow$
$Invest(|JS,\ Y,\ US\$\ 5{\cdot}10^3))$

Fig. 15. Datalog RuleML/XML for the investment-imitation example

While Fig. 14 uses hyperarc labels as a shorthand, Fig. 10 of Section 4 shows their labelnode normalization. Compositionality will then allow replacing, e.g., `Invest` with a 3-ary description as a λ-construction from other predicates, visualizing the `Lambda` definitions of Functional RuleML (http://ruleml.org/fun): `Invest` = λ`(X,Y,A) (Pay(X,Y,A)`$\wedge\exists$`(E) (Expect(X,Y,E)`$\wedge$ `Benefit(E,X))).`

8 Frames, Positional-Slotted Terms, and Psoa Rules

Frames of F-logic, W3C RIF, and PSOA RuleML, which can be regarded as 'logical records', associate slots with an OID-distinguished instance.

Fig. 16 introduces frames and gives a `Philosopher` sample fact whose OID `Socrates` is associated with a `Substance` and a `Teaching` slot.

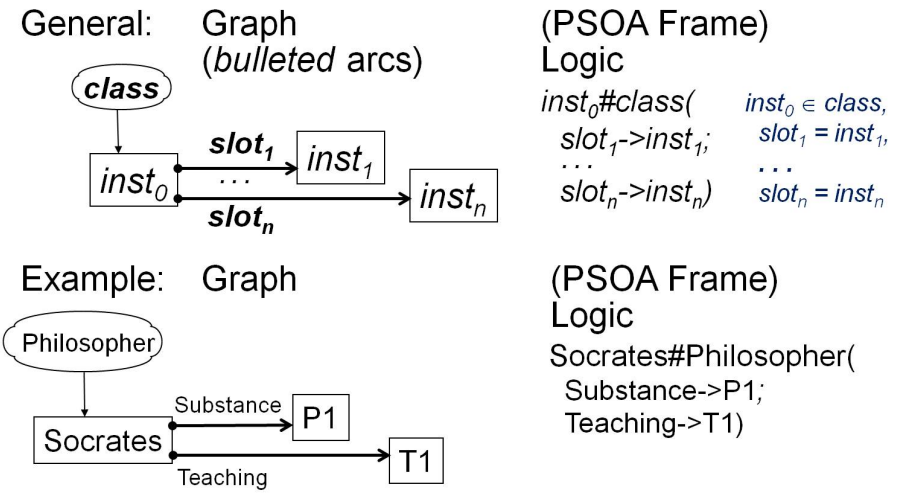


Fig. 16. General frame with philosopher example

Positional-slotted (psoa) terms [Bol11] can also associate tuples with an OID. Note that both slots and tuples use a bullet at the beginning of the arrow to distinguish their bullet-attached start node as playing the role of the OID.

Fig. 17 introduces single-tuple psoa terms and adds a lifetime tuple to the two slots of the `Philosopher` sample fact.

Psoa rules [Bol11] can be defined over psoa terms using existential OID variables in rule heads, which lead to new OID constants in derived psoa (e.g., frame) ground facts. PSOA RuleML provides such clauses via its PSOATransRun reference implementation (http://wiki.ruleml.org/index.php/PSOA_RuleML).

Fig. 18 adapts, from [Bol11], the psoa rule for deriving `family` frames based on two relational facts, where the shorthand $\underline{?}i$ is used for a head-existential variable `?`i: $\underline{?}$`1#family(...)` = `Exists ?1 (?1#family(...))`. The rule says:

"If `?Hu` is `married` with `?Wi` and `?Hu` or `?Wi` has `kid ?Ch`, then there exists a `family` OID `?1` with `husb ?Hu`, `wife ?Wi`, and `child ?Ch`."

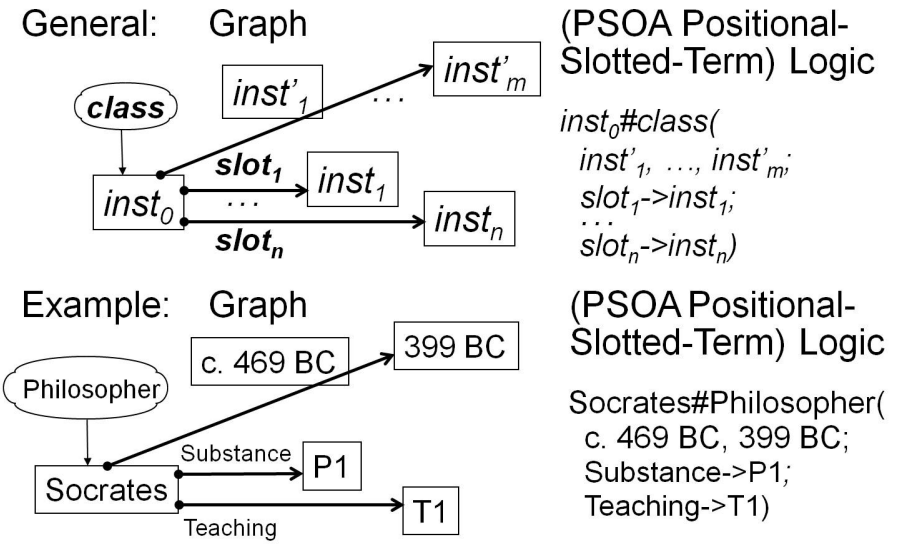


Fig. 17. Single-tuple psoa term with extended philosopher example

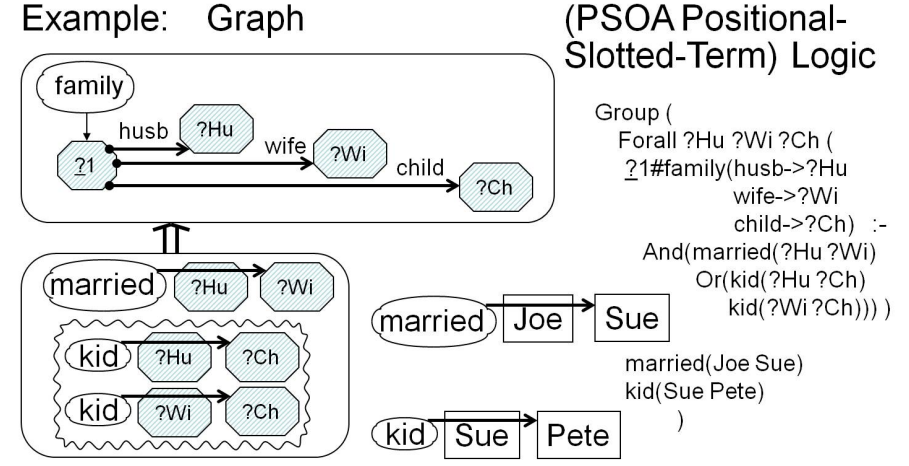


Fig. 18. Psoa family example with given relationships and rule

Fig. 19 shows the frame fact *deduced* from the relational facts by the psoa rule, where o is a new OID constant. Family frames can thus be derived 'on demand'. Conversely, a rule as in Fig. 18 is *inducible* from Fig. 19-like facts via association psoa-rule mining. A general psoa variant of the rule could keep the variables in the `husb` and `wife` slots positional, as in the relational `married` fact, and use only one (multi-valued) `child` slot: `?1#family(?Hu ?Wi child->?Ch) :- ...`.

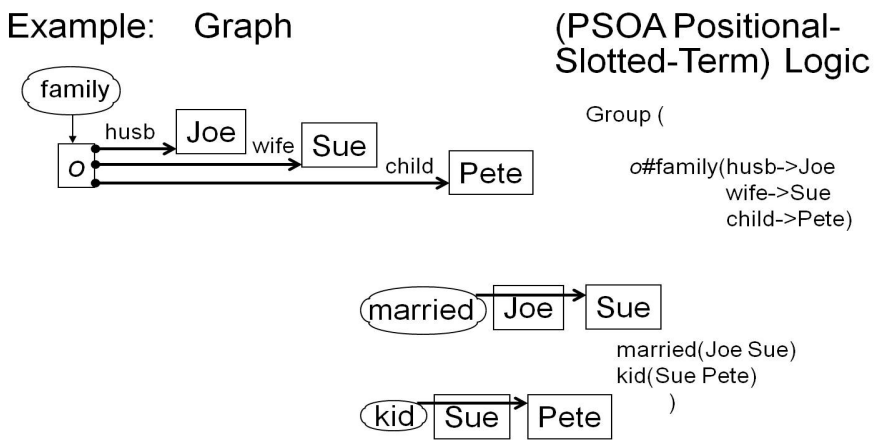


Fig. 19. Ground facts (given: relational, derived: frame) modeling family semantics

9 Conclusions

This paper discusses Grailog 1.0, which incorporates feedback on earlier versions. Grailog 1.0 introduces a novel box & arrow systematics using orthogonal features, including flexible name specification (xNS). The paper's focus is on intertranslating graphs with a symbolic-logic presentation syntax and RuleML/XML. Grailog use cases range from cognition to technology to business. The processing of the earlier Grailog-like DRLHs was studied in Lisp, FIT, and Relfun. This is now aligned with the Web-rule industry standard RuleML. Grailog has been serving as a vehicle for making central (description- and Horn-)logical notions of ontologies and rules accessible to newcomers to AI and the Semantic Web. The Grailog Initiative (http://wiki.ruleml.org/index.php/Grailog) is exploring Grailog in HyperGraphDB [Ior10], subsets, extensions, and use cases. E.g., box-boundary and arrow thickness can uniformly indicate *relevance*, with a 'tooltip' showing its (e.g., numerical) value, while an overt arrow label can indicate *certainty*.

The comparison of Grailog with other graph formalisms can build on DRLH [Bol92] comparisons. For Conceptual Graphs (CGs) [CM09], interoperation may be attempted, since a symbolic version of CGs, CGIF, was already compared with RuleML (http://philebus.tamu.edu/pipermail/cl/2010-October/002179.html), which acts as a symbolic version of Grailog. Since (model-theoretic) semantics is usually associated with symbolic presentation syntax (e.g., [Bol11]), Grailog's semantics is reached via graph-to-symbolic mapping. It will be interesting to apply semantic interpretations directly to Grailog graphs.

A Grailog tool library should be realized, e.g. for graph indexing and querying, graph transformations (normal form, typing homomorphism, merge, etc.), and advanced graph-theoretical operations (e.g., path tracing), exploiting possible parallelism. A Grailog structure editor should be developed, e.g. supporting autospecialize of neutral application/operator boxes for functions or relations, and a "highlight & visualize" for regions of large symbolic Grailog documents.

Stereo displays allow us to further proceed from the 2-dimensional (planar) Grailog 1.0 to a 3-dimensional (spatial) version, making Grailog data & knowledge 3D-visitable and -tangible for human analysts. Bidirectional 1D-2D-3D transitions between (zoomed) highlighted regions may themselves be visualized.

Acknowledgements. Thanks to Dietmar Dietrich for ICT invitations. Also, to students of my course "Logical Foundations of Cognitive Science", triggering early Grailog development (https://www.ict.tuwien.ac.at/lva/Boley_LFCS/index.html), and to students of "Semantic Web Techniques" for feedback on the Grailog versions since 2008. In particular, I thank the 2012 class, where Grailog subsets were implemented by Teams 1 and 8 (http://www.cs.unb.ca/~boley/cs6795swt/fall2012projects.html). Many thanks go to the colleagues at the venues of earlier Grailog presentations for Q & A discussions (http://www.cs.unb.ca/~boley/talks/RuleMLGrailog.pdf). Thanks also to the RuleML Steering Committee for encouraging Grailog-RuleML co-development, and to Gen Zou, Borislav Iordanov, and the RuleML 2013 reviewers for suggestions on this paper. I am grateful to Duncan Stewart for the inspiring NRC SDTech environment. Finally, NSERC is thanked for support through Discovery Grants.

References

[Ber73] Berge, C.: Graphs and Hypergraphs. North Holland (1973)

[Bol92] Boley, H.: Declarative Operations on Nets. In: Lehmann, F. (ed.) Semantic Networks in Artificial Intelligence. Computers & Mathematics with Applications, vol. 23, pp. 601–637. Pergamon Press (1992)

[Bol01] Boley, H.: Relationships Between Logic Programming and RDF. In: Kowalczyk, R., Loke, S.W., Reed, N.E., Graham, G. (eds.) PRICAI 2000 Workshop Reader. LNCS (LNAI), vol. 2112, pp. 201–218. Springer, Heidelberg (2001)

[Bol11] Boley, H.: A RIF-Style Semantics for RuleML-Integrated Positional-Slotted, Object-Applicative Rules. In: Bassiliades, N., Governatori, G., Paschke, A. (eds.) RuleML 2011 - Europe. LNCS, vol. 6826, pp. 194–211. Springer, Heidelberg (2011)

[BPS10] Boley, H., Paschke, A., Shafiq, O.: RuleML 1.0: The Overarching Specification of Web Rules. In: Dean, M., Hall, J., Rotolo, A., Tabet, S. (eds.) RuleML 2010. LNCS, vol. 6403, pp. 162–178. Springer, Heidelberg (2010)

[CM09] Chein, M., Mugnier, M.-L.: Graph-Based Knowledge Representation. Springer (2009)

[HSTC11] Howse, J., Stapleton, G., Taylor, K., Chapman, P.: Visualizing Ontologies: A Case Study. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part I. LNCS, vol. 7031, pp. 257–272. Springer, Heidelberg (2011)

[Ior10] Iordanov, B.: HyperGraphDB: A Generalized Graph Database. In: Shen, H.T., Pei, J., Özsu, M.T., Zou, L., Lu, J., Ling, T.-W., Yu, G., Zhuang, Y., Shao, J. (eds.) WAIM 2010. LNCS, vol. 6185, pp. 25–36. Springer, Heidelberg (2010)

[Woo07] Woods, W.A.: Meaning and Links. AI Magazine 28(4), 71–92 (2007)

Modeling Stable Matching Problems with Answer Set Programming*

Sofie De Clercq[1], Steven Schockaert[2], Martine De Cock[1], and Ann Nowé[3]

[1] Dept. of Applied Math., CS & Stats, Ghent University, Ghent, Belgium
{SofieR.DeClercq,Martine.DeCock}@ugent.be
[2] School of Computer Science & Informatics, Cardiff University, Cardiff, UK
S.Schockaert@cs.cardiff.ac.uk
[3] Computational Modeling Lab, Vrije Universiteit Brussel, Brussels, Belgium
ANowe@vub.ac.be

Abstract. The Stable Marriage Problem (SMP) is a well-known matching problem first introduced and solved by Gale and Shapley [7]. Several variants and extensions to this problem have since been investigated to cover a wider set of applications. Each time a new variant is considered, however, a new algorithm needs to be developed and implemented. As an alternative, in this paper we propose an encoding of the SMP using Answer Set Programming (ASP). Our encoding can easily be extended and adapted to the needs of specific applications. As an illustration we show how stable matchings can be found when individuals may designate unacceptable partners and ties between preferences are allowed. Subsequently, we show how our ASP based encoding naturally allows us to select specific stable matchings which are optimal according to a given criterion. Each time, we can rely on generic and efficient off-the-shelf answer set solvers to find (optimal) stable matchings.

1 Introduction

The Stable Marriage Problem (SMP) is a matching problem first introduced and solved by Gale and Shapley [7]. Starting from (i) a set of n men and n women, (ii) for each man a ranking of the women as preferred partners, and (iii) for each woman a ranking of the men as preferred partners, the SMP searches for a set of n couples (marriages) such that there are no man and woman who are in different marriages but both prefer each other to their actual partners. Such a man and woman are called a *blocking pair* and a matching without blocking pairs forms a *stable set* of marriages. Due to its practical relevance, countless variants on the SMP have been investigated, making the problem assumptions more applicable to a wider range of applications, such as kidney-exchange [12] and the hospital-resident problem [18]. Recently Roth and Shapley won the Nobel Prize for Economics for their theory of stable allocations and the practice of market design, work that has directly resulted from an application of the SMP.

* This research was funded by a Research Foundation-Flanders project.

L. Morgenstern et al. (Eds.): RuleML 2013, LNCS 8035, pp. 68–83, 2013.

In the literature, typically each time a new variant on the SMP is considered, a new algorithm is developed (see e.g. [10,13,20]). In this paper, we propose to use Answer Set Programming (ASP) as a general vehicle for modeling a large class of extensions and variations of the SMP. We show how an ASP encoding allows us to express in a natural way ties in the preferences of men and women, as well as unacceptability constraints (where certain people prefer to remain single over being coupled to undesirable partners). Furthermore, we illustrate how we can use our ASP encoding to find stable matchings that are optimal according to a certain criterion. Although the SMP has been widely investigated, and efficient approximation or exact algorithms are available for several of its variants (see e.g. [20]), to the best of our knowledge, our encoding offers the first exact implementation to find sex-equal, minimum regret, egalitarian or maximum cardinality stable sets for SMP instances with unacceptability and ties.

The paper is structured as follows. In Section 2 we give some background about the SMP and ASP. We introduce our encoding of the SMP with ASP and prove its correctness in the third section. In Section 4, we extend our encoding enabling it to find optimal stable sets. We explore several notions of optimality for stable matchings and show how optimal stable matchings can be found by solving the corresponding disjunctive ASP program. Finally we draw our conclusions. This paper contains only sketches of the proofs; a version of the same name, with additional examples and proofs, has been submitted on arXiv.org[1].

2 Background

2.1 The Stable Marriage Problem

To solve the standard SMP, Gale and Shapley [7] constructed an iterative algorithm —known as the Gale-Shapley algorithm, G-S algorithm or deferred-acceptance algorithm— to compute a particular solution of an SMP instance. The algorithm works as follows: in round 1 every man proposes to his first choice of all women. A woman, when being proposed, then rejects all men but her first choice among the subset of men who proposed to her. That first choice becomes her temporary husband. In the next rounds, all rejected men propose to their first choice of the subset of women by whom they were not rejected yet, regardless of whether this woman already has a temporary husband. Each woman, when being proposed, then rejects all men but her first choice among the subset of men who just proposed to her and her temporary mate. This process continues until all women have a husband. This point, when everyone has a partner, is always reached after a polynomial number of steps and the corresponding set of marriages is stable [7]. It should be noted, however, that only one of the potentially exponentially many stable matchings is found in this way. We formally define the SMP and introduce two variants that will be considered in this paper. We denote a set of men as $M = \{m_1, \ldots, m_n\}$ and a set of women as $W = \{w_1, \ldots, w_p\}$, with $n = p$ for the classical SMP. A set of marriages is a set of man-woman pairs such that all men and women occur in just one pair.

[1] `arxiv.org/pdf/1302.7251.pdf`

Definition 1 (Classical SMP). *An instance of the classical SMP is a pair* (S_M, S_W)*, with* $S_M = \{\sigma_M^1, \ldots, \sigma_M^n\}$ *and* $S_W = \{\sigma_W^1, \ldots, \sigma_W^n\}$ *sets of permutations of the integers* $1, \ldots, n$*. The permutations* σ_M^i *and* σ_W^i *are the preferences of man* m_i *and woman* w_i *respectively. If* $\sigma_M^i(j) = k$*, we say that woman* w_k *is the* j^{th} *most preferred woman for man* m_i*, and similarly for* $\sigma_W^i(j) = k$*. Man* m *and woman* w *form a blocking pair in a set of marriages* S *if* m *prefers* w *to his partner in* S *and* w *prefers* m *to her partner in* S*. A solution of an instance is a stable set of marriages, i.e. a set of marriages without blocking pairs.*

A first variant of the classical SMP allows men and women to point out unacceptable partners by not including them in their preference list. The number of men n can differ from the number of women p since men and women can remain single. A set of marriages is a set of singles (i.e. persons paired to themselves) and man-woman pairs such that all men and women occur in just one pair.

Definition 2 (SMP with unacceptability). *An instance of the SMP with unacceptability is a pair* (S_M, S_W)*,* $S_M = \{\sigma_M^1, \ldots, \sigma_M^n\}$*, and* $S_W = \{\sigma_W^1, \ldots, \sigma_W^p\}$*, with each* σ_M^i *a permutation of a subset of* $\{1, \ldots, p\}$ *and each* σ_W^j *a permutation of a subset of* $\{1, \ldots, n\}$*. If* $\sigma_M^i(j) = k$*, woman* w_k *is the* j^{th} *most preferred woman for man* m_i*, and similarly for* $\sigma_W^i(j) = k$*. If there is no* l *such that* $\sigma_M^i(l) = j$*, woman* w_j *is an unacceptable partner for man* m_i*, and similarly for no* l *such that* $\sigma_W^i(l) = j$*. A person* x *forms a blocking individual in a set of marriages* S *if* x *prefers being single to being paired with his or her partner in* S*. A solution of an instance is a stable set of marriages, i.e. a set of marriages without blocking pairs or individuals.*

The length of the permutation σ_M^i is denoted as $|\sigma_M^i|$. A stable matching for an SMP instance with unacceptability always exists and can be found in polynomial time [22] by a slightly modified G-S algorithm.

Example 1. Suppose $M = \{m_1, m_2, m_3\}$, $W = \{w_1, w_2, w_3, w_4\}$, $S_M = \{\sigma_M^1 = (4, 1, 3), \sigma_M^2 = (3, 2), \sigma_M^3 = (1, 3)\}$ and $S_W = \{\sigma_W^1 = (1, 3), \sigma_W^2 = (2), \sigma_W^3 = (3, 2), \sigma_W^4 = (2, 1)\}$. Hence woman w_1 prefers man m_1 to man m_3 while man m_2 is unacceptable. In this setting, there is exactly one stable set of marriages [22]: $\{(m_1, w_4), (m_2, w_3), (m_3, w_1), (w_2, w_2)\}$. Thus woman w_2 stays single.

The second variant of the SMP allows unacceptability and ties, i.e. the preferences do not have to be strict. For this variant there are several ways to define stability, but we will use the notion of weak stability [11].

Definition 3 (SMP with unacceptability and ties). *An instance of the SMP with unacceptability and ties is a pair* (S_M, S_W)*,* $S_M = \{\sigma_M^1, \ldots, \sigma_M^n\}$ *and* $S_W = \{\sigma_W^1, \ldots, \sigma_W^p\}$*. For every* $i \in \{1, \ldots, n\}$*,* σ_M^i *is a list of disjoint subsets of* $\{1, \ldots, p\}$*. Symmetrically* σ_W^i *is a list of disjoint subsets of* $\{1, \ldots, n\}$ *for every* $i \in \{1, \ldots, p\}$*. We call* σ_M^i *and* σ_W^i *the preferences of man* m_i *and woman* w_i *respectively. If* $k \in \sigma_M^i(j)$*, woman* w_k *is in man* m_i*'s* j^{th} *most preferred group of women. All the women in that group are equally preferred by* m_i*. The case* $k \in \sigma_W^i(j)$ *is similar. If there is no* l *such that* $j \in \sigma_M^i(l)$*, woman* w_j *is an*

unacceptable partner for man m_i, and similar for no l such that $j \in \sigma^i_W(l)$. For every k in the set[2] *$\sigma^i_M(|\sigma^i_M|)$, man m_i equally prefers staying single to being paired to woman w_k, and symmetrically for the preferences of a woman w_i. This is the only set in σ^i_M that might be empty, and similar for σ^i_W. Man m and woman w form a blocking pair in a set of marriages S if m strictly prefers w to his partner in S and w strictly prefers m to her partner in S. A blocking individual in S is a person who stricly prefers being single to being paired to his partner in S. A solution of an instance is a weakly stable set of marriages, i.e. a set of marriages without blocking pairs or individuals.*

A weakly stable matching always exists for an instance of the SMP with unacceptability and ties and it can be found in polynomial time by arbitrarily breaking the ties [14]. However, as opposed to the previous variant, the number of matched persons is no longer constant for every stable set in this variant. Note that the setting of Definition 3 generalizes the setting of Definition 2, which generalizes the setting of Definition 1. We introduce the notations

$$acceptable^i_M = \underbrace{\sigma^i_M(1) \cup \sigma^i_M(2) \cup \ldots \cup \sigma^i_M(|\sigma^i_M| - 1)}_{= preferred^i_M} \cup \underbrace{\sigma^i_M(|\sigma^i_M|)}_{= neutral^i_M}$$

Furthermore $unacceptable^i_M = \{1, \ldots, p\} \setminus acceptable^i_M$. We define the ordering $\leq^{m_i}_M$ on $\{w_j \mid j \in acceptable^i_M\} \cup \{m_i\}$ as $x \leq^{m_i}_M y$ iff m_i prefers person x at least as much as person y. The strict ordering $<^{m_i}_M$ is defined in the obvious way and analogous notations are used for σ^j_W.

Example 2. Suppose $M = \{m_1, m_2\}$, $W = \{w_1, w_2, w_3, w_4\}$ and $S_M = \{\sigma^1_M = (\{1,3\},\{4\}), \sigma^2_M = (\{2,3\},\{\})\}$. Hence man m_1 prefers women w_1 and w_3 to woman w_4. There is a tie between woman w_1 and w_3 as well as between woman w_4 and staying single. Woman w_2 is unacceptable for man m_1. Man m_2 prefers woman w_2 and w_3 to staying single, but finds w_1 and w_4 unacceptable. It holds that $w_1 <^{m_1}_M m_1$, i.e. m_1 prefers marrying w_1 over staying single, $acceptable^1_M = \{1,3,4\}$, $preferred^1_M = \{1,3\}$, $neutral^1_M = \{4\}$ and $unacceptable^1_M = \{2\}$.

2.2 Answer Set Programming

Answer Set Programming or ASP is a form of declarative programming [2]. Its transparence, elegance and ability to deal with Σ^P_2-complete problems make it an attractive method for solving combinatorial search and optimization problems. An ASP program is a finite collection of first-order rules

$$A_1 \vee \ldots \vee A_k \leftarrow B_1, \ldots, B_m, not\, C_1, \ldots, not\, C_n$$

with $A_1, \ldots, A_k, B_1, \ldots, B_m, C_1, \ldots, C_n$ predicates. The semantics are defined by the *ground version* of the program, consisting of all ground instantiations of the rules w.r.t. the constants that appear in it (see e.g. [2] for a good overview). This grounded program is a propositional ASP program. The building blocks of these programs are *atoms*, *literals* and *rules*. The most elementary are *atoms*,

[2] $|\sigma^i_M|$ denotes the length of the list σ^i_M.

which are propositional variables that can be true or false. A *literal* is an atom or a negated atom. Beside strong negation, ASP uses a special kind of negation, namely *negation-as-failure* (naf), denoted with '*not*'. For a literal a we call '*not* a' the naf-literal associated with a. The *extended literals* consist of all literals and their associated naf-literals. A *disjunctive rule* has the following form

$$a_1 \vee \ldots \vee a_k \leftarrow b_1, \ldots, b_m, not\, c_1, \ldots, not\, c_n$$

where $a_1, \ldots, a_k, b_1, \ldots, b_m, c_1, \ldots, c_n$ are literals from a fixed set $\mathcal{L}$, determined by a fixed set $\mathcal{A}$ of atoms. We call $a_1 \vee \ldots \vee a_k$ the head of the rule while the set of extended literals $b_1, \ldots, b_m, not\, c_1, \ldots, not\, c_n$ is called the *body*. The rule above intuitively encodes that a_1, a_2, ... or a_k is true when we have evidence that $b_1, \ldots, b_m$ are true and we have no evidence that at least one of $c_1, \ldots, c_n$ are true. When a rule has an empty body, we call it a *fact*; when the head is empty, we speak of a *constraint*. A rule without occurrences of *not* is called a *simple disjunctive rule*. A *simple disjunctive* ASP program is a finite collection of simple disjunctive rules and similarly a *disjunctive* ASP program $\mathcal{P}$ is a finite collection of disjunctive rules. If each rule head consists of at most one literal, we speak of a *normal* ASP program.

We define an *interpretation* I of a disjunctive ASP program $\mathcal{P}$ as a subset of $\mathcal{L}$. An interpretation I *satisfies* a simple disjunctive rule $a_1 \vee \ldots \vee a_k \leftarrow b_1, \ldots, b_m$ when $a_1 \in I \vee \ldots \vee a_k \in I$ or $\{b_1, \ldots, b_m\} \not\subseteq I$. An interpretation which satisfies all rules of a simple disjunctive program is called a *model* of that program. An interpretation I is an *answer set* of a simple disjunctive program $\mathcal{P}$ iff it is a minimal model of $\mathcal{P}$, i.e. no strict subset of I is a model of $\mathcal{P}$ [9]. The *reduct* $\mathcal{P}^I$ of a disjunctive ASP program $\mathcal{P}$ w.r.t. an interpretation I is defined as the simple disjunctive ASP program $\mathcal{P}^I = \{a_1 \vee \ldots \vee a_k \leftarrow b_1, \ldots, b_m \mid (a_1 \vee \ldots \vee a_k \leftarrow b_1, \ldots, b_m, not\, c_1, \ldots, not\, c_n) \in \mathcal{P}, \{c_1, \ldots, c_n\} \cap I = \emptyset\}$. An interpretation I of a disjunctive ASP program $\mathcal{P}$ is an answer set of $\mathcal{P}$ iff I is an answer set of $\mathcal{P}^I$.

Example 3. Let $\mathcal{P}$ be the ASP program with the following 4 rules:

$$man(john) \leftarrow, \quad person(john) \leftarrow, \quad person(fiona) \leftarrow$$
$$woman(X) \vee child(X) \leftarrow person(X), not\, man(X)$$

The last rule is grounded to 2 rules in which X is resp. replaced by $john$ and by $fiona$. We check that the interpretation $I = \{man(john), woman(fiona), person(john), person(fiona)\}$ is an answer set of the ground version of $\mathcal{P}$ by computing the reduct. The grounded rule with $X = john$ is deleted since $man(john)$ is in I. The reduct $\mathcal{P}^I$ is:

$$man(john) \leftarrow, \quad person(john) \leftarrow, \quad person(fiona) \leftarrow$$
$$woman(fiona) \vee child(fiona) \leftarrow person(fiona)$$

The first 3 rules are facts, hence their heads will be in any answer set. The fourth rule encodes that any person who is not a man, is a woman or child. It is clear that I is a minimal model of this simple program, so I is an answer set of $\mathcal{P}$. By replacing $woman(fiona)$ by $child(fiona)$ in I, another answer set is obtained.

To automatically compute the answer sets of the programs in this paper, we have used the ASP solver DLV[3], due to its ability to handle predicates, disjunction and numeric values (with some built-in aggregate functions). The numeric values are only used for grounding.

3 Modeling the Stable Marriage Problem in ASP

In this section we model variations and generalizations of the SMP with ASP. A few proposals of using nonmonotonic reasoning for modeling the SMP have already been described in the literature. For instance in [19] a specific variant of the SMP is mentioned (in which boys each know a subset of a set of girls and want to be matched to a girl they know) and in [4] an abductive program is used to find a stable set of marriages in which two fixed persons are paired, with strict, complete preference lists. To the best of our knowledge, beyond a few specific examples, no comprehensive study has been made of using ASP or related paradigms in this context. In particular, the generality of our ASP framework for weakly stable sets of SMP instances with unacceptablity and/or ties is a significant advantage. The expression $accept(m, w)$ denotes that a man m and a woman w accept each other as partners. The predicate $manpropose(m, w)$ expresses that man m is willing to propose to woman w and analogously $womanpropose(m, w)$ expresses that woman w is willing to propose to man m. Inspired by the Gale-Shapley algorithm, we look for an ASP formalisation to find the stable sets.

Definition 4 (ASP program induced by SMP with unacc. and ties). *The ASP program $\mathcal{P}$ induced by an instance $(\{\sigma_M^1, \ldots, \sigma_M^n\}, \{\sigma_W^1, \ldots, \sigma_W^p\})$ of the classical SMP with unacceptability and ties is the program containing for every $i \in \{1, \ldots, n\}, j \in \{1, \ldots, p\}$ the following rules:*

$$accept(m_i, w_j) \leftarrow manpropose(m_i, w_j), womanpropose(m_i, w_j) \quad (1)$$

$$accept(m_i, m_i) \leftarrow \{not\, accept(m_i, w_k) \mid k \in acceptable_M^i\} \quad (2)$$

$$accept(w_j, w_j) \leftarrow \{not\, accept(m_k, w_j) \mid k \in acceptable_W^j\} \quad (3)$$

and for every $i \in \{1, \ldots, n\}$, $j \in acceptable_M^i$:

$$manpropose(m_i, w_j) \leftarrow \{not\, accept(m_i, x) \mid x \leq_M^{m_i} w_j \text{ and } w_j \neq x\} \quad (4)$$

and for every $j \in \{1, \ldots, p\}$, $i \in acceptable_W^j$:

$$womanpropose(m_i, w_j) \leftarrow \{not\, accept(x, w_j) \mid x \leq_W^{w_j} m_i \text{ and } m_i \neq x\} \quad (5)$$

Intuitively (1) means that a man and woman accept each other as partners if they propose to each other. Due to (2), a man accepts himself as a partner (i.e. stays single) if no woman in his preference list is prepared to propose to him. Rule (4) states that a man proposes to a woman if he is not paired to a more or equally preferred woman. For $j \in neutral_M^i$ the body of (4) contains $not\, accept(m_i, m_i)$.

[3] Available from `www.dlvsystems.com`

Proposition 1. *Let (S_M, S_W) be an instance of the SMP with unacceptability and ties and let $\mathcal{P}$ be the corresponding ASP program. If I is an answer set of $\mathcal{P}$, then a weakly stable matching for (S_M, S_W) is given by $\{(x, y) \mid accept(x, y) \in I\}$.*

Sketch of the Proof. Let (S_M, S_W) and $\mathcal{P}$ be as described in the proposition and let I be an arbitrary answer set of $\mathcal{P}$. We prove this proposition in 4 steps.

1. *For every $i \in \{1, \ldots, n\}$, $j \in \{1, \ldots, p\}$ it holds that $accept(m_i, w_j) \in I$ implies that $j \in acceptable_M^i$ and $i \in acceptable_W^j$.*
   This can be proved by contradiction.
2. *For every man m_i there exists at most one woman w_j s.t. $accept(m_i, w_j) \in I$ and for every woman w_j there is at most one man m_i s.t. $accept(m_i, w_j) \in I$. Moreover, if $accept(m_i, m_i) \in I$ then $accept(m_i, w_j) \notin I$ for any w_j, and likewise when $accept(w_j, w_j) \in I$ then $accept(m_i, w_j) \notin I$ for any m_i.*
   This can be proved by contradiction.
3. *For every man m_i exactly one of the following conditions is satisfied*:
   (a) *there exists a woman w_j such that $accept(m_i, w_j) \in I$,*
   (b) *$accept(m_i, m_i) \in I$,*
   and similarly for every woman w_i.
   Unsatisfaction of the first condition implies satisfaction of the second.
4. The previous steps imply that I produces a set of marriages without blocking individuals. Weak stability also demands the absence of blocking pairs. Suppose by contradiction that there is a blocking pair (m_i, w_j), implying that there exist $i \neq i'$ and $j \neq j'$ such that $accept(m_i, w_{j'}) \in I$ and $accept(m_{i'}, w_j) \in I$ while $w_j <_M^{m_i} w_{j'}$ and $m_i <_W^{w_j} m_{i'}$. The rules of the form (1), the only ones with the literals $accept(m_i, w_{j'})$ and $accept(m_{i'}, w_j)$ in the head, imply that literals $manpropose(m_i, w_{j'})$ and $womanpropose(m_{i'}, w_j)$ should be in I. But since $w_j <_M^{m_i} w_{j'}$ and because of the form of the rules (4) there are fewer conditions to be fulfilled for $manpropose(m_i, w_j)$ to be in I than for $manpropose(m_i, w_{j'})$ to be in I. So $manpropose(m_i, w_j)$ should be in I as well. A similar reasoning implies that $womanpropose(m_i, w_j)$ should be in I. But now the rules of the form (1) imply that $accept(m_i, w_j)$ should be in I, contradicting step 2 since $accept(m_i, w_{j'})$ and $accept(m_{i'}, w_j)$ are already in I. □

Proposition 2. *Let (S_M, S_W) be an instance of the SMP with unacceptability and ties, and let $\mathcal{P}$ be the corresponding ASP program. If $\{(x_1, y_1), \ldots, (x_k, y_k)\}$ is a weakly stable matching for (S_M, S_W) then $\mathcal{P}$ has the following answer set I:*

$$\begin{aligned}
&\{manpropose(x_i, y) \mid i \in \{1, \ldots, k\}, x_i \in M, y <_M^{x_i} y_i)\} \\
\cup &\{womanpropose(x, y_i) \mid i \in \{1, \ldots, k\}, y_i \in W, x <_W^{y_i} x_i\} \\
\cup &\{accept(x_i, y_i) \mid i \in \{1, \ldots, k\}\} \\
\cup &\{manpropose(x_i, y_i) \mid i \in \{1, \ldots, k\}, x_i \neq y_i\} \\
\cup &\{womanpropose(x_i, y_i) \mid i \in \{1, \ldots, k\}, x_i \neq y_i\}
\end{aligned}$$

Sketch of the Proof. One can verify that I is indeed an answer set of $\mathcal{P}$ by verifying that I is a minimal model of the reduct $\mathcal{P}^I$. □

In [18] it is shown that the decision problem 'is the pair (m, w) stable?' for a given SMP instance with unacceptablity and ties is an NP-complete problem, even in the absence of unacceptability. A pair (m, w) is *stable* if there exists a stable set that contains (m, w). It is straightforward to see that we can reformulate this decision problem as 'does there exist an answer set of the induced normal ASP program $\mathcal{P}$ which contains the literal $accept(m, w)$?' (i.e. brave reasoning), which is known to be an NP-complete problem [1]. So our model forms a suitable framework for these kind of decision problems concerning the SMP.

4 Selecting Preferred Stable Sets

4.1 Notions of Optimality of Stable Sets

When several stable matchings can be found for an instance of the SMP, some may be more interesting than others. The stable set found by the G-S algorithm is *M-optimal* [22], i.e. every man likes this set at least as well as any other stable set. Exchanging the roles of men and women in the G-S algorithm yields a *W-optimal* stable set [7], optimal from the point of view of the women.

While some applications may require us to favour either the men or the women, in others it makes more sense to treat both parties equally. To formalize some commonly considered notions of fairness and optimality w.r.t. the SMP, we define the cost $c_x(S)$ of a stable set S to an individual x, where $c_x(S) = k$ if x has been matched with his or her k^{th} preferred partner. More precisely, for $x = m_i$ a man, we define $c_{m_i}(S) = |\{z : z <_M^{m_i} y\}|+1$ where y is the partner of x in S; for $x = w_j$ a woman, c_x is defined analogously. So in case of ties we assign the same list position to equally preferred partners, as illustrated in Example 4.

Example 4. Let $x = m_1$ be a man with preference list $\sigma_M^1 = (\{1\}, \{2, 3\}, \{4\})$ then w_1 as partner of x in some set of marriages S would yield $c_x(S) = 1$, w_2 and w_3 yield $c_x(S) = 2$ and w_4 yields $c_x(S) = 4$. If m_1 would be single in S, then the cost $c_x(S)$ is 4, since m_1 prefers women w_1, w_2 and w_3 to being single, but is indifferent between being paired to w_4 or staying single.

Definition 5. *For S a set of marriages,*

- *the sex-equalness cost is defined as* $c_{sexeq}(S) = |\sum_{x\in M} c_x(S) - \sum_{x\in W} c_x(S)|$,
- *the egalitarian cost is defined as* $c_{weight}(S) = \sum_{x\in M\cup W} c_x(S)$,
- *the regret cost is defined as* $c_{regret}(S) = \max_{x\in M\cup W} c_x(S)$, *and*
- *the cardinality cost is defined as* $c_{singles}(S) = |\{z : (z, z) \in S\}|$.

S is a sex-equal stable set iff S is a stable set with minimal sex-equalness cost. Similarly, S is an egalitarian (resp. minmum regret, maximum cardinality) stable set iff S is a stable set with minimal egalitarian (resp. regret or cardinality) cost.

A *sex-equal stable set* assigns an equal importance to the preferences of the men and women. An *egalitarian stable set* is a stable set in which the preferences of every individual are considered to be equally important. In [23] the use of an egalitarian stable set is proposed to optimally match virtual machines (VM) to

servers in order to improve cloud computing by equalizing the importance of migration overhead in the data center network and VM migration performance. A *minimum regret stable set* is optimal for the person who is worst off. A *maximal or minimal cardinality stable set* is a stable set with resp. as few or as many singles as possible. Examples of practical applications include an efficient kidney exchange program [21] and the National Resident Matching Program[4] [18]. Maximizing cardinality garantuees that as many donors as possible will get a compatible donor and as many medical graduates as possible will get a position.

Table 1 presents an overview of known complexity results[5] concerning finding an optimal stable set. Typically the presence of ties leads to an increase of complexity. Manlove et al. [17,18] proved that the problem of finding a maximum (or minimum) cardinality stable set for a given instance of the SMP with unacceptability and ties is NP-hard. Using this result, the problem of finding an egalitarian or minimum regret stable matching for a given SMP instance with ties is proved to be NP-hard [18], even if the ties occur on one side only and each tie is of length 2 (i.e. each set in a preference list has size at most 2). If there are no ties, the problem of finding an egalitarian or minimum regret stable set is solvable in polynomial time [13,10]. Since all stable sets consist of n couples in the classical SMP, the G-S algorithm trivially finds a maximum (or minimum) cardinality [7]. For an SMP instance with unacceptability the number of couples in a stable set is constant [8], so finding a maximum cardinality stable set reduces to finding a stable set, which is known to be solvable in polynomial time. Surprisingly, finding a sex-equal stable set for a classical SMP instance is NP-hard [16], even if the preference lists are bound in length by 3 [20].

Table 1. Literature complexity results for finding an optimal stable set

	sex-equal	egalitarian	min. regret	max. card.
SMP	NP-hard [16]	P ($O(n^4)$ [13])	P ($O(n^2)$ [10])	P ($O(n^2)$ [7])
SMP + unacc	NP-hard [20]			P [8]
SMP + ties		NP-hard [18]	NP-hard [18]	
SMP + {unacc,ties}				NP-hard [17,18]

Between brackets we mention in Table 1 the complexity of an algorithm that finds an optimal stable set if one exists, in function of the number of men n. To the best of our knowledge, the only exact algorithm tackling an NP-hard problem from Table 1 finds a sex-equal stable set for an SMP instance in which the strict preference lists of men and/or women are bounded in length by a constant [20]. To the best of our knowledge, no exact implementations exist to find an optimal stable set for an SMP instance with ties, regardless of the presence of unacceptability and regardless which notion of optimality from Table 1 is used. Our approach yields an exact implementation of all problems mentioned in Table 1.

[4] `www.nrmp.org`

[5] Throughout this paper we assume that P $\neq$ NP.

4.2 Finding Optimal Stable Sets Using Disjunctive ASP

As we discuss next, we can extend our ASP encoding of the SMP such that the optimal stable sets correspond to the answer sets of an associated ASP program. In particular, we use a saturation technique [5,1] to filter non-optimal answer sets. Intuitively, the idea is to create a program with 3 components: (i) a first part describing the solution candidates, (ii) a second part also describing the solution candidates since comparison of solutions requires multiple solution candidates within the same answer set whereas the first part in itself produces one solution per answer set, (iii) a third part comparing the solutions described in the first two parts and selecting the preferred solutions by saturation. It is known that the presence of negation-as-failure can cause problems when applying saturation. Therefore, we use a SAT encoding [15] of the ASP program in Definition 4 and define a disjunctive naf-free ASP program in Definition 6 which selects particular models of the SAT problem. This transformation can be found in our paper of the same name on arXiv and the correctness of Lemma 1 follows from [6,15].

Definition 6 (Induced disj. naf-free ASP program). *The disjunctive naf-free ASP program $\mathcal{P}_{disj}$ induced by an SMP instance (S_M, S_W) with unacceptability and ties contains the following rules for $i \in \{1,\dots,n\}, j \in \{1,\dots,p\}$:*

$$\neg accept(m_i, w_j) \vee manpropose(m_i, w_j) \leftarrow$$
$$\neg accept(m_i, w_j) \vee womanpropose(m_i, w_j) \leftarrow$$
$$accept(m_i, w_j) \vee \neg manpropose(m_i, w_j) \vee \neg womanpropose(m_i, w_j) \leftarrow$$

For every $i \in \{1,\dots,n\}$, $l \in unacceptable^i_M$, $j \in acceptable^i_M$, $x \leq^{m_i}_M w_j, x \neq w_j$ $\mathcal{P}_{disj}$ contains:

$$\bigvee_{k \in acceptable^i_M} accept(m_i, w_k) \vee accept(m_i, m_i) \leftarrow$$
$$\neg accept(m_i, m_i) \vee \neg accept(m_i, w_j) \leftarrow$$
$$\neg manpropose(m_i, w_j) \vee \neg accept(m_i, x) \leftarrow$$
$$\bigvee_{x \leq^{m_i}_M w_j, x \neq w_j} accept(m_i, x) \vee manpropose(m_i, w_j) \leftarrow$$
$$\neg manpropose(m_i, w_l) \leftarrow$$

and symmetrical for $j \in \{1,\dots,p\}$ and womanpropose.

Lemma 1. *Let $\mathcal{P}$ be the normal ASP program from Definition 4 and $\mathcal{P}_{disj}$ the disjunctive ASP program from Definition 6. It holds that for any answer set I of $\mathcal{P}$ there exists an answer set I_{disj} of $\mathcal{P}_{disj}$ such that the atoms of I and I_{disj} coincide. Conversely for any answer set I_{disj} of $\mathcal{P}_{disj}$ there exists an answer set I of $\mathcal{P}$ such that the atoms of I and I_{disj} coincide.*

4.3 ASP Program to Select Optimal Solutions

Let (S_M, S_W) be an SMP instance with unacceptability and ties, with $S_M = \{\sigma^1_M, \dots, \sigma^n_M\}$ and $S_W = \{\sigma^1_W, \dots, \sigma^p_W\}$, and let $\mathcal{P}_{norm}$ be the induced normal

ASP program from Definition 4. Our technique for extending this program to a program that can respectively optimize for the sex-equalness, egalitarian, minimum regret and maximum cardinality criterion is in each case very similar. We start by explaining it for the case of sex-equalness. Our first step is to add a set of rules that compute the sex-equalness cost of a set of marriages. For every man m_i and every woman w_j such that $j \in \sigma_M^i(k)$ we use the following rule to determine the cost for m_i if w_j would be his partner:

$$mancost(i,k) \leftarrow accept(m_i, w_j) \tag{6}$$

and similarly for every w_j and every m_i such that $i \in \sigma_W^j(k)$:

$$womancost(j,k) \leftarrow accept(m_i, w_j) \tag{7}$$

We also use the following rules with i ranging from 1 to n and j from 1 to p:

$$mancost(i, |\sigma_M^i|) \leftarrow accept(m_i, m_i) \tag{8}$$

$$womancost(j, |\sigma_W^j|) \leftarrow accept(w_j, w_j) \tag{9}$$

$$manweight(Z) \leftarrow \#sum\{B, A : mancost(A, B)\} = Z, \#int(Z) \tag{10}$$

$$womanweight(Z) \leftarrow \#sum\{B, A : womancost(A, B)\} = Z, \#int(Z) \tag{11}$$

$$sexeq(Z) \leftarrow manweight(X), womanweight(Y), Z = X - Y$$

$$sexeq(Z) \leftarrow manweight(X), womanweight(Y), Z = Y - X \tag{12}$$

Rules (8) and (9) state staying single leads to the highest cost. Rule (10) determines the sum of the male costs[6] and similarly (11) determines the sum of the female costs. According to Definition 5 the absolute difference of these values yields the sex-equalness cost, as determined by rules (12). Since numeric variables are restricted to positive integers in DLV, we omit conditions as '$X \geq Y$' or '$X < Y$'. The program $\mathcal{P}_{norm}$ extended with rules (6) – (12) is denoted $\mathcal{P}_{ext}^{sexeq}$. We construct a program $\mathcal{P}_{sexeq}$, composed by subprograms, that selects optimal solutions. Let $\mathcal{P}'_{disj}$ be the disjunctive naf-free ASP program, induced by the same SMP instance, in which a prime symbol is added to all literal names (e.g. *accept* becomes *accept'*). Define a new program $\mathcal{P}_{ext}^{\prime sexeq}$ with all the rules of $\mathcal{P}'_{disj}$ in which every occurrence of $\neg atom$ is changed into $natom$ for every atom $atom$, i.e. replace all negation symbols by a prefix 'n'. For every occurring atom $atom$ in $\mathcal{P}_{ext}^{\prime sexeq}$, add the following rule to exclude non-consistent solutions[7]:

$$sat \leftarrow atom, natom \tag{13}$$

Finally add rules (6) – (12) with prime symbols to the literal names to $\mathcal{P}_{ext}^{\prime sexeq}$ but replace rule (10) by:

$$mansum(n, X) \leftarrow mancost(n, X)$$

$$mansum(J, Z) \leftarrow mansum(I, X), mancost(J, Y), Z = X + Y, \#succ(J, I)$$

$$manweight(Z) \leftarrow mansum(1, Z) \tag{14}$$

[6] $\#sum$, $\#max$, $\#int$ and $\#count$ are DLV aggregate functions. The 'A' mentioned as variable in $\#sum$ indicates that a cost must be included for every person (otherwise the cost is included only once when persons have the same cost).

[7] For instance, $sat \leftarrow accept'(m_1, w_1), naccept'(m_1, w_1)$.

and analogously for (11) to avoid future cyclic dependencies of literals, involving aggregate functions. We define the ASP program $\mathcal{P}_{sexeq}$ as the union of $\mathcal{P}_{ext}^{sexeq}$, $\mathcal{P}_{ext}^{\prime sexeq}$ and $\mathcal{P}_{sat}$. The ASP program $\mathcal{P}_{sat}$ contains the following rules to select minimal solutions based on sex-equalness:

$$sat \leftarrow sexeq(X), sexeq'(Y), X \leq Y \tag{15}$$

$$\leftarrow not\ sat \tag{16}$$

$$\begin{aligned} mancost'(X,Y) &\leftarrow sat, manargcost'_1(X), manargcost'_2(Y) \\ womancost'(X,Y) &\leftarrow sat, womanargcost'_1(X), womanargcost'_2(Y) \\ manpropose'(X,Y) &\leftarrow sat, man(X), woman(Y) \\ womanpropose'(X,Y) &\leftarrow sat, man(X), woman(Y) \\ accept'(X,X) &\leftarrow sat, man(X) \\ accept'(X,X) &\leftarrow sat, woman(X) \end{aligned} \tag{17}$$

$$accept'(X,Y) \leftarrow sat, man(X), woman(Y) \tag{18}$$

and analogous to (18) a set of rules with prefix 'n' for the head predicates. Finally we add the facts[8] $manargcost'_1(1..n) \leftarrow$, $manargcost'_2(1..(p+1)) \leftarrow$, $womanargcost'_1(1..p) \leftarrow$, $womanargcost'_2(1..(n+1)) \leftarrow$, $man(x) \leftarrow$ for every man x and $woman(x) \leftarrow$ for every woman x to $\mathcal{P}_{sat}$. Intuitively the rules of $\mathcal{P}_{sat}$ express the key idea of saturation. First every answer set is forced to contain the atom sat by rule (16). Then the rules (17) – (18) and the facts make sure that any answer set should contain all possible literals with a prime symbol that occur in $\mathcal{P}_{sexeq}$. Rule (15) will establish that only optimal solutions will correspond to minimal models and thus lead to answer sets. For any non-optimal solution, the corresponding interpretation containing sat will never be a minimal model of the reduct. It is formally proved in Proposition 3 below that $\mathcal{P}_{sexeq}$ produces exactly the stable matchings with minimal sex-equalness cost.

Furthermore, only small adjustments to $\mathcal{P}_{sexeq}$ are needed to create programs $\mathcal{P}_{weight}$, $\mathcal{P}_{regret}$, and $\mathcal{P}_{singles}$ that resp. produce egalitarian, minimum regret and maximum cardinality stable sets. Indeed, the ASP program $\mathcal{P}_{weight}$ can easily be defined as $\mathcal{P}_{sexeq}$ in which the predicates $sexeq$ and $sexeq'$ are resp. replaced by $weight$ and $weight'$ and the rules (12) are replaced by (19), determining the egalitarian cost of Definition 5 as the sum of the male and female costs:

$$weight(Z) \leftarrow manweight(X), womanweight(Y), Z = X + Y \tag{19}$$

Similarly the ASP program $\mathcal{P}_{regret}$ is defined as $\mathcal{P}_{sexeq}$ in which the predicates $sexeq$ and $sexeq'$ are resp. replaced by $regret$ and $regret'$ and rules (10) – (12) are replaced by the following rules[9]:

$$manregret(Z) \leftarrow \#max\{B : mancost(A,B)\} = Z, \#int(Z) \tag{20}$$

$$womanregret(Z) \leftarrow \#max\{B : womancost(A,B)\} = Z, \#int(Z) \tag{21}$$

$$regret(X) \leftarrow manregret(X), womanregret(Y), X > Y$$

[8] The rule $manargcost'_1(1..n) \leftarrow$ is DLV-syntax for the n facts $manargcost'_1(1) \leftarrow$, $\ldots$, $manargcost'_1(n) \leftarrow$.

[9] As in (14) we adjust (20)–(21) and (23), for which we refer to the arXiv-version.

$$regret(Y) \leftarrow manregret(X), womanregret(Y), X \leq Y \tag{22}$$

Rule (20) determines the regret cost but only for the men. Similarly (21) determines the regret cost for the women. The regret cost as defined in Definition 5 is the maximum of these two values, determined by the rules in (22).

Finally we define the ASP program $\mathcal{P}_{singles}$ as $\mathcal{P}_{sexeq}$ in which the predicates $sexeq$ and $sexeq'$ are resp. replaced by $singles$ and $singles'$. Furthermore we replace rules (6) – (12) by (23), determining the number of singles:

$$singles(Z) \leftarrow \#count\{B : accept(B, B)\} = Z, \#int(Z) \tag{23}$$

Proposition 3. *Let the criterion crit be an element of* $\{sexeq, weight, regret, singles\}$. *For every answer set* I *of the program* $\mathcal{P}_{crit}$ *induced by an SMP instance with unacceptability and ties the set* $S_I = \{(m, w) \mid accept(m, w) \in I\}$ *forms an optimal stable set of marriages w.r.t. criterion crit and the optimal criterion value is given by the unique value* v_I *for which* $crit(v_I) \in I$. *Conversely for every optimal stable set* $S = \{(x_1, y_1), \ldots, (x_k, y_k)\}$ *with optimal criterion value* v *there exists an answer set* I *of* $\mathcal{P}_{crit}$ *such that* $\{(x, y) \mid accept(x, y) \in I\} = \{(x_i, y_i) \mid i \in \{1, \ldots, k\}\}$ *and* v *is the unique value for which* $crit(v) \in I$.

Sketch of the Proof. Let (S_M, S_W) be an SMP instance with unacc. and ties.

Answer set ⇒ Optimal stable set Let I be an arbitrary answer set of $\mathcal{P}_{crit}$ and let S_I be as in the proposition. It is clear that the only rules in $\mathcal{P}_{crit}$ that influence the literals of the form $manpropose(.,.)$, $womanpropose(.,.)$ and $accept(.,.)$ are the rules in $\mathcal{P}_{norm}$. Hence any answer set I of $\mathcal{P}_{crit}$ should contain an answer set I_{norm} of $\mathcal{P}_{norm}$ as a subset. Proposition 1 implies that I_{norm} corresponds to a stable set $S_I = \{(m, w) \mid accept(m, w) \in I_{norm}\}$. Moreover, the only literals of the form $manpropose(.,.)$, $womanpropose(.,.)$ and $accept(.,.)$ in I are those in I_{norm}, so $S_I = \{(m, w) \mid accept(m, w) \in I\}$. If $crit = sexeq$, it is straightforward to see that the literals of the form $accept(.,.)$ in I_{norm} uniquely determine which literals of the form $mancost(.,.)$, $womancost(.,.)$, $manweight(.)$, $womanweight(.)$ and $sexeq(.)$ should be in the answer set I. These literals do not occur in rules of $\mathcal{P}_{crit}$ besides those in $\mathcal{P}^{sexeq}_{ext}$. Notice that the rules which do contain these literals will imply that there will be just one literal of the form $sexeq(.)$ in I, namely $sexeq(v)$ with v the sex-equalness cost of S_I. Analogous results can be derived for $crit \in \{weight, regret, singles\}$. It remains to be shown that S_I is an optimal stable set. Suppose by contradiction that S_I is not optimal, so there exists a stable set S^* such that $v_I > v^*$, with v^* the criterion value of S^* to be minimized. We will prove that this implies that I cannot be an answer set of $\mathcal{P}_{crit}$, contradicting our initial assumption.

Proposition 2 and Lemma 1 imply that there exists an interpretation I^*_{disj} of the ASP program $\mathcal{P}_{disj}$ induced by (S_M, S_W) that corresponds to the stable set S^*. Moreover this interpretation is consistent, i.e. it will not contain $atom$ and $\neg atom$ for some atom $atom$. This implies that the interpretation I'_{disj} defined as I^*_{disj} in which $\neg atom$ is replaced by $natom$ for every atom $atom$ will falsify the body of the rules of the form (13) of $\mathcal{P}'^{crit}_{ext}$. An analogous reasoning as above yields that the literals of the form $accept'(.,.)$ in I'_{disj} uniquely determine which literals of the form $mancost'(.,.)$, $womancost'(.,.)$, $mansum'(.,.)$,

$womansum'(.,.)$, $manweight'(.)$, $womanweight'(.)$ and $sexeq'(.)$ should be in I'_{disj} for $crit = sexeq$ and analogously for the other criteria. With those extra literals added to I'_{disj}, I'_{disj} satisfies all the rules of $\mathcal{P}'^{crit}_{ext}$. Moreover, $crit(v^*)$ is the unique literal of the form $crit(.)$ in I'_{disj}. Notice that I'_{disj} does not contain the atom sat.

Define the interpretation $J = I_{norm} \cup I'_{disj}$. From the previous argument it follows that J will satisfy every rule of $\mathcal{P}^{crit}_{ext} \cup \mathcal{P}'^{crit}_{ext}$ since the predicates occurring in both programs do not overlap. Moreover J contains $crit(v_I)$ and $crit'(v^*)$ and these are the only literals of the form $crit(.)$ or $crit'(.)$. Since $v_I > v^*$ the rules of the form (15) will be satisfied by J since their body is always false. Call J' the set $J \cup \{a \,|\, (a \leftarrow) \in \mathcal{P}_{sat}\}$. Since J' does not contain sat, the rules of $\mathcal{P}_{sat}$ will all be satisfied by J', with exception of the rule $\leftarrow not\, sat$.

The rule of the form (16) implies that I as answer set of $\mathcal{P}_{crit}$ should contain sat. Now the set of rules (17) – (18) imply that I should also contain the literals $mancost'(.,.), womancost'(.,.)$ and $manpropose'(.,.)$, $womanpropose'(.,.)$, $accept'(.,.)$ with the corresponding literals prefixed by n for every possible argument stated by the facts in $\mathcal{P}_{sat}$. For $crit = sexeq$ rules (14) in $\mathcal{P}'^{crit}_{ext}$ garantuee that for all possible arguments $mansum'(.,.)$ and $manweight'(.)$ should be in I and analogously for $womansum'(.,.)$ and $womanweight'(.)$. Together with rules (12) this implies that I contains $crit(.)$ for every possible argument and analogously for the other criteria. So $I'_{disj} \subseteq I$. We already reasoned in the beginning of the proof that $I_{norm} \subseteq I$ holds so it follows that $J \subseteq I$. Since the literals of $J' \setminus J$ are stated as facts of $\mathcal{P}^{crit}_{ext}$, they should be in I, hence $J' \subseteq I$. Moreover $J' \subset I$ since $sat \in I \setminus J'$.

We use the notation $red(\mathcal{P}, I)$ to denote the reduct of an ASP program $\mathcal{P}$ w.r.t. an interpretation I. There is no rule in $\mathcal{P}'^{crit}_{ext}$ with negation-as-failure in the body, hence $red(\mathcal{P}'^{crit}_{ext}, I) = red(\mathcal{P}'^{crit}_{ext}, J') = \mathcal{P}'^{crit}_{ext}$. We already reasoned that J' satifies all the rules of the latter. We also reasoned that I does not contain any other literals of the form $accept(.,.)$ than those who are also in I_{norm}, and by construction the same holds for J'. Hence $red(\mathcal{P}^{crit}_{ext}, I) = red(\mathcal{P}^{crit}_{ext}, J')$ and by construction J' satisfies all the rules of this reduct. It is clear that $red(\mathcal{P}_{sat}, I)$ is $\mathcal{P}_{sat}$ without the rule $\leftarrow not\, sat$, since $sat \in I$. Again we already argued that J' satisfies $red(\mathcal{P}_{sat}, I)$. Hence J' satisfies all the rules of $red(\mathcal{P}_{crit}, I)$, implying that I, which strictly contains J', cannot be an answer set of $\mathcal{P}_{crit}$ since it is not a minimal model of the negation-free ASP program $red(\mathcal{P}_{crit}, I)$ [9].

Optimal stable set ⇒ Answer set This can be straightforwardly proved by writing down the interpretation which corresponds to S and checking that this is an answer set of $\mathcal{P}_{crit}$. Because of the considerable number of predicates, we refer the interested reader to the extended online paper of the same name on arXiv. □

If we delete from $\mathcal{P}_{sexeq}$ the rules (11) – (12) and replace rule (15) by the rule $sat \leftarrow manweight(X), manweight'(Y), X \leq Y$, then we obtain the M-optimal stable sets. Analogously we can obtain the W-optimal stable sets.

If a criterion is to be maximized, the symbol $\leq$ in rule (15) is simply replaced by $\geq$. E.g. for $crit = singles$ we will get minimum cardinality stable sets.

5 Complexity and Future Work

The NP-complete decision problem 'does there exist a stable set with cardinality $\geq k$ (resp. $\leq k$) for an SMP instance with unacceptability and ties with k a positive integer?' [17,18] has practical importance, e.g. in the National Resident Matching Program [18]. If we add a rule $sat \leftarrow singles(X), X \leq (n + p - 2k)$ to the extended induced program $\mathcal{P}_{ext}^{singles}$ defined in Subsect. 4.3, then this problem can be formulated as 'does there exist an answer set of the normal ASP program $\mathcal{P}_{ext}^{singles}$ which contains the literal *sat*?' (i.e. brave reasoning), another NP-complete problem [1]. So our model forms a suitable framework for these kind of decision problems concerning optimality of stable sets in the SMP.

Notice that the complexity of this kind of decision problem and the one mentioned in the last paragraph of Subsect. 3 are a good indication how hard it is to find an (optimal) stable set, as opposed to the problems 'does there exist an (optimal) stable set?', which tell us how hard it is to know whether there exists a solution but not necessarily how hard it is to find one.

Combining these problems leads to a new decision problem: 'is the pair (m, w) *optimally stable* for an instance of the SMP with unacceptability and ties?'. We define an *optimally stable pair* as a pair (m, w) for which there exists an optimal stable set in which m and w are matched. As far as we know this problem has not been studied yet, although it could be useful in practice, for instance if one wants to find a maximum cardinality matching but also wants to prioritize some couple or a person. Optimality is still desirable, because it ensures the others from not being put too much at a disadvantage. For instance in the kidney exchange problem, in which kidney patients with a willing but incompatible donor try to interchange each other's donors to get a transplant, this is a realistic situation: if two patients with intercompatible donors urgently need a transplant, they should get priority, but of course we still want to match as many patients to donors as possible. Considering the complexity of the separate decision problems, the combined problem might have a higher complexity, perhaps corresponding to the Σ_2^P-complexity of our grounded disjunctive normal ASP program with aggregate functions [1,3]. It should be noticed however that the addition of constraints not necessarily increases complexity and a precise classification of complexity is desirable.

6 Conclusion

We formalized and solved different variants of the SMP using ASP programs, which can easily be adapted to yet other variants. Moreover we applied saturation to compute optimal stable sets, with the advantage that these programs can be handled with the efficient off-the-shelf ASP solver DLV. To the best of our knowledge, our encoding offers the first exact implementation of finding sex-equal, egalitarian, minimum regret, or maximum cardinality stable sets for an instance of the SMP with unacceptability and ties. Hence, our general framework allows us to tackle a class of problems and requires only small adaptions to easily shift between them.

References

1. Baral, C.: Knowledge Representation, Reasoning, and Declarative Problem Solving. Cambridge University Press, New York (2003)
2. Brewka, G., Eiter, T., Truszczyński, M.: Answer set programming at a glance. Communications of the ACM 54(12), 92–103 (2011)
3. Dell'Armi, T., Faber, W., Ielpa, G., Leone, N., Pfeifer, G.: Aggregate functions in disjunctive logic programming: Semantics, complexity, and implementation in DLV. In: Gottlob, G., Walsh, T. (eds.) IJCAI, pp. 847–852. M. Kaufmann (2003)
4. Dung, P.: An argumentation-theoretic foundation for logic programming. The Journal of Logic Programming 22(2), 151–177 (1995)
5. Eiter, T., Gottlob, G., Mannila, H.: Disjunctive datalog. ACM Transactions on Database Systems 22(3), 364–418 (1997)
6. Erdem, E., Lifschitz, V.: Tight logic programs. Theory and Practice of Logic Programming 3, 499–518 (2003)
7. Gale, D., Shapley, L.: College admissions and the stability of marriage. The American Mathematical Monthly 69(1), 9–15 (1962)
8. Gale, D., Sotomayor, M.: Some remarks on the stable matching problem. Discr. Appl. Math. 11, 223–232 (1985)
9. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In: ICLP/SLP, pp. 1070–1080 (1988)
10. Gusfield, D.: Three fast algorithms for four problems in stable marriage. SIAM J. Comput. 16(1), 111–128 (1987)
11. Irving, R.: Stable marriage and indifference. Discr. Appl. Math. 48(3), 261–272 (1994)
12. Irving, R.: The cycle roommates problem: a hard case of kidney exchange. Inf. Process. Lett. 103(1), 1–4 (2007)
13. Irving, R., Leather, P., Gusfield, D.: An efficient algorithm for the "optimal" stable marriage. J. ACM 34(3), 532–543 (1987)
14. Iwama, K., Miyazaki, S.: A survey of the stable marriage problem and its variants. In: Proc. of ICKS 2008, pp. 131–136. IEEE Computer Society (2008)
15. Janhunen, T.: Representing normal programs with clauses. In: Proc. of the 16th European Conference on Artificial Intelligence, pp. 358–362. IOS Press (2004)
16. Kato, A.: Complexity of the sex-equal stable marriage problem. Japan Journal of Industrial and Applied Mathematics (JJIAM) 10, 1–19 (1993)
17. Manlove, D.: Stable marriage with ties and unacceptable partners. Tech. rep., University of Glasgow, Department of Computing Science (1999)
18. Manlove, D., Irving, R., Iwama, K., Miyazaki, S., Morita, Y.: Hard variants of stable marriage. Theoretical Computer Science 276(1-2), 261–279 (2002)
19. Marek, V., Nerode, A., Remmel, J.: A theory of nonmonotonic rule systems I. Ann. Math. Artif. Intell. 1, 241–273 (1990)
20. McDermid, E., Irving, R.: Sex-equal stable matchings: Complexity and exact algorithms. Algorithmica, 1–26 (2012)
21. Roth, A., Sömnez, T., Ünver, M.: Pairwise kidney exchange. J. Econ. Theory 125(2), 151–188 (2005)
22. Roth, A., Sotomayor, M.: Two-Sided Matching: A Study in Game-Theoretic Modeling and Analysis. Cambridge University Press (1990)
23. Xu, H., Li, B.: Egalitarian stable matching for VM migration in cloud computing. In: IEEE Computer Communication Workshop, pp. 631–636 (2011)

A Fuzzy, Utility-Based Approach for Proactive Policy-Based Management

Christoph Frenzel[1,2], Henning Sanneck[2], and Bernhard Bauer[1]

[1] Department of Computer Science, University of Augsburg, Augsburg, Germany
{frenzel,bauer}@informatik.uni-augsburg.de
[2] Nokia Siemens Networks Research, Munich, Germany
henning.sanneck@nsn.com

Abstract. Policy-based Management with rules is a wide-spread approach for operations automation. However, the continuous pressure for decreasing operational costs and increasing reliability of the systems lead to new challenges. Unfortunately, current Policy-based Management Systems lack the ability to act proactively along operational objectives in an autonomous manner in order to face these challenges. In this paper, we present a Policy-based Management System based on a Fuzzy Logic System that attempts to avoid problematic system states before they occur and that is guided by operator objectives expressed as utilities. Our approach can be seen as an extension of current rule-based Policy-based Management Systems, thus, requiring a reduced implementation effort.

Keywords: Policy-based Management, Proactive Management, Fuzzy Logic, Utility Theory, Rational Decision Making.

1 Introduction

Policy-based Management (PBM) is the continuous process of configuring the resources of a system such that the overall system performance satisfies the objectives of the operator. This process is controlled by a policy which encodes the technical knowledge and the preferences of the operator, usually as a set of rules like Event-Condition-Action (ECA) rules [3], [16]. This type of rule is triggered by an event which leads to the evaluation of the condition and, depending on the outcome, proposal of an action. The event is raised if the performance of the system is unacceptable which is usually determined through sharp constraints on Key Performance Indicators (KPIs). However, for an event there are usually several possible ways to react, i.e., several actions can be taken. In order to avoid these policy conflicts, human experts need to extend the rule conditions so that the PBM system selects the best of the possible actions in a specific situation based on the operator objectives and their technical experience [7]. As a result, the complexity of the policy is increased because the technical knowledge and operational objectives are mixed up.

In the future, the level of automation of PBM is required to be extended, driven by the increasing need for cost-efficient and reliable operations, as well as

L. Morgenstern et al. (Eds.): RuleML 2013, LNCS 8035, pp. 84–98, 2013.

the increasing complexity which prevents the understanding of the whole system by the operator. This leads to new challenges that a Policy-based Management System (PBMS) has to face. For instance, PBMSs are supposed to make more complex decisions automatically. This requires policies which directly express the operational objectives separately from the technical knowledge. This way, a PBMS is enabled to autonomously act towards achieving the objectives [10]. Furthermore, PBMSs must act proactively in order to avoid problems instead of reacting to problems that are already present. Hence, the sharp distinction between acceptable and unacceptable system states is not applicable anymore, and needs to be substituted with a fuzzy system state characterization that allows the system to react to gradually decreasing system performance. In summary, PBMSs need to proactively control the system guided by operational objectives.

In this paper, we present an approach for proactive PBM in order to face future challenges for PBM. Thereby, given the requirements for proactive PBM (Sec. 2), we outline the expected system behavior (Sec. 3), and then present the design of our solution (Sec. 4). The main idea behind the concept is to separate technical knowledge, expressed in fuzzy rules, from operator objectives, encoded in utilities, and utilize a fuzzy logic system in order to compute which actions satisfy the objectives the most. The approach can be seen as an extension of current rule-based PBMSs, thus, requiring a reduced initial implementation effort. The advantages of our system can be shown by comparing it with classical PBMSs (Sec. 5).

2 Problems of Classical PBM

It is assumed that the systems which are managed by a PBMS are composed of several system resources which are running autonomously controlled by some parameters. Driven by external or internal events, the operational state of the system, which is determined by the values of a set of KPIs and the presence of alarms that are produced by the system resources, can change. Thereby, the system can also transition into problematic operational states, e.g., if many users connect to a network at the same time, this can lead to an overload. In order to handle these undesired states, a PBMS, depicted in Fig. 1, continuously monitors the system using probes that identify unacceptable states, i.e., problems, and raise events that indicate the nature of the detected problems. Thereby, the distinction between acceptable and unacceptable states is sharp, usually defined through constraints, e.g., a KPI k representing a ratio could have a constraint requiring that $k > 2\%$.

The decision making component evaluates the policy and proposes actions in reaction to the events, thereby considering the operational context of the system. The latter can be any information about the system, e.g., the current time or Configuration Management (CM) data like the system topology. An action that is proposed can be, e.g., the setting of new configuration parameter values of a system resource or the call of some special system function that determines new parameters and configures a system resource accordingly.

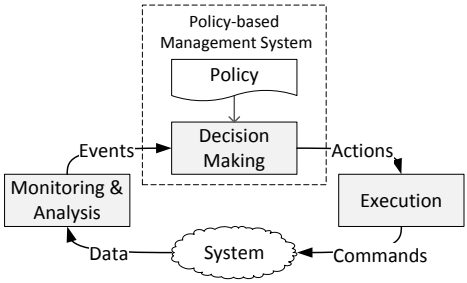


Fig. 1. The PBM process

Usually, the PBMS will propose several actions at the same time because, on the one hand, there can be several rules triggered by one event and, on the other hand, several events can be raised in parallel. A common challenge is that the actions might be in conflict [13], [3], i.e., they are somehow mutually exclusive. Policy designers try to detect these conflicts using sophisticated methods and resolve them by adapting the rule. Thereby, they often add further conditions to the rules which determine in which situations one action is preferred over another one based on the operational objectives. As a consequence, the policy mixes up both the technical knowledge about which action is a reasonable reaction for an event in a specific situation, and the operator objectives which describe the operator's preferences for the actions in specific situations. However, since it is not possible to detect and resolve all policy conflicts at design-time [13], there is still the need for an on-line conflict resolution. Unfortunately, current systems do not allow to encode complex operational objectives for this.

Figure 2 depicts an example from mobile networks management where a PBMS can distinguish between four operational states given two KPIs, the Dropped Call Rate (DCR) and the Call Setup Success Rate (CSSR). The DCR indicates unacceptable reliability problems if the DCR is high, whereas the CSSR indicates unacceptable availability problems if the CSSR is low. The two lines show the thresholds that are defined for the KPIs separating the acceptable from the unacceptable states. Specifically, Zone 1 is acceptable with respect to both KPIs, Zone 2 is unacceptable regarding DCR but acceptable regarding CSSR, Zone 3 is acceptable regarding DCR but unacceptable regarding CSSR, and Zone 4 is unacceptable for both KPIs. In a classical PBMS, a probe would raise an event as soon as an unacceptable state regarding a KPI is reached. Consequently, in Zone 4 two events indicating the DCR problem and CSSR problem would be triggered. Hence, the PBMS is required to perform some conflict resolution if the respective actions are in conflict.

The sharp thresholds in classical PBM do not allow to proactively react to possibly upcoming events. One solution to this is to lower the thresholds and, so, reduce the number of acceptable system states. In Fig. 2, this would mean to shrink Zone 1 to the dashed rectangle. However, then the system cannot distinguish between an event indicating a system state in Zone 1, i.e., a proactive

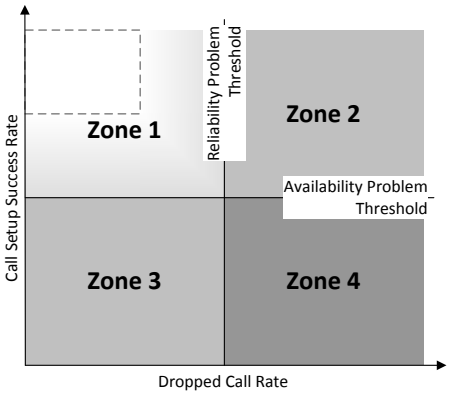


Fig. 2. Operational states for the KPIs DCR and CSSR

but not severe problem, and an event indicating a system state in Zone 2, i.e., a severe problem. As a result, the PBMS might select non-optimal actions.

If the level of automation in PBM needs to increase in the future, a classical PBMS as outlined in this section has two shortcomings:

- The sharp differentiation between acceptable and unacceptable operational states solely allows a reactive behavior, i.e., the problem must occur before the system can take countermeasures. However, it is desirable that the system avoids problematic state by proactively performing some action.
- The mingling of technical knowledge and operational objectives makes the maintenance of the policy costly. Both have different life cycles, i.e., the technical knowledge usually changes less frequently than the operational goals, thus, with every single change, the whole policy needs to be revised.

3 A Concept for Proactive PBM

In order to solve the issues with classical PBMSs, we model a utility-based rule system as a specification for a fuzzy logic system.

A utility-based rule system allows the separation of technical knowledge and operator objectives by expressing the former as rules, e.g., ECA rules, and the latter as utilities [7]. Upon a raised event, the rules are evaluated and applicable actions proposed, i.e., actions that treat the event from a purely technical point of view. Since these actions might be in conflict, a conflict resolution utilizes the utilities in order to calculate the *value* of each action based on the events an action handles, the *utility* of the events' treatment, and the severity of the events. Finally, an action is selected for execution based on its value, i.e., usually the action with the highest value. This can be seen as a kind of rational behavior [15].

A fuzzy logic system is a rule system that can perform logical inference with fuzzy sets, i.e., it can fuzzify input values into fuzzy sets, apply fuzzy rules

on these sets, and defuzzify the resulting sets into sharp output values [12]. The general idea is to replace boolean predicates used in classical rule systems with continuous memberships in the domain $[0, 1]$ representing the degree to which a predicate is true. Fuzzy logic systems provide a scientific and well-known foundation for decision making in uncertain or inaccurate domains. There are several implementations of fuzzy logic system available, e.g., jFuzzyLogic [5], and also a standard syntax, called Fuzzy Control Language [9], that is used in this paper.

The basic idea of combining a utility-based rule system and a fuzzy logic system is to

- fuzzify the outputs of the monitoring probes into fuzzy event levels, i.e., each system state has a fuzzy membership to the set of unacceptable states regarding each event in the interval $[0, 1]$,
- model the technical knowledge as fuzzy rules and weight them with the utilities, i.e., the preferences of the operator,
- utilize a fuzzy logic system to compute the value of each action, i.e., the degree to which an action satisfies the operator's preferences [7], and
- defuzzify the result such that conflicts between actions are resolved by selecting the actions according to their value.

In order to outline the expected system behavior, consider Fig. 2 again. The sharp constraints distinguishing acceptable and unacceptable states are substituted by fuzzy event levels, i.e., fuzzy memberships to the unacceptable zone indicated by the events. Therefore, the Zone 1 is divided into two sub-zones:

- A zone indicating a perfect system state, i.e., a state where the system is perfectly working and no problem is supposed to arise soon. In this zone, depicted as the dashed rectangle, all fuzzy event levels are 0.
- A jeopardy zone [1], i.e., states where the system performance is still acceptable but there is the danger of running into a faulty state. Therefore, the system should react in order to avoid the unacceptable state. In this zone, indicated by the gradient, at least one fuzzy event level is greater than 0.

Given the fuzzy event levels, the system selects the best action with respect to the level of the events and the utility the action has to the operator, i.e., how important the treatment of the problem is according to the operator goals.

4 Design of the Fuzzy PBMS with Utilities

The reasoning process of the fuzzy PBMS with utilities, depicted in Fig. 3, is performed in three steps which are spread over the monitoring and analysis as well as the decision making phase of the PBM process depicted in Fig. 1:

Fuzzification is performed by the monitoring probes in the monitoring and analysis phase. Thereby, sharp thresholds for the decision to raise an event are replaced by fuzzy membership functions defined by the system operator in the event specification. As a result, the probes are raising fuzzy events, i.e., events containing a fuzzy event level representing the membership.

Fuzzy inference is performed as first step of the decision making phase. Based on fuzzy events and fuzzy context information, the fuzzy rules are evaluated. The rules, which are provided by the system administrator, define possible actions in reaction to some event in a specific operational context. As a result, this process provides the value of all actions based on the system objectives which are provided as utilities by the operator.

Defuzzification is the second step of the decision making phase. The proposed actions are analyzed for conflicts that are defined in the conflict specification given by the administrator. The detected conflicts are resolved by selecting actions for execution based on the value.

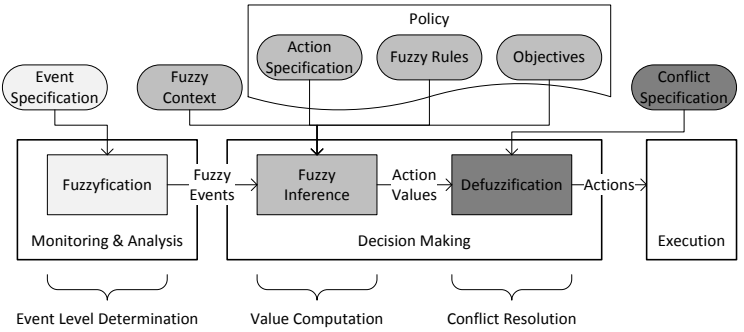


Fig. 3. The reasoning process of the fuzzy PBMS with utilities

4.1 Fuzzification

The fuzzification extends the monitoring probes with the ability to annotate the events with an event level in order to turn them into fuzzy events. This level has the semantics of the degree to which the operator wants to handle the event in order to avoid a possible negative system state in the future. The event levels are computed by membership functions, e.g., sigmoid or linear functions, which determine the membership degree of a system state to the fuzzy set raised for an event. The membership functions form the event specification provided by the operator. They are determined through an analysis of the system behavior, e.g., whether the system state changes rapidly, and the trade-off between the benefits of reacting proactively and the efforts of executing potentially more actions.

The thresholds used by classical monitoring probes define two classes of system states: the acceptable state and the unacceptable state. In fuzzy monitoring probes, the membership functions define three classes of system states as shown in Fig. 4: the acceptable state, the unacceptable state, and the jeopardy state in between. In the concrete example, the PBMS should not react if the DCR is below 1%, because the system works perfectly fine. This is indicated by an event level of 0 and, so, the event would not be raised at all. Between a DCR of 1% and 2% the event level linearly increases from 0 to 1. Hence, in this jeopardy

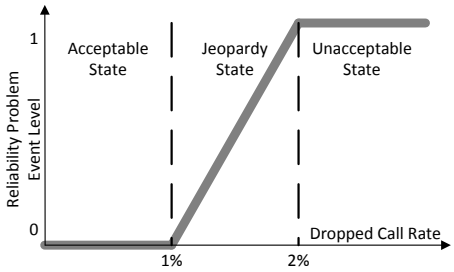


Fig. 4. The three classes of system states determined by fuzzy monitoring probes

state the handling of the problem becomes more and more urgent. Finally, at a DCR of 2% and above, the system is in an unacceptable state. Hence, the event level is 1 in this range. Note that a classical threshold would be a DCR of 2%.

4.2 Fuzzy Inference

The fuzzy inference is the core of the system and determines the value of the actions as shown in Fig. 3. Besides the fuzzy events, it has four inputs:

- A fuzzy context that provides contextual information about the system.
- An action specification defining the actions the system can take.
- A set of operator objectives including their utilities.
- A set of fuzzy rules that define the possible reactions to a fuzzy event in a specific context from a technical point of view.

Fuzzy Context. The operational context of the system contains information about, e.g., the system topology, system configuration, current system status, and current time and date. In principle, the context needs to provide all the information that is necessary for evaluating the conditions of the fuzzy rules.

The context is fuzzified by using membership functions just like the events. For instance, instead of representing the time with a concrete hour and minute, it is possible to define fuzzy sets like daytime and nighttime. Note that the membership functions can also define crisp sets.

Objectives. Utility theory [15] provides a framework to represent and reason with complex objectives by using a single measure called utility. In other words, the utility represents the degree to which an operational state satisfies the objectives, so, allowing to compare different states in order to make decisions. The elicitation of utilities is a non-trivial process and an active field of research [6].

In the presented approach, each fuzzy rule has an assigned objective and each of these objectives is mapped to a utility, i.e., a real value. The semantics of a rule is that if the action of the rule is executed in response to the event under the operational context determined by the rule, then it fulfills the assigned objective

and produces the respective utility. Technically, the objectives and utilities are a set of variables and their respective real values in the fuzzy logic system.

Although this approach allows a fine-granular rule–objective mapping, it is not recommended to assign different objectives to every rule due to the costly utility elicitation. An example for a reasonable objective assignment is the definition along the events that can be raised. That means that there is an objective for every event and the utility represents the importance of the event resolution.

Action Specification. The action specification defines the values of the output variable of the fuzzy logic system, i.e., the representation of the possible actions. Specifically, it defines the output variable action as well as a crisp singleton set, i.e., a single value, for each action on that variable. Although singletons are not common in fuzzy logic systems, it is a reasonable modeling approach since the single values represent discrete actions on the continuous variable action. For instance, consider the two actions Mobility Robustness Optimization (MRO) and Mobility Load Balancing (MLB) that are treating a reliability problem and an availability problem respectively. For each, a singleton set is created on the variable action, e.g., MRO is linked to a singleton set with the value 1 and MLB is linked to a singleton set with value 2. Note that, depending on the defuzzification method, the values of the actions can represent preferences (cf. Sec. 4.3).

Fuzzy Rules. The fuzzy rules express the expected behavior of the PBMS from a technical point of view. Hence, these rules should be designed based on the technical knowledge of the rule designers without considering the operational goals. Specifically, these rules can contain policy conflicts. We consider Mamdani-style [12] fuzzy rules with the following general structure:

IF event IS raised AND condition **THEN** action IS singleton **WITH** objective

The form of the rules is aligned with the ECA rule pattern. It has three parts:

IF part or antecedent consists of two components which are connected by a conjunction. First, the evaluation of a single event level, i.e., the membership of event to the set raised. Second, a condition part which can be any fuzzy logic formula over the operational context.

THEN part or consequent proposes a single action by setting the value of the output variable action to a singleton set representing the proposed action.

WITH part weights the rule with the utility of the assigned objective.

Note that this general rule structure also allows to model complex technical knowledge as well as operator objectives (cf. [7]). For instance, by extending the WITH part to be the product of the objective's utility and an *effectiveness*, one can represent a confidence in the action, i.e., the likeliness that the action will treat the event correctly and generate the utility. Furthermore, imagine that actions have fixed costs: this can be modeled by creating rules which have an empty antecedent, i.e., they are always triggered, and a cost instead of an objective which is some negative number.

System operators working with PBMS are used to ECA rules and so, they might be unfamiliar with fuzzy rules. Furthermore, the structure of the fuzzy rules is inconvenient since it is driven by the technical capabilities of a fuzzy logic system. However, a simple transformation is able to translate ECA rules into fuzzy rules. Thereby, it is necessary to group the ECA rules into policy groups [16] which model the objectives the rules are fulfilling.

Inference. A fuzzy logic system evaluates each fuzzy rule and calculates the output membership of the proposed action, i.e., the *expected utility*. Thereby, the fuzzy logic semantics is aligned with the ECA paradigm, i.e., AND, OR, and NOT are interpreted as usual as minimum, maximum, and complement [12]. The expected utility of a rule r, which proposes an action, is calculated as

$$U_{\text{exp}}(r) = W_{\text{act}}(r)U(r) \tag{1}$$

with, $W_{\text{act}}(r)$ being the rule activation, i.e., the combined fuzzy membership of the antecedent, and $U(r)$ being the utility of the objective of the rule.

After that, the expected utilities of the fuzzy rules must be combined in order to compute the value of each action. This combination is defined by the *accumulation method* in a fuzzy logic system. A common approach is to sum up the utilities that an action produces, however, this requires mutual preferential independence [15] between the objectives, i.e., the utility of an objective for an action is independent of the expected utilities of the other objectives for that action. This assumption is implied by the structure of the objectives and fuzzy rules, specifically the direct rule–objective assignment. However, the expected utilities cannot be simply summed up since several rules can propose the same action with the same objective. Hence, the accumulation needs to distinguish between rules assigned to the same objective and rules assigned to different objectives as outlined by the following example.

Consider the fuzzy rules r_{1a} and r_{1b} that are both assigned to objective o_1, and rule r_2 which is assigned to objective o_2. All three rules are triggered by the event e and propose the action a. In this setting, the expected utilities for o_1 from r_{1a} and r_{1b} should count once since, even though the action has been proposed twice, it can satisfy o_1 only once. However, the expected utilities for o_1 and o_2 are summed up since the action satisfies both objectives in parallel.

Formally, the value V of a action a is defined as:

$$V(a) = \sum_{\rho \in R(a)/\sim} \max(\{U_{\text{exp}}(r) | r \in \rho\}) \tag{2}$$

whereby $R(a)/\sim$ is the partition of the set of all rules proposing a with respect to equal objectives. Hence, the value of an action is determined by, first, calculating the maximum of the expected utilities for each set of rules that have the same objective and, second, summing up these maxima.

Technically, the accumulation can be implemented by grouping the fuzzy rules assigned to the same objective into one Fuzzy Control Language rule group.

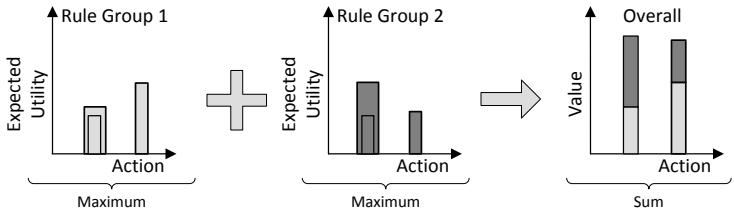


Fig. 5. The accumulation of the expected utilities to the value

Within a rule group, the expected utilities of the rules are accumulated by selecting the highest one, whereas the combined expected utilities of different rule groups are summed up as shown in Fig. 5. This functionality is not standardized and, so, this usually requires an adaptation of the fuzzy logic system. Note that this adaptation can be avoided if it can be ensured that no two rules that propose the same action for the same objective are ever triggered together.

4.3 Defuzzification

All actions that have a value greater than 0 are seen as proposed for execution. However, not all can be executed in parallel because there might be conflicts between them which are modeled in the conflict specification shown in Fig. 3. Hence, the defuzzification selects the best actions that can be executed based on their values. The actual output of the defuzzification are real numbers which represent values of the variable action. They can be translated into the actions by using the action specification for the fuzzy inference.

In simple cases, all actions are in conflict, i.e., only one action can be selected at a time. Then, the best action is the one with the highest value since it satisfies the most severe and important objectives. It can be selected from the fuzzy variable action using a standard defuzzify method which selects the singleton set with the highest membership degree. Thereby, the operator can prioritize actions in the action specification if there are ties. For instance, if ties are broken by selecting the action with the smallest value of the variable action, the operator should assign smaller variable values to the actions that are more preferred.

More sophisticated defuzzification methods can also determine a set of best actions if the conflicts are more complex. Suppose that the conflict specification is a set of pairs of actions that cannot be executed together. So, the defuzzification needs to find a combination of actions the maximizes the overall value produced by the selected actions. This assignment problem can be formulated as a constraint optimization problem which can be solved with a constraint optimizer.

5 Case Study

In the following case study, we show the application of the fuzzy PBMS with utilities in mobile networks management and evaluate its performance using a demonstrator system based on jFuzzyLogic [5].

5.1 Scenario

The scenario is a 3rd Generation Partnership Project (3GPP) Long Term Evolution (LTE) network which is managed as a Self-Organizing Network (SON), i.e., the network has self-configuration, self-optimization, and self-healing features [8]. A SON is characterized by autonomous SON functions which continuously monitor the network and trigger the execution of algorithms in order to resolve detected problems. Thereby, the SON functions can influence each other, e.g., via common configuration parameters (output of SON functions) or commonly influenced KPIs (as input to SON functions), which leads to conflicts. These conflicts are detected and resolved by the SON coordination function.

The PBMS for the case study is a simplified SON coordination. The SON functions can be seen as monitoring probes which trigger the execution of some resolution action. Since the monitoring of the functions is quite radio specific, the details of the fuzzification are not presented here. Instead, we concentrate on the decision making component, i.e., the SON coordination, and consider the fuzzy events as given. We assume that, in discrete time intervals, all raised fuzzy events are passed to the SON coordination which determines suitable actions to resolve the problems, analyzes their values with respect to the operator objectives, and triggers the best action. Thereby, all actions are assumed to be mutually in conflict in order to keep the case study simple.

5.2 System Model

Figure 6 depicts the technical knowledge that is encoded in the fuzzy rules for the fuzzy logic system. There are five fuzzy events that can be raised and six actions that can be triggered. The arrows between the events and actions outline the rules of the system, i.e., for each event e and action a pair which is connected by an arrow, there is a rule which proposes a if e is present. However, a rule can also contain a context condition which is not depicted. For instance, the arrow between reliability problem and Remote Electrical Tilt (RET) optimization represents the following fuzzy rule:

IF reliability_problem IS raised AND ret_available IS true
THEN action IS ret_optimization **WITH** objective_dcr

Note that the rule evaluates the context whether the RET feature is available. The events in SON encode specific network problems and, so, we defined the objectives with respect to the fuzzy events, i.e., there is an objective for each event. The utilities of these objectives are normalized, i.e., each utility is in the range $[0, 1]$ and the sum of the utilities is 1.

The fuzzy PBMS with utilities is evaluated against two simpler PBMSs:

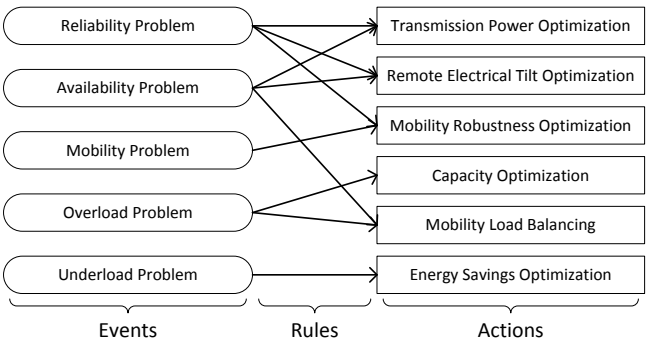


Fig. 6. Events, actions, and fuzzy rules in the case study

Classical PBMS considers neither fuzzy event levels nor operator objectives. Hence, the event fuzzification is actually a sharp threshold which is set to the border between jeopardy state and good state. Furthermore, the utility of all objectives is equal.

Fuzzy PBMS uses fuzzy events in order to take the severity of the events into consideration. However, the utilities of all objectives are equal.

Fuzzy PBMS with utilities considers both fuzzy events and utilities.

Operators adopting the fuzzy PBMS with utilities will likely start with the classical approach and introduce the fuzzy PBMS as an intermediate step. After gaining some experience with the system, the operator will adapt the utilities to meet the true objectives and, so, switch to the fuzzy PBMS with utilities.

5.3 Evaluation

The evaluation is performed by comparing the performance of the three PBMSs regarding the average value of the actions they select. Several sets of operator objectives, i.e., utilities, and problem situations, i.e., event levels and contexts, are created randomly. Thereby, the probabilities for an event level of 0 or 1 is 0.25 each, whereas, the event levels in $]0, 1[$ are uniformly probable. Finally, each combination of operator objectives and problem situation is fed into the three PBMSs and the values of the proposed actions with respect to the operator objectives and the problem situation are recorded.

Figure 7 depicts the average value of the selected actions for 100 different objective sets, each evaluated with 1000 different problem situations. As expected, the classical PBMS has the lowest average value since it utilizes the least information, i.e., it neglects the fuzzy event levels and utilities. As a result, this system always picks the action which resolves most of the raised events. The fuzzy PBMS performs better since it utilizes the event level information. Hence, it neglects events with low levels, i.e., that are less important, and concentrates on the events with high levels. In summary, this system picks the action with the

highest sum of the levels of the resolved events. However, the best performance is shown by the fuzzy PBMS with utilities because it knows the objectives of the operators and, so, concentrates on the events with high levels that are also valuable to the operator. Actually, the fuzzy PBMS with utilities always selects the optimal action in this scenario where the actions are mutually conflicting.

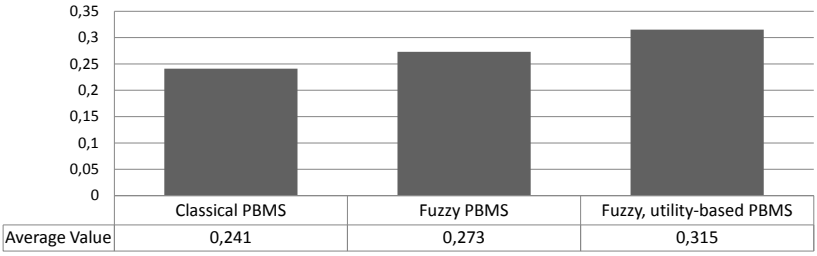


Fig. 7. Average value produced by the three PBMSs

Although the ranking of the PBMSs is not surprising, it is interesting that the fuzzy PBMS performs 13% better than the classical system and the fuzzy PBMS with utilities performs 15% better than the fuzzy system. This significant performance increase between these two evolutionary steps indicates that a gradual adoption of the fuzzy PBMS with utilities is a reasonable approach.

6 Related Work

Using a fuzzy logic system for systems management is not a new idea. For instance, Kousaridas and Nguengang [11] utilize two fuzzy logic systems in sequence: the first estimates the network state given low-level technical measurements and the second proposes actions to be performed based on the fuzzy network state. The fuzzy membership degree is used to compute a confidence in the decision. However, the approach neither separates the operator objectives from the technical knowledge nor considers values.

Fuzzy logic systems have also been used for multi-criteria decision making [14], [2], [17], i.e., the evaluation of preferences for specific decision options with respect to some objectives. Thereby, the system is given a fuzzy satisfaction value for each action regarding each objective. Based on these values, the system selects the best action using the Max-Min selector, i.e., it determines the minimal satisfaction value regarding an objective for each action and selects the action with the highest minimal value. Boutalis and Schmidt [4] present a nice application for a fuzzy discrete event system in the domain of mobile robots. However, the pessimistic multi-criteria decision making approach is not applicable for PBM: usually no action is satisfying all objectives at once and, thus, the minimal satisfaction value for all actions would be 0. Hence, we adopt a more decision theoretic approach [15] which can be seen as a Sum-Max selector. Nevertheless,

by changing the accumulation methods (cf. Sec. 4.2) appropriately, one can also use the pessimistic Max-Min selector with our approach.

In [7], we have presented a PBM approach that considers uncertainty in the inputs and operational goals. The Rational Policy System extends an ECA rule system with probabilistic events and a dynamic action conflict resolution that is based on the value of the actions. Hence, the approach makes a clear distinction between technical knowledge, represented in ECA rules, and operator objectives, encoded in utilities. However, there are some differences between the Rational Policy System and the presented approach. First, the reasoning of the Rational Policy System is hard coded in program code and, so, less flexible to adapt than the fuzzy logic problem formulation of our approach, e.g., if an operator decides to use the Max-Min selector. Second, in contrast to the Rational Policy System which solely allows for probabilistic events, the fuzzy logic approach allows both fuzzy events and fuzzy context information. Third, the semantics of probabilistic events used in the Rational Policy System is different from fuzzy events.

Another approach for PBM with utilities and uncertainty is presented by Bartolini et al. [1]. In order to keep the efforts for system modeling low, they introduce a jeopardy system state for each KPI, i.e., a special acceptable state in which the system soon runs into an unacceptable state with some probability. The system selects the best action based on the system state, a specification of the probabilistic action effects, and the operator objectives. However, modeling the system behavior as probabilistic action effects instead of rules is a complex mental process for policy designers who are used to rule-based PBMSs.

7 Conclusion

In this paper, we presented a fuzzy PBMS with utilities which enables proactive PBM. It determines the best action in response to fuzzy event with respect to operator objectives. Thereby, technical knowledge, expressed in fuzzy rules, and operator objectives, encoded in the utilities, are separated to facilitate different life-cycles of both information models. The focus has been on providing an extension of classical PBM which requires little modeling effort.

In the future, it seems very promising to add a learning component to the system which can estimate the effectiveness of the rules by observing the system behavior. Furthermore, the system needs to avoid the recurring execution of actions if they fail to work. This can be done by keeping a history of the executed actions and avoid them.

References

1. Bartolini, C., Sallé, M., Trastour, D.: IT service management driven by business objectives: an application to incident management. In: Proc. 10th IEEE/IFIP Network Operations and Management Symposium (NOMS 2006), pp. 45–55. IEEE, Vancouver (2006)
2. Bellman, R.E., Zadeh, L.A.: Decision-Making in a Fuzzy Environment. Tech. Rep. NASA CR-1594, National Aeronautics and Space Administration (1970)

3. Boutaba, R., Aib, I.: Policy-based Management: A Historical Perspective. Journal of Network and Systems Management 15(4), 447–480 (2007)
4. Boutalis, Y., Schmidt, K.: Multi-objective decision making using fuzzy discrete event systems: A mobile robot example. In: Proc. 18th Mediterranean Conference on Control and Automation (MED 2010), pp. 575–580. IEEE, Marrakech (2010)
5. Cingolani, P., Alcala-Fdez, J.: jFuzzyLogic: a robust and flexible Fuzzy-Logic inference system language implementation. In: Proc. IEEE International Conference on Fuzzy Systems (Fuzz 2012), pp. 1–8. IEEE, Brisbane (2012)
6. Domshlak, C., Hüllermeier, E., Kaci, S., Prade, H.: Preferences in AI: An overview. Artificial Intelligence 175(7-8), 1037–1052 (2011)
7. Frenzel, C., Sanneck, H., Bauer, B.: Rational Policy System for Network Management. In: Proc. International Symposium on Integrated Network Management (IM 2013). IEEE, Ghent (2013)
8. Hämäläinen, S., Sanneck, H., Sartori, C.: LTE Self-organizing Networks (SON): Network Management Automation for Operational Efficiency. Wiley, Chichester (2011)
9. International Electronical Commission (IEC): IEC 1131 - Programmable Controllers: Part 7 - Fuzzy Control Programming (January 1997), `http://www.fuzzytech.com/binaries/ieccd1.pdf`
10. Kephart, J., Walsh, W.: An artificial intelligence perspective on autonomic computing policies. In: Proc. IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY 2004), pp. 3–12. IEEE, Yorktown Heights (2004)
11. Kousaridas, A., Nguengang, G.: Deliverable D2.3: Final Report on Self-Management Artefacts. Tech. rep., Self-Management of Cognitive Future InterNET Elements, Self-NET (2010)
12. Kruse, R., Gebhardt, J., Klawonn, F.: Foundations of Fuzzy Systems. Wiley, New York (1994)
13. Lupu, E., Sloman, M.: Conflicts in policy-based distributed systems management. IEEE Transactions on Software Engineering 25(6), 852–869 (1999)
14. O'Hagan, M.: A Fuzzy Decision Maker, `http://www.fuzzysys.com/fdmtheor.pdf`
15. Russell, S.J., Norvig, P.: Artificial Intelligence: A Modern Approach, 2nd edn. Prentice Hall, Upper Saddle River (2003)
16. Strassner, J.: Policy-Based Network Management: Solutions for the Next Generation. Morgan Kaufmann, Amsterdam (2004)
17. Zimmermann, H.J.: Fuzzy Programming and Linear Programming with Several Objective Functions. Fuzzy Sets and Systems 1(1), 45–55 (1978)

Picking Up the Best Goal
An Analytical Study in Defeasible Logic

Guido Governatori[1,4], Francesco Olivieri[1,2,4], Antonino Rotolo[3],
Simone Scannapieco[1,2,4], and Matteo Cristani[2]

[1] NICTA, Queensland Research Laboratory, Australia
[2] Department of Computer Science, University of Verona, Italy
[3] CIRSFID and DSG, University of Bologna, Italy
[4] Institute for Integrated and Intelligent Systems, Griffith University, Australia

Abstract. In this paper we analyse different notions of the concept of goal starting from the idea of sequences of "alternative acceptable outcomes". We study the relationships between goals and concepts like agent's beliefs, norms, and desires, and we propose a formalisation using Defeasible Logic that will be able to provide a computationally feasible approach. The resulting system captures various nuances of the notion of goal against different normative domains, for which the right decision is not only context-dependent, but it must be chosen from a pool of alternatives as wide as possible.

1 Motivation and Basic Intuitions

The BDI architecture is a prominent approach to model rational agents [1–3]. BDI agents are means-ends reasoners equipped with: (i.) Desires, Goals, Intentions (or Tasks); (ii.) a description of the current state of the environment (Beliefs); (iii.) Actions. The key tenet of this architecture is that the agent's behaviour is the outcome of a rational balance among different mental states. Previous seminal works on the BDI paradigm [2–4], or implementing the BDI architecture in Defeasible Logic [5–7] have assumed that mental states are primitive and independent from each other, even though some mutual influences are considered (e.g., intentions are seen as desires satisfied up to commitment).

We work here on a different perspective to provide a fresh and efficient rule-based framework that considers goals, desires, and intentions as facets of the same phenomenon (all of them being *goal-like attitudes*): the notion of *outcome*, which is simply something an agent would like or is expected to achieve. An advantage of the proposed framework is that it allows agents to compute different degrees of motivational attitudes, and degrees of commitment that take into account other factors, such as beliefs and norms.

While different schemas for generating and filtering agents' outcomes are possible, we will restrict ourself to schemas where we adopt the following principles:

- When an agent faces alternative outcomes in a given context, it is natural to rank them in a preference order;
- Beliefs prevail over conflicting motivational attitudes, thus avoiding various cases of wishful thinking [8, 9];

L. Morgenstern et al. (Eds.): RuleML 2013, LNCS 8035, pp. 99–113, 2013.

- Norms and obligations are used to filter social motivational states (*social intentions*) and compliant agents [9, 6];
- Goal-like attitudes can be derived via *conversion* using other mental states, such as beliefs (e.g., believing that Madrid is in Spain may imply that the goal to go to Madrid implies the goal to go to Spain) [6].

Consider the following example, Alice, during her holidays, plans to pay a visit to her friend John, who lives close to her parents. The plan can be described by the sentence *I shall come over to John's place to visit him on Monday, but if he is not home or the visit is not possible, I am going to visit my parents. If this is not possible as well, I shall take some rest at home.* This idea can be easily implemented by building for each alternative a sequence of other alternatives $A_1, \ldots, A_n$ that are preferred when the first choice is no longer feasible. Normally, each set of alternatives is the result of a specific context C. This scenario can be represented as $C \Rightarrow A_1, \ldots, A_n$ which is closely related of contrary-to-duty obligations [10], where a norm is represented by an Obligation rule of the type:

$$r_1 : drive\_car \Rightarrow_{\mathsf{O}} \neg damage \odot compensate \odot foreclosure.$$

Rule r_1 states that, if an agent drives a car, she has the obligation not to cause any damage to others; if this happens, she is obliged to compensate; if she fails to compensate, there is an obligation of foreclosure. The previous setting can be rewritten as:

$$r_2 : holiday \Rightarrow_{\mathsf{U}} visit\_friend \odot visit\_parents \odot stay\_home.$$

where r_2 is a rule introducing the oUtcome mode. In both examples, the sequences express a preference ordering among alternatives, which means that also *stay_home* and *foreclosure*, though not the best options, still correspond to acceptable situations.

Besides rules for outcomes and obligations, we also have rules for beliefs such as

$$r_3 : friend\_away \Rightarrow_{\mathsf{B}} \neg visit\_friend$$

for which we assume there is no preference ordering, since they do not express expected outcomes but simply describe how the world is.

These building blocks allow us to introduce different types of goal-like attitudes and degrees of commitment to outcomes: desires, goals, intentions, and social intentions.

Desires as Acceptable Outcomes. Suppose an agent is equipped with the following outcome rules expressing two preference orderings:

$$r : a_1, \ldots, a_n \Rightarrow_{\mathsf{U}} b_1 \odot \cdots \odot b_m \qquad s : a'_1, \ldots, a'_n \Rightarrow_{\mathsf{U}} b'_1 \odot \cdots \odot b'_k$$

and that the situation described by $a_1, \ldots, a_n$ and $a'_1, \ldots, a'_n$ are mutually compatible but b_1 and b'_1 are not, namely $b_1 = \neg b'_1$. In this case $b_1, \ldots, b_m, b'_1, \ldots, b'_k$ are anyway all *acceptable outcomes*, including the incompatible outcomes b_1 and b'_1. *Desires* are expected or acceptable outcomes, independently of whether they are compatible with other expected or acceptable outcomes.

Goals as Preferred Outcomes. For rule r alone the preferred outcome is b_1, and for rule s alone it is b_1'. But if both rules are applicable, then a state where both b_1 and b_1' hold is not possible: the agent would not be rational if she considers both b_1 and $\neg b_1$ as her preferred outcomes. Hence, the agent has to decide if she prefers a state where b_1 holds to one where b_1' (i.e., $\neg b_1$) holds, or *vice versa*. If the agent cannot make up her mind, i.e., she has no way to decide which is the most suitable option for her, then neither the chain of r nor that of s can produce preferred outcomes.

Suppose that the agent opts for the latter option; this can be done if the agent establishes that the second rule overrides the first one, i.e., $s > r$. Accordingly, the preferred outcome is b_1' for the chain of outcomes defined by s, and b_2 is the preferred outcome of r. b_2 is the second best alternative according to rule r: in fact b_1 has been discarded as an acceptable outcome given that s prevails over r.

Two Degrees of Commitment: Intentions and Social Intentions. The next issue is to clarify which are the acceptable outcomes for an agent to commit to. Naturally, if the agent values some outcomes more than others, she should strive for the best, i.e., for the most preferred outcomes.

Let us start by considering the case where only rule r applies. Here, the agent should commit to the outcome she values the most, i.e., b_1. But what if the agent *believes* that b_1 cannot be achieved in the environment where she is currently situated in, or she knows that $\neg b_1$ holds? Committing to b_1 would result in a waste of agent's resources; rationally, she should target the next best outcome, in this case b_2. Suppose, now, that b_2 is *forbidden*, and the agent is social (an agent is social if the agent would not knowingly commit to anything that is forbidden [6]). Once again, in this situation the agent has to lower her expectation and settle for b_3, which is the next acceptable outcome.

To complete the analysis, consider the situation where both rules r and s apply and the agent prefers s to r. As we have seen before, $\neg b_1$ (b_1') and b_2 are the preferred outcomes based on the preference of the agent over the two rules. Assume that, this time, the agent knows she cannot achieve $\neg b_1$ (or equivalently, b_1 holds). If the agent is rational, she cannot commit to $\neg b_1$. Thus, the best option for her is to commit to b_2' and b_1, where she is guaranteed to be successful. In this scenario, the best course of action for the agent is where she commits herself to some outcomes that are not her preferred ones, or even that she would consider not acceptable based only on her preferences, but such that they influence her decision process given that they represent relevant external factors (either her beliefs or the norms that apply to her).

The layout of the paper is as follows. Section 2 presents the new logical framework. Section 3 describes the algorithms to prove that the logic has linear complexity. Section 4 ends the paper with some conclusions and a discussion of related work.

2 Logic

Defeasible Logic (DL) [11] is a simple but flexible and efficient rule based non-monotonic formalism. The strength of DL lays in the constructive proof theory, which has an argumentation-like structure and allows us to draw meaningful conclusions from (potentially) conflicting and incomplete knowledge bases. The framework provided by

the proof theory accounts for the possibility of extensions of the logic, in particular extensions with modal operators. Several extensions have been proposed, which resulted in applications in the area of normative reasoning [12], modelling agents [6, 13, 7], and business process compliance [14], as well as efficient implementations of the logic (including the modal variants), able to handle very large knowledge bases [15–17].

2.1 Language

The main aim of this subsection is to establish an inference process to compute factual knowledge, desires, intentions, goals and obligations from existing facts, primitive desires, intentions, goals and unconditional obligations. As a first step, we introduce the language adopted. Let PROP be a set of propositional atoms, $\mathrm{MOD} = \{\mathsf{B}, \mathsf{O}, \mathsf{D}, \mathsf{G}, \mathsf{I}, \mathsf{SI}\}$ the set of modal operators[1] and Lbl be a set of arbitrary labels. The set $\mathrm{Lit} = \mathrm{PROP} \cup \{\neg p | p \in \mathrm{PROP}\}$ denotes the set of *literals*. The *complementary* of a literal q is denoted by $\sim q$; if q is a positive literal p, then $\sim q$ is $\neg p$, and if q is a negative literal $\neg p$ then $\sim q$ is p. The set of *modal literals* is $\mathrm{ModLit} = \{\Box l, \neg\Box l | l \in \mathrm{Lit}, \Box \in \{\mathsf{O}, \mathsf{D}, \mathsf{G}, \mathsf{I}, \mathsf{SI}\}\}$. We assume that the "$\Box$" modal operator for belief B is the empty modal operator, thus a modal literal $\mathsf{B}l$ is equivalent to literal l. Accordingly, we state that the complementary of $\mathsf{B}\sim l$ as well as $\neg\mathsf{B}l$ is $\sim l$.

We define a *defeasible theory* D as a structure $(F, R, >)$, where (i.) F is a set of *facts* or indisputable statements, (ii.) R contains three sets of *rules*: for beliefs, obligations, and outcomes and (iii.) $> \subseteq R \times R$ is a *superiority relation* to determine the relative strength of conflicting rules. *Belief rules* are used to relate the factual knowledge of an agent (her vision of the environment), and defines the relationships between states of the world. As such, provability for beliefs does not generate modal literals. *Obligation rules* determine when and which obligations are in force. The conclusions generated by obligation rules are modalised with obligation. Finally, *outcome rules* establish the possible outcomes of an agent depending on the particular context. Apart from obligation rules, outcome rules are used to derive conclusions for all modes representing possible types of outcomes: desires, goals, intentions, and social intentions.

Following ideas given in [10], rules can gain more expressiveness when a *preference operator* $\odot$ is used: an expression like $a \odot b$ means that if a is possible, then a is the first choice and b is the second one; if $\neg a$ holds, then the first choice is not attainable and b is the actual choice. This operator is used to build chains of preferences, called $\odot$-*expressions*. The formation rules for $\odot$-expressions are: (i.) every literal is an $\odot$-expression, (ii.) if A is an $\odot$-expression and b is a literal then $A \odot b$ is an $\odot$-expression. In addition we stipulate that $\odot$ obeys to the following properties: (i.) $a \odot (b \odot c) = (a \odot b) \odot c$ (associativity); (ii.) $\bigodot_{i=1}^{n} a_i = (\bigodot_{i=1}^{k-1} a_i) \odot (\bigodot_{i=k+1}^{n} a_i)$ where exists j such that $a_j = a_k$ and $j < k$ (duplication and contraction on the right). $\odot$-expressions are given by the agent designer, or obtained through *construction rules* based on the particular logic [10].

In this paper we exploit the classical definition of *defeasible rule* in DL [11]. A defeasible rule is an expression $r : A(r) \Rightarrow_{\Box} C(r)$, where

[1] The reading of the modal operators is B for *belief*, O for *obligation*, D for *desire*, I for *intention* and SI for *social intention*.

1. $r \in \text{Lbl}$ is the name of the rule;
2. $A(r) = \{a_1, \ldots, a_n\}$ with $a_i \in \text{Lit} \cup \text{ModLit}$ is the set of the premises (or the *antecedent*) of the rule;
3. $\Box \in \{\mathsf{B}, \mathsf{O}, \mathsf{U}\}$ represents the *mode* of the rule (from now on, we omit the subscript B in rules for beliefs, i.e., $\Rightarrow$ is used as a shortcut for $\Rightarrow_{\mathsf{B}}$);
4. $C(r)$ is the *consequent* (or *head*) of the rule, which is a single literal if $\Box = \mathsf{B}$, or an $\odot$-expression otherwise[2].

We use the following abbreviations on sets of rules: $R^{\Box}$ ($R^{\Box}[q]$) denotes all rules of mode $\Box$ (with consequent q), and $R[q] = \bigcup_{\Box \in \{\mathsf{B},\mathsf{O},\mathsf{U}\}} R^{\Box}[q]$. $R[q,i]$ denotes the set of rules whose head is $\odot_{j=1}^{n} c_j$ and $c_i = q$, with $1 \leq i \leq n$.

Most of the terminology defined so far appears in [6], where an extension of DL with modal operators is introduced to differentiate modal and factual rules. However, labelling the rules of DL produces nothing more but a simple treatment of the modalities, thus two interaction strategies between modal operators are analysed.

Rule Conversions. It is sometimes meaningful to use rules for a modality X as they were for another modality Y, i.e., to convert one type of conclusions into a different one. For example, if 'a car industry has the purpose of assembling perfectly working cars' and 'it is known that in every working car there is a working engine', then 'a car industry has also the purpose of assembling working engines in every car produced'. Formally, we define an asymmetric binary relation $\text{Convert} \subseteq \text{MOD} \times \text{MOD}$ such that $\text{Convert}(X,Y)$ means 'a rule of mode X can be used also to produce conclusions of mode Y'. This intuitively corresponds to the following logical schema:

$$\frac{Ya_1, \ldots, Ya_n \quad a_1, \ldots, a_n \Rightarrow_X b}{Yb} \ \text{Convert}(X,Y).$$

In our framework obligations and goal-like attitudes cannot change what the agent believes or how she perceives the world, thus we only consider conversion with mode for belief as the first element of the relation (i.e., $\text{Convert}(\mathsf{B}, X)$ with $X \in \{\mathsf{O}, \mathsf{D}, \mathsf{G}, \mathsf{I}, \mathsf{SI}\}$).

Conflict-Detection/Resolution. It is crucial to identify criteria for detecting and solving conflicts between different modalities. Formally, we define an asymmetric binary relation $\text{Conflict} \subseteq \text{MOD} \times \text{MOD}$ such that $\text{Conflict}(X,Y)$ means 'modes X and Y are in conflict and mode X prevails over Y'. Consider the following theory:

$$F = \{\, sunny\_day,\ school\_day \,\},$$
$$R = \{\, r_1 : Sunny\_day \Rightarrow_{\mathsf{U}} go\_outside,\ r_2 : school\_day \Rightarrow_{\mathsf{O}} \neg go\_outside \,\}.$$

Even if there is a sunny day, a responsible parent would not go outside and play with her kid but will bring him to school; this behaviour is captured by $\text{Conflict}(\mathsf{O}, \mathsf{SI})$, which means that the rule that forbids to go outside prevents the agent from obtaining the (social) intention of going outside, and the parent will not derive the (social) intention.

In our framework, we consider conflicts between beliefs and intentions, beliefs and social intentions, and obligations and social intentions. In other words, we have:

[2] It is worth noting that modal literals can occur only in the antecedent of rules: the reason is that the rules are used to derive modal conclusions and we do not conceptually need to iterate modalities. The motivation of a single literal as a consequent for belief rules is dictated by the intended reading of the belief rules, where these rules are used to describe the environment.

- Conflict(B,I), Conflict(B,SI) meaning that the agents are realistic (cf. [9]), and
- Conflict(O,SI) meaning that the agents are social (cf. [6]).

Observation 1. Convert *and* Conflict *relations behave differently in our framework than the usual deployed in the literature [6]. Typically, there is a bijective correspondence between a* mode *and the* type of rule *"representing" it. For example, there are rules with mode* O *to derive obligations, or rules with mode* I *to derive intentions. This is not the case in our logic where outcome rules are used to derive conclusions for all goal-like attitudes. Thus, we can have* Conflict(O,SI) *exhibiting the sociality of the agent but not* Conflict(O,U) *since desires and obligation do not attack each other.*

There are two applications of the *superiority relation*: the first considers rules of the same mode; the latter compares rule of different mode. Given $r \in R^X$ and $s \in R^Y$, notice that $r > s$ iff r converts X into Y, or s converts Y into X, i.e., the superiority relation is used when rules, each with a different mode, are used to produce complementary conclusions of the same mode. Consider the following theory with Convert(B,G):

$$
\begin{aligned}
F &= \{\, go\_to\_Rome, parent\_anniversary, August \,\},\\
R &= \{\, r_1 : go\_to\_Rome \Rightarrow_{\mathsf{B}} go\_to\_Italy\\
&\qquad r_2 : parent\_anniversary \Rightarrow_{\mathsf{U}} go\_to\_Rome\\
&\qquad r_3 : August \Rightarrow_{\mathsf{U}} \neg go\_to\_Italy \,\},\\
> &= \{\, (r_1, r_3) \,\}.
\end{aligned}
$$

Typically, I have the goal not to go to Italy in August since the weather is too hot and it is too crowded. However, it is my parents' anniversary and they are going to celebrate it this August in Rome, which is the capital of Italy. Nonetheless, I have the goal to go to Italy for my parents' wedding anniversary, since I am a good son. Here, the superiority applies because we use r_1 through a conversion from belief to goal.

2.2 Inferential Mechanism

A *proof* P of *length* n is a finite sequence $P(1),\ldots,P(n)$ of *tagged literals* of the type $+\partial_X q$ and $-\partial_X q$, where $X \in \mathrm{MOD}$. The proof conditions below define the logical meaning of such tagged literals. As a conventional notation, $P(1..i)$ denotes the initial part of the sequence P of length i. Given a defeasible theory D, $+\partial_X q$ means that q is defeasibly provable in D with the mode X, and $-\partial_X q$ that it has been proved in D that q is not defeasibly provable in D with the mode X. As usual, we use $D \vdash \pm\partial_\Box l$ iff there is a proof P in D such that $P(n) = \pm\partial_\Box l$ for an index n.

In order to characterise the notions of provability for beliefs ($\pm\partial_{\mathsf{B}}$), obligations ($\pm\partial_{\mathsf{O}}$), desires ($\pm\partial_{\mathsf{D}}$), goals ($\pm\partial_{\mathsf{G}}$), intentions ($\pm\partial_{\mathsf{I}}$) and social intentions ($\pm\partial_{\mathsf{SI}}$), it is essential to define when a rule is *applicable* or *discarded*. To this end, the preliminary notion of when a rule is *body-applicable/discarded* must be introduced, stating that each literal in the body of the rule must be proved/rejected with the suitable mode.

Definition 1. *Let* P *be a proof and* $\Box \in \{\mathsf{O},\mathsf{D},\mathsf{G},\mathsf{I},\mathsf{SI}\}$. *A rule* $r \in R$ *is* body-applicable *(at step* $n+1$*) iff for all* $a_i \in A(r)$*:*

1. *if* $a_i = \Box l$ *then* $+\partial_\Box l \in P(1..n)$,

2. *if* $a_i = \neg\Box l$ *then* $-\partial_\Box l \in P(1..n)$,
3. *if* $a_i = l \in \text{Lit}$ *then* $+\partial l \in P(1..n)$.

A rule $r \in R$ *is* body-discarded *(at step* $n+1$*) iff there is* $a_i \in A(r)$ *such that*

1. $a_i = \Box l$ *and* $-\partial_\Box l \in P(1..n)$, *or*
2. $a_i = \neg\Box l$ *and* $+\partial_\Box l \in P(1..n)$, *or*
3. $a_i = l \in \text{Lit}$ *and* $-\partial l \in P(1..n)$.

As already stated, belief rules allow us to derive literals with different modes. The applicability mechanism must take into account this constraint.

Definition 2. *Let P be a proof. A rule* $r \in R$ *is 1.* Conv-applicable*, 2.* Conv-discarded *(at step* $n+1$*) for X iff*

1. $r \in R^{\mathsf{B}}$, $A(r) \neq \emptyset$ *and for all* $a \in A(r)$, $+\partial_X a \in P(1..n)$;
2. $r \notin R^{\mathsf{B}}$ *or* $A(r) = \emptyset$ *or* $\exists a \in A(r)$, $-\partial_X a \in P(1..n)$.

Let us consider the following theory

$$F = \{a, b, \mathsf{O}c\}, \quad R = \{r_1 : a \Rightarrow_{\mathsf{O}} b, r_2 : b, c \Rightarrow d\},$$

r_1 is applicable, while r_2 is not since c is not proved as a belief. Instead, r_2 is *Conv-applicable* in the condition for $\pm\partial_{\mathsf{O}}$, since $\mathsf{O}c$ is a fact and r_1 proves $\mathsf{O}b$.

The notion of applicability gives guidelines on how to consider the next element in a given chain. Since a rule for belief cannot generate reparative chains but only single literals, we can conclude that the applicability condition for belief collapses into body-applicability. The same happens to desires, where we also consider the Convert relation. For obligations, each element before the current one must be a violated obligation. A literal is a candidate to be a goal only if none of the previous elements in the chain have been proved as a goal. For intentions, the elements of the chain must pass the wishful thinking filter, while social intentions are also constrained not to violate any norm.

Definition 3. *Given a proof P,* $r \in R[q,i]$ *is* applicable *(at index i and step* $n+1$*) for*

1. B *iff* $r \in R^{\mathsf{B}}$ *and is body-applicable.*
2. O *iff either: (2.1.1)* $r \in R^{\mathsf{O}}$ *and is body-applicable, (2.1.2)* $\forall c_k \in C(r), k < i$, $+\partial_{\mathsf{O}} c_k \in P(1..n)$ *and* $-\partial c_k \in P(1..n)$*, or (2.2) r is Conv-applicable.*
3. D *iff either: (3.1)* $r \in R^{\mathsf{U}}$ *and is body-applicable, or (3.2) Conv-applicable.*
4. X, $X \in \{\mathsf{G}, \mathsf{I}, \mathsf{SI}\}$ *iff either: (4.1.1)* $r \in R^{\mathsf{U}}$ *and is body-applicable, (4.1.2)* $\forall c_k \in C(r), k < i$, $+\partial_Y \sim c_k \in P(1..n)$ *for some Y such that* Conflict(Y, X) *and* $-\partial_X c_k \in P(1..n)$ *or (4.2) r is Conv-applicable.*
   For G *there are no conflicts; for* I *we have* Conflict(B, I)*, and for* SI *we have* Conflict$(\mathsf{B}, \mathsf{SI})$ *and* Conflict$(\mathsf{O}, \mathsf{SI})$.

Conditions to establish that a rule is discarded correspond to the constructive failure to prove that the same rule is applicable, and follow the principle of *strong negation*.[3]

[3] The strong negation principle is closely related to the function that simplifies a formula by moving all negations to an inner most position in the resulting formula, and replaces the positive tags with the respective negative tags, and the other way around [18, 7].

We can now describe the proof conditions for the various modal operators; we start with those for desires:

$+\partial_{\mathsf{D}}$: If $P(n+1) = +\partial_{\mathsf{D}} q$ then
(1) $\mathsf{D}q \in \mathrm{F}$ or
(2) (2.1) $\neg \mathsf{D}q \notin \mathrm{F}$ and
(2.2) $\exists r \in R[q,i]$: r is applicable for D and
(2.3) $\forall s \in R[\sim q, j]$ either
(2.3.1) s is discarded for D, or
(2.3.2) $s \not> r$.

We say that a *desire* is each element in a chain of an outcome rule for which there is no stronger argument for the opposite desire. The proof conditions for $+\partial_X$, with $X \in \{\mathsf{B}, \mathsf{O}, \mathsf{G}, \mathsf{I}, \mathsf{SI}\}$ are as follows:

$+\partial_X$: If $P(n+1) = +\partial_X q$ then
(1) $Xq \in \mathrm{F}$ or
(2) (2.1) $\neg Yq \notin \mathrm{F}$ for $Y = X$ or $\mathrm{Convert}(Y,X)$ and
(2.2) $\exists r \in R[q,i]$: r is applicable for X and
(2.3) $\forall s \in R^Y[\sim q, j]$ either
(2.3.1) s is discarded for Y, or
(2.3.2) $\exists t \in R^T[q,k]$: t is applicable for T and either
(2.3.2.1) $t > s$ if $Y = T$, $\mathrm{Convert}(Y,T)$, or $\mathrm{Convert}(T,Y)$; or
(2.3.2.2) $\mathrm{Conflict}(T,Y)$.

To show that a literal q is defeasibly provable with modality X we have two choices: (1) modal literal Xq is a fact; or (2) we need to argue using the defeasible part of D. In this case, we require that a complementary literal (of the same modality, or of a conflictual modality) does not appear in the set of facts (2.1), and that there must be an applicable rule for q for mode X (2.2). Moreover, each possible attack brought by a rule s for $\sim q$ has to be either discarded (3.1), or successfully counterattacked by another stronger rule t for q (2.3.2). We recall that the superiority relation combines rules of the same mode, rules with different modes that produce complementary conclusion of the same mode through conversion (both considered in clause (2.3.2.1)), and conflictual modalities (clause 2.3.2.2). Obviously, if $\Box = \mathsf{B}$, then the proof conditions reduce to those of classical defeasible logic [11].

Again, the negative counterparts ($-\partial_{\mathsf{D}}$ and $-\partial_X$) are derived by strong negation applied to conditions for $+\partial_{\mathsf{D}}$ and $+\partial_X$, respectively. As an example, consider the theory:

$$F = \{\neg b_1, \mathsf{O}\neg b_2, \mathsf{SI}b_4\} \quad R = \{r: \Rightarrow_{\mathsf{U}} b_1 \odot b_2 \odot b_3 \odot b_4\}.$$

Then r is trivially applicable for D and $+\partial_{\mathsf{D}} b_i$ holds, for $1 \leq i \leq 4$. Moreover, we have $+\partial_{\mathsf{G}} b_1$ and r is discarded for G after b_1. Since $+\partial \neg b_1$, $-\partial_{\mathsf{I}} b_1$ holds (as well as $-\partial_{\mathsf{SI}} b_1$); the rule is applicable for I and b_2, and we are able to prove $+\partial_{\mathsf{I}} b_2$, thus the rule becomes discarded for I after b_2. Given that $\mathsf{O}\neg b_2$ is a fact, r is discarded for SI and b_2 and $-\partial_{\mathsf{SI}} b_2$ is proved, which in turn makes the rule applicable for SI at b_3, proving $+\partial_{\mathsf{SI}} b_3$. As we have argued before, this would make the rule discarded for b_4. Nevertheless, b_4 is still provable with mode SI (in this case because it is a fact, but in other theories there could be more rules with b_4 in their head).

The logic resulting enjoys properties describing the appropriate behaviour of the modal operators.

Definition 4. *A defeasible theory $D = (\mathrm{F}, R, >)$ is* consistent *iff $>$ is acyclic and* F *does not contain pairs of complementary (modal) literals, that is pairs like (i.) l and $\sim l$, (ii.) $\Box l$ and $\neg\Box l$, $\Box \in$ MOD, and (iii.) $\Box l$ and $\Box\sim l$, $\Box \in \mathrm{MOD} \setminus \{\mathsf{D}\}$.*

Proposition 1. *Let D be a consistent modal defeasible theory. For any literal l, it is not possible to have both*

1. *$D \vdash +\partial_\Box l$ and $D \vdash -\partial_\Box l$ with $\Box \in$ MOD;*
2. *$D \vdash +\partial_\Box l$ and $D \vdash +\partial_\Box \sim l$ with $\Box \in \mathrm{MOD} \setminus \{\mathsf{D}\}$.*

Moreover, given $\Box \in \mathrm{MOD} \setminus \{\mathsf{D}\}$, then:

3. *if $D \vdash +\partial_\Box l$, then $D \vdash -\partial_\Box \sim l$.*

Proof. *Omissis.*

3 Algorithmic Results

We now present the algorithms apt to compute the *extension* of a *finite* defeasible theory, i.e., with finite set of facts and rules, in order to bind the complexity of the logic introduced in the previous sections. The algorithms are inspired by ideas of [19, 20].

For the sake of clarity, from now on $\blacksquare$ denotes a generic mode in MOD, $\Diamond$ a generic mode in $\mathrm{MOD} \setminus \{\mathsf{B}\}$, and $\Box$ a fixed mode chosen in $\blacksquare$. Moreover, we will treat literals $\Box l$ and l as synonyms whenever $\Box = \mathsf{B}$. To accommodate the Convert relation to the algorithms, we denote with $R^{\mathsf{B},\Diamond}$ the set of belief rules with non-empty body that can be used for a conversion to mode $\Diamond$. Furthermore, for each literal l, $l_\blacksquare$ is the set (initially empty) such that $\pm\Box \in l_\blacksquare$ iff $D \vdash \pm\partial_\Box l$. Given a modal defeasible theory D, a set of rules R, and a rule $r \in R^\Box[l]$, we expand $>$ by incorporating the Conflict relation to ease the computation. Then, we define: (i.) $r_{sup} = \{s \in R : (s,r) \in >\}$ and $r_{inf} = \{s \in R : (r,s) \in >\}$ for any $r \in R$; (ii.) HB_D as the set of literals such that the literal or its complement appears in D, where 'appears' means that it is a sub-formula of a modal literal occurring in D; (iii.) the modal Herbrand Base of D as $HB = \{\Box l |\ \Box \in \mathrm{MOD}, l \in HB_D\}$. Accordingly, the extension of a defeasible theory is defined as follows.

Definition 5. *Given a modal defeasible theory D, the* defeasible extension *of D is defined as $E(D) = (+\partial_\Box, -\partial_\Box)$ where $\pm\partial_\Box = \{l \in HB_D : D \vdash \pm\partial_\Box l\}$ with $\Box \in$ MOD. Two defeasible theories D and D' are* equivalent *whenever $E(D) = E(D')$.*

The next definition extends the concept of complement presented in Section 2 for modal literals and establishes the logical connection among proved and refuted literals.

Definition 6. *The* complement *of a given modal literal l, denoted by $\tilde{l}$, is:*

1. *if $l = \mathsf{D}m$, then $\tilde{l} = \{\neg\mathsf{D}m\}$;*
2. *if $l = \Box m$, then $\tilde{l} = \{\neg\Box m, \Box\sim m\}$, with $\Box \in \{\mathsf{O}, \mathsf{G}, \mathsf{I}, \mathsf{SI}\}$;*
3. *if $l = \neg\Box m$, then $\tilde{l} = \{\Box m\}$.*

Truncation and *removal* are two syntactical operations on the consequent of rules.

Definition 7. *Let $c_1 = a_1 \odot \cdots \odot a_{i-1}$ and $c_2 = a_{i+1} \odot \cdots \odot a_n$ be two (possibly empty) $\odot$-expressions such that a_i does not occur in them, and $c = c_1 \odot a_i \odot c_2$ is an $\odot$-expression. Let r be a rule with form $A(r) \Rightarrow_X c$. We define the*

- truncation *of the consequent c at a_i as $A(r) \Rightarrow_X c!a_i = A(r) \Rightarrow_X c_1 \odot a_i$;*
- removal *of a_i from the consequent c as $A(r) \Rightarrow_X c \ominus a_i = A(r) \Rightarrow_X c_1 \odot c_2$.*

Given $\Box \in \mathrm{MOD}$, the sets $\pm\partial_\Box$ denote the global sets of defeasible conclusions (i.e., the set of literals for which condition $\pm\partial_\Box$ holds), while $\partial^{\pm}_\Box$ are the corresponding temporary sets. Moreover, to simplify the calculus we do not operate on outcome rules: for each rule $r \in R^{\mathsf{U}}$ we create instead a new rule for all the other goal-like modes (resp. r^{D}, r^{G}, r^{I}, and r^{SI}). Consequently, we will use expressions like "the intention rule" as a shortcut for "the clone of outcome rule used to derive intentions".

The idea of all algorithms is to use the operations of truncation and elimination in order to obtain, step after step, a simpler but equivalent theory. Indeed, proving a literal does not give just local information about the element itself, but reveals which rules will be applicable, discarded, or reduced in their head or tail.

Observation 2. *Assume that, at a given step, the algorithm proves l. At the next step,*

1. *the applicability of any rule r with l in its antecedent $A(r)$ does not depend on l any longer. Accordingly, we can safely remove l from $A(r)$.*
2. *Any rule s where $\widetilde{l}$ is in its antecedent $A(s)$ is discarded. Consequently, any superiority tuple involving this rule is now meaningless and can be removed from the superiority relation as well.*
3. *We can shorten chains by exploiting conditions of Definition 3. For example, if $l = \mathsf{O}m$, we can truncate chains for obligations at $\sim m$ and eliminate $\sim m$.*

Algorithm 1 DEFEASIBLEEXTENSION is the core algorithm to compute the extension of a defeasible theory. The first part (lines 1–4) sets up the data structure needed for the computation. Lines 5–8 handle facts as immediately provable literals. The main idea of the algorithm is to check whether there are rules whose body is empty. Since defeasible rules can have $\odot$-expressions as their head, the literal we are interested in is the first element of the $\odot$-expression (loop **for** at lines 16–33 and **if** condition at line 17). Such rules are clearly applicable and they can produce conclusions with the right modality. However, before asserting that the first element of the conclusion is provable, we have to check whether there are no rules for the complement (again with the appropriate mode), otherwise such rules for the complement must be weaker than the applicable rules. This information is stored in R_{infd} inspired by the technique of [20]. If no rule stronger than the current one exists, the complementary conclusion must be refuted by condition (2.3) of $-\partial_\Box$ (line 25). A straightforward consequence of $D \vdash -\partial_\Box l$ is that literal l is also refutable in D with any modality conflicting with $\Box$ (line 26). Notice that this reasoning does not hold for desires: since we can have $\mathsf{D}l$ and $\mathsf{D}\sim l$ at the same time, when $\Box = \mathsf{D}$ the algorithm invokes procedure 2 PROVED (line 23).

The next step is to check whether there exist rules for the complement of the literal with the same (or conflicting) mode. The rules for the complement should not be defeated by an applicable rule, i.e., they should not be in R_{infd}. If all these rules are defeated by r (line 27), then conditions for deriving $+\partial_\Box$ are satisfied. If a literal is

Algorithm 1 . DEFEASIBLEEXTENSION

1: $+\partial_{\blacksquare}, \partial^{+}_{\blacksquare} \leftarrow \emptyset$; $-\partial_{\blacksquare}, \partial^{-}_{\blacksquare} \leftarrow \emptyset$
2: $R \leftarrow R \cup \{r^{\Box} : A(r) \Rightarrow_{\Box} C(r) | r \in R^{\mathsf{U}}\} \setminus R^{\mathsf{U}}$, with $\Box \in \{\mathsf{D}, \mathsf{G}, \mathsf{I}, \mathsf{SI}\}$
3: $R^{\mathsf{B},\Diamond} \leftarrow \{r^{\Diamond} : \Diamond a_1, \ldots, \Diamond a_n \Rightarrow_{\Diamond} C(r) | r \in R^{\mathsf{B}}, A(r) \neq \emptyset, A(r) \subseteq \mathrm{Lit}, a_i \in A(r)\}$
4: $> \leftarrow > \cup \{(r^{\Diamond}, s^{\Diamond}) | r^{\Diamond}, s^{\Diamond} \in R^{\mathsf{B},\Diamond}, r > s\} \cup \{(r,s) | r \in R^{\blacksquare}, s \in R^{\Diamond} \cup R^{\mathsf{B},\Diamond}, \mathrm{Conflict}(\blacksquare, \Diamond)\}$
5: **for** $l \in F$ **do**
6: **if** $l = \Box m$ **then** PROVED$(m, \Box)$
7: **if** $l = \neg\Box m \wedge \Box \neq \mathsf{D}$ **then** REFUTED$(m, \Box)$
8: **end for**
9: $+\partial_{\blacksquare} \leftarrow +\partial_{\blacksquare} \cup \partial^{+}_{\blacksquare}$; $-\partial_{\blacksquare} \leftarrow -\partial_{\blacksquare} \cup \partial^{-}_{\blacksquare}$
10: $R_{infd} \leftarrow \emptyset$
11: **repeat**
12: $\partial^{+}_{\blacksquare} \leftarrow \emptyset$; $\partial^{-}_{\blacksquare} \leftarrow \emptyset$
13: **for** $\Box l \in HB$ **do**
14: **if** $R^{\Box}[l] \cup R^{\mathsf{B},\Box}[l] = \emptyset$ **then** REFUTED$(l, \Box)$
15: **end for**
16: **for** $r \in R^{\Box} \cup R^{\mathsf{B},\Box}$ **do**
17: **if** $A(r) = \emptyset$ **then**
18: $r_{inf} \leftarrow \{r \in R : (r,s) \in >, s \in R\}$; $r_{sup} \leftarrow \{s \in R : (s,r) \in >\}$
19: $R_{infd} \leftarrow R_{infd} \cup r_{inf}$
20: Let l be the first literal of $C(r)$ in HB
21: **if** $r_{sup} = \emptyset$ **then**
22: **if** $\Box = \mathsf{D}$ **then**
23: PROVED(m, D)
24: **else**
25: REFUTED$(\sim l, \Box)$
26: REFUTED$(\sim l, \Diamond)$ for $\Diamond$ s.t. Conflict$(\Box, \Diamond)$
27: **if** $R^{\Box}[\sim l] \cup R^{\mathsf{B},\Box}[\sim l] \cup R^{\blacksquare}[\sim l] \setminus R_{infd} \subseteq r_{inf}$, for $\blacksquare$ s.t. Conflict$(\blacksquare, \Box)$ **then**
28: PROVED$(m, \Box)$
29: **end if**
30: **end if**
31: **end if**
32: **end if**
33: **end for**
34: $\partial^{+}_{\blacksquare} \leftarrow \partial^{+}_{\blacksquare} \setminus +\partial_{\blacksquare}$; $\partial^{-}_{\blacksquare} \leftarrow \partial^{-}_{\blacksquare} \setminus -\partial_{\blacksquare}$
35: $+\partial_{\blacksquare} \leftarrow +\partial_{\blacksquare} \cup \partial^{+}_{\blacksquare}$; $-\partial_{\blacksquare} \leftarrow -\partial_{\blacksquare} \cup \partial^{-}_{\blacksquare}$
36: **until** $\partial^{+}_{\blacksquare} = \emptyset$ and $\partial^{-}_{\blacksquare} = \emptyset$
37: **return** $(+\partial_{\blacksquare}, -\partial_{\blacksquare})$

assessed to be provable (with the appropriate modality) the algorithm calls procedure 2 PROVED, otherwise the procedure 3 REFUTED is invoked. The algorithm finally returns the extension of the input theory when no modifications are done on sets $\partial^{\pm}_{\blacksquare}$.

Algorithm 2 PROVED is invoked when literal l is proved with modality $\Box$. The computation starts by updating the relative positive extension set for modality $\Box$ and the local information on literal l (line 2); $\Box l$ is then removed from HB at line 3. Proposition 1 Part *3.* defines the modalities literal $\sim l$ can be refuted with (**if** condition at line 4). Lines 5 to 7 modifies the sets of rules R and $R^{\mathsf{B},\Box}$, and the superiority relation accordingly to ideas of Observation 2.

Depending on the modality $\Box$ of l, we have to perform some specific operations on chains (condition **switch** at lines 8–27). Entering into the detail of each **case** would be redundant without giving more information than conditions of a rule being applicable or

Algorithm 2 . PROVED

1: **procedure** PROVED($l \in$ Lit, $\Box \in$ MOD)
2: $\quad \partial^+_\Box \leftarrow \partial^+_\Box \cup \{l\}$; $l_\blacksquare \leftarrow l_\blacksquare \cup \{+\Box\}$
3: $\quad HB \leftarrow HB \setminus \{\Box l\}$
4: $\quad$ **if** $\Box \neq \mathsf{D}$ **then** REFUTED($\sim l$, $\Box$)
5: $\quad R \leftarrow \{r : A(r) \setminus \{\Box l, \neg\Box\sim l\} \hookrightarrow C(r) |\ r \in R,\ A(r) \cap \widetilde{\Box} l = \emptyset\}$
6: $\quad R^{\mathsf{B},\Box} \leftarrow \{r : A(r) \setminus \{\Box l\} \hookrightarrow C(r) | r \in R^{\mathsf{B},\Box},\ A(r) \cap \widetilde{\Box} l = \emptyset\}$
7: $\quad > \leftarrow > \setminus \{(r,s), (s,r) \in >\ |\ A(r) \cap \widetilde{\Box} l \neq \emptyset\}$
8: $\quad$ **switch** ($\Box$)
9: $\quad\quad$ **case** B**:**
10: $\quad\quad\quad R^X \leftarrow \{A(r) \Rightarrow_X C(r)!l |\ r \in R^X[l,n]\}$ with $X \in \{\mathsf{O}, \mathsf{I}\}$
11: $\quad\quad\quad R^X \leftarrow \{A(r) \Rightarrow_X C(r) \ominus \sim l |\ r \in R^X[\sim l,n]\}$ with $X \in \{\mathsf{I}, \mathsf{SI}\}$
12: $\quad\quad\quad$ **if** $+\mathsf{O} \in \sim l_\blacksquare$ **then** $R^{\mathsf{O}} \leftarrow \{A(r) \Rightarrow_{\mathsf{O}} C(r) \ominus \sim l |\ r \in R^{\mathsf{O}}[\sim l,n]\}$
13: $\quad\quad\quad$ **if** $-\mathsf{O} \in \sim l_\blacksquare$ **then** $R^{\mathsf{SI}} \leftarrow \{A(r) \Rightarrow_{\mathsf{SI}} C(r)!l |\ r \in R^{\mathsf{SI}}[l,n]\}$
14: $\quad\quad$ **case** O**:**
15: $\quad\quad\quad R^{\mathsf{O}} \leftarrow \{A(r) \Rightarrow_{\mathsf{O}} C(r)!\sim l \ominus \sim l |\ r \in R^{\mathsf{O}}[\sim l,n]\}$
16: $\quad\quad\quad R^{\mathsf{SI}} \leftarrow \{A(r) \Rightarrow_{\mathsf{SI}} C(r) \ominus \sim l |\ r \in R^{\mathsf{SI}}[\sim l,n]\}$
17: $\quad\quad\quad$ **if** $-\mathsf{B} \in l_\blacksquare$ **then** $R^{\mathsf{O}} \leftarrow \{A(r) \Rightarrow_{\mathsf{O}} C(r) \ominus l |\ r \in R^{\mathsf{O}}[l,n]\}$
18: $\quad\quad\quad$ **if** $-\mathsf{B} \in \sim l_\blacksquare$ **then** $R^{\mathsf{SI}} \leftarrow \{A(r) \Rightarrow_{\mathsf{SI}} C(r)!l |\ r \in R^{\mathsf{SI}}[l,n]\}$
19: $\quad\quad$ **case** D**:**
20: $\quad\quad\quad$ **if** $+\mathsf{D} \in \sim l_\blacksquare$ **then**
21: $\quad\quad\quad\quad R^{\mathsf{G}} \leftarrow \{A(r) \Rightarrow_{\mathsf{G}} C(r)!l \ominus l |\ r \in R^{\mathsf{G}}[l,n]\}$
22: $\quad\quad\quad\quad R^{\mathsf{G}} \leftarrow \{A(r) \Rightarrow_{\mathsf{G}} C(r)!\sim l \ominus \sim l |\ r \in R^{\mathsf{G}}[\sim l,n]\}$
23: $\quad\quad\quad$ **end if**
24: $\quad\quad$ **otherwise:**
25: $\quad\quad\quad R^{\Box} \leftarrow \{A(r) \Rightarrow_{\Box} C(r)!l |\ r \in R^{\Box}[l,n]\}$
26: $\quad\quad\quad R^{\Box} \leftarrow \{A(r) \Rightarrow_{\Box} C(r) \ominus \sim l |\ r \in R^{\Box}[\sim l,n]\}$
27: $\quad$ **end switch**
28: **end procedure**

discarded in Section 2. Therefore, we propose one significative example by considering the scenario where l has been proved as a belief (**case** at lines 9–13). Here, chains of obligation (resp. intention) rules can be truncated after l since are discarded for all following elements (line 10). Analogously, condition (4.1.2) of Definition 3 allows us to eliminate $\sim l$ from intention and social intention rules (line 11). If $+\partial_{\mathsf{O}} \sim l$ has been already proved, then we eliminate $\sim l$ since it represents a violated obligation. *Vice versa*, if $-\partial_{\mathsf{O}} \sim l$ is the case, then each element after l cannot be a social intention (resp. **if** conditions at lines 12 and 13).

Algorithm 3 REFUTED performs all necessary operations in case literal l is refuted with modality $\Box$. The initialisation steps at lines 2–6 follow the same schema exploited at lines 2–7 of Algorithm 2 PROVED. Again, the operations to be performed on chains vary according to the current mode $\Box$ (**switch** at lines 7–19). For example, if $\Box = \mathsf{B}$ (lines 8–11), then applicability condition (4.1.2) for $\pm\partial_{\mathsf{I}}$ cannot be satisfied for any literal after $\sim l$ in chains for intentions, and such chains can be truncated at $\sim l$. Furthermore, if the algorithm has already proven $+\partial_{\mathsf{O}} l$, then l represents a violated obligation. Thus, l can be removed from all chains for obligations. If instead $-\partial_{\mathsf{O}} l$ holds, then the elements after $\sim l$ in chains for social intentions do not satisfy applicability condition (4.1.2) of $\pm\partial_{\mathsf{SI}}$ for $\sim l$, and the algorithm removes them.

We conclude by showing the computational properties of the algorithms proposed.

Algorithm 3 . REFUTED

1: **procedure** REFUTED($l \in$ Lit, $\Box \in$ MOD)
2: $\partial_{\Box}^{-} \leftarrow \partial_{\Box}^{-} \cup \{l\}$; $l_{\blacksquare} \leftarrow l_{\blacksquare} \cup \{-\Box\}$
3: $HB \leftarrow HB \setminus \{\Box l\}$
4: $R \leftarrow \{r : A(r) \setminus \{\neg\Box l\} \hookrightarrow C(r) | \, r \in R, \, \Box l \notin A(r)\}$
5: $R^{\mathsf{B},\Box} \leftarrow R^{\mathsf{B},\Box} \setminus \{r \in R^{\mathsf{B},\Box} : \Box l \in A(r)\}$
6: $> \leftarrow > \setminus \{(r,s),(s,r) \in > | \Box l \in A(r)\}$
7: **switch** ($\Box$)
8: **case** B**:**
9: $R^{\mathsf{I}} \leftarrow \{A(r) \Rightarrow_{\mathsf{I}} C(r)!\sim l | r \in R^{\mathsf{I}}[\sim l,n]\}$
10: **if** $+\mathsf{O} \in l_{\blacksquare}$ **then** $R^{\mathsf{O}} \leftarrow \{A(r) \Rightarrow_{\mathsf{O}} C(r) \ominus l | r \in R^{\mathsf{O}}[l,n]\}$
11: **if** $-\mathsf{O} \in l_{\blacksquare}$ **then** $R^{\mathsf{SI}} \leftarrow \{A(r) \Rightarrow_{\mathsf{SI}} C(r)!\sim l | r \in R^{\mathsf{SI}}[\sim l,n]\}$
12: **case** O**:**
13: $R^{\mathsf{O}} \leftarrow \{A(r) \Rightarrow_{\mathsf{O}} C(r)!l \ominus l | r \in R^{\mathsf{O}}[l,n]\}$
14: **if** $-\mathsf{B} \in l_{\blacksquare}$ **then** $R^{\mathsf{SI}} \leftarrow \{A(r) \Rightarrow_{\mathsf{SI}} C(r)!\sim l | r \in R^{\mathsf{SI}}[\sim l,n]\}$
15: **case** D**:**
16: $R^{X} \leftarrow \{A(r) \Rightarrow_{X} C(r) \ominus l | r \in R^{X}[l,n]\}$ with $X \in \{\mathsf{D},\mathsf{G}\}$
17: **otherwise:**
18: $R^{\Box} \leftarrow \{A(r) \Rightarrow_{\Box} C(r) \ominus l | r \in R^{\Box}[l,n]\}$
19: **end switch**
20: **end procedure**

Theorem 1. *Algorithm 1* DEFEASIBLEEXTENSION *terminates and its computational complexity is* $O(|R| * |HB|)$.

Proof Sketch. Termination is ensured since at every iteration either no modification occurs and line 36 ends the computation, or a literal is removed from HB. This set is finite, since the sets of facts and rules are finite, thus the process eventually empties HB. This bounds also the complexity to the number of rules and literals in HB, since each modal literal is processed once, and every time we scan the set of rules.

Theorem 2. *Algorithm 1* DEFEASIBLEEXTENSION *is sound and complete.*

Proof. *Omissis.* A similar result can be found in [21].

4 Conclusions and Related Work

This article provides a fresh characterisation for motivational states as the concepts of desire, goal, intention, and social intention obtained through a deliberative process based on various types of preferences among desired outcomes. In this sense, this contribution has strong connections with [5–7] but presents significant improvements in at least two respects. First, while in those works the agent deliberation is simply the result of the derivation of mental states from *precisely* the corresponding rules of the logic, here the proof theory is much more aligned with the BDI intuition, according to which intentions and goals are the results of desire manipulation. This allowed us to encode this idea within a logical language and a proof theory, by exploiting the different interaction patterns between the basic mental states, as well as the derived ones. In this perspective, our framework is significantly more expressive than the one in BOID [9], which uses different rules to derive the corresponding mental states and proposes simple criteria to solve conflicts between rule types.

Second, the framework proposes a rich language expressing two orthogonal concepts of preference among motivational attitudes. One is encoded within $\odot$ sequences, which state reparative orders among homogeneous mental states or motivations, and which are contextual. The second type of preference is encoded via the superiority relation between rules which can work locally, as well as via the Conflict relation. The interplay between these two preference mechanisms can help us to isolate different and complex ways for deriving mental states, but the resulting logical machinery is still computationally tractable.

Since the preferences allow us to determine what preferred outcomes can be chosen by an agent (in a specific scenario) when previous goals in $\odot$-sequences are not (or no longer) feasible, our logic in fact provides an abstract semantics for several types of goal and intention reconsideration. Intention reconsideration was expected to play a crucial role in the BDI paradigm [22, 2] since intentions obey the law of inertia and resist retraction or revision, but they can be reconsidered when new relevant information comes in [22]. Despite that, the problem of revising intentions in BDI frameworks has received little attention. A very sophisticated exception is [23], where revisiting intentions mainly depends on the dynamics of beliefs but the process is incorporated in a very complex framework for reasoning about mental states. Recently, [24] discussed how to revise the commitments to planned activities because of mutually conflicting intentions, which interestingly has connections with our work. How to employ our logic to give a semantics for intention reconsideration is not the main goal of the paper and is left to future work.

Finally, we recall that our focus was on the deliberation aspects of an agent determining what are her mental states in a given moment. That is the case, our investigation is orthogonal with respect to the works of [25–27].

Acknowledgements. NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

References

1. Bratman, M.E., Israel, D.J., Pollack, M.E.: Plans and resource-bounded practical reasoning. Computational Intelligence 4, 349–355 (1988)
2. Cohen, P.R., Levesque, H.J.: Intention is choice with commitment. Artificial Intelligence 42(2-3), 213–261 (1990)
3. Rao, A.S., Georgeff, M.P.: Modeling rational agents within a BDI-architecture. In: Allen, J.F., Fikes, R., Sandewall, E. (eds.) KR, pp. 473–484. Kaufmann, M. (1991)
4. Rao, A.S., Georgeff, M.P.: Decision procedures for bdi logics. Journal of Logic and Computation 8(3), 293–342 (1998)
5. Dastani, M., Governatori, G., Rotolo, A., van der Torre, L.: Programming cognitive agents in defeasible logic. In: Sutcliffe, G., Voronkov, A. (eds.) LPAR 2005. LNCS (LNAI), vol. 3835, pp. 621–636. Springer, Heidelberg (2005)
6. Governatori, G., Rotolo, A.: BIO logical agents: Norms, beliefs, intentions in defeasible logic. Journal of Autonomous Agents and Multi-Agent Systems 17(1), 36–69 (2008)

7. Governatori, G., Padmanabhan, V., Rotolo, A., Sattar, A.: A defeasible logic for modelling policy-based intentions and motivational attitudes. Logic Journal of the IGPL 17(3), 227–265 (2009)
8. Thomason, R.H.: Desires and defaults: A framework for planning with inferred goals. In: Cohn, A.G., Giunchiglia, F., Selman, B. (eds.) KR 2000. Morgan Kaufmann (2000)
9. Broersen, J., Dastani, M., Hulstijn, J., van der Torre, L.: Goal generation in the BOID architecture. Cognitive Science Quarterly 2(3-4), 428–447 (2002)
10. Governatori, G., Rotolo, A.: Logic of violations: A Gentzen system for reasoning with contrary-to-duty obligations. Australasian Journal of Logic 4, 193–215 (2006)
11. Antoniou, G., Billington, D., Governatori, G., Maher, M.J.: Representation results for defeasible logic. ACM Transactions on Computational Logic 2(2), 255–287 (2001)
12. Governatori, G.: Representing business contracts in RuleML. International Journal of Cooperative Information Systems 14(2-3), 181–216 (2005)
13. Kravari, K., Papatheodorou, C., Antoniou, G., Bassiliades, N.: Reasoning and proofing services for semantic web agents. In: Walsh, T. (ed.) IJCAI 2011, pp. 2662–2667 (2011)
14. Governatori, G., Sadiq, S.: The journey to business process compliance. In: Handbook of Research on BPM, pp. 426–454. IGI Global (2008)
15. Lam, H.-P., Governatori, G.: The making of SPINdle. In: Governatori, G., Hall, J., Paschke, A. (eds.) RuleML 2009. LNCS, vol. 5858, pp. 315–322. Springer, Heidelberg (2009)
16. Bassiliades, N., Antoniou, G., Vlahavas, I.: A defeasible logic reasoner for the semantic web. Int. J. Semantic Web Inf. Syst. 2(1), 1–41 (2006)
17. Tachmazidis, I., Antoniou, G., Flouris, G., Kotoulas, S., McCluskey, L.: Large-scale parallel stratified defeasible reasoning. In: De Raedt, L., Bessière, C., Dubois, D., Doherty, P., Frasconi, P., Heintz, F., Lucas, P.J.F. (eds.) ECAI 2012, pp. 738–743. IOS Press (2012)
18. Antoniou, G., Billington, D., Governatori, G., Maher, M.J., Rock, A.: A family of defeasible reasoning logics and its implementation. In: ECAI 2000, pp. 459–463 (2000)
19. Maher, M.J.: Propositional defeasible logic has linear complexity. Theory and Practice of Logic Programming 1(6), 691–711 (2001)
20. Lam, H.-P., Governatori, G.: What Are the Necessity Rules in Defeasible Reasoning? In: Delgrande, J., Faber, W. (eds.) LPNMR 2011. LNCS, vol. 6645, pp. 187–192. Springer, Heidelberg (2011)
21. Governatori, G., Olivieri, F., Rotolo, A., Scannapieco, S.: Computing strong and weak permissions in defeasible logic. CoRR abs/1212.0079 (2012)
22. Bratman, M.E.: Intentions, Plans and Practical Reason. Harvard University Press (1987)
23. van der Hoek, W., Jamroga, W., Wooldridge, M.: Towards a theory of intention revision. Synthese 155(2), 265–290 (2007)
24. Shapiro, S., Sardina, S., Thangarajah, J., Cavedon, L., Padgham, L.: Revising conflicting intention sets in BDI agents. In: AAMAS 2012, pp. 1081–1088. IFAAMS (2012)
25. Winikoff, M., Padgham, L., Harland, J., Thangarajah, J.: Declarative & procedural goals in intelligent agent systems. In: Fensel, D., Giunchiglia, F., McGuinness, D.L., Williams, M. (eds.) KR 2002, pp. 470–481. Morgan Kaufmann (2002)
26. Dastani, M., van Riemsdijk, M.B., Meyer, J.J.C.: Goal types in agent programming. In: Nakashima, H., Wellman, M.P., Weiss, G., Stone, P. (eds.) AAMAS, pp. 1285–1287. ACM (2006)
27. van Riemsdijk, M.B., Dastani, M., Meyer, J.J.C.: Goals in conflict: Semantic foundations of goals in agent programming. Journal of Autonomous Agents and Multi-Agent Systems 18(3), 471–500 (2009)

Computing Temporal Defeasible Logic

Guido Governatori[1] and Antonino Rotolo[2]

[1] NICTA, Australia
[2] CIRSFID and DSG, University of Bologna, Italy

Abstract. We investigate the complexity of temporal defeasible logic, and propose an efficient algorithm to compute the extension of any theory. The logic and algorithm are discussed in regard to modeling deadlines and normative retroactivity.

1 Introduction

Defeasible Logic (DL) [25,3] is historically the first of a family of approaches based on the idea of logic programming without negation as failure. DL is a simple, efficient but flexible non-monotonic formalism capable of dealing with many different intuitions of non-monotonic reasoning [5]. The logic was designed to be easily implementable right from the beginning, unlike most other approaches, and has a linear complexity [23]. Recent implementations include DR-Prolog [2], DELORES [22], DR-DEVICE [8] and SPINdle [20].

DL proved to be modular and flexible. In particular, propositional DL has been recently extended in two different directions.

In a first case, different types of modal operators (capturing notions such as directed and undirected deontic statements, actions, beliefs, and intentions) have been embedded within propositional DL [12,13]. The result was a number of logics having still linear complexity and being able to model the deliberation of cognitive agents and their interplay with normative systems. Some implementations have been recently developed for such logics [9,19,20].

DL has been also extended to capture temporal aspects of normative reasoning [17,14] several specific phenomena, such as legal positions [17] and modifications [11,14], deadlines [10]. Although Temporal Defeasible Logic (TDL) proved to be sufficiently expressive for those purposes, and many variants of it have been proposed accordingly, no systematic investigation on the proof-theoretic and computational properties of TDL has been so far carried out. This paper is a first step in this direction. In particular, we will present a conceptually expressive variant of TDL, which is computationally feasible. This variant is able to represent different types of deadlines and capture backward causation and normative retroactivity. We will prove that it is always computationally feasible to compute the complete set of consequences of any given TDL theory, thus preserving the nice computational features of standard DL.

The layout of the paper is as follows. Section 2 describes the variant of TDL studied in the paper. Section 3 investigates the complexity of the logic, and proposes an algorithm to compute the extension of any theory. Section 4 discusses the approach and considers how to model retroactivity. A section on related work ends the paper.

L. Morgenstern et al. (Eds.): RuleML 2013, LNCS 8035, pp. 114–128, 2013.

2 Temporal Defeasible Logic (TDL)

Consider a set $\mathscr{P}$ of atomic propositional literals. The language of TDL is based on the concept of *temporalised literal*, which is an expression such as l^t (or its negation, $\neg l^t$), where l is a literal and t is an element of a discrete totally ordered set $\mathscr{T}$ of instants of time $\{t_1, t_2, \dots\}$: l^t intuitively means that l holds at time t. Given a temporalised literal l the complement $\sim l$ is $\neg p^t$ if $l = p^t$, and p^t if $l = \neg p^t$.

A *rule* is an expression $lbl: A \hookrightarrow^x m$, where lbl is a unique label of the rule, A is a (finite, possibly empty) set of temporalised literals, $\hookrightarrow \in \{\rightarrow, \Rightarrow, \rightsquigarrow\}$, m is a temporalised literal and x is either π or τ signaling whether we have a *persistent* or *transient* rule. *Strict rules*, marked by the arrow $\rightarrow$, support indisputable conclusions whenever their antecedents, too, are indisputable. *Defeasible rules*, marked by $\Rightarrow$, can be defeated by contrary evidence. *Defeaters*, marked by $\rightsquigarrow$, cannot lead to any conclusion but are used to defeat some defeasible rules by producing evidence to the contrary. A *persistent* rule is a rule whose conclusion holds at all instants of time after the conclusion has been derived, unless interrupting events occur; a *transient* rule establishes its conclusion only for a specific instant of time. Thus $ex_1: p^5 \Rightarrow^\pi q^6$ means that if p holds at 5, then q defeasibly holds at time 6 and continues to hold after 6 until some event or fact overrides it. The rule $ex_2: p^5 \Rightarrow^\tau q^6$ means that, if p holds at 5, then q defeasibly holds at time 6 but we do not know whether it will persist after 6.

Example 1. Let us consider two examples from the normative domain. Suppose Bob buys today a car. If so, he will have the obligation to pay for that. Clearly, this obligation holds today and will persist afterwards (until Bob pays for the car). Hence:

$$r_1: Bob\_Car^{today} \Rightarrow^\pi \mathrm{OBL}\, Pay^{today}$$

Suppose now that Bob enters a Catholic church. As long as he is in the church, he is obliged not speak loudly. In this case, the obligation does not persist, but only holds insofar as Bob is in the church:

$$r_2: Bob\_Church^x \Rightarrow^\tau \mathrm{OBL} \neg Speak\_Loudly^x$$

We use some abbreviations. Given a rule r and a set R of rules, $A(r)$ denotes the antecedent of r while $C(r)$ denotes its consequent; R^π denotes the set of persistent rules in R, R^τ the set of transient rules in R; $R[\psi]$ is the subset of R of rules with consequent ψ. R_s, R_{sd} and R_{dft} are respectively the set of strict rules, the set of strict and defeasible rules, and the set of defeaters in R.

Note that we assume that persistent literals, once they are blocked at a certain time t, no longer hold after t unless other applicable rules reinstate them. Indeed, this is the usual rendering of the common sense law of inertia [28]. Accordingly, we assume that defeaters are only transient: if a persistent defeasible conclusion is blocked at t by a defeater, there is no need that such a defeater is still applicable after t.

Example 2. Consider again the rule:

$$r_1: Bob\_Car^{today} \Rightarrow^\pi \mathrm{OBL}\, Pay^{today}$$

If Bob pays, then he is no longer obliged to pay:

$$r_3: Pay^x, \mathrm{OBL}Pay^x \rightsquigarrow^{\tau} \neg \mathrm{OBL}Pay^{x+1}$$

There is no reason here to say that this obligation to pay will sooner or later come back: if this happens, it implies that something new has happened in the meantime that has triggered again the obligation. Hence, there are good reasons to state that defeaters have not to be persistent, at least in the domain of normative reasoning.

Remark 1. Impeding reasons work as depicted in the example above in many contexts other than the normative domain: again, this is the usual rendering of the common sense law of inertia mentioned above. However, there are examples where a reason r to block the persistence of a literal l (for example, state of affairs) is effective with respect to this literal only as long as r's effect persists too: when such an r no longer applies, then l comes back by inertia without any further positive reason. Consider the example of Bob, a person who sleeps because he under the effect of of a drug. Bob is asleep as long as the drug is active: when it is no longer active, Bob wakes up. Intuitively, a simple way to model this scenario in TDL is as follows:

$$r_4: \neg Drug\_Active^x \Rightarrow^{\pi} Awake^x$$
$$r_5: Drug\_Active^y \rightsquigarrow^{\mathbf{z}} \neg Awake^y$$

Should r_5 be persistent or transient? If persistent, it would block the inertia of *Awake* indefinitely, but this is not required: in this context, we can simply say that a precondition for Bob to be awake is that no drug is active on him. So we can capture the scenario by stating that $z = \tau$.

There are three kinds of features in TDL: facts, rules, and a superiority relation among rules. Facts are indisputable statements, represented by temporalised literals. The superiority relation ($\succ$) provides information about the relative strength of rules, i.e., about which rules can overrule which other rules. A knowledge base that consists of these items is called a TDL theory.

Definition 1. *A* TDL theory *is a structure $(F, R, \succ)$, where F is a finite set of facts, R is a finite set of rules and $\succ$ is an acyclic binary relation over R.*

Given a rule r, $r_{inf} = \{s : r \succ s\}$ and $r_{sup} = \{s : s \succ r\}$.

TDL adopts a constructive inference mechanism based on tagged conclusions. Proof tags indicate the strength of conclusions. The strength depends on whether conclusions are indisputable (the tag is Δ), namely obtained by using facts and strict rules, or they are defeasible (the tag is ∂).

Provability is defined below grounded on the concept of a derivation (or proof) in a TDL theory D.

Definition 2. *Given a TDL theory D, a* proof *P from D is a finite sequence of tagged temporalised literals such that:*

1. *Each tag is one of the following: $+\Delta$, $-\Delta$, $+\partial$, $-\partial$;*
2. *The proof conditions* Definite Provability *and* Defeasible Provability *given below are satisfied by the sequence P.*

Given a proof P we use $P(n)$ to denote the n-th element of the sequence, and $P[1..n]$ denotes the first n elements of P. The meaning of the proof tags is as follows:

- $+\Delta p^{t_p}$: we have a definite derivation of p holding at time t_p;
- $-\Delta p^{t_p}$: we can show that it is not possible to have a definite derivation of p holding at time t_p;
- $+\partial p^{t_p}$: we have a defeasible derivation of p holding at time t_p; in other terms p is provable at time t_p.
- $-\partial p^{t_p}$: we can show that it is not possible to have a defeasible derivation of p holding at time t_p; thus p is refuted at time t_p.

Note that the inference conditions for negative proof tags ($-\Delta$ and $-\partial$) are derived from the inference conditions for the corresponding positive proof tags by applying the Principle of Strong Negation introduced by [4]: the strong negation of a formula is closely related to the function that simplifies it by moving all negations to an innermost position in the resulting formula and replace the positive tags with the respective negative tags and viceversa.

Definite Provability

If $P(n+1) = +\Delta p^{t_p}$, then
1) $p^{t_p} \in F$; or
2) $\exists r \in R_s^x[p^{t'_p}]$ such that
$\forall a^{t_a} \in A(r)$: $+\Delta a^{t_a} \in P[1..n]$.

If $P(n+1) = -\Delta p^{t_p}$, then
1) $p^{t_p} \notin F$; and
2) $\forall r \in R_s^x[p^{t'_p}]$,
$\exists a^{t_a} \in A(r)$: $-\Delta a^{t_a} \in P[1..n]$.

where:

(a) if $x = \pi$, then $t'_p \leq t_p$;
(b) if $x = \tau$, then $t'_p = t_p$.

If the rule used to derive p is transient (if $x = \tau$), the above conditions are the standard ones for definite proofs in DL, which are just monotonic derivations using forward chaining. If the rule is persistent ($x = \pi$), p can be obtained at t_p or, by persistence, at any time t'_p before t_p. Finally, notice that facts lead to strict conclusions, but are taken not to be persistent. This condition can be relaxed or we could introduce persistent and well as transient facts.

Defeasible Provability

If $P(n+1) = +\partial p^{t_p}$, then
1) $+\Delta p^{t_p} \in P[1..n]$ or
2) $\exists r \in R_d^x[p^{t'_p}]$ such that
1) $-\Delta {\sim} p^{t_{\sim p}} \in P[1..n]$ and
2) $\forall a^{t_a} \in A(r)$: $+\partial a^{t_a} \in P[1..n]$, and
3) $\forall s \in R[{\sim} p^{t_{\sim p}}]$ either
1) $\exists b^{t_b} \in A(s), -\partial b^{t_b} \in P[1..n]$ or
2) $\exists w \in R[p^{t_{\sim p}}]$ such that $\forall c^{t_c} \in A(w)$:
$+\partial c^{t_c} \in P[1..n]$ and $w \succ s$.

If $P(n+1) = -\partial p^{t_p}$, then
1) $-\Delta p^{t_p} \in P[1..n]$ and
2) $\forall r \in R_d^x[p^{t'_p}]$ either
1) $+\Delta {\sim} p^{t_{\sim p}} \in P[1..n]$ or
2) $\forall r \in R_d^x[p^{t'_p}]$ or
3) $\exists s \in R[{\sim} p^{t_{\sim p}}]$ such that
1) $\forall b^{t_b} \in A(s), +\partial b^{t_b} \in P[1..n]$ and
2) $\forall w \in R[p^{t_{\sim p}}]$ $\exists c^{t_c} \in A(w)$:
$-\partial c^{t_c} \in P[1..n]$ or $w \not\succ s$.

where

(1) if $x = \pi$, then $t'_p \leq t_{\sim p} \leq t_p$;
(2) if $x = \tau$, then $t'_p = t_{\sim p} = t_p$.

Defeasible derivations run in three phases. In the first phase we put forward a supported reason (rule) for the conclusion we want to prove. Then in the second phase we consider all possible (actual and not) reasons against the desired conclusion. Finally, in the last phase, we have to rebut all the counterarguments. This can be done in two ways: we can show that some of the premises of a counterargument do not obtain, or we can show that the argument is weaker than an argument in favour of the conclusion. If $x = \tau$, the above conditions are essentially those for defeasible derivations in DL. If $x = \pi$, a proof for p can be obtained by using a persistent rule which leads to p holding at t_p or at any time t'_p before t_p. In addition, for every instant of time between t'_p and t_p, p should not be terminated. This requires that all possible attacks were not triggered (clause 2.3.1) or are weaker than some reasons in favour of the persistence of p (clause 2.3.2).

As usual we write $D \vdash \pm\#l^t$ ($\# \in \{\Delta, \partial\}$) if there is a derivation P for $\pm\#l^t$ in D. Given a rule r and a derivation P, we will say that r is applicable in P, or simply applicable when the derivation is clear from the context, iff $\forall a^t \in A(r), +\partial a^t \in P[1..n]$ for some $n \in \mathbb{N}$.

Example 3. Consider the following theory, where $t_1 < t_2 < t_3 < t_4$:

$$\begin{array}{l}(F = \{a^{t_1},\ b^{t_3},\ c^{t_3},\ d^{t_4}\},\\ \quad R = \{r_1\colon a^{t_1} \Rightarrow^{\pi} e^{t_1}, \quad r_2\colon b^{t_3} \Rightarrow^{\pi} \neg e^{t_3}, \quad r_3\colon c^{t_3} \rightsquigarrow^{\tau} e^{t_3}, \quad r_4\colon d^{t_4} \Rightarrow^{\tau} \neg e^{t_4}\},\\ \quad \succ = \{r_3 \succ r_2,\ r_1 \succ r_4\})\end{array}$$

At time t_1, r_1 is the only applicable rule; hence, we derive $+\partial e^{t_1}$. At time t_2 no rule is applicable, and the only derivation permitted is the one of $+\partial e^{t_2}$ by persistence. At time t_3 both r_2 and r_3 are applicable, but r_4 is not. If r_2 prevailed, then it would terminate e. However, it is rebutted by r_3, so we derive $+\partial e^{t_3}$. At time t_4, rule r_4 is applicable, thus we derive $+\partial \neg e^{t_4}$ and $-\partial e^{t_4}$, which means that r_4 terminates e. Even if r_4 is weaker than r_1, the latter is not applicable at t_4, thus it does not offer any support to maintain e.

Proposition 1. *Let D be a TDL theory.*

1. *It is not possible that both $D \vdash +\#p^t$ and $D \vdash -\#p^t$ (for $\# \in \{\Delta, \partial\}$);*
2. *if $D \vdash +\partial p^t$ and $D \vdash +\partial \sim p^t$, then $D \vdash +\Delta p^t$ and $D \vdash +\Delta \sim p^t$.*

The proof of Proposition 1 is a simple extension of the ones for Theorems 1 and 2 in [13] and is omitted for space reasons. Proposition 1 shows the soundness of TDL: it is not possible to derive a tagged conclusion and its opposite, and that we cannot defeasibly prove both p and its complementary unless the definite part of the theory proves them; this means that inconsistency can be derived only if the monotonic part of the theory is inconsistent, and even in this case the logic does not collapse to the trivial extensions (i.e., everything is provable).

Definition 3. *Let HB_D be the Herbrand Base for a TDL theory D. The extension of D (denoted by E^D) is the 4-tuple $(\Delta^+, \Delta^-, \partial^+, \partial^-)$, where*

$$\partial^+ = \{p^t \mid p \in HB_D, D \vdash +\partial p^t, t \in \mathscr{T}\}, \quad \partial^- = \{p^t \mid p \in HB_D, D \vdash -\partial p^t, t \in \mathscr{T}\},$$
$$\Delta^+ = \{p^t \mid p \in HB_D, D \vdash +\Delta p^t, t \in \mathscr{T}\}, \quad \Delta^- = \{p^t \mid p \in HB_D, D \vdash -\Delta p^t, t \in \mathscr{T}\}.$$

We will refer to Δ^+ and Δ^- as the definite extension, to ∂^+ and ∂^- as the defeasible extension, to Δ^+ and ∂^+ as the positive extension, and to Δ^- and ∂^- as the negative extension.

3 Computing Consequences in TDL

In this section we present an algorithm to compute the extension of a TDL theory. We show that the time complexity of the algorithm remains computationally feasible and it is proportional to the size of the theory (number of symbols in a theory). Following the idea of [23] the algorithm is based on a series of (theory) transformations that allow us (1) to assert whether a literal is provable or not (and the strength of its derivation) and (2) to progressively reduce and simplify a theory.

At this point we have to make precise what we mean for two theories to be equivalent.

Definition 4. *Two theories D and D' are* equivalent *if and only if they have the same extension, namely $D \equiv D'$ iff $E^D = E^{D'}$.*

The key ideas behind the approach depends on the following properties that allow us to transform theories into 'simpler' ones (i.e., either with rules with less elements in their antecedent or with less rules).

Proposition 2. *Let $D = (\emptyset, R, \succ)$ be a temporal defeasible theory*[1]*. Then*

1. *If $r\colon \rightarrow^{\pi} p^t \in R$ then $D \vdash +\Delta p^{t'}$, $\forall t' \geq t$.*
2. *If $r\colon \rightarrow^{\tau} p^t \in R$ then $D \vdash +\Delta p^t$.*
3. *If $D \vdash +\Delta p^t$, then $D \cup \{r\colon a_1^{t_1}, \ldots, a_n^{t_n}, p^t \rightarrow c^{t_c}\} \equiv D \cup \{r\colon a_1^{t_1}, \ldots, a_n^{t_n} \rightarrow c^{t_c}\}$*
4. *Let $t_{min} = \min\{t\colon R_s[p^t] \neq \emptyset\}$, then $D \vdash -\Delta p^{t'}$ $\forall t' < t_{min}$.*
5. *Let $t_\pi = \min\{t\colon R_s^{\pi}[p^t]\}$,[2] $t_\tau^- = \min\{t\colon R_s^{\tau}[p^t] \neq \emptyset, t > t_{min}\}$, $t_\tau^+ = \max\{t\colon R_s^{\tau}[p^t] \neq \emptyset, t < t_\pi\}$. Then $D \vdash -\Delta p^{t'}$, $\forall t'$ such that either $t_{min} < t' < t_\tau^-$ or $t_\tau^+ < t' < t_\pi$.*
6. *If $D \vdash -\Delta p^t$, then $D \cup \{r\colon a_1^{t_1}, \ldots, a_n^{t_n}, p^t \rightarrow c^{t_c}\} \equiv D \cup \{r\colon a_1^{t_1}, \ldots, a_n^{t_n}, p^t \Rightarrow c^{t_c}\}$*

The properties in Proposition 2 give us an alternative characterisation of the proof theory for definite conclusions. Properties 1 and 2 provide conditions under which we are allowed to positively assert a definite conclusion. Properties 3 presents the condition for the transformation. In fact it tells us that we can remove already proved temporal literals from the body of other (strict) rules without affecting the conclusions we can derive from a theory. Thus we can use Properties 1–3 to transform a theory into a simpler but equivalent theory.

Properties 4–6, on the other hand, are related to transformations and conditions to derive negative definite conclusions. Properties 4, establishes that for all times less than the minimum time appearing associated to a literal in the conclusion of a rules, we fail to prove literal, thus we can assert the literal with $-\Delta$. The meaning of Property 5 is similar to that of Property 4, but this time, the focus is for the times in an interval; in particular it states that for all instants between two persistent rules (for a literal p) such that no other rule is in that interval (provided that the smallest instant for a persistent rule for the same literal is after the last of the right extreme of the interval), we can assert that the literal is provable with $-\Delta$. Similarly for an interval between the smallest instant

[1] Notice that the restriction to theories with an empty set of facts is not a limitation. For any theory $D = (F, R, \succ)$ there is an equivalent theory $D' = (\emptyset, R \cup \{\rightarrow^{\tau} a^t\colon a^t \in F\}, \succ)$ where the set of facts is empty.

[2] If $\{t\colon R_s^{\pi}[p^t]\} = \emptyset$, then we assume t_π to the be the minimum of the temporal order $\mathscr{T}$.

for a persistent rule and a transient rule. Finally, Property 6 tells us that after we have assessed that a strict rule is not applicable for the computation of definite conclusions, we can transform the rule into a defeasible rule.

Proposition 3. *Let D be a theory in TDL. For $y \in \{\pi, \tau\}$:*
(1) If $D \vdash +\partial p^t$, then $D \cup \{r: p_1^{t_1}, \ldots, p_n^{t_n}, p^t \Rightarrow^y q\} \equiv D \cup \{r: p_1^{t_1}, \ldots, p_n^{t_n} \Rightarrow^y q\}$.
(2) if $D \vdash -\partial p^t$, then $D \cup \{r: p_1^{t_1}, \ldots, p_n^{t_n}, p^t \Rightarrow^y q\} \equiv D$.

The meaning of (1) in the above proposition is that once we have established that a temporalised literal is positively provable we can remove it from the body of rules without affecting the set of conclusions we can derive from the theory. Similarly (2) states that we can safely remove rules from a theory when one of the elements in the body of the rules is negatively provable.

To give conditions under which we can conclude that a temporal literal is defeasibly provable, we use the notion of *inferiorly defeated* set introduced in [21]. The set of inferiorly defeated rules, R_{infd}, is thus defined $R_{infd} = \{r: \exists s, s \succ r, \text{ and } A(s) = \emptyset\}$.

Proposition 4. *Let D be a TDL theory. If $r: \Rightarrow^x p^t \in R$ and $R[\sim p^t] \subseteq R_{infd}$, then $D \vdash +\partial p^t$ and $D \vdash -\partial \sim p^t$.*

This proposition gives us the main criterion to assess whether we can defeasibly prove a literal.

We compute the extension of a TDL theory in two phases:

1. in the first phase we compute the definite extension;
2. in the second phase we use the theory from the first phase to generate the theory to be used to compute the defeasible extension.

3.1 Computing the Definite Extension

Before giving the algorithm to compute the extension we have to introduce some auxiliary notation to refer to the data structures needed for the algorithms that compute the extension.

Definition 5. *Let D be a TDL theory and H_D be the set of literals in D. For each $a \in H_D$ we have the following sets:*

- $ptimes(a) = \{t: \exists r \in R^{\pi}[a^t]\}$;
- $ttimes(a) = \{t: \exists r \in R^{\tau}[a^t]\}$;
- $times(a) = ptimes(a) \cup ttimes(a)$.

We use the same abbreviations as those of Section 2. Thus, e.g., $ptimes_s(a)$ is the set of instants associated to $R_s[a]$.

In the presentation of the algorithms we use intervals to give a compact representation for sets of contiguous instants. We will use both proper intervals, i.e., intervals with both start and end time, and punctual intervals, i.e., intervals corresponding to singletons. We will use $[t, t']$ for a proper interval and $[t]$ for a punctual interval.

Definition 6. *Given an interval I we say that $t^* \in I$ iff*
(1) if $I = [t,t']$, $t < t'$ and $t \leq t^* < t'$ or
(2) if $I = [t]$ and $t^* = t$.

We adopt a compact but isomorphic representation for the extensions, namely, elements of extensions are now pairs (l,I) were l is a literal and I is an interval; thus (l,I) corresponds to the set of temporalised literals l^t such that $t \in I$.

Algorithm 1. *ComputeDefinite*

Input: A Temporal Defeasible Theory $D = (F,R,\succ)$
$\Delta^+ \leftarrow \{(a,[t]) : a^t \in F\}$
$\Delta^- \leftarrow \{(a,[0,\infty]) : R_s[a] = \emptyset)\}$
$H_D \leftarrow H_D \setminus \{a : R_s[a] = \emptyset\}$
while $\Delta_+ \neq \emptyset$ **do**
  $\Delta_+ \leftarrow \emptyset$
  for $a \in H_D$ **do**
    $R_s \leftarrow R_s \setminus \{s : a^{t'} \in A(s), t' < \min(ptimes(a) \text{ and } t' \notin ttimes(a))\}$
    if $times(a) = \emptyset$ **then**
      $H_d \leftarrow H_d \setminus \{a\}$
  for $r \in R_s$ **do**
    if $A(r) = \emptyset$ ***and*** $C(r) = a^t$ **then**
      if $r \in R^\pi$ **then**
        $\Delta_+ \leftarrow \Delta_+ \cup \{(a,[t,\infty])\}$
        $R_s \leftarrow R_s \setminus \{s : C(s) = a^{t'}, t' \geq t\}$
        $R_s \leftarrow \{r : A(r) \setminus \{l^{t'}\} \rightarrow C(s) : r \in R_s, t' \geq t\}$
      if $r \in R^\tau$ **then**
        $\Delta_+ \leftarrow \Delta_+ \cup \{(a,[t])\}$
        $R_s \leftarrow R_s \setminus \{s : C(s) = a^t\}$
        $R_s \leftarrow \{r : A(r) \setminus \{l^t\} \rightarrow C(s) : r \in R_s\}$
  $\Delta^+ \leftarrow \Delta^+ \cup \Delta_+$
for $a \in H_D$ **do**
  $t_{left} \leftarrow \max(0, \{t : (a,[t',t]) \in \Delta^-\})$
  $t_{right} \leftarrow \min(0, \{t : (a,[t,\infty]) \in \Delta^+\})$
  $T_\tau \leftarrow \{t : (a,[t]) \in \Delta^+, t_{left} < t < t_{right}\}$
  $T_\tau \leftarrow T_\tau \cup \{t : C(r) = a^t, r \in R_s^\tau[a], t_{left} < t < t_{right}\}$
  $T \leftarrow T_\tau \cup \{t : C(r) = a^t, r \in R_s^\pi[a], t_{left} < t < t_{right}\}$
  while $T_\tau \neq \emptyset$ **do**
    $t_{min} \leftarrow \min(T_\tau)$
    $T_\tau \leftarrow T_\tau \setminus \{t_{min}\}$
    if $t_{min} + 1 \notin T$ ***and*** $t_{min} + 1 < t_{right}$ **then**
      $\Delta^- \leftarrow \Delta^- \cup \{(a,[t_{min}])\}$

At each cycle the algorithm *ComputeDefinite* scans the set of literals in search of temporalised literals for which there are no rules for them. This happens in two cases: (i) there are no rules for a temporalised literal or (ii) all the persistent rules for the literal have a greater time. For each of such temporalised literals *ComputeDefinite* adds them to the negative definite extension of the theory, and removes all rules where at least one of these literals occurs.

Then *ComputeDefinite* scans the set of rules in search of rules with an empty body. In case of a positive match the algorithm adds the conclusion of the rule to the positive definite extension (with an open ended interval for a persistent rule and with a punctual interval otherwise). Finally the algorithm removes from the body of rules the temporalised literals matching the newly added conclusions.

We repeat the cycle until (1) there are no more literals to be examined, or (2) the set of strict rules is empty, or (3) no addition to the extension happened in the cycle.

Proposition 5. *ComputeDefinite is correct.*

Proof. The correctness of *ComputeDefinite* follows immediately from Proposition 2. The transformations in *ComputeDefinite* correspond to the properties listed in Proposition 2.

3.2 Computing the Defeasible Extension

We are now ready to give the algorithm that computes the defeasible extension of a theory. We first give some subroutines corresponding to theory transformations to be used in the main algorithm. ∂_+ and ∂_- are sets of accumulators for the conclusions proved in each cycle of the main routine.

The first algorithm we consider is concerned with literals to be tagged with $-\partial$.

Algorithm 2. *discard*(l,I)

Input: a literal l and an interval I

1 $\partial_- \leftarrow \partial_- \cup \{(l,I)\}$
2 $S \leftarrow \{s : l^t \in A(s), t \in I\}$
3 $R \leftarrow R \setminus S$
4 $\succ \leftarrow \succ \setminus \{(r,s),(s,r) : s \in S\}$
5 *persistence*(S)

The algorithm *discard* adds a literal to the negative defeasible extension and then removes rules for which we have already proved that some literal in the antecedent of the rules is not provable. The literal is parametrised by an interval. This means that the operation is performed for all instances of the literal temporalised with an instant in the interval. The transformation corresponding to it is justified by Proposition 3.(2). The algorithm further calls the subroutine *persistence* that updates the state of the extension of a theory.

Algorithm *proved* (Algorithm 3) concerns defeasible provable literals.

Algorithm 3. *proved*(l,r,I)

Input: a literal l, a rule r, and and interval I

1 $\partial_+ \leftarrow \partial_+ \cup \{(l,I)\}$
2 *discard*$(\sim l,I)$
3 **for** $s \in R$ **do**
4 **if** $l^t \in A(s)$ ***and*** $t \in I$ **then**
5 $A(s) \leftarrow A(s) \setminus \{l^t\}$
6 $R \leftarrow R \setminus \{r\}$

It first inserts a provable literal in the positive defeasible extension of the theory. Then *proved* calls *discard* with the complementary literal. The next step is to remove all the

instances of the literal temporalised with an instant in the interval I from the body of rules. Finally it removes the rule for the set of rules. The transformations implemented by this algorithm are justified by Propositions 3.(1), and 4.

Algorithm *persistence* updates the state of literals in the extension of a theory after we have removed rules we know cannot longer be fired (i.e., at least one literal in the antecedent of the rule is provable with $-\partial^x$).

Algorithm 4. *persistence*(S)

```
   Input: a set of rules S
1  for (l,[t,t']) ∈ ∂+ do
2      if s ∈ S and C(s) = ∼l^t' then
3          if t* = min{k ∈ ttimes(∼l): k > t'} then
4              ∂+ ← (∂+ \ {(l,[t,t'])}) ∪ {(l,[t,t*])}
5              proved(l,∅,[t',t*])
```

As we have seen in Section 2 a conclusion proved using a persistent rule persists until it is terminated by another (applicable) rule for the complement of the conclusion, thus an entry $(l,[t,t'])$ in ∂^+ means that l holds from t to $t'-1$. Hence, there is a rule for $\sim l^{t'-1}$. When we insert $(l,[t,t'])$ in ∂^+ we do not know if the rule for $\sim l^{t'-1}$ is applicable or not. The set S passed as parameter to the algorithm is the set of rules we have discovered to be no longer applicable. At this point we can update the entry for l in ∂^+, and we set it to t'', where t'' is the next instant for which we have a rule for $\sim l$. Consider, e.g., a theory where the rules for p and $\neg p$ are:

$$r\colon \Rightarrow^{\pi} p^1,$$
$$s\colon q^5 \Rightarrow^{\tau} \neg p^{10},$$
$$v\colon \Rightarrow^{\pi} \neg p^{15}.$$

Here, we can prove $+\partial p^t$ for $1 \leq t < 10$, no matter whether q is provable at 5 or not. Suppose we discover that $-\partial q^5$. Then we have to remove rule s. In the resulting theory can prove $+\partial p^t$ for $1 \leq p < 15$. Thus we can update the entry for l from $(l,[1,10])$ to $(l,[1,15])$.

For *ComputeDefeasible* lines 4–11 are essentially the same as the main loop in *ComputeDefinite*, lines 4–27 (with the difference that when we eliminate a rule we update the state of the extension instead of waiting to the end as we did for the definite extension).

From line 12 we search for rules with empty body. Suppose we have one of such rules, let us say a rule for l^t. If there are no stronger rules for the opposite, no matter what type of rule we have, we know that the complement of l, i.e., $\sim l$, cannot be proved at t. So we call *discard* with parameter $(\sim l,[t])$, and remove rules where $\sim l^t$ appears in the body. At this stage we still have to determine whether we can insert l in ∂^+ and the instant/interval associated to it. We have a few cases. The rule is a defeater. Defeaters cannot be used to prove conclusions, so in this case, all we can do is to insert all rules weaker than the defeater in the set of inferiorly defeated rules (line 24). If the rule is transient, then it can prove the conclusion only at t, and we have to see if there are transient rules for $\sim l^t$ or persistent rules for $\sim l^{t'}$ such that $t' \leq t$. If there are we have to wait to see if we can discard such rules. Otherwise, we can add $(l,[t])$ to ∂^+ and carry

Algorithm 5. *ComputeDefesible*

Input: A TDL theory $(F, R, \prec)$

1 $\partial^+ \leftarrow F$
2 $\partial^- \leftarrow \{(a,[t]) : \sim a^t \in F\}$
3 $R \leftarrow R \setminus \{r \in R : \sim C(r) \in F\}$
4 $\prec \leftarrow \prec \setminus \{(r,s),(r,s) : s \in R, \sim C(s) \in F\}$
5 **repeat**
6 $\quad \partial_+ \leftarrow \emptyset$
7 $\quad \partial_- \leftarrow \emptyset$
8 $\quad S = \{s \in R : \exists a^{t'} \in A(s) : t' < \min(ptimes(a)), t' \notin ttimes(a)\}$
9 $\quad R \leftarrow R \setminus S$
10 $\quad persistence(S)$
11 $\quad$ **for** $a \in H_D$ **do**
12 $\quad\quad$ **if** $R[l] = \emptyset$ **then**
13 $\quad\quad\quad discard(a,[0,\infty])$
14 $\quad\quad$ **if** $\exists r \in R[l] : A(r) = \emptyset$ ***and*** $C(r) = a^t$ **then**
15 $\quad\quad\quad$ **if** $r_{sup} = \emptyset$ **then**
16 $\quad\quad\quad\quad discard(\sim a,[t])$
17 $\quad\quad\quad\quad$ **if** $r \in R_d$ *and* $R[\sim l] - R_{infd} \subseteq r_{inf}$ **then**
18 $\quad\quad\quad\quad\quad$ **if** $r \in R_d^\tau$ **then**
19 $\quad\quad\quad\quad\quad\quad proved(a,[t])$
20 $\quad\quad\quad\quad\quad$ **else if** $r \in R_d^\pi[a]$ **then**
21 $\quad\quad\quad\quad\quad\quad t^* = \min(\{k \in times(\sim a)\} \cup \{\infty\})$
22 $\quad\quad\quad\quad\quad\quad proved(a,[t,t*])$
23 $\quad\quad\quad$ **else**
24 $\quad\quad\quad\quad R_{infd} \leftarrow R_{infd} \cup r_{inf}$
25 **until** $\partial_+ = \emptyset$ *and* $\partial_- = \emptyset$
26 $\partial^+ \leftarrow \partial^+ \cup \partial_+$
27 $\partial^- \leftarrow \partial^- \cup \partial_-$

over the related housekeeping. Finally, in the last case the rule is persistent, and, again the second part of the condition in line 15 holds, what we have to do in this case is to search for the minimum time greater or equal to t in the rules for $\sim l$, and we can include $(l,[t,t'])$ in ∂^+ and again perform the related housekeeping.

Proposition 6. *Let $D = (\emptyset, R, \succ)$ be a TDL theory such that there is no strict rule with empty body in R; then the transformation ComputeDefeasible(D) is correct.*

3.3 Computing the Extension

To compute the full extension of a TDL theory D we use the following series of transformation:

1. Let D' be the theory obtained from *ComputeDefinite*(D)
2. The defeasible extension is obtained from *ComputeDefeasible*(D').

We call the transformation *ComputeExtension.*

Theorem 1. *ComputeExtension is correct.*

Proof. The result follows from the correctness of the two steps (Propositions 5 and 6).

Theorem 2. *Given a TDL theory D, the extension of D can be computed in* $O(|R| * |H_D| * |\mathscr{T}_D|)$ *time, where* $\mathscr{T}_D$ *is the set of distinct instants in D.*

Proof. First of all we notice that for each literal a the set $R[a]$ can be implemented as an hash table with pointers to the rules, where each rule is implemented as an ordered list of pairs. The sets of *ptimes* and *ttimes* can be constructed as indexes: the information stored in them can be accessed (in the form required by *ComputeDefeasible*) in linear time.

For *ComputeDefeasible* we have a loop over the set of literals, and inside a loop over the set of rules. After a successful execution of the loop we reduce the complexity of the theory and keep trace of the literals and rules for which we have to repeat the cycle. Every time we remove a rule in the algorithm *ComputeDefeasible*, we call *persistence* to update the extension. For each cycle the number of time we call the procedure is bounded by the number of instants such that there is a particular literal with that instant in the head of a rule. The number of operations we are going to perform is bounded by the number of symbols in the theory (we either remove a rule or a literal from a rule). The total number of symbols in a theory is bounded by the product of the number of rules, the number of atoms and the distinct instant of time in the theory.

Thus, the complexity of *ComputeExtension* is $O(|R| * |H_D| * |\mathscr{T}_D|)$.

4 Discussion and Implementation

TDL is an extension of basic defeasible logic [3] for which [23] proved that the complexity is linear. The extension is twofold: on the syntactic side, literals are labelled with timestamps, on the conceptual side TDL introduces persistent and transient conclusions. The idea of persistent conclusions is that once a conclusion has been classified as persistent then it continues to hold until there are some reasons to terminate it. From a computational point of view, we can propagate persistent conclusions from one instant to the successive instant unless there are some reasons that prevent the propagation. Based on this intuition, if we restrict the language to rules with the form

$$a_1^{t_1}, \ldots, a_n^{t_n} \Rightarrow b^t$$

such that

$$\max(\{t_1, \ldots, t_n\}) \leq t$$

then we can devise the following procedure [18] to compute the extension of a theory.

- At time 0, consider the sub-theory restricted to the rules whose consequent is labelled by 0. Then use the algorithms given in [23] to compute the extension of the sub-theory at time 0.
- At time $n+1$, consider the extension at time n. Then for each positive conclusion (i.e., conclusion whose proof tag is $+\partial$) p_i:
  - introduce a rule $r_{p_i}^n$: $\Rightarrow^{\tau} p_i$
  - introduce an instance of the superiority relation $r_{p_i}^n \prec s$ for each s such that $C(s) = \sim p_i^{n+1}$;
  - remove p_i^n from the body of rules where it occurs;

For each negative conclusion q_j remove rules where q_j appears in the body. Compute the extension for the sub-theory restricted to the rules whose consequent is labelled with $n+1$.

Thus, we compute the extension of a theory D in the interval $[0,t]$ in $O(|D| * t)$-time (where $|D|$ is the number of instances of literals in it).

The above procedure applies an incremental swap over the time-line, and the problem with it is that it cannot look backward. Therefore the major limitation is that it cannot handle rules where the time of the conclusion precedes the time of some of its antecedents. This is a drawback, since rules of that form are able to capture interesting phenomena, such as backward causation and normative retroactivity. While it is debatable whether effects can precede the causes (a discussion is reported in [18]), normative retroactivity occurs indeed in law. This means that a norm is introduced at a particular time, but its normative effects must be considered at times preceding the validity. In fact, this happens, e.g., of taxation law, where it is possible to claim a tax benefit from a date in the past. Theories containing rules such as $\{a^{10} \Rightarrow^{\tau} b^{10}, b^{10} \Rightarrow^{\pi} c^{0}, c^{3} \Rightarrow^{\pi} a^{0}\}$ raise computational problems: clearly, it is not possible to handle the computation by sequentially running the algorithms of [23] for each time-slice. The procedure proposed in this paper handles these cases by keeping the complexity under control. An efficient Java implementation of TDL based on the algorithms presented in this paper is discussed in [27,26]. Initial experiments with the implementation confirm the scalability of the approach. The implementation has been tested on both synthetic theories (designed to test particular features of TDL) and concrete theories obtained from real life scenarios.

On the synthetic side, TDL and the proposed algorithms can account for compact encodings and efficient computations. For example consider the theories:

$$T_1\colon \Rightarrow^{\pi} p^0 \qquad\qquad T_2\colon \Rightarrow^{\tau} p^0, p^i \Rightarrow^{\tau} p^{i+1} \text{ for } 0 \leq i \leq 99$$

The two theories are equivalent (i.e., they generate the same extension) in the time interval $[0, 100]$, but, trivially, our algorithms compute the extension much more quickly when given the first theory as input than when the second theory is used as input. On the practical side, of particular interest is the formalisation in TDL of the Road Traffic Restriction Regulation of the Italian town of Piacenza [26,16].

5 Related Work

Typically there are two mainstream approaches to reasoning with and about time. A point based approach, as in the present paper, and an interval based approach [1]. Notice that the current approach is able to deal with constituents holding in an interval of time: an expression $\Rightarrow a^{[t_1,t_2]}$ meaning that a holds between t_1 and t_2 can just be seen as a shorthand of the pair of rules $\Rightarrow^{\pi} a^{t_1}$ and $\rightsquigarrow^{\tau} \neg a^{t_2}$.

Non-monotonicity and temporal persistence are covered by a number of different formalisms, some of which are quite popular and mostly based on variants of Event Calculus or Situation Calculus combined with non-monotonic logics (see, e.g., [28,29]). TDL has some advantages over many of them. While TDL is able to cover many different aspects (in particular retroactivity), its time complexity is linear: to the best of our knowledge, no logic with the same coverage of TDL is so efficient.

Anyway, we would like to point out that interval and duration based defeasible reasoning has been developed by [6,18]. [18] focus on duration and periodicity and relationships with various forms of causality: no complexity result was presented there. [6] integrates temporal reasoning in argumentation theory and proposes a sophisticated interaction of defeasible reasoning and standard temporal reasoning (i.e., mutual relationships of intervals and constraints on the combination of intervals). In [6] constraint-based temporal reasoning combining intervals and instants are integrated in an argumentation framework. Predicates as *Holds*, *Occurs* and *Do* are used to associate constrained temporal information to logic formulas representing properties, events and actions. From the general point of view, [6] develops a very expressive (but not necessarily tractable) argumentation system: indeed, no complexity results are available for this work, but the system deals with very complex temporal structures without any apparent computational concern.

Other interesting approaches integrating temporal reasoning in argumentation theory include Mann and Hunter's [24] and Barringer and Gabbay's [7]. The last one proposes temporal and modal languages to represent arguments in the nodes of a network and give a Kripke semantics. Mann and Hunter in [24] encode temporal information via formulas of the form Holds(α, i) to express that α holds at interval i and propose a translation into classical propositional calculus.

Acknowledgements. NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

A previous version of this work was presented at NMR'2010 [15]. We are grateful to the referees for their valuable comments on earlier versions.

References

1. Allen, J.: Towards a general theory of action and time. Artificial Intelligence 23, 123–154 (1984)
2. Antoniou, G., Bikakis, A.: Dr-prolog: A system for defeasible reasoning with rules and ontologies on the semantic web. IEEE Trans. Knowl. Data Eng. 19(2), 233–245 (2007)
3. Antoniou, G., Billington, D., Governatori, G., Maher, M.J.: Representation results for defeasible logic. ACM Transactions on Computational Logic 2, 255–287 (2001)
4. Antoniou, G., Billington, D., Governatori, G., Maher, M.J.: Embedding defeasible logic into logic programming. Theory and Practice of Logic Programming 6, 703–735 (2006)
5. Antoniou, G., Billington, D., Governatori, G., Maher, M.J., Rock, A.: A family of defeasible reasoning logics and its implementation. In: Proceedings of the 14th European Conference on Artificial Intelligence, pp. 459–463. IOS Press (2000)
6. Augusto, J., Simari, G.: Temporal defeasible reasoning. Knowledge and Information Systems 3, 287–318 (2001)
7. Barringer, H., Gabbay, D.M.: Modal and temporal argumentation networks. In: Manna, Z., Peled, D.A. (eds.) Time for Verification. LNCS, vol. 6200, pp. 1–25. Springer, Heidelberg (2010)
8. Bassiliades, N., Antoniou, G., Vlahavas, I.: A defeasible logic reasoner for the Semantic Web. International Journal on Semantic Web and Information Systems 2, 1–41 (2006)

9. Dimaresis, N., Antoniou, G.: Implementing modal extensions of defeasible logic for the semantic web. In: AAAI 2007, pp. 1848–1849 (2007)
10. Governatori, G., Hulstijn, J., Riveret, R., Rotolo, A.: Characterising deadlines in temporal modal defeasible logic. In: Orgun, M.A., Thornton, J. (eds.) AI 2007. LNCS (LNAI), vol. 4830, pp. 486–496. Springer, Heidelberg (2007)
11. Governatori, G., Palmirani, M., Riveret, R., Rotolo, A., Sartor, G.: Norm modifications in defeasible logic. In: JURIX 2005, pp. 13–22. IOS Press, Amsterdam (2005)
12. Governatori, G., Rotolo, A.: BIO logical agents: Norms, beliefs, intentions in defeasible logic. Journal of Autonomous Agents and Multi Agent Systems 17, 36–69 (2008)
13. Governatori, G., Rotolo, A.: A computational framework for institutional agency. Artif. Intell. Law 16(1), 25–52 (2008)
14. Governatori, G., Rotolo, A.: Changing legal systems: Legal abrogations and annulments in defeasible logic. Logic Journal of IGPL 18(1), 157–194 (2010)
15. Governatori, G., Rotolo, A.: On the complexity of temporal defeasible logic. In: NMR 2010 (2010)
16. Governatori, G., Rotolo, A., Rubino, R.: Implementing temporal defeasible logic for modeling legal reasoning. In: Nakakoji, K., Murakami, Y., McCready, E. (eds.) JSAI-isAI 2009. LNCS (LNAI), vol. 6284, pp. 45–58. Springer, Heidelberg (2010)
17. Governatori, G., Rotolo, A., Sartor, G.: Temporalised normative positions in defeasible logic. In: ICAIL 2005, pp. 25–34. ACM Press (2005)
18. Governatori, G., Terenziani, P.: Temporal extensions to defeasible logic. In: Orgun, M.A., Thornton, J. (eds.) AI 2007. LNCS (LNAI), vol. 4830, pp. 476–485. Springer, Heidelberg (2007)
19. Kontopoulos, E., Bassiliades, N., Governatori, G., Antoniou, G.: A modal defeasible reasoner of deontic logic for the semantic web. Int. J. Semantic Web Inf. Syst. 7(1), 18–43 (2011)
20. Lam, H.-P., Governatori, G.: The making of SPINdle. In: Governatori, G., Hall, J., Paschke, A. (eds.) RuleML 2009. LNCS, vol. 5858, pp. 315–322. Springer, Heidelberg (2009)
21. Lam, H.-P., Governatori, G.: What are the necessity rules in defeasible reasoning? In: Delgrande, J.P., Faber, W. (eds.) LPNMR 2011. LNCS, vol. 6645, pp. 187–192. Springer, Heidelberg (2011)
22. Maher, M.J., Rock, A., Antoniou, G., Billington, D., Miller, T.: Efficient defeasible reasoning systems. International Journal of Artificial Intelligence Tools 10(4), 483–501 (2001)
23. Maher, M.J.: Propositional defeasible logic has linear complexity. Theory and Practice of Logic Programming 1, 691–711 (2001)
24. Mann, N., Hunter, A.: Argumentation using temporal knowledge. In: COMMA 2008, pp. 204–215. IOS Press (2008)
25. Nute, D.: Defeasible logic. In: Handbook of Logic in Artificial Intelligence and Logic Programming, vol. 3, pp. 353–395. OUP (1993)
26. Rubino, R.: Una implementazione della logica defeasible temporale per il ragionamento giuridico. PhD thesis, CIRSFID, University of Bologna (2009)
27. Rubino, R., Rotolo, A.: A Java implementation of temporal defeasible logic. In: Governatori, G., Hall, J., Paschke, A. (eds.) RuleML 2009. LNCS, vol. 5858, pp. 297–304. Springer, Heidelberg (2009)
28. Shanahan, M.: Solving the Frame Problem: A Mathematical Investigation of the Common Sense Law of Inertia. MIT Press (1997)
29. Turner, H.: Representing actions in logic programs and default theories: A situation calculus approach. Journal of Logic Programming 31(1-3), 245–298 (1997)

Efficient Persistency Management in Complex Event Processing: A Hybrid Approach for Gamification Systems

Philipp Herzig[1], Bernhard Wolf[1], Svenja Brunstein[1], and Alexander Schill[2]

[1] SAP Research, Chemnitzer Straße 46, 01187 Dresden, Germany
philipp.herzig@sap.com
[2] Technische Universität Dresden, Institute of Systems Architecture, Nöthnitzer Straße 46, 01187 Dresden, Germany
alexander.schill@tu-dresden.de

Abstract. Complex Event Processing (CEP) has been successfully applied in various domains. As of today, the management of external, durable, and encapsulated state in such systems has received little attention in research. An emerging kind of rule and event-based systems are platforms for gamification. These systems require an efficient management of entities containing state. In this paper, we are proposing a hybrid system capable of fast event processing on the one hand and global state, entity, and persistency management on the other hand. Moreover, we present and evaluate different synchronization strategies between an event processor and a business entity provider. We demonstrate that our extensions outperform conventional CEP solutions in terms of state persistency and ex post analytics by adding just a marginal performance overhead.

Keywords: Complex Event Processing, Production and ECA Rules, State Management, Hybrid Systems, Gamification.

1 Introduction

Complex Event Processing (CEP) is used to detect patterns in complex event clouds or continuous streams of events, for instance, "*total value of all withdrawals in the last 10 days exceeds 30*" [1]. CEP has been applied successfully in different domains such as banking, insurance, or healthcare [2]. More recently, numerous researchers raised the question how likely it is that critical decisions are taken "*merely on event patterns such as: event a is followed by event b in last 10 seconds?*" [3, p. 123]. For some scenarios, additional knowledge has to be processed in conjunction with events in order to detect noteworthy patterns.

In this paper, we consider knowledge as state encapsulated in durable entities representing, for example, users or physical devices. While the state of entities may change through arbitrary external influences, in CEP applications the state also changes upon consequences or updates of detected event patterns. Besides state, entities contain behavior. The behavior determines the next state after

L. Morgenstern et al. (Eds.): RuleML 2013, LNCS 8035, pp. 129–143, 2013.

updates have been received by the entity. In either case, the current state of entities has to be processed upon event arrival within an event processor. Although this management of additional context and external state is considered as a key part for event processing applications [4], its investigation in rule-based CEP systems has received little attention so far [5].

One domain that needs efficient state management in conjunction with real-time event processing is gamification. Gamification is defined as the "use of game design elements in non-game contexts" [6] and has emerged as recent information system trend. In the enterprise domain the intention is to introduce game mechanics on the job in order to foster higher levels of employee engagement which, in turn, positively influences organizational outcomes, such as organizational commitment or job performance. Enterprise information systems (EIS), e.g., ERP, CRM, or SCM may act as mediator to introduce game mechanics such as goals, rules, or badges [7] on top of existing business processes. Recent research has shown that gamification yields the desired improvements from a psychological perspective [8].

1.1 Scenario

In Figure 1, the general scenario of the gamification platform is shown which is used as example throughout this paper. An enterprise may operate a number of enterprise systems, e.g., enterprise resource planning (ERP), portal, or customer relationship management (CRM) systems. These systems are intended to be gamified. To support the introduction of gamification, another system, called the *gamification platform*, is used. This platform comprises an *event processing agent* (EPA) and an entity manager.

At design time, designers or psychologists define the gamification logic in the *EPA* using a declarative rule language. Moreover, entities such as badges, points, or levels have to be defined using the *entity manager*. Furthermore, the enterprise

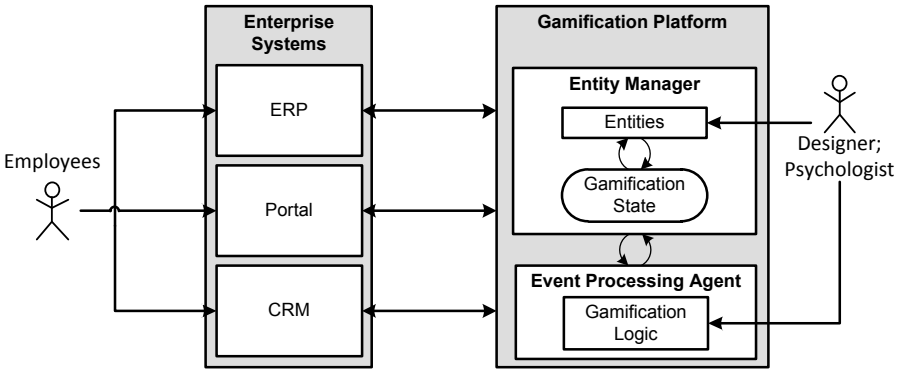


Fig. 1. General Scenario

systems have to be configured to send events to the platform, for example, using a BPM solution or other event driven components.

At runtime, employees interact with the enterprise systems. Accordingly, the enterprise systems send events to the EPA of the gamification platform. Based on the gamification rules deployed in the EPA, the rule engine calculates feedback mechanisms such as point or badge instances for the respective users and updates the entities' states using the entity manager. The enterprise systems may retrieve the users' current state, e.g., current points or leaderboards, from the entity manager. For example, the gamification platform may comprise rules that give the users of a CRM system several points for each customer they maintain in the system. Furthermore, these users may obtain badges once they have reached a particular amount of points and the like.

1.2 Requirements

We already discussed the quality attributes for a generic gamification platform in [9]. These requirements are summarized in the following. *Flexibility:* The gamification platform, especially its logic, has to be changeable to react to new business requirements. *Real-time event correlation and pattern detection*: Events sent to the platform in the form of user actions have to be arbitrarily correlated by causal, temporal, or spatial operators. If patterns have been detected, feedback has to be given at least in soft real-time. *Persistency and consistency:* Merits, such as points or badges, have to be stored and made available for users themselves as well as for other users at any time. Moreover, the data must be highly consistent, i.e., the stored data must be free of conflicts and contradictions. *Manageability:* The persisted data must be manageable across the gamifications' life cycle. For example, data has to be aggregated or anonymized after a particular time to ensure legal compliance in some countries or domains. *Analyzability:* In order to monitor the impact of the gamification, data must be analyzable by its stakeholders over certain dimensions such as time, users, or groups. Moreover, data mining might be applied ex post on the data to predict future user behavior. These predictions, in turn, may lead to a modification of the gamification logic.

Based on these requirements, we are assessing traditional approaches in Section 2. Due to mutual limitations of existing solutions, we are proposing a hybrid concept in Section 3 based on [5]. The efficiency of the proposed approach, however, highly depends on the utilized communication paradigms and trade-offs can be found which are evaluated and discussed in Section 4. We conclude our paper with an outlook on future research.

2 Solution Approaches

In this section, we describe the advantages and disadvantages of using relational database or complex event processing technologies as stand-alone solutions.

2.1 Databases

As first solution approach, we highlight databases. Due to ACID (atomicity, consistency, isolation, durability [10]) properties these systems comply with *persistency* and *consistency*. Moreover, databases allow the definition of transient queries on persistent data. This permits the definition of ad-hoc queries and ex post analysis and complies with *analyzability* requirements. Furthermore, well-established mechanisms for application-level entity management, e.g., object relational mapping exist (*manageability*).

However, databases typically fall short when large amounts of continuous data such as events have to be analyzed. Without index the correlation/joining of many events is very slow[1]. While the introduction of indexes may provide significant speed up[2], updating these indexes can be costly and inefficient on large event rates. Events leading to an index update might be never used again. Thus the cost-benefit ratio of the indexes might be large. Moreover, the efficient definition of indexes is not possible when the nature of events cannot be assumed or query statistics are not available. Hence, *real-time* analysis requirements are insufficiently supported. Finally, different types of events have different signatures. In databases, different signature types need to be represented by separate tables. Altering the signature of events or introducing new event types requires the restructuring of the database schema. This limits their *flexibility* with regard to design changes.

2.2 Complex Event Processing

"Complex Event Processing enables to identify causal, temporal, and spatial relationships between events. Those relationships specify patterns that are analyzed in real-time using event-pattern matching" [12]. In the last years, CEP has evolved as the overarching discipline of prior work such as production rule systems, active databases, and event/action/transition logic systems [13].

In contrast to databases, persistent queries are issued on transient and continuous event or data streams, i.e., data tuples are available for analysis for a particular time only. In production rule systems or event-condition-action (ECA) rule systems, fast pattern matching algorithms are available (e.g., RETE [14]). In event stream processing (ESP), fast aggregation is realized through window operators supporting at least soft *real-time* requirements. Due to reactive nature, zero latency between event arrival and processing can be assumed, i.e., events are processed immediately. Moreover, temporal operators such as *after*, *before*, *during* [15] are supporting the need of temporal relationships. Finally, the definition of production or ECA rules allows for a high degree of *flexibility* due to their dynamic structure.

However, in contrast to databases, the data has to reside within the working memory. Besides the fact that data in working memory is volatile, all modern

[1] $\mathcal{O}(\Pi_{i=0}^{p} r_i)$ or at least $\mathcal{O}(\sum_{i=0}^{p} r_i * log(r_i))$ for merge joins with r_i being the number of pages for the ith relation attribute [11].

[2] e.g., $\mathcal{O}(\sum_{i=0}^{p} r_i)$ assuming at least one equijoin [11].

CEP technologies have to fulfill either implicit memory management (e.g., using temporal constraints or limited windows sizes) or the explicit declaration of event expiration durations to reduce memory footprint. For most practical scenarios, this makes ex post analytics and ad-hoc queries impossible. Finally, implementing entity management logic directly within rules, easily bloats an application and makes it difficult to read and maintain. This strongly suggests for a decoupling and encapsulation of logic in entities [5].

3 Hybrid Approach

In order to compensate the disadvantages of both alternatives, we are proposing a hybrid concept in the following section.

3.1 General Case

The general idea of a hybrid system is shown in Figure 2. The system comprises several event sources, event sinks, one EPA as well as one business entity provider (BEP).

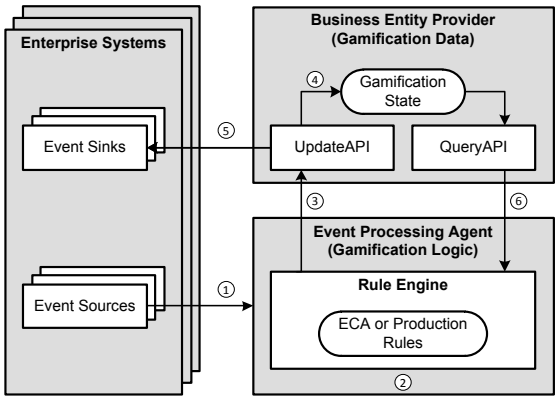


Fig. 2. Complex Event Processing with Business Entities (based on [5])

The BEP is a container managing so-called business entities which can be considered as typed data structures containing state and relationships to other entities. The entities are mapped and stored into relations of the underlying database. For example, the BEP manages user entities holding status information for all users of the gamification platform ($\{U_1, ..., U_k\}$; Table 1). Furthermore, the BEP offers an update interface that allows for updates to the state of the entities. For instance, the BEP may receive low-level updates such as points p or badges b through the update interface. A query interface might be utilized to retrieve the entities' current state (e.g., the user's average points $\overline{\mathcal{P}_{U_i}}$). Finally,

the BEP may provide derived state in transient data structures upon queries, e.g., an individual high-score for the user or several leaderboards of multiple users (Table 1).

Table 1. Business Entity Provider Examples for Gamification

Example	**Formal Notation**
User	U_1
Set of users	$\{U_1, ..., U_k\}$
Set of points for User U_i	$\{p_{i1}, ..., p_{in}\}$
Set of badges for User U_i	$\{b_{i1}, ..., b_{im}\}$
Sum of points for User U_i at timestamp t	$\mathcal{P}_{t,U_i} = \sum_{j=0}^{n} p_{ij}$
Average of points for User U_i	$\overline{\mathcal{P}_{U_i}} = \frac{1}{n} \sum_{j=0}^{n} p_{ij}$
Individual high score for user U_i based on $\mathcal{P}$	$(\mathcal{P}_{t,U_i}, ..., \mathcal{P}_{t+k,U_i})$
Leaderboard between users	$(U_1, ..., U_k)$

Within the EPA, multiple rules, e.g. gamification rules, are deployed. In this paper, rule examples are based on production rules with event processing capabilities, i.e., we consider a left-hand side (LHS) and right-hand side (RHS). In Table 2, we give examples for the rules' LHS. Besides typical event operations such a event occurrences[3] (a), boolean (b) and temporal (c) operators, or event aggregation (d), we also consider rules requiring additional non-event data in the LHS. This context might be processed upon event arrival such as in ECA rules (e), e.g., if an event e_i occurs in conjunction with the user's average points $\overline{\mathcal{P}_{U_i}}$ being equal or greater than 20. Furthermore, context might be processed without explicit event occurrence such as in production rules (f), e.g., if a user U_i holds any of the badges b_{i1} or b_{i2}.

On the RHS of rules (Table 3), either complex events (g), domain data (h) or combinations thereof (i) might be generated.

According to Figure 2, the EPA may receive events from arbitrary event sources in step ①. Assuming that the BEP does not contain data at the beginning, only rules of types (a) through (d) are activated. Activated rules may cause the generation of new events (g), domain data (h), or combinations thereof (i) in step ② (Fig. 2) . While the creation of new (possibly complex) events can be processed directly, domain data needs to by synchronized with the entities in the BEP. The rules' action clause, therefore, may call the update interface of the BEP in step ③. After updates have been received, validated, and stored by the BEP in step ④, data can be forwarded to various event sinks deployed in

[3] Note, that we presume interval-based event semantics in the notation instead of point-based semantics [16].

Table 2. LHS Examples for Gamification

LHS Types	**Example**
(a) Simple Event / Event Rule	$e_1 \rightarrow ...$
(b) Boolean event correlation	$e_1 \wedge e_2 \rightarrow ...$
(c) Temporal event operators	$e_1\ during\ e_2 \rightarrow ...$
(d) Event Aggregation	$\frac{1}{n}\sum_{i=0}^{n} e_i^{value} > 20 \wedge e_2 \rightarrow ...,\ n = sizeof(window)$
(e) Event with Context	$e_1 \wedge \mathcal{P}_{t,U_i} \geq 20 \rightarrow ...$
(f) Context only	$U_i \wedge (b_{i1} \vee b_{i2}) \rightarrow ...$

Table 3. RHS Examples for Gamification

RHS Types	**Example**
(g) Multiple Events	$... \rightarrow e_2, e_3$
(h) Multiple Data (e.g., Point or Badge)	$... \rightarrow p_{i1}, b_{i2}$
(i) Multiple Data and Events	$... \rightarrow p_{i2}, b_{i1}, e_2, e_3$

the enterprise in step ⑤, e.g., to notify the user on the successful completion of a task. Some rules in the EPA (namely (e) and (f)) may need state data (plain or derived) from the BEP. Hence, the event processor may retrieve and evaluate state by utilizing the BEP's query interface in step ⑥. This, in turn, may lead to the activation of new rules which closes the processing cycle starting again from step ①.

Based on this general scheme, we can map the rule types to the respective parts of the architecture (Table 4).

While the presented hybrid system might be effective in general, the efficiency of the approach, however, mainly depends on the communication paradigm that is used to communicate between the CEP and BEP. Therefore, we investigate different communication solutions in the next sections and describe their advantages and disadvantages.

3.2 Synchronous Context-Update

In Figure 3, the synchronous communication is shown. Events arrive at the EPA in step ①. Only rules of type (a) through (d) are activated in step ②. If at least one rule of type (h) or (i) is activated, the update interface of the BEP is called synchronously in step ③, i.e., the engine halts until the BEP

Table 4. Rule Types and Business Entity Provider

Scenario	Types of Rules
BEP only	Table 1
CEP only	Table 2 (a), (b), (c), (d), and (h); Table 3 (g)
Query Interface required	Table 2 (e) and (f)
Update Interface required	Table 3 (h) and (i)

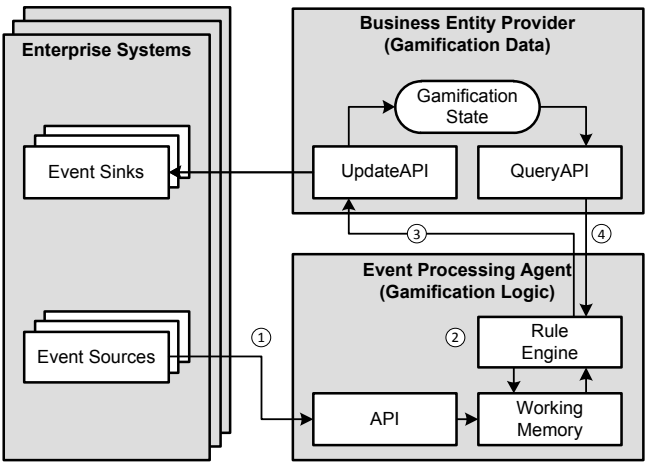


Fig. 3. Synchronous Context-Update

acknowledges the message. Since the respective entity received low-level updates, another evaluation cycle is triggered. This leads to the potential activation of rule types (e) and (f) as the rule engine utilizes the BEP's query interface while evaluating the rules' LHS in step ④.

Two main problems arise out of this architecture. First, due to blocking behavior of synchronous calls, the rule execution and update operations are delayed, especially when sampling rates of the event sources are high, i.e., there are many events to be processed concurrently. Second, situations occur where events may cause the query of the BEP although no update took place. This may result in many unnecessary calls to the query interface. Although the synchronous approach works in general, the solution is very inefficient, especially when large amounts of events have to be processed or many rules of type (e),(f),(h), and (i) reside within the rule engine. Thus, the approach heavily contradicts the real-time requirements.

3.3 Asynchronous Context-Update

Alternatively, the communication can be conducted asynchronously (Fig. 4). While this may speed up communication in general, race conditions leading to wrong results and inconsistencies may occur when domain data from the BEP is required. Therefore, we introduce a proxy component (BEP proxy) within the EPA that manages the data between CEP and BEP more efficiently and acts as central synchronization point. This proxy implements the interface specification of the update and query interfaces defined by the BEP. Moreover, this proxy takes care of the same entity management functionality as the BEP except persistency.

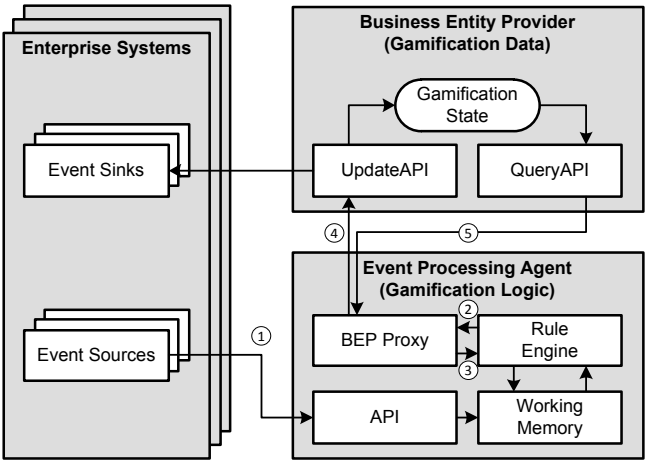


Fig. 4. Asynchronous Context-Update

Compared to the synchronous case, the rule engine only communicates with the local BEP proxy. All updates and queries are issued against the proxy. In the update step ②, the proxy stores the data internally and lazily replicates the data to the BEP in step ④. Vice versa, query results might be retrieved directly from the proxy ③. In the background, the proxy synchronizes with the BEP ⑤, for example, when the business entities in the BEP change their states due to exogenous influences to achieve consistency.

While this approach avoids race conditions and heavily accelerates the communication by several orders of magnitude, the low-level updates in the proxy force a constantly growing memory. Hence, there is a space-time trade-off which we characterize in the next section.

4 Evaluation

For the evaluation of the concept, the use case covers the gamification of a networking application which automatically matches users for lunches, coffee

breaks, or other networking opportunities. Users are assigned missions to engage them in exploring and using all aspects of the application. Production rules are used to reward various actions, e.g., the user gains points for each added colleague or accepted meeting. If a specific amount of points has been achieved, the user completes a particular mission, receives a badge, and is assigned another mission. For example, after accepting the first meeting, the user completes the *Accept First Meeting* mission, receives the corresponding badge and gets the next mission, e.g., *Host a Group Meeting*.

4.1 Experimental Setup

Overall the evaluation is based on 46 production rules with event processing capabilities. Given the types of rules from Section 3, we are using the rule quantities per type as shown in Table 5.

Table 5. Quantities of Rule Types

RHS \ LHS	(a)-(d)	(e)-(f)
(g)	1	0
(h)-(i)	15	30

The experiments are performed for 2^n experimental users with $n = 3, ..., 12$. Each user creates on average 0.67 events/second which is based on current usage statistics of the application[4]. Eight different event types are created for all possible actions users can perform in the application. For the experiment, each event type is triggered with a certain probability defined by the application scenario. These probabilities are fixed for each users, and thus, in an observation time of 5 minutes, the average number of events in total is

$$2^n \text{ Users} \times 0.67 \frac{\text{Events}}{\text{User} \times \text{s}} \times 300s = 201 \times 2^n \text{ Events}. \quad (1)$$

Experiments were executed on a machine with 8GB RAM, two 6-core Intel Xeon L5640 processors in hyper-threading mode. As reference implementation we used and measured Drools Fusion in *Cloud* mode.

4.2 Experimental Results

Besides the synchronous and asynchronous strategies from Section 3, we additionally present a modification to the synchronous approach using the context

[4] to test in fast motion, we extrapolated user behavior by replacing *day* with *seconds* at constant event probabilities.

locally at the EPA. Moreover, we present a plain CEP implementation for comparison, i.e., without additional overhead. Figure 5a and 5b present the median and maximum response times for all four approaches.

In the median case, the asynchronous strategy excels the synchronous strategies significantly up to seven orders of magnitude (1024 users). It is important to note that beyond 1024 experimental users, the synchronous case is not only significantly slower, but due to limited queues, the server starts to reject requests after a particular timeout which results in errors and makes results incomparable. Therefore, we only present synchronous results up to 1024 users. For the asynchronous case, event rates beyond 4096 users reached the capacities of our simulation environment. Until then, our asynchronous approach is only slightly slower compared to the plain CEP approach (0.98ms-4.31ms).

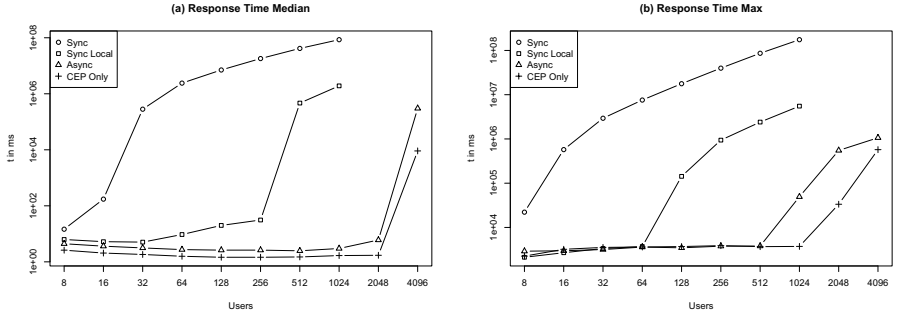


Fig. 5. Experimental Max and Median Response Times (in ms)

In the maximum case, the difference between the synchronous and asynchronous approach is smaller (four orders of magnitude for 1024 users). However, analyzing the underlying response time distributions yields that large response times are much more likely in the synchronous case (e.g., 95% of responses are slower than 10^6ms) compared to the asynchronous one (e.g., 1% of responses is slower than 10^5ms but 99% are faster than 100ms).

Furthermore, we can distinguish two phases in the median case. First, the phase in which the systems runs normally. In this phase, the data distribution is highly right-skewed as shown in Figure 6a. Second, the phase where the system runs under high load. This phase is characterized through an increasing frequency of high response times (e.g., Fig. 6b - 6d). For example, in the synchronous case, the first phase persists until 16 users. The second phase starts at 32 users. In the asynchronous case, the first phase persists until 2048 users, i.e., two orders of magnitude later with regards to the amount of events (Eq. 1).

This statement is further supported by Figure 7a, where we considered a response time threshold of 500ms. Already for 8 users, 8% of all events are too late in the synchronous case. For 256 users, already 95% of the events are above the threshold. In the asynchronous case, events are delayed starting at 512 experimental users (3%).

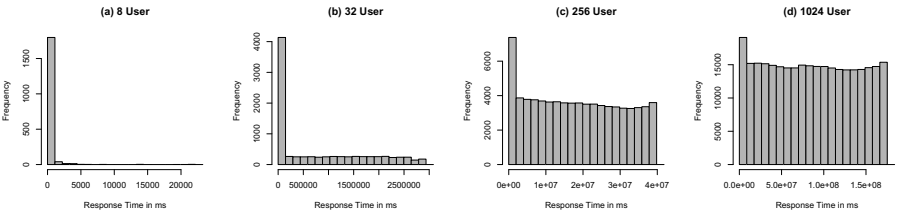


Fig. 6. Evolution of Response Times for Synchronous Communication

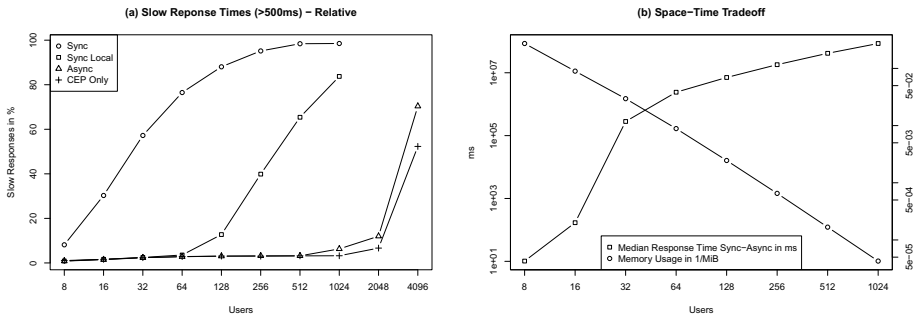


Fig. 7. Slow Reponse Times and Space-Time Tradeoff

However, the speed-up in the asynchronous case comes at the cost of growing memory in the EPA. The resulting space-time tradeoff is presented in Figure 7b where the speed-up is compared to the inverse memory consumption of the asynchronous approach. As presented, the intersection can be found around 32 experimental users, i.e., for small workloads the synchronous case might be preferred. Larger workloads may be deployed in an asynchronous setting. This trade-off, however, does not take into account the individual costs for performance or memory. For example, at the trade-off point already 60% of events are highly delayed in the synchronous case which might not be acceptable either. Thus, a cost-benefit trade-off might look different. Overall, this tradeoff is not generalizable but depends on the workload, rule, and deployment model of the experimental setup described above.

4.3 Discussion

As shown above, the plain synchronous system is up to seven orders of magnitude slower and runs already under overload at 32 users in the experiment. Moreover, data tuples are discarded under overload conditions leading even to wrong results in the BEP. Holding the context local in the EPA with synchronous communication, shifts the overload phase to 512 users in our experiments. Therefore,

the synchronous communication is not acceptable in all cases with regards to performance.

The asynchronous case with local BEP proxy is able to compete with our reference CEP implementation in the median case. The overload phase occurs at 4069 users. The same holds for the pure CEP implementation. Hence, we argue that our hybrid concept with asynchronous communication serves all the requirements of Section 1.2.

However, the BEP Proxy has to implement the query and update interfaces of the BEP. Hence, additional effort is needed for reimplementing and maintaining these interfaces. Hence, future research may address an automatic translation of BEP semantics.

Furthermore, the experimental maximum results for the asynchronous solution suggest room for improvements. Currently, we hold fine grained domain data within the proxy. For example, the user's points are stored atomically in collections or hashmaps. Instead of providing fine grained domain data, this data might be aggregated according to the semantics defined in the entities. Such an aggregation may improve maximum response times in general and bring the overall performance closer to the pure CEP measures. Additionally, this also reduces memory consumption and improves the space-time trade-off of Figure 7b.

5 Related Work

The idea of hybrid concepts for state management as presented in this paper are not completely new. Already [3] noted that *"[...] for some applications patterns such as event a is followed by event b in last 10 seconds are expressive enough; however, for knowledge-rich applications, they are certainly not. In such applications real-time actions are triggered not only by events, but also upon additional knowledge"*.

Based on this observation, [5] introduced the general idea of entity management with so-called business entities within a business entity provider. While their concept exactly targets the problem described herein, their paper is of conceptual nature and even states that their solution runs in a *"synchronous execution mode to the expense of overall event-processing performance"*. Our results demonstrate that synchronous execution may not acceptable at all, depending on the types of rules deployed in the EPA. Thus, not considering the execution mode may work, but contradicts with requirements in the CEP space. The introduction of asynchronous communication serves these requirements but requires additional concepts as described in this paper.

FlexStreams [17] allows the encapsulation of state in procedural definitions locally available to the stream operators. Well-encapsulated state management has been proposed by Kozlenkow et al. [4]. The state is maintained at the context in which it is processed. State deltas are managed and distributed by a state transformer. However, both approaches require a certain style of programming paradigm, i.e., a procedural or functional paradigm respectively.

Teymourian et al. [18] proposed semantic CEP for knowledge rich event processing, i.e., the usage of ontologies together with rules in order to allow rules such as *"buy Stocks of Companies, who have production facilities in Europe and produce products from Iron and [...] and their price/volume increased stable in the past 5 minutes"*. Their main assumption is that the ontology does not fit into the working memory and that the ontology is rarely updated by an external person. Based on this assumption, they proposed several strategies to query the ontology. However, besides a *polling* strategy, other strategies are tailored to the specifics of semantic query languages.

6 Summary and Outlook

In this paper, we presented and evaluated a hybrid system comprising an event processor and business entity provider which is capable of fast event processing on the hand and ex post analytics on the other. We demonstrated that the hybrid approach extends classical CEP approaches with novel features, such as, state persistency or ex post analytics. Nevertheless, its performance can still compete with traditional CEPs since just a minimal processing overhead is added. The results presented are based on a real-life product scenario. To generalize the performance and memory costs we are currently formalizing the approach in a sophisticated performance model. Moreover, we plan to conduct further studies to validate the approach and identify limitations, additional requirements, and challenges. Future research also includes the investigation of the impact of external changes to the entity manager which have to be efficiently synchronized with the event processor.

References

1. Adi, A., Botzer, D., Nechushtai, G., Sharon, G.: Complex event processing for financial services. In: Services Computing Workshops, SCW 2006, pp. 7–12. IEEE (September 2006)
2. Magid, Y., Sharon, G., Arcushin, S., Ben-Harrush, I., Rabinovich, E.: Industry experience with the ibm active middleware technology (amit) complex event processing engine. In: Proceedings of the Fourth ACM International Conference on Distributed Event-Based Systems, DEBS 2010, pp. 140–149. ACM, New York (2010)
3. Anicic, D., Rudolph, S., Fodor, P., Stojanovic, N.: Retractable Complex Event Processing and Stream Reasoning. In: Bassiliades, N., Governatori, G., Paschke, A. (eds.) RuleML 2011 - Europe. LNCS, vol. 6826, pp. 122–137. Springer, Heidelberg (2011)
4. Kozlenkov, A., Jeffery, D., Paschke, A.: State management and concurrency in event processing. In: Proceedings of the Third ACM International Conference on Distributed Event-Based Systems, DEBS 2009, pp. 23:1–23:4. ACM, New York (2009)
5. Obweger, H., Schiefer, J., Suntinger, M., Thullner, R.: Entity-Based State Management for Complex Event Processing Applications. In: Bassiliades, N., Governatori, G., Paschke, A. (eds.) RuleML 2011 - Europe. LNCS, vol. 6826, pp. 154–169. Springer, Heidelberg (2011)

6. Deterding, S., Dixon, D., Khaled, R., Nacke, L.: From Game Design Elements to Gamefulness: Defining Gamification. In: MindTrek 2011 Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments, pp. 9–15. ACM (2011)
7. Flatla, D., Gutwin, C., Nacke, L., Bateman, S., Mandryk, R.: Calibration Games: Making Calibration Taks Enjoyable by Adding Motivating Game Elements. In: UIST (2011)
8. Herzig, P., Strahringer, S., Ameling, M.: Gamification of ERP Systems - Exploring Gamification Effects on User Acceptance Constructs. In: Multikonferenz Wirtschaftsinformatik, GITO, 793–804 (2012)
9. Herzig, P., Ameling, M., Schill, A.: A Generic Platform for Enterprise Gamification. In: Software Architecture (WICSA) and European Conference on Software Architecture (ECSA), pp. 219–223. IEEE (2012)
10. Bell, D., Grimson, J.: Distributed database systems. Addison-Wesley Longman Publishing Co., Inc. (1992)
11. Date, C.J.: An Introduction to Database Systems, 8th edn. Pearson Addison-Wesley, Boston (2004)
12. Luckham, D.: The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems. Addison-Wesley, New York (2007)
13. Paschke, A., Kozlenkov, A.: Rule-Based Event Processing and Reaction Rules. In: Governatori, G., Hall, J., Paschke, A. (eds.) RuleML 2009. LNCS, vol. 5858, pp. 53–66. Springer, Heidelberg (2009)
14. Forgy, C.L.: Rete: A fast algorithm for the many pattern/many object pattern match problem. Artificial Intelligence 19(1), 17–37 (1982)
15. Allen, J.: Maintaining Knowledge about Temporal Intervals. Communications of the ACM 26(11), 832–843 (1983)
16. Chakravarthy, S., Adaikkalavan, R.: Events and streams: harnessing and unleashing their synergy? In: Proceedings of the Second International Conference on Distributed Event-Based Systems, pp. 1–12. ACM (2008)
17. Sybase Inc.: Flexstreams (2013), `http://www.sybase.de/products/financialservices-solutions/complex-event-processing`
18. Teymourian, K., Rohde, M., Paschke, A.: Fusion of background knowledge and streams of events. In: Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems, pp. 302–313. ACM (2012)

Ontology Patterns for Complex Activity Modelling

Georgios Meditskos, Stamatia Dasiopoulou, Vasiliki Efstathiou, and Ioannis Kompatsiaris

Information Technologies Institute
Centre of Research & Technology - Hellas
{gmeditsk,dasiop,vefstathiou,ikom}@iti.gr

Abstract. In this paper we propose an activity patterns ontology to formally represent the relationships that drive the derivation of complex activities in terms of the activity types and temporal relations that need to be satisfied. The patterns implement the descriptions and situations (DnS) ontology pattern of DOLCE Ultra Lite, modelling activity classes of domain ontologies as instances. The aim is to allow the formal representation of activity interpretation models over activity classes that are generally characterized by intricate temporal associations, and where it is often the case that the aggregation of individual activities entails the existence of a new (composite) activity. Due to the expressive limitations of OWL, these semantics are often defined outside the ontology language, e.g. they are encapsulated in rules and they are tightly-coupled with implementation frameworks, hindering the interoperability and reuse of the underlying knowledge. As a proof of concept, we describe the implementation of the activity pattern semantics using dynamically generated SPARQL rules in terms of `CONSTRUCT` graph patterns.

Keywords: ontologies, patterns, activity modelling, SPARQL rules.

1 Introduction

In the past five years, ontologies have been gaining increasing attention as means for modelling and reasoning over contextual information, and human activities in particular. Their expressiveness and level of formality, make them particularly well-suited for the open nature of context-aware computing, where information at various levels of abstraction and completeness has to be integrated. Low-level information acquired from detectors, such as video cameras and contact sensors, is mapped to ontological representations; high-level activity interpretations are then inferred through the use of background knowledge specific to the domain in order to elicit an understanding of the situation and to afford services tailored to the user needs. For example, inferring that an Alzheimer's disease patient left the kitchen to answer an incoming call but failed to resume lunch afterwards comprises critical contextual information, in which case a respective reminder needs to be issued.

L. Morgenstern et al. (Eds.): RuleML 2013, LNCS 8035, pp. 144–157, 2013.

Plenty of ontology-based models have been developed for complex activity recognition. Central to all, yet primary cause of variation, is the approach taken to capture *activity patterns*, namely the structure of complex activities that are built from atomic or other complex activities. Roughly speaking, two strands permeate the relevant literature. The one adopts an *a-temporal* viewpoint, where contextual activity information has no temporal extension, and complex activities are defined as the intersection of their constituent parts [21][6]. Though relevant to applications where suitable time-windows can be reliably defined, *a-temporal* approaches fall short when more intricate activity patterns are involved, requiring, for instance, the discrimination of sequential and interleaved activities. The second strand embraces hybrid ontology-based models that manage temporal information by means of rule-based reasoning [7], Semantic Complex Event Processing [27] and RDf stream reasoning [26], thus affording far more expressive and flexible solutions than their *a-temporal* counterparts.

However, no consensual activity recognition model exists that may be broadly reused. This is largely due to semantics encapsulated in the implementation rather than the models developed for activity recognition. A prominent example is the assertion of new named individuals for representing inferred complex activities, e.g. assert a *tea preparation* instance that is inferred on the basis of *heat water* and *use tea bag* instances. Thus, applications that share similar purpose and scope cannot directly avail of existing ontologies, unless specific implementation details are made available.

In order to promote a well-defined description of patterns for high-level activity interpretation tasks and achieve a high degree of interoperability, we propose an activity patterns ontology that formally captures the structure of complex activities. The activity patterns introduced follow the DnS design pattern [10] of DOLCE+DnS Ultralite (DUL) [9] and make use of the OWL 2 metamodelling capabilities (*punning*). Currently, two activity patterns have been implemented; one for formalising the notion of *composition* of an activity based on its constituent activities and one for formalising the notion of *specialisation* of an activity within the activity hierarchy. We also present the implementation of the activity pattern semantics in terms of dynamically generated SPARQL `CONSTRUCT` rules (SPIN rules [12]), exemplifying the way the proposed patterns can be used in the domain of rule-based activity recognition.

The rest of the paper is structured as follows. Section 2 reviews existing ontology-based activity modelling and processing frameworks. Section 3 provides an overview of the lightweight vocabulary used by the activity patterns ontology to represent domain activities, e.g. actors, places, etc. Section 4 introduces the core activity pattern for representing complex activity-related conceptualisations, along with its two implementations defined within the pattern ontology for addressing the requirements of specialisation and composition of complex activities. Section 5 presents an example use of the ontology patterns in the domain of rule-based activity recognition, implementing the activity pattern semantics as SPARQL `CONSTRUCT` graph patterns. Conclusions and future directions are presented in Section 6.

2 Related Work

Ontologies have been increasingly explored for modelling and reasoning about complex activities and situations in context-aware applications.

In [6][21][7][22] ontologies are used to recognise typical activities of daily living (e.g. making tea), through snapshot-like definitions that capture the associated contextual elements (e.g. turning the kettle on, place tea bag to mug), but fall short to cover the underlying temporal correlations. In [20], activity definitions are endowed with temporal properties, such as *recently used* and *second last activity*, while several approaches have investigated hybrid frameworks, combining ontologies with rules [16][13][28], as well as other formalisms, such as event calculus [19][18], to support reasoning over the temporal structure of activities. In addition, extensions to the SPARQL language have been proposed for working with temporal RDF streaming data [27][26][5][3].

Ontologies have been also explored as means to capture in a declarative way meta-knowledge. In [29] a top-level ontology is proposed to model the semantics common to all dimensions of an information space, i.e. levels of granularity, conflicting, and overlapping relationships that can be used to evaluate and compare concepts and terms of the ontologies built upon them. In [14] an ontology-based framework, based on the Event-Condition-Action (ECA) pattern, is presented in order to integrate heterogenous semantic web services via rule definition, in [4] an ontology is used to model different types of event rules in order to enable automatic service discovery, while in [8], a Rule Management Ontology is presented to support the representation of event-based rules that trigger specific actions in a context-aware recommender system.

Similar to [8], we aim to promote reusable and interoperable contextual activity models. However, unlike [8] that focuses on the definition of a vocabulary for representation of event-based rules, we use the DnS ontology pattern to formalise abstract activity descriptions. As such, the underlying semantics of the activity patterns can be reused in already existing frameworks for activity modelling and processing, such as the aforementioned one.

3 The Domain Activity Ontology

In order to enable the definition of patterns for the description of the contextual conditions and the temporal relations that drive the derivation of *complex activities*, there is a need for a core vocabulary for the representation of basic activity information. The Domain Activity Ontology depicted in Figure 1 serves this purpose. Both the atomic (i.e. asserted) and complex activities of the domain are represented as instances of the `Activity` class and they are linked to ranges of time through the use of the `hasStartTime` and `hasEndTime` datatype properties. Actors are defined using the `hasActor` property, whereas relevant participating entities in an activity, such as objects or other persons, are represented using the `hasParticipant` property. Spatial information can be associated with the activities using the `hasArea` property, e.g. the room where an activity takes

place. Finally, the ontology supports the correlation of activities through the `hasSubActivity` and `isSpecialisedBy` properties for the representation of the corresponding semantics of the two proposed patterns. The aforementioned modelling capabilities have been designed with a minimum of semantic commitment to guarantee maximal interoperability. As such, the Domain Activity Ontology can be fully aligned with relevant foundational ontologies, such as SEM [11] and Ontonym [25], reusing existing vocabularies for modelling different aspects of activities, e.g. entities, places, etc.

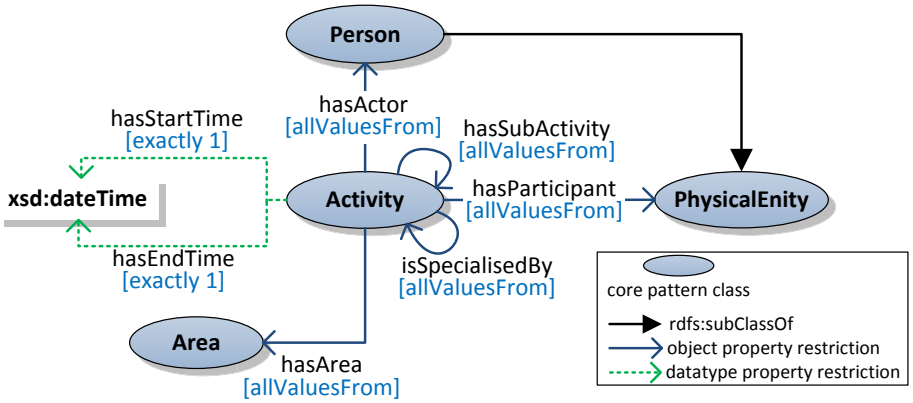


Fig. 1. The core vocabulary of the Domain Activity Ontology

4 Design of Activity Patterns

To promote a well-defined description of complex activities structure and achieve a better degree of knowledge sharing and reuse, an activity patterns ontology has been developed as specialized instantiations of the descriptions and situations (DnS) ontology pattern that is part of DOLCE+DnS Ultralite. The developed activity patterns treat domain activity concepts as instances to allow property assertions to be made among activity types. In that way, they enable the representation of contextualised views on complex activities, and afford reusable pieces of knowledge that cannot otherwise be directly expressed by the standard ontology semantics, e.g. temporal correlations among activities that are not connected in a tree-like manner [15].

In the following, we introduce the core activity pattern and its two current instantiations, namely the *specialisation* and *composition* patterns. The specialisation pattern formalises complex activities, whose derivation concerns an already asserted activity instance; the composition pattern formalises complex activities, whose derivation is based on the aggregation of other activities and requires the assertion of new activity instances.

4.1 Core Meta-Activity Pattern

The DnS design pattern provides a principled approach to context reification through a clear separation of states-of-affairs, i.e. a set of assertions, and their interpretation based on a non-physical context, called a description [10]. Intuitively, DnS axioms try to capture the notion of "*situation*" as a unitarian entity out of a state of affairs, with the unity criterion being provided by a "*description*". In that way, when a description is applied to a state of affairs, a situation emerges. We use DnS to formally provide precise representations of contextualized situations and descriptions on activity concepts of the Domain Activity Ontology, describing the different activity types and temporal relations that can be associated with complex domain activities.

The implementation of DnS in DUL allows the relation of situations and descriptions with individuals of the `dul:Event` and `dul:EventType` classes, respectively. For example, Event-Model-F [23] implements a number of instantiations on top of the DnS pattern to describe relations among *asserted* events (instances of the `dul:Event` class), such as causality and correlation. The scope of the core activity pattern, however, is to conceptually describe the activity context that defines complex activities at the class level, and not to represent relations directly among activity instances. To this end, the core activity pattern allows the representation of the following activity-related conceptualisations (Figure 2):

- **Activity situations**: An activity situation defines a set of activity classes that are involved in a specific pattern instantiation and they are interpreted on the basis of an activity description. The classified activity classes are treated as instances of the `MetaActivity` class and they are associated with situations through the `isMetaActivityIncludedIn` property. Moreover, each situation satisfies only a single description (`hasDescription` property).
- **Activity descriptions**: An activity description serves as the descriptive context of an activity situation, defining the activity types (`definesActivityType` property) that classify the domain activities of a specific pattern instantiation, creating views on situations.
- **Activity types**: Activity types are DUL concepts that classify activity classes, i.e. they treat domain activity classes as instances, describing how they should be interpreted in a particular situation. These descriptions mainly involve the specification of the temporal constraints that characterise the respective contextual activities, reusing the temporal property assertions provided by the OWL-Time ontology in terms of the `time:TemporalEntity` class [17].
- **Meta-activities**: The domain activity classes, i.e. the classes of the Domain Activity Ontology, are instances of the `MetaActivity` class.

Currently, two implementations of the core activity pattern are provided that satisfy common interpretation requirements in complex activity recognition applications, namely *specialisation* and *composition*.

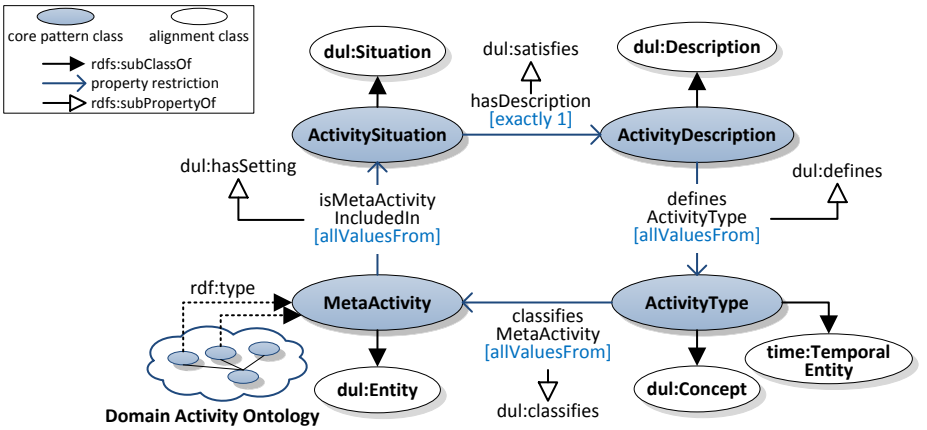


Fig. 2. The core meta-activity pattern and the main alignments with DUL's DnS vocabulary

4.2 Activity Specialisation Pattern

The activity specialisation pattern enables to formally capture complex activities that are defined as further specialisations of a given atomic or other complex activity. As shown in Figure 3, a definition of this type is expressed by a `SpecialisationSituation` that satisfies a `SpecialisationDescription`. The situation includes the descriptive context that admits the specialisation, namely the activity that is subject to further specialisation, one or more associated activities, and their pertinent temporal correlations. The classes `SpecialisedActivityType` and `SpecialisationType` express the asserted and derived activity respectively, while the class `ContextType` allows to express activities comprising the descriptive context. Temporal correlations among activities are expressed through their associated `SpecialisedActivityType`, `SpecialisationType` and `ContextType` classifications that subsume the `ActivityType` class, and thus `time:TemporalEntity`, as depicted in Figure 2.

4.3 Activity Composition Pattern

The activity composition pattern enables to formally capture complex activities that are defined as the composition of atomic or other complex activities. As shown in Figure 4, a composite activity definition is expressed by a `CompositionSituation` that satisfies a `CompositionDescription`. The situation includes the descriptive context that admits the composition, namely the composite activity, its constituent activities and their pertinent temporal correlations. The classes `CompositeType` and `SubActivityType` express the complex activity to be inferred and its constituent activities respectively. Similar to the specialisation pattern, the temporal correlations among the involved activities are expressed through their associated `CompositeType` and `SubActivityType` classifications that subsume the `ActivityType` class.

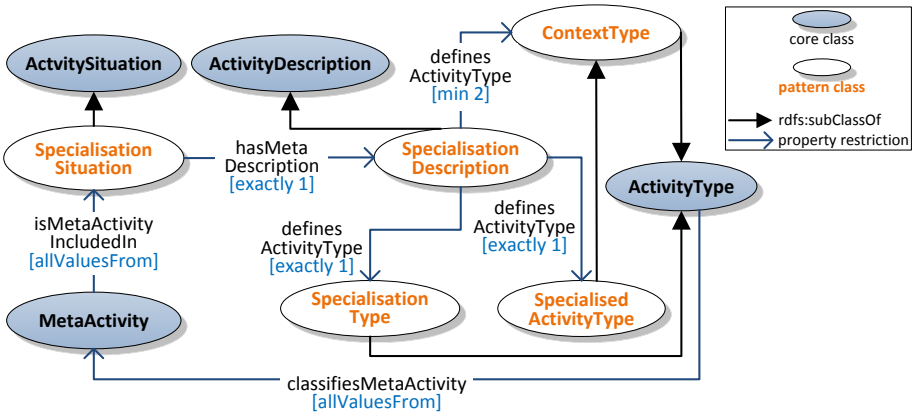


Fig. 3. The activity specialisation pattern

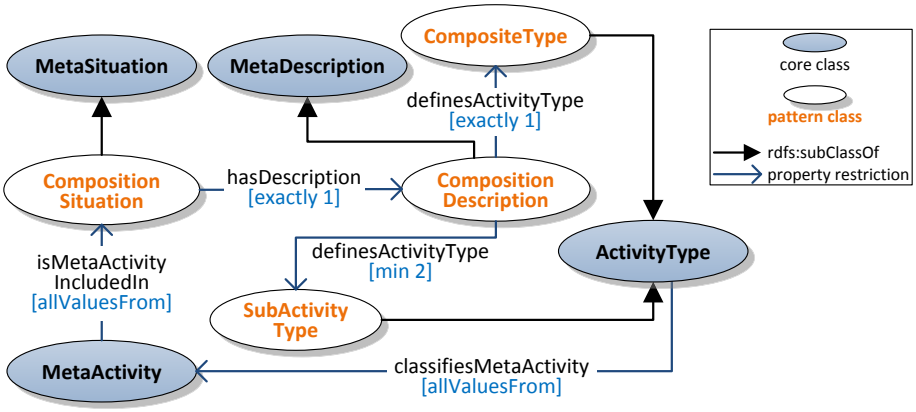


Fig. 4. The activity composition pattern

4.4 Example

In the following we describe example instantiations of the activity patterns in the home-based healthcare domain for specifying the activity contexts that define the derivation of nocturia incidences. Our scenario involves the following low-level (atomic) activities that are represented in terms of the Domain Activity Ontology (see section 3) and they are made available directly through monitoring, e.g. by video processing and sensor-based modules, such as sleep monitoring devices[1]:

– **Night sleep**: The overall night sleep activity is represented as instance of the `NightSleep` class.

[1] The Renew SleepClock™ is an example of a monitoring device that records, among others, the sleep onset and waking times.

– **Out of bed**: Instances of the `OutOfBed` class represent periods of time when the person is out of the bed.
– **In bathroom**: Instances of the `InBathroom` class represent periods of time when the person is in the bathroom.

The scope of the activity pattern instantiations is to describe the way the aforementioned low-level activities can be aggregated and correlated so as to derive the following complex (inferred) activities of the domain:

– **Bed exit**: A bed exit (instance of the `BedExit` class) represents an out of bed activity that is performed during the night sleep.
– **Nocturia**: Nocturia represents a new activity element (instance of the `Nocturia` class), derived by the aggregation of bed exit and in bathroom activities.

Figure 5 depicts the instantiation of the specialisation pattern for describing bed exits. The example defines `ActivityType` instances for the classification of activity classes: the `NightSleep` and `OutOfBed` classes are classified by `ContextTypes`, the `OutOfBed` class is further classified by a `SpecialisedType` and `BedExit` is classified by a `SpecialisationType` instance. Moreover, the `NightSleep` class is temporally related (`time:intervalContains`) to the `OutOfBed` class through the corresponding activity type instances that classify them. These relations are encapsulated as property assertions in the `bed_exit_desc` description which is related to the `bed_exit_sit` situation. Intuitively, the instantiation of the pattern defines that an `OutOfBed` instance is further specialised as `BedExit`, if it is temporally contained in a `NightSleep` activity.

Figure 6 shows the instantiation of the composition pattern for describing `Nocturia` activities. Both the `InBathroom` and `BedExit` classes are classified by `SubActivityTypes`, `Nocturia` is classified as the `CompositeType`, and `BedExit` is temporally related (`time:intervalContains`) to `InBathroom` through their pertinent classifying instances. Intuitively, the example defines `Nocturia` as a

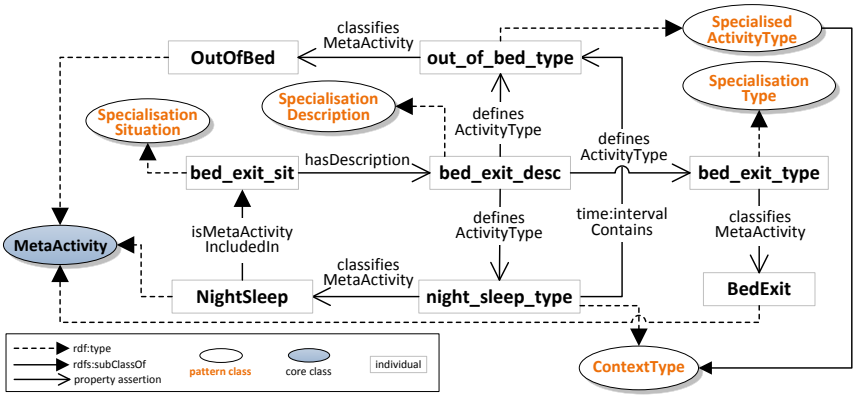


Fig. 5. The instantiation of the specialisation pattern for `BedExit`

composite activity whose instances derive based on the temporal dependencies between `BedExit` and `InBathroom` activities. Additionally, the `nocturia_type` instance is temporally related to the `in_bathroom_type` and `bed_exit_type` instances, so as to define the temporal boundaries of the new composite activity: nocturia is defined to start when the bed exit starts (startedBy) and finishes together with the in bathroom activity (finishedBy).

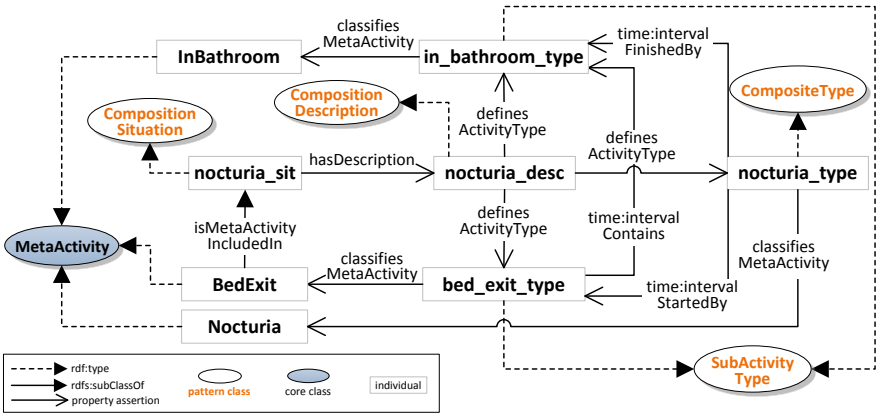


Fig. 6. The instantiation of the composition pattern for `Nocturia`

5 Transforming Activity Patterns into SPARQL

The purpose of the activity patterns is to describe the contextual conditions and the temporal relations that drive the derivation of complex activities, through well-defined and reusable activity models. As such, the encapsulated semantics can be shared across applications with similar scope but different implementation frameworks. In this section, we describe a prototype implementation of a transformation framework that dynamically generates SPARQL rules that implement the semantics of the composition and specialisation patterns.

The abstract architecture of the prototype framework is depicted in Figure 7. More specifically, the semantics of the Domain Activity Ontology for representing domain activities, and the semantics of the instantiated patterns, e.g. the property restrictions, sub-properties, inverse properties, etc. that are defined by the DnS implementation in DUL, are handled by an OWL ontology reasoner (e.g. [24]). The ontology model is then used by the SPARQL Generator to dynamically generate SPARQL rules, based on the provided pattern instantiations.

5.1 SPARQL Rules

The rules in our framework are defined in terms of domain-specific SPARQL `CONSTRUCT` graph patterns (SPIN rules [12]), tailored to the semantics of the provided pattern instantiations. A SPARQL rule is defined in terms of a `CONSTRUCT`

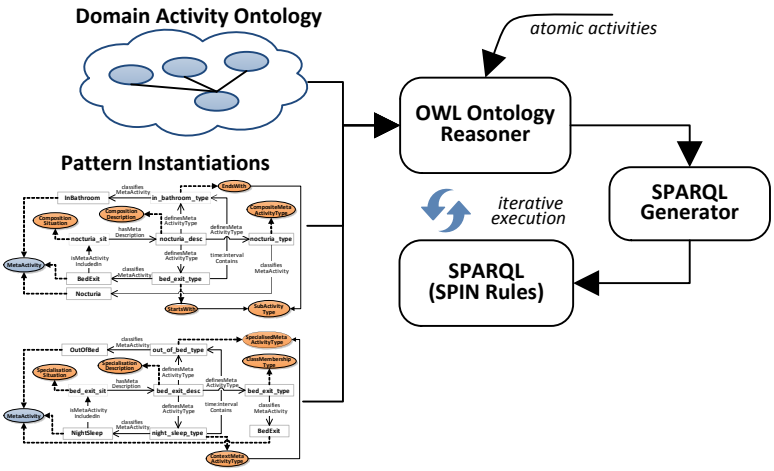


Fig. 7. The prototype framework for generating and executing SPARQL rules

and a `WHERE` clause: the former defines the set of triple patterns that should be added to the underlying activity model upon the successful pattern matching of the triple patterns in the `WHERE` clause. The generation of such rules in involves the mapping of the conceptual knowledge provided by the activity patterns on SPARQL triple patterns and functions.

In the case of the specialisation pattern, the activity classes that are classified by `ContextType`s are used to define the triple patterns that match the corresponding activity instances in the `WHERE` clause. Additionally, the `SpecialisationType` that classifies the class of the specialisation is used to define the triple patterns in the `CONSTRUCT` clause that specify the additional class type of the specialised instance. In the case of the composition pattern, the triple patterns that match activity instances in the `WHERE` clause are determined based on the `SubActivityType` classifications. The `CompositeType` classification is used to define the activity type of the new composite activity that is generated in the `CONSTRUCT` clause. In both activity patterns, the temporal constraints among activities are checked using SPIN functions[2] that implement basic temporal relations (Allen's temporal operators [1]).

Figure 8 shows the SPARQL rules that are generated for the recognition of bed exit and nocturia activities performed by a specific patient (actor), based on the pattern instantiations described in Section 4.4. In Figure 8 (a), the out of bed activity is further specialised as a bed exit, whereas in Figure 8 (b) a new nocturia instance is generated by aggregating bed exit and in bathroom activities. The start time of the composite activity is associated with the start

[2] SPIN functions are reusable SPARQL queries that can be referenced inside SPARQL FILTER and BIND functions.

time of the bed exit activity, whereas the end time is associated with the end time of the in bathroom activity, as the example in Figure 6 models through the temporal-related property assertions `startsBy` and `finishedBy`. The SPARQL function `contains` checks whether the first time interval (`[?st1, ?et1]`) contains the second (`[?st2, ?et2]`). The `newURI` SPARQL function generates a unique URI for the new nocturia by concatenating its sub-activity URIs to ensure the termination of the reasoning procedure, since always the same URI is generated from the same sub-activities `?x` and `?y`. Also note that the rules further correlate the activity instances in terms of the Domain Activity Ontology properties discussed in Section 3. In that way, we materialise information relevant to the activities that participated in the specialisation of an instance (`isSpecialisedBy` property) and to the sub-activities of composite activities (`hasSubActivity` property).

```
CONSTRUCT {
  ?y a BedExit; //SpecialisationType
    isSpecialisedBy ?x.
}
WHERE{
  ?x a NightSleep; //ContextType
    hasStartTime ?st1;
    hasEndTime ?et1;
    hasActor ?p.
  ?y a OutOfBed; //SpecialisedType
    hasStartTime ?st2;
    hasEndTime ?et2;
    hasActor ?p.
  FILTER(:contains(?st1, ?et1, ?st2, ?et2))
}
```

(a)

```
CONSTRUCT {
  ?new a Nocturia; //CompositeType
    hasStartTime ?st1;
    hasEndTime ?et2;
    hasActor ?p;
    hasSubActivity ?x;
    hasSubActivity ?y.
}
WHERE{
  ?x a BedExit; //SubActivityType
    hasStartTime ?st1;
    hasEndTime ?et1;
    hasActor ?p.
  ?y a InBathroom; //SubActivityType
    hasStartTime ?st2;
    hasEndTime ?et2;
    hasActor ?p.
  FILTER(:contains(?st1, ?et1, ?st2, ?et2))
  BIND(:newURI(?x, ?y) as ?new)
}
```

(b)

Fig. 8. (a) The specialisation rule for deriving bed exits, (b) the composition rule for inferring nocturia instances

6 Conclusions

In this paper, we presented an activity patterns ontology that serves as a meta-model over domain activity classes, capturing the structural notions of atomic and compound activities, based on the DnS pattern implementation in DUL. The aim is to allow the formal representation of activity interpretation models over activity classes that are generally characterized by intricate temporal associations, and where it is often the case that the aggregation of individual activities entails the existence of a new (composite) activity. To this end, two types of implemented patterns were presented, namely specialisation and composition that provide well-defined and interoperable activity models that satisfy common interpretation requirements in activity recognition domains. We also

elaborated on the implementation of a prototype framework for the generation of domain-dependent instantiations of the activity patterns in the form of SPARQL `CONSTRUCT` graph patterns (SPIN rules).

Future work involves the further extension of the activity patterns to capture more complex activities whose recognition requires additional information, such as cardinality and ordering (e.g. sequential, interleaving) constraints. In the near future, we also plan to provide an ontology API (e.g. in Sesame[3]) for enabling users to define pattern instantiations without going into the details of the implementation of the activity patterns. As far as the prototype framework is concerned, we are currently investigating the possibility of integrating and adapting the SPARQL generation algorithm in existing SPARQL-oriented complex activity processing and recognition frameworks, such as in [27] that combines SPARQL and Prova rules[4] and in [2] for processing RDF streams. In that way, the SPARQL query sets of such frameworks can be dynamically extended and adapted to reflect changes in the recognition logic, e.g. after learning new patterns.

Acknowledgement. This work has been supported by the EU FP7 project Dem@Care: Dementia Ambient Care – Multi-Sensing Monitoring for Intelligent Remote Management and Decision Support under contract No. 288199.

References

1. Allen, J.F.: Towards a general theory of action and time. Artif. Intell. 23(2), 123–154 (1984), http://dx.doi.org/10.1016/0004-3702(84)90008-0
2. Anicic, D., Fodor, P., Rudolph, S., Stojanovic, N.: Ep-sparql: a unified language for event processing and stream reasoning. In: Proceedings of the 20th International Conference on World Wide Web, WWW 2011, pp. 635–644. ACM, New York (2011), http://doi.acm.org/10.1145/1963405.1963495
3. Barbieri, D.F., Braga, D., Ceri, S., Valle, E.D., Grossniklaus, M.: Querying rdf streams with c-sparql. SIGMOD Rec. 39(1), 20–26 (2010)
4. Beltran, V., Arabshian, K., Schulzrinne, H.: Ontology-based user-defined rules and context-aware service composition system. In: García-Castro, R., Fensel, D., Antoniou, G. (eds.) ESWC 2011 Workshops. LNCS, vol. 7117, pp. 139–155. Springer, Heidelberg (2012)
5. Bolles, A., Grawunder, M., Jacobi, J.: Streaming SPARQL - extending SPARQL to process data streams. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 448–462. Springer, Heidelberg (2008)
6. Chen, L., Nugent, C.D.: Ontology-based activity recognition in intelligent pervasive environments. International Journal of Web Information Systems 5(4), 410–430 (2009)
7. Chen, L., Nugent, C.D., Wang, H.: A knowledge-driven approach to activity recognition in smart homes. IEEE Trans. on Knowl. and Data Eng. 24(6), 961–974 (2012)

[3] http://www.openrdf.org/
[4] https://prova.ws/

8. Debattista, J., Scerri, S., Rivera, I., Handschuh, S.: Ontology-based rules for recommender systems. In: SeRSy, pp. 49–60 (2012)
9. DOLCE+DnS Ultralite (DUL) ontology, http://www.loa.istc.cnr.it/ontologies/DUL.owl
10. Gangemi, A., Mika, P.: Understanding the semantic web through descriptions and situations. In: Meersman, R., Schmidt, D.C. (eds.) CoopIS/DOA/ODBASE 2003. LNCS, vol. 2888, pp. 689–706. Springer, Heidelberg (2003)
11. van Hage, W.R., Malaisé, V., Segers, R., Hollink, L., Schreiber, G.: Design and use of the Simple Event Model (SEM). J. Web Sem. 9(2), 128–136 (2011)
12. Knublauch, H., Hendler, J.A., Idehen, K.: SPIN - overview and motivation. W3C member submission, World Wide Web Consortium (February 2011)
13. Maria, K., Vasilis, E., Grigoris, A.: S-CRETA: Smart classroom real-time assistance. In: Novais, P., Hallenborg, K., Tapia, D.I., Rodríguez, J.M.C. (eds.) Ambient Intelligence - Software and Applications. AISC, vol. 153, pp. 67–74. Springer, Heidelberg (2012), http://dx.doi.org/10.1007/978-3-642-28783-1_9
14. May, W., Alferes, J.J., Amador, R.: An ontology- and resources-based approach to evolution and reactivity in the semantic web. In: Meersman, R. (ed.) OTM 2005, Part II. LNCS, vol. 3761, pp. 1553–1570. Springer, Heidelberg (2005)
15. Motik, B., Cuenca Grau, B., Sattler, U.: Structured objects in OWL: representation and reasoning. In: Proceedings of the 17th International Conference on World Wide Web (WWW 2008), pp. 555–564. ACM, New York (2008)
16. Okeyo, G., Chen, L., Hui, W., Sterritt, R.: A hybrid ontological and temporal approach for composite activity modelling. In: Min, G., Wu, Y., Liu, L.C., Jin, X., Jarvis, S.A., Al-Dubai, A.Y. (eds.) TrustCom, pp. 1763–1770. IEEE Computer Society (2012)
17. Pan, F., Hobbs, J.R.: Time ontology in OWL. W3C working draft, W3C (September 2006), http://www.w3.org/TR/2006/WD-owl-time-20060927/
18. Patkos, T., Chibani, A., Plexousakis, D., Amirat, Y.: A production rule-based framework for causal and epistemic reasoning. In: Bikakis, A., Giurca, A. (eds.) RuleML 2012. LNCS, vol. 7438, pp. 120–135. Springer, Heidelberg (2012)
19. Patkos, T., Chrysakis, I., Bikakis, A., Plexousakis, D., Antoniou, G.: A reasoning framework for ambient intelligence. In: Konstantopoulos, S., Perantonis, S., Karkaletsis, V., Spyropoulos, C.D., Vouros, G. (eds.) SETN 2010. LNCS, vol. 6040, pp. 213–222. Springer, Heidelberg (2010), http://dx.doi.org/10.1007/978-3-642-12842-4_25
20. Riboni, D., Pareschi, L., Radaelli, L., Bettini, C.: Is ontology-based activity recognition really effective? In: 2011 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), pp. 427–431 (March 2011)
21. Riboni, D., Bettini, C.: Cosar: hybrid reasoning for context-aware activity recognition. Personal Ubiquitous Comput. 15(3), 271–289 (2011), http://dx.doi.org/10.1007/s00779-010-0331-7
22. Riboni, D., Bettini, C.: Owl 2 modeling and reasoning with complex human activities. Pervasive and Mobile Computing 7(3), 379–395 (2011)
23. Scherp, A., Franz, T., Saathoff, C., Staab, S.: A core ontology on events for representing occurrences in the real world. Multimedia Tools and Applications 58(2), 293–331 (2012)
24. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. Web Semantics: Science, Services and Agents on the World Wide Web 5(2), 51–53 (2011)

25. Stevenson, G., Knox, S., Dobson, S., Nixon, P.: Ontonym: a collection of upper ontologies for developing pervasive systems. In: Proceedings of the 1st Workshop on Context, Information and Ontologies, CIAO 2009, pp. 9:1–9:8. ACM, New York (2009)
26. Tappolet, J., Bernstein, A.: Applied Temporal RDF: Efficient Temporal Querying of RDF Data with SPARQL. In: Aroyo, L., et al. (eds.) ESWC 2009. LNCS, vol. 5554, pp. 308–322. Springer, Heidelberg (2009)
27. Teymourian, K., Rohde, M., Paschke, A.: Fusion of background knowledge and streams of events. In: DEBS, pp. 302–313 (2012)
28. Wessel, M., Luther, M., Wagner, M.: The difference a day makes - recognizing important events in daily context logs. In: C&O:RR (2007)
29. Ye, J., Stevenson, G., Dobson, S.: A top-level ontology for smart environments. Pervasive and Mobile Computing 7(3), 359–378 (2011)

A Rule-Based Contextual Reasoning Platform for Ambient Intelligence Environments*

Assaad Moawad[1], Antonis Bikakis[2], Patrice Caire[1], Grégory Nain[1], and Yves Le Traon[1]

[1] University of Luxembourg, SnT
firstname.lastname@uni.lu
[2] Department of Information Studies, University College London
a.bikakis@ucl.ac.uk

Abstract. The special characteristics and requirements of intelligent environments impose several challenges to the reasoning processes of Ambient Intelligence systems. Such systems must enable heterogeneous entities operating in open and dynamic environments to collectively reason with imperfect context information. Previously we introduced Contextual Defeasible Logic (CDL) as a contextual reasoning model that addresses most of these challenges using the concepts of *context*, *mappings* and *contextual preferences*. In this paper, we present a platform integrating CDL with Kevoree, a component-based software framework for Dynamically Adaptive Systems. We explain how the capabilities of Kevoree are exploited to overcome several technical issues, such as communication, information exchange and detection, and explain how the reasoning methods may be further extended. We illustrate our approach with a running example from Ambient Assisted Living.

Keywords: contextual reasoning, distributed reasoning, Ambient Intelligence, system development.

1 Introduction

Ambient Intelligence (AmI) constitutes a new paradigm of interaction among agents acting on behalf of humans, smart objects and devices. Its goal is to transform our living and working environments into *intelligent spaces* able to adapt to changes in contexts and to their users' needs and desires. This requires augmenting the environments with sensing, computing, communicating and reasoning capabilities. AmI systems are expected to support humans in their every day tasks and activities in a personalized, adaptive, seamless and unobtrusive fashion [8]. Therefore, they must be able to reason about their *contexts*, i.e. with any information relevant to the interactions between the users and system.

The imperfect nature of context, and the special characteristics of AmI environments impose several challenges in the reasoning tasks. Henricksen and

* The present research is supported by the National Research Fund, Luxembourg, CoPAInS project (code: CO11/IS/1239572).

L. Morgenstern et al. (Eds.): RuleML 2013, LNCS 8035, pp. 158–172, 2013.

Indulska [15] characterize four types of imperfect context: *unknown*, *ambiguous*, *imprecise*, and *erroneous*. Sensor or connectivity failures, which are inevitable in wireless connections, result in situations that not all context data is available at any time. When data about a context property comes from multiple sources, then context may become ambiguous. Imprecision is common in sensor-derived information, while erroneous context arises as a result of human or hardware errors. Context is typically distributed among agents with different views of the environment that use different languages to describe it. Due to the highly dynamic and open nature of the environments and the unreliable wireless communications, agents do not typically know a priori all other entities present at a specific time instance, nor can they communicate directly with all of them. Despite these restrictions, they must be able to reach common conclusions about the state of the environment and collectively take context-aware decisions.

In previous works we introduced a new nonmonotonic logic, *Contextual Defeasible Logic* (CDL), tailored to the specific requirements of AmI systems. We presented its language and argumentation semantics, proved its formal properties [4], and developed algorithms for distributed query evaluation [7]. This paper is a further step towards reasoning with CDL and making it real in an AmI environment. Here, we focus on scenarios from *Ambient Assisted Living* (AAL), though the same findings may be applied to any subfield of AmI. AAL is a relatively new research and application domain focused on the technologies and services to enhance the quality of life of people with reduced autonomy, such as the elderly. In the framework of the CoPAInS project[1], such problematic is studied to evaluate the tradeoffs to be made in AAL systems [21], particularly as they pertain to conviviality, privacy and security [9]. Creating this bridge between CDL and AAL requires addressing issues, such as dynamicity, adaptability, detection and communication between devices, and is therefore not trivial. This prompts our research question: *How to deploy CDL in real AmI environments?*

To address this question, we created a platform that maps the CDL model to the business view of AAL using Kevoree [10]. Kevoree is designed to facilitate the development of Distributed Dynamically Adaptive Systems. Particular features of Kevoree make it a suitable for our needs: ability to implement heterogeneous entities as independent *nodes*; use of communication *channels* to enable message exchanges between nodes; support for shared models to enable common representations for different types of nodes; adaptive capabilities to fit with the open and dynamic nature of AAL.

The contribution of this work is twofold: (a) we describe solutions for the deployment of a theoretical model (CDL) in real AmI environments; and (b) we provide an AAL platform, usable by anyone to test and implement AAL scenarios. The rest of the paper is structured as follows. Section 2 describes our running AAL example. Sections 3 and 4 present the theoretical and technical background of this work, namely CDL and Kevoree. Section 5 describes the integration of CDL with Kevoree in our novel AAL platform. Section 6 presents related work, and Section 7 concludes and presents our plans for future work.

[1] `http://wwwen.uni.lu/snt/research/serval/projects/copains`

2 An Ambient Assisted Living Example

In this section, we present an Ambient Assisted Living (AAL) scenario, part of a series of scenarios validated by HotCity, the largest WI-FI network in Luxembourg, in the Framework of our CoPAInS project (Conviviality and Privacy in Ambient Intelligence Systems).

In our scenario, visualized in Figure 1, the eighty-five years old Annette is prone to heart failures. The hospital installed a Home Care System (HCS) at her place. One day, she falls in her kitchen and cannot get up. The health bracelet she wears gets damaged and sends erroneous data, e.g., heart beat and skin temperature, to the HCS. Simultaneously, the system analyzes Annette's activity captured by the Activity Recognition Module (ARM). Combining all the information to Annette's medical profile, and despite the normal values transmitted by Annette's health bracelet, the system infers an emergency situation. It contacts the nearby neighbors asking them to come and help.

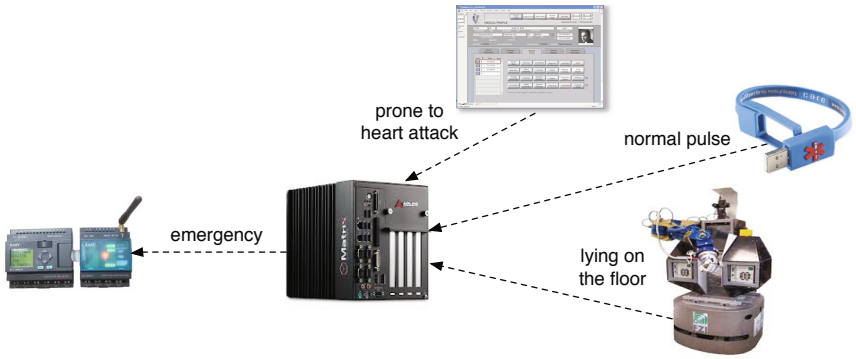


Fig. 1. Context Information flow in the scenario

This scenario exemplifies challenges raised when reasoning with the available context information in Ambient Intelligence environments. Furthermore, it highlights the difficulties in making correct context-dependent decisions.

First, context knowledge may be erroneous. In our example, the values transmitted by the health bracelet for Annette's heart beat and skin temperature, are not valid, thereby leading to a conflict about Annette's current condition. Second, local knowledge is incomplete, in the sense that none of the agents involved has immediate access to all the available context information. Third, context knowledge may be ambiguous; in our scenario, the HCS receives mutually inconsistent information from the ARM and the health bracelet. Fourth, context knowledge may be inaccurate; for example, Annette's medical profile may contain corrupted information. Finally, devices communicate over a wireless network. Such communications are unreliable due to the nature of wireless networks, and are also restricted by the range of the network. For example, the health bracelet may not be able to transmit its readings to HCS due to a damaged transmitter.

In the next section, we analyze how CDL enables devices to model and reason with such imperfections.

3 Contextual Defeasible Logic

Contextual Defeasible Logic (*CDL*) is a distributed rule-based approach for contextual reasoning, which was recently proposed as a reasoning model for Ambient Intelligence systems [4]. CDL adopts ideas from:

- *Defeasible Logic* [3] - it is rule-based, skeptical, and uses priorities to resolve conflicts among rules;
- *Multi-Context Systems* (MCS, [13,12]) - logical formalizations of distributed context theories connected through mapping rules, which enable information flow between contexts. In MCS, a *context* can be thought of as a logical theory - a set of axioms and inference rules - that models local knowledge.

Below, we present the representation model of CDL and explain how it fits with the special characteristics and requirements of Ambient Intelligence environments. We also present an operational model of CDL in the form of a query evaluation algorithm, which we implemented in Kevoree.

3.1 Representation Model

In CDL, the MCS model is extended with defeasible rules and a preference relation on the system contexts. In CDL, a MCS C is a set of contexts C_i: A context C_i is defined as a tuple of the form (V_i, R_i, T_i), where V_i is the vocabulary of C_i, R_i is a set of rules, and T_i is a preference ordering on C.

V_i is a set of positive and negative literals of the form $(c_i : a_i)$ and $\sim (c_i : a_i)$, which denotes the negation of $(c_i : a_i)$. Each context uses a distinct vocabulary, i.e. $V_i \cap V_j = \emptyset$ *iff* $i \neq j$. This reflects the fact that each entity (e.g. device in an Ambient Intelligence environment) may use its own terminology. We should note, though, that the proposed model may also enable different contexts to use common literals (e.g. URIs) by adding a context identifier, e.g. as a prefix, in each such literal and using appropriate mappings to associate them to each context.

R_i consists of a set of *local rules* and a set of *mapping rules*. The body of a local rule is a conjunction of *local* literals (literals that are contained in V_i), while its head is labeled by a local literal. There are two types of local rules:

- Strict rules, of the form

$$r_i^l : (c_i : a^1), \ldots, (c_i : a^{n-1}) \rightarrow (c_i : a^n)$$

  They express sound local knowledge and are interpreted in the classical sense: whenever the literals in the body of the rule $((c_i : a^1), \ldots, (c_i : a^{n-1}))$ are strict consequences of the local theory, then so is the conclusion of the rule $((c_i : a^n))$. Strict rules with empty body denote factual knowledge.
- Defeasible rules, of the form

$$r_i^d : (c_i : a^1), \ldots, (c_i : a^{n-1}) \Rightarrow (c_i : a^n)$$

  They are used to express uncertainty: a defeasible rule cannot be applied to support its conclusion if there is adequate contrary evidence.

Mapping rules associate local literals of C_i with literals from the vocabularies of other contexts (*foreign literals*). The body of each such rule is a conjunction of local and foreign literals, while its head is labeled by a local literal. Mapping rules are modeled as defeasible rules of the form:

$$r_i^m : (c_j : a^1), \ldots, (c_k : a^{n-1}) \Rightarrow (c_i : a^n)$$

A mapping rule associates literals from different contexts (e.g. $(c_j : a^1)$ from C_j and $(c_k : a^{n-1})$ from C_k), with a local literal of the context that has defined r_i, which labels the head of the rule (here $(c_i : a^n)$). By representing mappings as defeasible rules, we can deal with ambiguities and inconsistencies caused by importing mutually conflicting information from different contexts.

Finally, each context C_i defines a strict total preference ordering T_i on C to express its confidence in the knowledge it imports from other contexts:

$$T_i = [C_k, C_l, ..., C_n]$$

C_k is preferred to C_l by C_i, if C_k precedes C_l in T_i. The strict total ordering enables resolving all potential conflicts that may arise from the interaction of contexts through their mapping rules. In a later version of CDL [6], T_i is defined as a partial preference order on C, which enables handling incomplete preference information. For sake of simplicity, we adopt here the original definition of T_i.

Example. The scenario described in Section 2 may be modeled as follows in CDL. We consider 5 different contexts: *sms* for the SMS system, *hcs* for the Home Care Syste, *arm* for the activity recognition module, *br* for the bracelet, and *med* for the medical profile. *sms* has only one mapping rule according to which, when the Home Care System detects an emergency situation, the SMS system dispatches messages to a prescribed list of mobile phone numbers:

$r_{sms}^m : (hcs : emergency) \Rightarrow (sms : dispatchSMS)$

The Home Care system imports information from the activity recognition module, the bracelet and Annete's medical profile to detect emergency situations using two mapping rules:

$r_{hcs}^{m1} : (br : normalPulse) \Rightarrow \neg(hcs : emergency)$
$r_{hcs}^{m2} : (arm : lyingOnFloor), (med : proneToHA) \Rightarrow (hcs : emergency)$

The factual knowledge of the other three modules is modeled using local rules with empty body:

$r_{br}^l :\rightarrow (br : normalPulse)$
$r_{arm}^l :\rightarrow (arm : lyingOnFloor)$
$r_{med}^l :\rightarrow (med : proneToHA)$

The Home Care System is configured to give highest priority to information imported by the medical profile and lowest priority to the bracelet:

$T_{hcs} = [med, arm, br]$

3.2 Distributed Query Evaluation

In [4] we presented an argumentation semantics of CDL, while in [7] we provided four algorithms for query evaluation. $P2P\_DR$ is one of these algorithms, which is called when a context C_i is queried about the truth value of one of its local literals $(c_i : a_i)$, and roughly proceeds as follows:

Algorithm 1. $P2P\_DR$

if $(c_i : a_i)$ (**or** $\neg(c_i : a_i)$) is derived as a conclusion of the local rules of C_i **then**
  return *true* (or *false* resp.)
else
  for all rules r_i in c_i that have $(c_i : a_i)$ or $\neg(c_i : a_i)$ in their heads **do**
    if r_i is applicable **then**
      Compute the Supportive Set of r_i, SS_{r_i}
  Compute the Supportive Sets of $(c_i : a_i)$, $SS_{(c_i:a_i)}$, and $\neg(c_i : a_i)$, $SS_{\neg(c_i,a_i)}$
  if $SS_{(c_i:a_i)}$ is *stronger* than $SS_{\neg(c_i,a_i)}$ with respect to T_i **then**
    return *true*
  else
    return *false*

We should note that a rule r_i is applicable when for all its body literals we have obtained positive truth values. SS_{r_i} is the union of the foreign literals with the Supportive Sets of the local literals contained in the body of r_i, while $SS_{(c_i:a_i)}$ is the *strongest* between the Supportive Sets of the rules with head $(c_i : a_i)$. A set of literals S_1 is *stronger* than set S_2 w.r.t. T_i *iff* there is a literal l in S_2, such that all literals in S_1 are stronger than l w.r.t. T_i. A literal $(c_k : a)$ is stronger than literal $(c_l : b)$ w.r.t. T_i *iff* C_k precedes C_l in T_i.

Example (continued). In our running example, a query to *sms* about $(sms : dispatchSMS)$ will initiate a second query to *hcs* about $(hcs : emergency)$. The second query will in turn initiate three more queries: a query to *br* about $(br : normalPulse)$; a query to *arm* about $(arm : lyingOnFloor)$; and a query to *med* about $(med : proneToHA)$. $P2P\_DR$ will return *true* for the latter three queries, and will compute $SS_{(hcs:emergency)} = \{(arm : lyingOnFloor), (med : proneToHA)\}$ and $SS_{\neg(hcs:emergency)} = \{(br : normalPulse)\}$. W.r.t. T_{hcs}, all elements of $SS_{(hcs:emergency)}$ are stronger than $(br : normalPulse)$, therefore $P2P\_DR$ will return a positive truth value for $(hcs : emergency)$, and the same value for $(sms : dispatchSMS)$ too, as there is no rule that supports its negation.

As shown in the example above, CDL may deal with several of the challenges of Ambient Intelligence environments, such as uncertainty, ambiguity, and erroneous data. There are still, though, some questions that need to be addressed in order to fully deploy CDL in real environments: how do the devices actually detect and communicate with each other? and how can we achieve dynamicity and adaptability? In the next sections, we describe how we addressed such questions by integrating CDL in the software platform of Kevoree.

4 Kevoree - A Component Based Software Platform

On the one hand, in Ambient Assisted Living (AAL), systems need to be adapted to users preferences and contexts. They also need to combine various data and reason about it, but the imperfect nature of context makes this task very challenging. Returning to our use case, the HCS receives data from different devices, and many situations may occur causing the data to be erroneous, e.g., Annette may have left her health bracelet next to her bed instead of wearing it, or the battery capability may be weak and preventing the bracelet from transmitting any data.

On the other hand, CDL allows to manage uncertainty and reason about it. The problem remains to apply such theoretical tools to the AAL domain in order to solve the very concrete challenges affecting patients. In this section, we present the Kevoree environment, which we use to address such issues by implementing the CDL reasoning model. This is illustrated in Figure 2.

Fig. 2. Kevoree bridges the AAL needs to the theoretical model of CDL

4.1 Kevoree: Modeling Framework and Components

Kevoree [10] is an open-source environment that provides means to facilitate the design and deployment of Distributed Dynamically Adaptive Systems, taking advantage of Models@Runtime [22] mechanisms throughout the development process.

This development platform is made of several tools, among which the Kevoree Modeling Framework (KMF) [11], a model editor (to assemble components to create an application), and several runtime environments, from Cloud to JavaSE or Android platforms. The component model of Kevoree defines several concepts. The rest of this section describes the most interesting ones in relation to the content of this paper.

The **Node** (in grey in figure 3) is a topological representation of a Kevoree runtime. There exist different types of nodes (e.g.: JavaSE, Android, etc.) and a system can be composed of one, or several distributed heterogeneous instances of execution nodes.

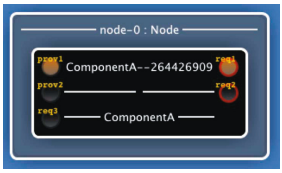


Fig. 3. A component instance, inside the node instance on which it executes

Component instances are deployed and run on a node instance, as presented on figure 3. Components may also be of different types, and one or more, heterogeneous or not, component instances may run on

a single node. Components declare **Ports** (rounds on left and right sides of the component instance) for provided and required services, and input and output messages. The ports are used to communicate with other components of the system.

Groups (top shape in figure 4) are used to share models (at runtime) between execution platforms (i.e. nodes). There are different types of Groups, each of which implements a different synchronization / conciliation / distribution algorithm. Indeed, as the model shared is the same for all the nodes, there may be some concurrent changes on the same model, that have to be dealt with.

Finally (for the scope of this paper), **Channels** (bottom shape in figure 4) handle the semantics of a communication link between two or more components. In other words, each type of channel implements a different means to transport a message or a method call from component A to component B, including local queued message list, TCP/IP sockets connections, IMAP/SMTP mail communications, and various other types of communication.

Fig. 4. An instance of Group on top, of Channel on the bottom

4.2 Kevoree Critical Features

Kevoree appears to be an appropriate choice to provide solutions for the development of Ambient Intelligence systems, as it can deal with their dynamic nature. In such systems, agents are often autonomous, reactive and proactive in order to collaborate and fulfil tasks on behalf of their users.

In Kevoree, an agent is represented as a node that hosts one or more component instances. The node is responsible for the communication with other nodes by making use of the synchronization Group. Some group types implement algorithms with auto-discovery capabilities, making nodes and their components dynamically appear in the architecture model of the overall system. The fact that a new node appears in the model means that an agent is reachable, but it does not necessarily mean that it participates in any interaction. The component instances of a node provide the services for the agent. Therefore, for an agent to take part in a collaborative work, the ports of the component instances it hosts have to be connected to some ports of other agents' components.

Some features of Kevoree make it particularly suitable for our needs. First, it enables the implementation, deployment and management of heterogeneous entities as independent *nodes*. Second, it uses communication *channels* to enable the exchange of messages among the distributed components. Third, it offers a common and shared representation model for different types of nodes. Finally, it is endowed with adaptive capabilities and auto-discovery, which fit with the open and dynamic nature of AmI environments.

In the next section, we detail how we exploit the features of Kevoree and integrate them with CDL to create our AAL platform.

5 The AAL Platform

In this section, we explain how CDL and Kevoree are integrated in our AAL platform. We should note that the parts of CDL, which were not directly mapped to existing elements Kevoree, were implemented in Java.

5.1 Query Component

In our platform, the notion of context, as this is described in Section 3, is implemented by a new component type that we developed, called *Query Component*. This component has two inputs: *Console In* and *Query In*, and two outputs: *Console out* and *Query Out*. The Query Component has three properties: a *Name*, an *initial preference address* and an *initial knowledge base address*. In Kevoree, each instance must have a unique name. In our platform, we use this unique name to specify the sender or the recipient of a query. The preference address and the knowledge base address contain the addresses of the files to be loaded when the component starts. The knowledge base file contains the rule set of a context, while the preference file contains the preference order of the context implemented as a list.

Each component has two console (in/out) and two query (in/out) ports. The console input port is used to send commands to the component, e.g. to update its knowledge base or change its preference order. The outputs of the commands are sent out to the console output port. The query in/out ports are used when a component is sending/receiving queries to/from other components. Queries are sent via the "Query out" port and responses are received via "Query In".

Internally, the Query Component has some private variables, which represent its knowledge base, the preference order and a list of query servant threads currently running on it. When the component receives a new query, it creates a new query servant thread dedicated to solve the query and adds it to the list of currently running query threads. When this thread reports back the result of the query, it is killed and removed from the list.

5.2 Query Servant

When a query servant thread is created, it is always associated with an ID and with the query containing the literal to be solved, and it is added to the list of running threads of the query component. In accordance with the $P2P\_DR$ algorithm that we described in Section 3, the query servant model works as follows:

1. The first phase consists of trying to solve the query locally using the local knowledge base of the query component. If a positive/negative truth value is derived locally, the answer is returned and the query servant terminates.
2. The second phase consists of enumerating the rules in the knowledge base that support the queried literal as their conclusion. For each such rule, the query servant initiates a new query for each of the literals that are in the body

of the rule. For foreign literals, the queries are dispatched to the appropriate remote components. After initiating the queries, the query servant goes into an idle state through the java command "wait()".

3. When responses are received, the query servant thread is notified. Phase two is repeated again, but this time using the rules that support the negation of the queried literal.
4. The last step is to resolve the conflict by comparing the Supportive Sets of the rules that support the queried literals with the Supportive Sets of the rules that support its negation using the preference order. The result is reported back to the query component.

5.3 Query Class and Loop Detection Mechanism

The query Java class that we developed for our platform has the following attributes: the queried literal, the name of the component that initiated this query (*query owner*), the name of the component to which the query is addressed (*query recipient*), the id of the query servant thread that is responsible for solving this query, a set of supportive sets, and a list that represents the history of the query. The history is used to track back to the origin of the query by a loop detection mechanism, which we have integrated in *P2P_DR*.

As *P2P_DR* is a distributed algorithm, we cannot know a-priori whether a query will initiate an infinite loop of messages. The loop detection mechanism that we developed detects and terminates any infinite loops. The simple case is when a literal $(c_i : a)$ in component C_i depends on literal $(c_k : b)$ of component C_k, and vice-versa. The loop detection mechanism works as follows: each time the query servant inquires about a foreign literal to solve the current query, it first checks that the foreign literal in question does not exist in the history of the current query, and if not, it generates a new query for the foreign literal by integrating the history of the current query into the history of the new one. This way, a query servant is only allowed to inquire about new literals.

5.4 Running Example

Applying the above methodology on the running example described in section 2, we created 5 Query Component instances, each one representing one of the devices or elements of the scenario: the sms module, the bracelet, the medical profile, the ARM and the Home Care System. According to the scenario, the sms module must determine whether to send messages to the neighbors according to a predefined set of rules. Using a console component of Kevoree that we attached to the sms module, we were able to initiate queries on the sms module.

Figure 5 shows our experimental setup, which involves the 5 query components and the console component connected to the sms module (*FakeConsole*). Note that, all query input and output ports of the query components are connected to each other via the same message channel called *queryChannel*, to allow any component to communicate and send queries to any other component.

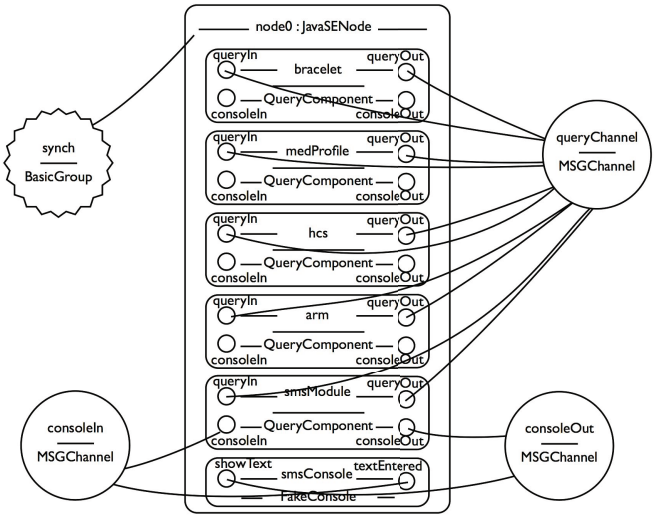


Fig. 5. The running example implemented, a snapshot of the Kevoree Editor

Before pushing the model from the Kevoree editor to the Kevoree runtime (i.e.: the node that will host the instances), we setup the properties of the components to initialize their knowledge bases and preference orders as described in Table 1, and according to the CDL representation model that we present in Section 3. For instance, the *sms* component is initiated with a knowledge base containing one mapping rule ($M1$) that states that if ($hcs : emergency$) of *hcs* is true, then ($sms : dispatchSMS$) of the sms module will also be true. HCSPref.txt contains the preference order of *hcs*, according to which the information imported by the medical profile is preferred to that coming from the ARM, which is in turn preferred to that coming from the bracelet.

After pushing the model to the Kevoree runtime, a console appears allowing us to interact with the sms module. We initiate a query about ($sms : dispatchSMS$), and we get *true* as a response. In fact, what happens in the

Table 1. Initialization of the components of the running example

File Name	**File contents**
smsModuleKB.txt	M1: (hcs:emergency) → (sms:dispatchSMS)
BraceletKB.txt	L1: → (br:normalPulse)
MedProfileKB.txt	L1: → (med:proneToHA)
ArmKB.txt	L1: → (arm:lyingOnFloor)
HCSKB.txt	M1: (br:normalPulse) ⇒ ¬(hcs:emergency) M2: (arm:lyingOnFloor), (med:proneToHA) ⇒ (has:emergency)
HCSPref.txt	med, arm, br

back-end is that a query servant starts on the sms module to solve the query. The query servant initiates, then, a new query for (hcs : $emergency$). In the knowledge base of hcs, there is one rule supporting this literal, and another one supporting its negation. hcs evaluates both rules and resolves the conflict using its preference order. Finally, it sends back the result of the query to the first query servant, which in turn computes and returns a positive truth value for (sms : $dispatchSMS$). The full interaction is displayed in figure 6.

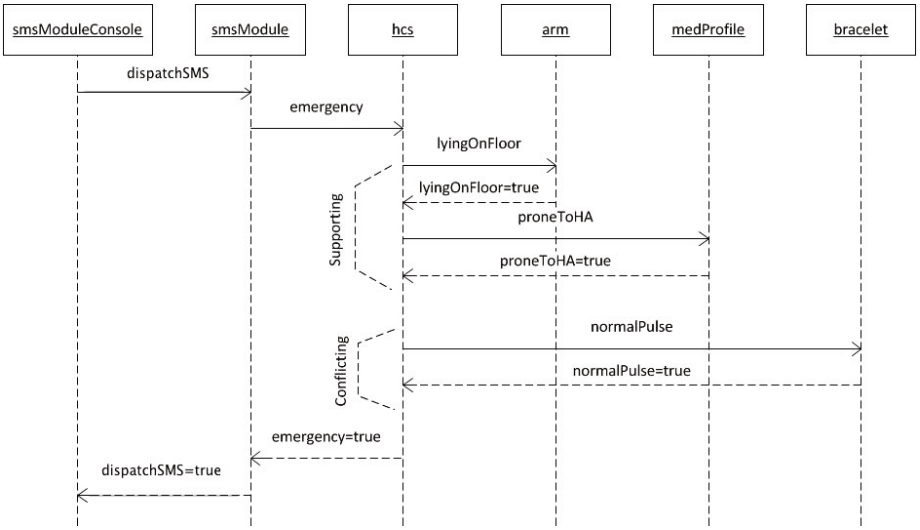


Fig. 6. Execution of the running example

5.5 Limitations

Our platform still has some technical limitations. As it deals with real components, we must assume limited memory, battery, computation and power resources. These limitations vary widely from a component to another depending on the nature of the component, its size and its technical complexity. For the current implementation, we have limited the knowledge base size to a maximum of 500 literals and rules. We have also limited the time-out for 10 seconds, so that if a component does not receive an answer to its query within 10 seconds, the corresponding thread server will send a time-out response, and the query will automatically expire. This limits the maximum number of hops that a query can make before it expires, which in turn limits the communication resources, as some communication channel might not be free (over sms for example). With the current settings, we can easily implement small-scale AAL scenarios. However, dealing with more complex scenarios requires a more scalable methodology. To address such needs, we are already working on solutions that offer trade-offs between computation time, memory and communication between devices, and we are redesigning our algorithms so that they able to adapt between different strategies depending on the available resources.

6 Related Work

Rule-based approaches offer several benefits with respect to reasoning about context in Ambient Intelligence environments, such as *simplicity and flexibility*, *formality*, *expressivity*, *modularity*, *high-level abstraction* and *information hiding* [5]. Various logics have been proposed so far for such purposes including: First Order Logic [24,14], Logic Programming [25,1], Answer Set Programming [20] and Defeasible Logic [2]. Classical reasoning approaches (e.g. First Order Logic) are based on the assumption of perfect knowledge of context, which, as we explained in Section 1, is not valid in Ambient Intelligence environments. Nonmonotonic approaches, including the one that we propose in this paper, enable reasoning with imperfect context, adding though additional complexity overhead to the reasoning process. Although the complexity of the algorithm that we use for query evaluation is exponential [4], it is among our future plans to design new algorithms that will exploit the linear complexity of Defeasible Logic.

Although Ambient Intelligence environments are distributed by nature, all systems that are cited above are based on centralized architectures: a central entity is responsible for collecting relevant context data from all sensors and devices operating in the same environment, and for conducting the contextual reasoning tasks. The *shared memory* and *blackboard* models that were used by other systems (e.g [16,18,17]) are also based on the assumption of a central place where all relevant data is collected and processed. However, in such environments context changes may be very frequent, devices may join or disconnect at random times and without prior notice, while wireless communications are unreliable and restricted by the range of the transmitters. Moreover, privacy restrictions may be applied by the users, according to which part of the data stored in a device must remain local. Therefore, a totally distributed model, such as the one that we propose here, fits better with such requirements and needs.

7 Conclusion

In this paper, we address some of the challenges imposed by the special characteristics and requirements of intelligent environments to the reasoning processes of Ambient Intelligence systems. Such systems must enable heterogeneous entities, which operate in open and dynamic environments, to collectively reason with imperfect context information. We build on previous work, in which we introduced Contextual Defeasible Logic as a contextual reasoning model to address most of these challenges using the concepts of *context*, *mappings* and *contextual preferences*. In this paper, we first introduce the implementation of the logic in Kevoree, a component-based software platform for Distributed Dynamically Adaptive Systems. Second, we present an Ambient Assisted Living (AAL) scenario, which we use as a running example to present the main aspects of our platform. We describe how we implemented the reasoning model of CDL in Kevoree, and explain how the capabilities of Kevoree are exploited to overcome several technical issues, such as communication, information exchange and detection. Third, we discuss the additional technical issues that arise from the

deployment of CDL in real environments, and propose ways to resolve them. Finally, we emphasize that we provide a platform, which anyone may use to test and implement scenarios from any field of Ambient Intelligence.

In the future, we plan to extend CDL to support shared pieces of knowledge, which are directly accessible by all system contexts, and implement this extension in Kevoree using its *groups* feature (see section 4). This will enable different devices operating in an Ambient Intelligence environment to maintain a common system state. We also plan to develop and implement reactive (bottom-up) reasoning algorithms, which will be triggered by certain events or changes in the environment. Such types of algorithms fit better with the adaptive nature of Ambient Intelligence systems, and may be particularly useful in AAL contexts. We will also study the integration of a low-level context layer in our platform, which will process the available sensor data and feed the rule-based reasoning algorithms with appropriate values for the higher-level predicates. For this layer, we will investigate the Complex Event Processing (*CEP*) methodology [19], which combines data from multiple sources to infer higher-level conclusions, and we will build on top of previous works that study the integration of CEP and reaction rules [23]. We will test and evaluate all our deployments and extensions to our platform in the Internet of Things Laboratory of the Interdisciplinary Centre for Security, Reliability and Trust (SnT) in Luxembourg. It is also among our plans to use our platform to evaluate tradeoffs among requirements of AAL systems, e.g., privacy, security, usability/conviviality and performance. Finally, we plan to investigate how the same reasoning methods may be applied to other application areas with similar requirements, such as the Semantic Web and Web Social Networks.

References

1. Agostini, A., Bettini, C., Riboni, D.: Experience Report: Ontological Reasoning for Context-aware Internet Services. In: Proceedings of PERCOMW 2006. IEEE Computer Society, Washington, DC (2006)
2. Antoniou, G., Bikakis, A., Karamolegou, A., Papachristodoulou, N., Stratakis, M.: A context-aware meeting alert using semantic web and rule technology. International Journal of Metadata Semantics and Ontologies 2(3), 147–156 (2007)
3. Antoniou, G., Billington, D., Governatori, G., Maher, M.J.: Representation results for defeasible logic. ACM Transactions on Computational Logic 2(2), 255–287 (2001)
4. Bikakis, A., Antoniou, G.: Defeasible Contextual Reasoning with Arguments in Ambient Intelligence. IEEE Trans. on Knowledge and Data Engineering 22(11), 1492–1506 (2010)
5. Bikakis, A., Antoniou, G.: Rule-based contextual reasoning in ambient intelligence. In: Dean, M., Hall, J., Rotolo, A., Tabet, S. (eds.) RuleML 2010. LNCS, vol. 6403, pp. 74–88. Springer, Heidelberg (2010)
6. Bikakis, A., Antoniou, G.: Partial preferences and ambiguity resolution in contextual defeasible logic. In: Delgrande, J.P., Faber, W. (eds.) LPNMR 2011. LNCS, vol. 6645, pp. 193–198. Springer, Heidelberg (2011)
7. Bikakis, A., Antoniou, G., Hassapis, P.: Strategies for contextual reasoning with conflicts in Ambient Intelligence. Knowledge and Information Systems 27(1), 45–84 (2011)
8. Cook, D.J., Augusto, J.C., Jakkula, V.R.: Ambient intelligence: Technologies, applications, and opportunities. Pervasive and Mobile Computing, 277–298 (2009)

9. Efthymiou, V., Caire, P., Bikakis, A.: Modeling and evaluating cooperation in multi-context systems using conviviality. In: Proceedings of BNAIC 2012 The 24th Benelux Conference on Artificial Intelligence, pp. 83–90 (2012)
10. Fouquet, F., Barais, O., Plouzeau, N., Jézéquel, J.M., Morin, B., Fleurey, F.: A Dynamic Component Model for Cyber Physical Systems. In: 15th International ACM SIGSOFT Symposium on Component Based Software Engineering, Bertinoro, Italie (July 2012), http://hal.inria.fr/hal-00713769
11. Fouquet, F., Nain, G., Morin, B., Daubert, E., Barais, O., Plouzeau, N., Jézéquel, J.-M.: An Eclipse Modelling Framework Alternative to Meet the Models@Runtime Requirements. In: France, R.B., Kazmeier, J., Breu, R., Atkinson, C. (eds.) MODELS 2012. LNCS, vol. 7590, pp. 87–101. Springer, Heidelberg (2012)
12. Ghidini, C., Giunchiglia, F.: Local Models Semantics, or contextual reasoning=locality+compatibility. Artificial Intelligence 127(2), 221–259 (2001)
13. Giunchiglia, F., Serafini, L.: Multilanguage hierarchical logics, or: how we can do without modal logics. Artificial Intelligence 65(1) (1994)
14. Gu, T., Pung, H.K., Zhang, D.Q.: A Middleware for Building Context-Aware Mobile Services. In: Proceedings of the IEEE Vehicular Technology Conference (VTC 2004), Milan, Italy (May 2004)
15. Henricksen, K., Indulska, J.: Modelling and Using Imperfect Context Information. In: Proceedings of PERCOMW 2004, pp. 33–37. IEEE Computer Society, Washington, DC (2004)
16. Khushraj, D., Lassila, O., Finin, T.: sTuples: Semantic Tuple Spaces. In: First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous 2004), pp. 267–277 (August 2004)
17. Korpipaa, P., Mantyjarvi, J., Kela, J., Keranen, H., Malm, E.J.: Managing Context Information in Mobile Devices. IEEE Pervasive Computing 02(3), 42–51 (2003)
18. Krummenacher, R., Kopecký, J., Strang, T.: Sharing Context Information in Semantic Spaces. In: Meersman, R., Tari, Z., Herrero, P. (eds.) OTM Workshops 2005. LNCS, vol. 3762, pp. 229–232. Springer, Heidelberg (2005)
19. Luckham, D.C.: The power of events - an introduction to complex event processing in distributed enterprise systems. ACM (2005)
20. Mileo, A., Merico, D., Pinardi, S., Bisiani, R.: A logical approach to home healthcare with intelligent sensor-network support. Comput. J. 53(8), 1257–1276 (2010)
21. Moawad, A., Efthymiou, V., Caire, P., Nain, G., Le Traon, Y.: Introducing conviviality as a new paradigm for interactions among IT objects. In: Proceedings of the Workshop on AI Problems and Approaches for Intelligent Environments, vol. 907, pp. 3–8. CEUR-WS.org (2012)
22. Morin, B., Barais, O., Nain, G., Jezequel, J.M.: Taming dynamically adaptive systems using models and aspects. In: Proceedings of the 31st International Conference on Software Engineering, ICSE 2009, Washington, DC, USA, pp. 122–132 (2009), http://dx.doi.org/10.1109/ICSE.2009.5070514
23. Paschke, A., Vincent, P., Springer, F.: Standards for complex event processing and reaction rules. In: Olken, F., Palmirani, M., Sottara, D. (eds.) RuleML - America 2011. LNCS, vol. 7018, pp. 128–139. Springer, Heidelberg (2011)
24. Ranganathan, A., Campbell, R.H.: An infrastructure for context-awareness based on first order logic. Personal Ubiquitous Comput. 7(6), 353–364 (2003)
25. Toninelli, A., Montanari, R., Kagal, L., Lassila, O.: A Semantic Context-Aware Access Control Framework for Secure Collaborations in Pervasive Computing Environments. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 473–486. Springer, Heidelberg (2006)

Extending an Object-Oriented RETE Network with Fine-Grained Reactivity to Property Modifications

Mark Proctor[1,2], Mario Fusco[2], and Davide Sottara[3]

Dept. of Electrical & Electronic Engineering, Imperial College London, London (UK)
m.proctor13@imperial.ac.uk
JBoss, a Division of Red Hat Inc.
mfusco@redhat.com
Biomedical Informatics Dept., Arizona State University, Scottsdale (AZ)
davide.sottara@asu.edu

Abstract. Managing rule chaining, especially in presence of recursion, is a common difficulty when authoring rule based applications. For this reason, production systems implement strategies such as refraction to control rule activation repeatability. In this paper, we present a related extension for an object-oriented version of the RETE algorithm, called property-based reactivity. This extension provides additional, finer grained control of repeatable rules, at the object property level. Patterns have control over which properties they will react to: by default, this is properties the pattern constrains on, but additional properties may be included or excluded using appropriate annotations in the rule base. The engine enhancement is based on a compile-time analysis of the rule base that minimizes the overhead on the language and the runtime execution. The correlation between the performance impact due to the strategy and the structure of the rules has been analyzed and benchmarked using an implementation based on the open source rule engine Drools.

Keywords: RETE, RETE-OO, production rules, pattern matching, rule engines, refraction.

1 Introduction

Reactive production rule systems typically offer very limited control in how rules that are being reevaluated are selected for firing. [3]. For lack of well defined, decoupled and configurable strategies, implementing appropriate semantics, users are forced to mix business and control logic, polluting their data models and rules. This approach, however, does not scale well with the complexity of the application: not only are the results less robust, with increased maintenance, but they are also inefficient. Control logic uses computational resources for evaluations that could otherwise be avoided. As the applications grow in size the ratio of business logic to control logic degrades: in severe cases, the author can spend more time telling a rulebase what not to do, rather than what to do. Different

L. Morgenstern et al. (Eds.): RuleML 2013, LNCS 8035, pp. 173–187, 2013.

solutions exist for this problem, depending on the nature of the underlying engine. This paper focuses its discussion on rule engines based on RETE networks [8], which are among the most used algorithm in production systems implementations. Moreover, it assumes an object-oriented rule engine using the semantics of the Java language in terms of classes and properties, as well as inheritance and subsumption testing. The engine is based on an adaptation of the RETE algorithm to work with Java and its type system, called RETE-OO, implemented in the open source rule engine Drools Expert[1]. In a RETE-OO network the type discrimination nodes come first, after the root node. In RETE-OO these are referred to as `ObjectTypeNodes` and their discrimination obeys Java `instanceof` behaviour. The `ObjectType` may be any Java class or interface. Thus a single fact may propagate to multiple `ObjectTypeNodes` depending on its inheritance hierarchy and implemented interfaces. By knowing the `ObjectType` for the propagation the engine can ensure type safety for both alpha and beta expression evaluation, an important advantage for any object-oriented RETE implementation. RETE-OO also provides the mapping and evaluation of alpha and beta node constraints to Java properties. Both simple and compound expressions are supported. Unlike other Java based production systems such as Jess[2], RETE-OO is able to work directly with Java objects, and does not require any form of shadowing or additional state saving for objects. The engine supports reactive (event) condition action rules, providing the usual working memory operations `insert`, `retract` and `modify` (update), redefined to use objects as working memory elements. In the basic version of the engine, the modification of a fact (object) always triggers a new propagation, rematching all the patterns whose object type subsumes at least one of types of the modified object itself. The modify is implemented as an `update-in-place` [11], rather than the traditional RETE [8] implementation of `retract` followed by `insert`. This paper proposes an algorithm that will allow the RETE network to react to working memory changes with a finer and more precise granularity: in particular, reevaluating only the patterns matching for those properties of objects that have been actually modified. This can be achieved by performing a compile-time analysis of the rule base and using the results to enrich the nodes of the RETE network with additional information. In turn, this will be used at runtime, to efficiently discriminate irrelevant or undesired modifications. The paper is organized as follows. Section 2 discusses related works and shows how property reactivity relates to execution control strategies such as refraction. Section 3 describes the general idea of property reactivity and its proposed integration in a RETE-OO network. Finally, Section 4, analyzes the impact of its implementation on the performance of a rule engine, presenting some benchmarks developed for this purpose.

[1] `http://www.jboss.org/drools`
[2] `http://herzberg.ca.sandia.gov/`

2 Related Works

A generic (production) rule engine executes a recognize-act cycle, where (i) facts present in the working memory are matched against rules, generating rule instance activations which are placed into an agenda, (ii) a conflict resolution strategy is used to choose one activation for execution, (iii) the actions defined by the selected rule are applied to the working memory [10]. These working memory actions may cause new activations, or even cancel previous ones, allowing rules to be "forward" chained for more complex inferences. A formal definition of this execution process is presented in [3]: a rule instance execution r is modeled as a transition $\langle E_j, s_j \rangle \rightarrow_r \langle E_{j+1}, s_{j+1} \rangle$, where E_j is the set of eligible rule activations (according to an eligibility strategy $\mathcal{E}$) in a given rule base and working memory state s_j. The execution of a rule program is then seen as a sequence of state transitions, as determined by a selection (conflict resolution) strategy $\mathcal{S}$ which, in any state, chooses the actual rule instance $r \in E_j \cap A_j$ for execution between those rule instances which are both eligible and applicable in that state. This sequence can potentially be infinite if, given the chosen $\mathcal{E}$ and $\mathcal{S}$, the execution of rule r in a state $\langle E_j, s_j \rangle$ will necessarily lead to a state $\langle E_k, s_k \rangle_{,k>j}$ where r will be selected again.

While infinite loops should be avoided, controlled loops may be more or less desirable, depending on the use case. Consider for example the rules in 1.1[3]: the former is supposed to be applied only once for each employee, but may potentially result in an endless loop. The latter instead, is actually expected to be activated a variable, but finite, number of times.

Listing 1.1. Looping Rule

```
rule   Award senior employees
when $e : Employee( hiringYear < 2010 )
then modify( $e ) { setSalary( $e.getSalary() * 1.05 ) };

rule   Adequate salaries by fixed increments
when $e : Employee( salary < 1000 )
then modify( $e ) { setSalary( $e.getSalary() + 100 ) };
```

For lack of a better strategy in the execution engine, one of the most common workarounds to prevent infinite loops is to enrich the domain objects with one or more additional properties, which have the purpose of keeping track of which rules have been applied to a given working memory fact [5]. Rule 1.2 is an example. This solution, despite being widely used, has many drawbacks. First, unlike "local" control strategies [9], where the control logic is domain-specific and thus part of the domain model itself, this approach blurs the model with non-business related information, mixing domain and control logic. Sometimes, like in the example rule 1.2, the control structures (the `awarded` flag) may actually be meaningful in the application domain, but be otherwise useless from the business logic perspective, effectively masking the problem. Secondly, it becomes

[3] Rules are represented using the Drools Rule Language (DRL) [**?**].

a responsibility of the user to define the control flow of the patterns to be matched and the rules to be executed, instead of leaving it to the rule engine, as it should be expected of a declarative system. The solution also does not scale with the number of rules: it is impractical to keep a data structure for each rule that could potentially be applied to an object. Moreover, a further performance-related issue should be considered: the first modification would at least cause a second, useless, reevaluation of the rule's premise that could be safely avoided.

Listing 1.2. Fixed Rule

```
rule   Award senior employees (fixed)
when $e : Employee( hiringYear < 2010, awarded == false )
then modify( $e ) { setSalary( $e.getSalary() * 1.05 ),
 setAwarded(true) };
```

Another common workaround is to split the classes of the domain model into more granular entities, having a biunivocal relationship with the original object. However this solution is undesirable for many reasons. It defeats the point of using an object-oriented model, forcing to change the domain model only for implementation reasons; moreover it adds an unnecessary performance overhead since the RETE-OO network will now require additional runtime joins to reconstruct the original piece of information [11]. This approach is implicitly adopted in CLIPS, where "objects and their attributes and values were transformed into object-attribute-value triplets, and these triplets handled exactly like simple working memory elements" [2]. In this way, when the values of some properties of an object change, only the modified triplets are sent to the pattern matcher that will have to reevaluate them. A much cleaner approach would be based on the implementation of a proper strategy. Production systems such as Drools and Jess provide the "no-loop" rule annotation, which prevents the creation additional instances of the same rule for the same set of facts. It can help avoid self-loops, but does not work with more complex loops involving two or more rules. This feature can be seen as an approximate implementation of the conflict resolution algorithms known as *refraction* (or *refractoriness*), which were part of the original OPS5 LEX/MEA strategies [7]. The principle has been restated and standardized as a part of the W3C RIF-PRD specification [6] : "a given instance of a rule must not be fired more than once as long as the reasons that made it eligible for firing hold", then used for formalization in [3]. However, conflicting definitions have also been reported, such as "the same rule instantiation should not be executed multiple times" [1] or "the same rule cannot be applied to the same working memory elements multiple times" [10]. In an object oriented context, their interpretation depends on two aspects. First, two object may be the same - by identity, or by equality on a subset of key attributes - even if some of their (non key) slots have been modified. Second, while definitions such as the one provided by [10] are focused on the facts and changes in their internal state, the standard definition is focused on constraints and changes in the truth state of their evaluation, as a possible consequence of a change in the internal state of an object. So, for example, the second rule of rule set 1.1 could fire multiple times according to [10] but not according to [6]. In fact,

RIF-PRD allows rules to be marked as "repeatable" to disable constraint-based refraction. Regardless of its definition, refraction is a strategy that is applied during the conflict resolution phase, when the matches have been reevaluated due to the modifications in the previous action phase. This proposal, instead, is aimed at minimizing the pattern evaluations to what is strictly necessary during the match phase. Even if, in an object-oriented production system, a working memory element is an object, it is not considered an atomic entity when it is modified. A rule consequence normally modifies only a subset of the properties of an object, so only the portion of the network effectively involved by those modifications should be notified and updated. This strategy is compatible with refraction, even in absence of a proper refraction implementation. It will prevent the selection of a rule instance when, from that rule's perspective, a working memory element has not changed: in fact, the rule is not even evaluated for further instantiation. It should be noted that similar functionalities might exist in other rule engines: namely, Jess, [**?**] offers a feature called "slot-specific updates", which is presented as an implementation of refraction but functionally is much more similar to our proposal. Due to its closed source nature, however, its architecture is not documented and no methodology is provided that can be applied in different contexts. YES/OPS [11] also describes something it calls "new triggering conditions" but with no implementation details. It further adds the control symbol '!' on slots to allow "triggering on any change". This provides a functionality which is similar to the `@Watch` annotation defined later in this paper. Like Jess, YES/OPS is closed source and no additional information can be found. Clips COOL exhibits similar behaviour but, as previously mentioned, its implementation is different and more computationally expensive as all slots are considered individual working memory elements.

3 Property Reactivity

The idea behind property based reactivity is that, whenever a working memory element (an object) is updated, patterns should not be reevaluated on that object, unless they involve constraints on properties that are known to have changed since the initial insertion or the latest modification, whichever happened later. This information can be obtained statically at compile time, performing an analysis of the rule base, and can be used at runtime to optimize the behavior of the execution engine. In systems such as CLIPS, Jess or Drools, an object type (class) C is defined by an identifying name, one parent class E_C and a set of slots (also called fields, properties or attributes) S_C. Slots are identified by a name, a range type and their cardinality. Inheritance works with the usual semantics: type and slots are inherited transitively, so that the effective set of slots available to a type is $S^\star_C = S_C \cup S^\star_{E_C}$. Slot names are unique within the scope of $S^\star_C$; their ranges may be primitives (using datatypes such as strings, numbers, dates, etc...) or object types. Unlike the others, Drools uses Java classes or interfaces natively: in this case, Java inheritance is used and slots are mapped to public getter/setter pairs. This also implies that, while class inheritance is

single, type inheritance may be multiple when interfaces are used. Individual objects `x` are instances of one class $C(\mathtt{x})$, so they have one or more declared or inherited types, and their fields are either populated using appropriate values from the field's range or the special value `null`. The usual working memory actions - insertion (I), retraction (R) and update/modification (M) - are instance oriented, involving a single object at a time, and shallow, i.e. for any two objects `x` and `y`, for any $op \in \{I, R, M\}$, if `y` is the value of a slot of `x`, $op(\mathtt{x})$ does not imply $op(\mathtt{y})$. Modifications of a working memory object `x` usually involve only a subset $S'(M(\mathtt{x}))$ of all the slots $S^{\star}_{C(\mathtt{x})}$ made available to `x` by its class $C(\mathtt{x})$. The premise (left hand side) of rule R is composed by a sequence of n object patterns $P_R[k]_{k:0..n-1}$, usually but not necessarily combined by conjunction. Other operators and quantifiers might be used, but this would not be relevant from the point of view of property reactivity. An object pattern P[4], then, is a logic formula involving the conjunction of (i) a type constraint (for any X to be be matched, $C(X) \sqsubseteq D_P$, for the one type literal D_P used in P) and (ii) another sub-formula which is a combination of zero or more atomic slot constraints, boolean expressions using the *constrained* slots in $S_c(P) \subseteq S^{\star}_{D_P}$. The set $S_c(P)$ can be further customized using a pattern annotation called `@Watch`. The result is the final set of properties the pattern will react to, called $S'(P)$. Its content is defined by the regular expression `('!'? '*')? ('!'? slotName)*` . `@Watch(*)` forces the pattern to respond to any modification, effectively initializing $S'(P) \doteq S^{\star}_{D_P}$, while `@Watch(!*)` prevents reactiveness to modifications by initializing $S'(P) \doteq \emptyset$. After this override, any additional slot specified in the annotation is added back to (resp. removed from) the set of watched (resp. negative) properties $S_w(P)$ (resp. $S_n(P)$), depending on the use of the negation operator `'!'`. So, $S'(P)$ is effectively computed as:

$$S'(P) = [S_c(P)|S^{\star}_{D_P}|\emptyset] \cup S_w(P) \setminus S_n(P)$$

When a modification $M(\mathtt{x})$ is executed, for any pattern P whose type D_P is a supertype of `x`, there are two possible situations : either $S'(M(\mathtt{x})) \cup S'(P) = \emptyset$ or not. In the former case, from the perspective of rule R, the pattern would be reevaluated using the same working memory element as before the modification: any activation thus generated would be a candidate for cancellation due to refraction, so there is little point in performing the evaluation and generating the activation in the first place. The latter, instead, may effectively change the eligibility state of a rule, so reevaluation cannot be avoided and the decision about the eligibility of any new activation has to be delegated to the agenda and its strategies. Property reactivity, then, is the modification of the engine behavior to deal with modification actions, so that a fact `x` is reevaluated in the context of a pattern only when $S'(M(\mathtt{x})) \cup S'(P) \neq \emptyset$, i.e. if that pattern is attempting a match on one or more of the properties that have been actually modified. Notice that, being based on the static declaration of the modified slots, a "false update" (i.e. assigning to a slot the same value it already has) would be treated exactly like a real modification.

[4] Context indexes are dropped for simplicity.

3.1 Property Reactive RETE

To implement property reactivity, it is necessary to know which object's properties have been modified during the execution of a rule instance ($S'(M(\mathtt{x}))$) and on which properties each node of the RETE network does pattern matching. To this end, $S'(P)$ is further partitioned between the nodes that concur in the evaluation of the pattern. This information needs to be encoded in a compact way that also allows an efficient comparison. Whenever a consequence modifies an object, it can then propagate the metadata about the affected properties, so that any node can compare that set with the one describing its local properties, in order to determine whether it should be affected by that change - reevaluating its constraints and possibly propagating the object - or if it can safely ignore and discard it. For efficiency reasons, the sets $S'(M)$ and $S'(P)$ are precomputed statically, performing a compile-time analysis of the rule base. In particular, $S'(P)$ can be inferred by processing each pattern in the LHS of each rule R; The set, then, can be "manually" tweaked using annotations, which allow to add or remove other properties which are not explicitly involved in the pattern's constraints. Any property in the final set will be referred to as being *watched* by (the nodes implementing) a pattern. Likewise, $S'(M)$ can be inferred by processing each modify instruction in a rule's RHS.

Bit Mask Generation for Property Sets. A bit mask is used to encode these sets. Since a bit mask is ordered, an arbitrary but deterministic order between properties is assumed, sorting the elements of $S^\star$ and allowing it to be modeled using a Bitset. However, since the masks are computed based on the pattern type, not the actual object type, this order has to take into account the inheritance level where each property is defined, allowing patterns to match instances of subclasses of their declared type. We group the properties based on the level in the classes' hierarchy where they are declared (the higher in the hierarchy come first) and then use a lexicographic order for each group. So, for example, if a class `MyCls` had properties `a`, `b`, `c`, `d` and `e`, the property set $\{\mathtt{c}, \mathtt{d}, \mathtt{a}\}$ would first be reordered as $\{\mathtt{a}, \mathtt{c}, \mathtt{d}\}$ and, eventually, the bit mask `10110` would be generated. In this way, since each property corresponds to a bit according to its index, both $S'(M)$ and $S'(P)$ can be represented using bit masks, called modification bit mask and watched properties bit mask respectively. Their intersection can be computed in a very efficient way using bitwise ANDs. The procedure to generate a bit mask for a given class and a list of properties is reported in Listing 3.1.

RETE Network Building and Configuration. Since RETE matches working memory elements using a data-flow network, where different nodes concur to the evaluation of a pattern, the mask has to be allocated appropriately to optimize the propagation. More over, the masks have to be compatible with node sharing to preserve the structural optimization provided by the algorithm. In general, there are five main aspects to be considered when building a RETE graph, namely (i) the alpha nodes tree, (ii) the beta nodes left inputs (including

Algorithm 1. Generate Bit Mask from a list of Properties

```
function genBitMask( t : Type, l : PropertyList )
    settableProperties ← t.settableProperties
    bitmask ← newbit[settableProperties.length]
    sort(settableProperties)
    for all propertypinl do
        i ← settableProperties.indexOf( p )
        bitmask[i] ← 1
    end forreturn bitmask
end function
```

the (iii) first beta node left input in every beta node sub-tree), the (iv) beta nodes right inputs and (v) the rule terminal nodes left inputs. Any property-based reactive modify is scoped to a single object, so mask based filtering is only needed during an object's propagation through the alpha network, at the junction between the alpha network and the beta network or in the left input of a terminal node. The masks have no other impact during or after beta network join propagations. So, there are three main parts of the beta network the alpha network can connect to (i) Beta nodes right inputs, (ii) Root Beta nodes left inputs and (iii) "Root" Terminal nodes left inputs. Beta nodes are created only when a rule has two or more patterns: in this case, alpha nodes are connected to the left input of the first beta node, and to the right input of each beta node. If a rule has a single pattern, there are no joins and thus no beta nodes, so the alpha network flows directly into the terminal node. Masks, then, are required in the alpha network and in these junction points. Three types of bit masks are distinguished in the system: *declared*, *negative* and *inferred*. The declared mask represents properties the current node wishes to react to. In alpha nodes, this is based on the property[5] used in the node's constraint. The declared mask of a beta right input is created using the node's join constraints and the right pattern's watched properties, if any is defined using `@Watch`. Beta left inputs anf terminal nodes have no constraints, but their declared mask may still derive from the use of `@Watch`. The negative mask, instead, represents the properties to be ignored, again as defined by `@Watch`. These two masks are not used at runtime, but they are used to build the inferred mask, which in turn will be used to filter propagations in each node. The beta and terminal nodes left inputs can be treated as a special case of beta nodes right input configuration, so the discussion will be focused only on the construction of the alpha nodes and the beta nodes right inputs.

Alpha Node Mask Generation. The alpha network construction is incremental. When adding a pattern, it starts from the object type node and adds each alpha node in turn. The last alpha node in a chain is connected to a beta node's right input or, as appropriate, to a beta or terminal node's left input. Due to

[5] The discussion generalizes easily to complex constraints, involving more than one property.

node sharing, an alpha node may have more than one children, creating a tree structure. However, the path from a beta node to its root object type node is always linear. An alpha node inferred mask, then, is a bitwise `inclusive or` accumulation of the declared masks of all its parents and all its children, down to and including the beta nodes inputs. In general, a node can never be less permissive than its child - otherwise the propagation would be blocked, before reaching the node that would allow it to pass. Likewise, a node must inherit its parent's permissions to forward the propagation. The construction is incremental: as the node is added to the network, its inferred mask is updated using the parents' masks, using the function `updateAlphaInferredMask` in Algorithm 3.1. The update based on the children has to be delayed until the completion of the portion of that alpha network.

Algorithm 2. Incremental computation of an Alpha node Inferred mask

```
function accAlphaDeclaredMasksFromParents( alpha : AlphaNode )
    if alpha.parent.type ≡ ALPHA then
return node.declaredMask ∨ accAlphaDeclaredMasksFromParents( alpha.parent )
    else
return node.declaredMask
    end if
end function
function updateAlphaInferredMask( alpha : AlphaNode, mask : Long )
    alpha.inferredMask ← accAlphaDeclaredMasksFromParents( alpha ) ∨ mask
    if alpha.parent.type ≡ ALPHA then
        updateAlphaInferredMask( alpha.parent, alpha.inferredMask )
    end if
end function
```

Beta Node Masks. The beta node is connected to the graph after all its alpha nodes have been created. Upon connection, it triggers the process necessary for each alpha node to determine its inferred mask from its children. The beta node right declared mask is a combination of the properties used by the beta constraint and the additional properties to react on, as possibly declared using `@Watch`. Beta nodes may have an additional right negative mask which lists the properties on which reactions are to be ignored. In the case of the beta left and the terminal node left inputs, where there are no join constraints, the declared mask is simply derived from the `@Watch`. The mask is added to the left inputs to maximize node sharing when patterns use the same constraints but different `@Watch` declarations. However, this implies that beta nodes which would normally be shared due to using same constraints, cannot be shared if they have different `@Watch` declarations. The beta node right inferred mask is computed in two steps, as shown in Algorithm 3.1. First, an intermediary value is computed from the bitwise `inclusive or` accumulation of all parent alpha node declared mask and the beta node declared mask. Notice that a beta node can only have a single parent alpha node, and each alpha node can only have a

single parent alpha node; this forms a single line path, in comparison to the alpha nodes descendant potential tree shape. The inferred value is then computed from the bitwise `and` of the intermediary value and the `not` of the negated mask.

Algorithm 3. Incremental computation of an Beta node Inferred mask

function CALCULATEBETAMASK(*beta* : *BetaNode*)
  beta.declaredMask ←
genBitMask(beta.typeClass, beta.constrained)
∨ genBitMask(beta.typeClass, beta.watched)
  beta.negativeMask ← genBitMask(beta.typeClass, beta.ignored)
  beta.intermediaryMask ←
accAlphaDeclaredMasksFromParents(beta.leftInput) ∨ *beta.declaredMask*
  beta.inferredMask ← beta.intermediaryMask ∧ ¬*beta.negativeMask*
end function

Dynamic Rule Addition and Removal. The proposed method is incremental and can be applied even if rules are dynamically added to an already complete RETE network. The same data structures, however, can also be used for rule removal. The process is subtractive and requires the recalculation of masks for any node which was shared between the removed rule and any other rule still part of the network. A high level specification for the procedure to handle the resetting and recalculation of the inferred masks is shown in Algorithm 3.1.

Listing 1.3. Example Rules

```
rule R1 when
    Dummy( $v : v )
    MyCls( c >= 1, a >= 1, e >= $v )
then ... end

rule R2 when
    Dummy( $v : v )
    MyCls( c >= 1, d >= 1, b >= $v )
then ... end

rule R3 when
    Dummy( )
    MyCls( c >= 1 ) @Watch( f, d, !c )
then ... end
```

3.2 Example

As an example, consider the three rules in Listing 1.3. It focuses, again, on the beta right input configuration; as this is the most full featured of the network configurations. The example uses a class named `MyCls`, which has 6 (integer) properties `a, b, c, d, e and f` and three rules `R1, R2 and R3`. Rule `R3` uses

Algorithm 4. Reset masks on Rule removal

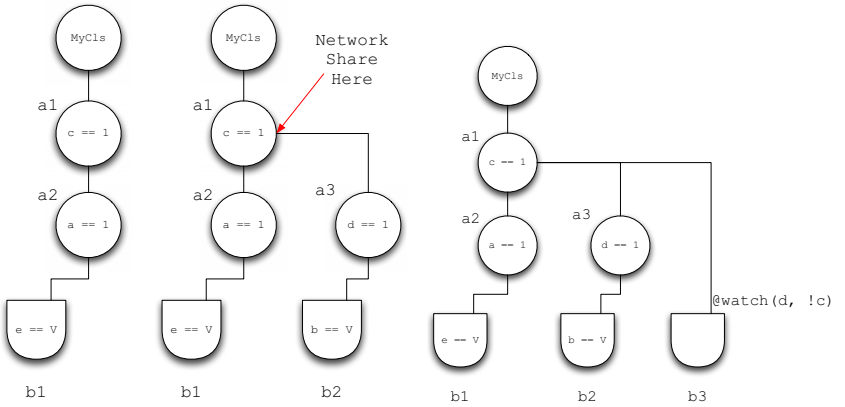
```
function updateMasksOnRuleRemoval( rule : Rule )
    alphaRoots ← getRootAlphaNodes( rule )
    remove( rule )
    betas ← new List()
    for all alpha ∈ alphaRoots do
        if inUse(alpha ) then
            for all beta ∈ getAllChildBetaNodes( alpha ) do
                betas.add( beta )
            end for
        end if
    end for
    for all beta ∈ betas do
        for all alpha ∈ getAllAlphas( beta ) do
            alpha.declaredMask
← accAlphaDeclaredMasksFromParents( alpha.parent )
        end for
    end for
    for all beta ∈ betas do
        updateAlphaInferredMask( beta.rightParent )
        calculateBetaMask (beta)
    end for
end function
```

the `@Watch` annotation, but otherwise has no other constraint, so it would normally result in a full cross product. The diagram in figure 1 shows the RETE network as each rule is added. Once built, Node `a1` is shared with three child outputs. Applying the proposed algorithm, the masks are computed incrementally and the final results are shown in Table 1.

Table 1. Bitmasks after adding R3

	a1	a2	a3	b1	b2	b3
declared	000100	000001	001000	010000	000010	101000
negative				000000	000000	000100
inferred	111111	010101	001110	010101	001110	101000

A boolean flag table is used to indicate whether a node would block or propagate a modification, based on various possible modification scenarios. We assume that two facts are inserted: `Dummy(0)` and `MyCls(1,1,1,1,1,1)`. The values ensure that all constraints evaluate to true on insertion. Each row in Table 3.2 represents a different modification of the `MyCls` object involving different properties, assuming that no updated value would cause any constraint to be evaluated as false. The first column shows the properties being used in each scenario, while the second column shows the corresponding modification bit mask.



(a) Two Beta RETE (b) Three Beta RETE

Fig. 1. Example RETE networks with masks

The remaining columns represent each node in the RETE network: a 0 indicates that the node will block the modification, while 1 accounts for propagations.

Table 2. Modification Truth Table

fields	modify mask	a1	a2	a3	b1	b2	b3
fedcba	111111	1	1	1	1	1	1
a	000001	1	1	0	1	0	0
b	000010	1	0	1	0	1	0
c	000100	1	1	1	1	1	0
d	001000	1	0	1	0	1	1
e	000001	1	1	0	1	0	0
f	000001	1	0	0	0	0	1
b a	001010	1	1	1	1	1	0
f e	110000	1	1	0	1	0	1
d e	011000	1	1	1	1	1	1
f d e	011000	1	1	1	1	1	1

4 Benchmarking

The computation of the bit masks can be performed during the compilation phase of the RETE network: the only operation that needs to be executed at runtime is a bitwise AND between the modification bit mask and the inferred bit masks of the traversed nodes. This operation adds negligible overhead and allows the CPU to save time by avoiding unnecessary propagations into the network and their related constraint evaluations. In order to quantify this performance

improvement, a synthetic benchmark suite has been developed. Best practices in writing a Java [4] benchmarking suite have been applied. All tests introduce a warm up phase long enough to allow the JVM HotSpot to perform the optimizations necessary to allow a consistent, normalized and optimal set of results. Each benchmark has been run 10 times, discarding the worst and best results, then average and standard deviation are computed. The results have been collected on a machine equipped with a i7-2820QM@2.30GHz Quad core CPU with HT and 8Gb of RAM, running an OpenJDK 1.7 on top of Ubuntu Quetzal 64bit operating system. A mock data model with two classes was used, `A` and `B`, with two properties each: `a1`, `a2` and `b1`, `b2` respectively, each of type `int`.

The first benchmark, shown in rule 1.4 is minimal and compares the use of property reactivity with (local) refraction. It inserts 1 million instances of `A`, with initial values for `a1` and `a2` set to 1. The rule checks the former property and modifies the latter. Without property reactivity, `no-loop` is required to prevent infinite recursion. With property reactivity, instead, the reevaluation of the modified instances is blocked before reaching the agenda, so the control attribute is not needed. This benchmark takes, on average, 2.275 ± 0.080 seconds to run using `no-loop`, while, when property reactivity is enabled, it runs in 2.265 ± 0.047 seconds, with a difference of only the 0.44%.

Listing 1.4. Benchmark #1

```
rule R0 //no-loop
when
    $a: A( $a1 : a1 < 10 )
then
    modify( $a ) { setA2( $a1 + 1 ) };
end
```

The second benchmark evaluated the benefits of property reactivity with respect to an ad-hoc control constraint, using a rule base containing the single rule `R1`, shown in Listing 1.5. As previously, it inserts a million instances of `A(1,1)` and a single instance of `B(0,2)`. The rule modifies the instances of `A` so that, during the reevaluation in absence of property reactivity, the join constraint `b2 > $a.a2` will block the propagation. This benchmark takes 2.562 ± 0.137 seconds to run. With property reactivity, the control constraint can be removed and the benchmark runs, on average, in 2.483 ± 0.155 seconds, with a 3.08% improvement.

Listing 1.5. Benchmark #2

```
rule R1 when
    $a: A( a1 < 10 )
    $b: B( b1 < $a.a1, b2 > $a.a2 )
then
    modify( $a ) { setA2( $b.getB2() + 1 ) };
end
```

The third benchmark evaluated the correlation between performance gains and the number of rules potentially triggered by a modification. To this end,

Class `A` has been modified with a third property `a3`, and created a new rule base with *R1* and `R2` (see Listing 1.6). The benchmark takes 5.187 ± 0.398 seconds without property reactivity, while it takes only 4.690 ± 0.413 seconds with property reactivity enabled, yielding a 9.58% improvement. This gain, due to the fact that rules are not reevaluated, scales almost linearly with the number of rules affected. Two additional properties were `a4` and `a5` to `A` and other 2 rules using the same structure as in `R1` and `R2`, obtaining a benchmark with a total of 4 rules that runs in 12.476 ± 0.565 seconds without property reactivity and in 9.015 ± 0.445 with property reactivity, with a gain of 27.74%.

Listing 1.6. Benchmark #3 (partial)

```
rule R2 when
    // avoid alpha node sharing
    $a: A( a1 < 11 )
    $b: B( b1 < $a.a1, b2 > $a.a3 )
then
    modify( $a ) { setA3( $b.getB2() + 1 ) };
end
```

The last benchmark evaluated the correlation between performance improvement and the distance between the pattern containing the bit mask that blocks the propagation and the one containing a control constraint. For this measurement, it has been introduced an increasing number of types `C`, `D`, etc..., again following the same pattern of `A` and `B`. Different rule bases have then been created with rules such as `R3` (Listing 1.7), for an increasing number of patterns of 2, 4, 8 and 16. The results have been reported in Table 4.

Listing 1.7. Benchmark #4

```
rule R3 when
    $a: A( a1 < 10 )
    $b: B( b1 < $a.a1 )
    ...
    $x: X( x1 < $w.w1, x2 > $a.a2 )
then
    modify( $a ) { setA2( $c.getC2() + 1 ) };
end
```

Table 3. Benchmark #4 results

	2 patterns		4 patterns		8 patterns		16 patterns	
	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.
PR Off	3.276	±0.240	3.940	±0.273	6.118	±0.364	9.896	±0.604
PR On	3.141	±0.172	3.367	±0.225	4.839	±0.296	7.512	±0.306
Gain	4.12%		14.54%		20.91%		24.09%	

5 Conclusions and Future Works

We presented an optimization algorithm, based on the static analysis of an object-oriented production rule base, which provides better control over recursive rule activations due to working memory element modifications and, at the same time, allows to improve the runtime performance. The approach is more declarative than traditional flow-control attributes and agenda groups, and allows to write cleaner rules, resulting in rule bases which are easier to maintain. The optimization is applied at compile time, during the RETE construction phase, and is compatible with the incremental addition and removal of rules. It can be enabled, disabled or customized as needed with the use of annotations. Functionally, it implements a limited form of preventive refraction, which applies during the match phase instead of the conflict resolution phase. It remains to be seen whether additional optimizations could be introduced. At the moment, any "virtual" property constraint configured using `@Watch` is collapsed into the beta nodes. While preserving alpha node sharing, it limits beta node sharing, since beta nodes with the same constraints but different masks can not be considered equivalent. Moreover, it delays the filtering process, impacting performance. So, it is planned to investigate the possibility of splitting or pushing back part of the masks into the alpha network while still optimizing the degree of node sharing.

References

1. Anderson, J.R.: Rules of the Mind. Lawrence Erlbaum Associates, Hillsdale (1993)
2. Auburn, M.: Integrating an object system into CLIPS: Language design and implementation issues, Tech. report, NASA (1990)
3. Berstel-Da Silva, B.: Formalizing both refraction-based and sequential executions of production rule programs. In: Bikakis, A., Giurca, A. (eds.) RuleML 2012. LNCS, vol. 7438, pp. 47–61. Springer, Heidelberg (2012)
4. Boyer, B.: Robust java benchmarking (2008), `http://www.ibm.com/developerworks/java/library/j-benchmark1/index.html`
5. Brownston, L., Farrell, R., Kant, E., Martin, N.: Programming expert systems in OPS5: an introduction to rule-based programming. Addison-Wesley Longman Publishing Co., Inc., Boston (1985)
6. de Sainte Marie, C., Hallmark, G., Paschke, A.: Rule interchange format, production rule dialect. Recommendation, w3c (2013), `http://www.w3.org/TR/rif-prd/`
7. Forgy, C.: Ops5 users's manual, Tech. report (1981)
8. Forgy, C.: RETE: A fast algorithm for the many pattern/many object pattern match problem. Artificial Intelligences 19, 17–37 (1982)
9. Jackson, P.: Introduction to Expert Systems, 3rd edn. Addison-Wesley (1998)
10. Jones, G., Ritter, F.E.: Production systems and rule-based inference. John Wiley and Sons, Ltd. (2006)
11. Schor, M.I., Daly, T., Lee, H.S., Tibbitts, B.: Advances in RETE pattern matching. In: Proceedings of the Fifth National Conference on Artificial Intelligence, pp. 225–232 (1986)

Computing the Stratified Semantics of Logic Programs over Big Data through Mass Parallelization

Ilias Tachmazidis and Grigoris Antoniou

University of Huddersfield, UK

Abstract. Increasingly huge amounts of data are published on the Web, and generated from sensors and social media. This Big Data challenge poses new scientific and technological challenges and creates new opportunities - thus the increasing attention in academia and industry. Traditionally, logic programming has focused on complex knowledge structures/programs, so the question arises whether and how it can work in the face of Big Data. In this paper, we examine how stratified semantics of logic programming, equivalent to the well-founded semantics for stratified programs, can process huge amounts of data through mass parallelization. In particular, we propose and evaluate a parallel approach using the MapReduce framework. Our experimental results indicate that our approach is scalable and that stratified semantics of logic programming can be applied to billions of facts.

1 Introduction

A huge amount of data is being generated at an increasing pace by sensor networks and social media. In addition, this data is heterogeneous, and needs often to be combined with other information, including database and web data, to become more useful. This *big data* challenge is at the core of many contemporary scientific, technological and business developments.

The question arises whether the reasoning community, as found in the areas of knowledge representation, rule systems, logic programming and semantic web, can connect to the big data wave. On the one hand, there is clear application scope, e.g. for data cleaning, deriving higher-level knowledge, assisting decision support etc. But on the other hand, there are significant challenges arising from the area's traditional focus on rich knowledge structures instead of large amounts of data, and its reliance on in-memory methods. The best approach for enabling reasoning with big data is parallelization, as established e.g. by the Larkc project[1] [7].

Parallel reasoning can be achieved by distributing the computation among nodes. Such distribution can be based either on rule partitioning or on data partitioning [12]. For rule partitioning, the computation of each rule is assigned to a node in the cluster. Thus, the workload for each rule (and node) depends on the structure of the rule set, which in general does not lead to a balanced workload. On the contrary, in case of data partitioning, data is divided in chunks and each chunk is assigned to a node, allowing more balanced distribution of the computation among nodes.

[1] `www.larkc.eu`

L. Morgenstern et al. (Eds.): RuleML 2013, LNCS 8035, pp. 188–202, 2013.

Parallel reasoning, based on data partitioning, has been studied in [16,23,9]. Several aspects, such as highly uneven distribution of Semantic Web data, have been pointed out and addressed in [13]. In terms of scalability, parallel reasoning based on the MapReduce framework has been scaled up to 100 billion triples [22].

Although parallelization approaches have mainly focused on monotonic reasoning, such as RDFS and OWL-horst, there has been work on nonmonotonic reasoning. Specifically, in [20] authors proposed an approach for scalable reasoning, using the MapReduce framework, over defeasible logic with unary predicates. Experimental evaluation presented that defeasible reasoning can be performed over billions of facts.

Defeasible reasoning has been extended for predicates of arbitrary arity in [21]. In particular, authors presented that, under the assumption of stratification, defeasible reasoning can be performed over millions of facts and has the potential to scale up to billions of facts.

In this paper, we propose an approach for stratified semantics of logic programming, equivalent to the well-founded semantics for stratified programs. Stratification is a well-known notion applied in many areas of knowledge representation in order to provide efficient reasoning. It has been used in various problems such as tractable RDF query answering [19], Description Logics [3,10,15] and nonmonotonic formalisms [4].

The rest of the paper is organized as follows. Section 2 introduces briefly the MapReduce framework and the stratified semantics of logic programming. The algorithm for the stratified semantics over MapReduce is described in Section 3, while Section 4 presents our experimental evaluation. We conclude in Section 5.

2 Preliminaries

In this Section, we describe the basic notions of: (a) the MapReduce framework and (b) the stratified semantics of logic programming.

2.1 MapReduce Framework

MapReduce is a framework for parallel processing over huge datasets [5]. Processing is carried out in two phases, a map and a reduce phase. For each phase, a set of user-defined map and reduce functions are run in a parallel fashion. The former performs a user-defined operation over an arbitrary part of the input and partitions the data, while the latter performs a user-defined operation on each partition.

MapReduce is designed to operate over key/value pairs. Specifically, each Map function receives a key/value pair and emits a set of key/value pairs. All key/value pairs produced during the map phase are grouped by their key and passed to the reduce phase. During the reduce phase, a $Reduce$ function is called for each unique key, processing the corresponding set of values.

Probably the most well-known MapReduce example is the *wordcount* example. In this example, we take as input a large number of documents and the final result is the calculation of the number of occurrences of each word. The pseudo-code for the Map and $Reduce$ functions is depicted below.

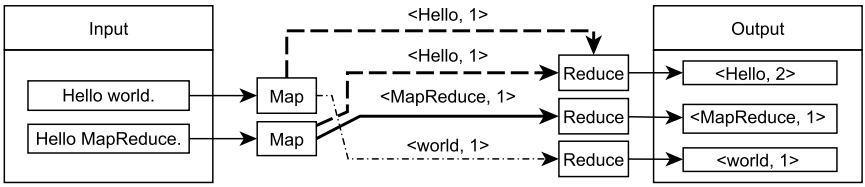


Fig. 1. Wordcount example

```
map(Long key, String value):
  // key: position in document
  // value: document line
  for each (word w in value) do
    EmitIntermediate(w, "1");

reduce(String key, Iterator values):
  // key: a word
  // values : list of counts
  int count = 0;
  for each (v in values) do
    count += ParseInt(v);
  Emit(key, count);
```

Consider as input the lines "Hello world." and "Hello MapReduce.". Figure 1 depicts the whole process. During the map phase, each map operation gets as input a line of a document. The Map function extracts words from each line and emits that word w occurred once ("1"). Here we do not use the position of each line in the document, thus the *key* in Map is ignored. As mentioned above, the MapReduce framework will group and sort pairs by their key. The $Reduce$ function has to sum up all occurrence values for each word emitting a pair containing the word and the final number of occurrences for this word. The final result for each word will be <Hello, 2>, <MapReduce, 1> and <world, 1>.

2.2 Stratified Semantics

In this paper, we describe the *stratified semantics* of logic programming as they were defined in [17] for stratified programs of the well-founded semantics. We impose several restrictions in order to achieve parallelization using the MapReduce framework.

Definition 1. *[17] A* general logic program *is a finite set of* general rules*, which may have both positive and negative subgoals. A general rule is written with its* head*, or conclusion on the left, and its subgoal (body), if any to the right of the symbol "←", which may be read "if". For example,*

$$p(X) \leftarrow a(X), \textbf{ not } b(X).$$

is a rule in which p(X) *is the head,* a(X) *is a* positive subgoal*, and* b(X) *is a* negative subgoal*. This rule may be read as "*p(X) *if* a(X) *and not* b(X)*". A* Horn rule *is one with no negative subgoals, and a* Horn logic program *is one with only Horn rules.* □

We use the following conventions. A logical variable starts with a capital letter while a constant or a predicate starts with a lowercase letter. Note that functions are not allowed. A predicate of arbitrary arity will be referred as a *literal*. If p is a *positive* literal then $\neg p$ is its *negative* literal, p and $\neg p$ are *complements* of each other. Constants, variables and literals are *terms*. A *ground term* is a term with no variables. The *Herbrand universe* is the set of constants in a given program. The *Herbrand base* is the set of ground terms that are produced by the substitution of variables with constants in the Herbrand universe.

Definition 2. *[17] A program is* stratified *if all of its predicates can be assigned a* rank *such that*

- *no predicate depends positively on one of greater rank, and*
- *no predicate depends negatively on one of* equal *or greater rank*

in any rule. □

Definition 3. *[17] Given a program* **P***, a* partial interpretation I *is a consistent set of literals whose atoms are in the Herbrand base of* **P***. A* total interpretation *is a partial interpretation that contains every atom of the Hebrand base or its negation. We say a ground (variable-free) literal is* true in I *when it is in* I *and say it is* false in I *when its complement is in* I*. Similarly, we say a conjunction of ground literals is* true *in* I *if all of the literals are true in* I*, and is* false *in* I *if any of its literals is false in* I □

According to the *stratified semantics* literals are classified as positive or negative as follows:

1. Facts are classified as positive literals.
2. All inferences for rank 0 are classified as positive literals, while whose not inferred are classified as negative.
3. If all literals up to rank $k-1$ are classified either as positive or negative, then we can perform reasoning for rank k. All inferences for rank k are then classified as positive literals, while whose not inferred are classified as negative.

Definition 4. *[17] Let a program* **P***, its associated Herbrand base* H *and a partial interpretation* I *be given. We say* A $\subseteq$ H *is an* unfounded set *(of* **P***) with respect to* I *if each atom* p $\in$ A *satisfies the following condition: For each instantiated rule* R *of* **P** *whose head is* p*, (at least) one of the following holds:*

1. *Some (positive or negative) subgoal q of the body is false in* I*.*
2. *Some positive subgoal of the body occurs in* A*.*

A literal that makes (1) or (2) above true is called a witness of unusability *for rule* R *(with respect to* I*).* □

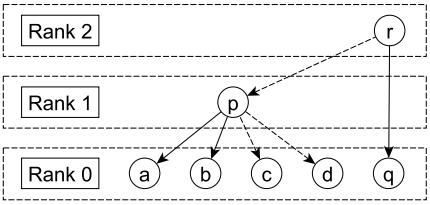


Fig. 2. Predicates assigned to ranks

3 Algorithm Description

In this section we present a parallel solution for stratified programs, address several special cases and discuss arising challenges.

According to the Definition 2, for a stratified program, literals that are given as input (facts) are assigned rank 0. Predicates that are supported by rules containing no negative subgoals are assigned rank 0, as well. Subsequently, literals depending negatively only on rank 0, are assigned rank 1. In general, literals depending negatively on rank $k-1$, are assigned rank k. Thus, stratification is finished when all predicates are assigned a rank.

Consider the following program:

r(X,Y) ← q(X,Y), **not** p(X,Y).
p(X,Y) ← a(X,Z), b(Z,Y), **not** c(X,Z), **not** d(Y).

Figure 2 shows how predicates are assigned to ranks, which in our example is as follows:

rank 0: *a*, *b*, *c*, *d*, *q* rank 1: *p* rank 2: *r*

plain lines represent a positive dependency, while dashed lines represent a negative dependency.

Once predicates have been assigned to ranks, we proceed with reasoning according to the following algorithm:

```
Overall reasoning process:
  Set KB = facts;
  for (i=0; i<=N; i++) do
    do
      new_KB = Perform_reasoning(i, KB);
      KB += new_KB;
    while (new_KB != null) \\ check if fixpoint is reached
  done
```

Initially, we add facts to our knowledge base (KB). Subsequently, for each rank (0 to *N*, where *N* is the highest rank for a given program) we perform reasoning as described later in this section (Perform_reasoning(i, KB)), by taking into consideration the rank we are reasoning for (i) and the current knowledge base (KB), while classifying all

the inferred literals as positive. Note that for each rank, reasoning is iterated until no new conclusion is derived (while (new_KB != null)). Once all inferences are computed for rank i, we continue with the next rank, until all ranks are evaluated. Finally, when reasoning is finished, literals that are contained in our KB are classified as positive, while whose that are not are classified as negative.

Consider the following rule from the aforementioned program:

$$\text{p(X,Y)} \leftarrow \text{a(X,Z), b(Z,Y), } \textbf{not}\ \text{c(X,Z), } \textbf{not}\ \text{d(Y).}$$

here *p(X,Y)* is our *final goal*, *a(X,Z)* and *b(Z,Y)* are positive subgoals, and **not** *c(X,Z)* and **not** *d(Y)* are negative subgoals. We will group all positive subgoals into a *positive goal*. A positive goal consists of a new predicate (say *ab*) that contains as arguments all the arguments of the final goal (X,Y) plus all the common arguments with the negative subgoals (X,Z,Y), namely we need to compute *ab(X,Z,Y)*. However, we need to ensure that for each rule, all arguments of the final goal (X,Y) can be found in a positive subgoal. In addition, we need to ensure that for each rule, the set of arguments that can be found in all the negative subgoals (X,Z,Y) is a subset ($\subseteq$) of the arguments that can be found in all the positive subgoals (X,Z,Y). In order to compute the final goal (p(X,Y)) we retain all values of the positive goal (ab(X,Z,Y)) that have no equal values with any negative subgoal (**not** *c(X,Z)* and **not** *d(Y)*) on their common arguments (X,Z and Y respectively).

As a general guideline, we perform a *single join* or *multiple joins* (see Subsection 3.1) in order to calculate positive goals, and *anti-joins* (see Subsection 3.2) to calculate the final goal. However, special cases may apply to certain programs (see Subsection 3.3).

3.1 Positive Goals Calculation

Consider the following program:

$$\text{p(X,Y)} \leftarrow \text{a(X,Z), b(Z,Y).}$$

and the facts for *a* and *b*

$$\text{a(1,2)}\quad \text{a(1,3)}\quad \text{b(2,4)}\quad \text{b(3,4)}$$

A single join can be performed either in Map or $Reduce$. However, the basic idea remains the same since in both cases we join *a(X,Z)* and *b(Z,Y)* on their common argument *Z* in order to produce *p(X,Y)*.

Single Join in Reduce. We will first describe how joins (for the aforementioned rule) can be performed in $Reduce$ according to the following pseudo-code:

```
map(Long key, String value):
  // key: position in document (irrelevant)
  // value: document line (positive literal)
  if (value.predicate == "a") then
    emit(value.Z, {value.predicate,value.X});
  else if (value.predicate == "b") then
    emit(value.Z, {value.predicate,value.Y});
```

```
reduce(String key, Iterator values):
  // key: matching argument
  // values: positive literals for matching
  for each (value in values) do
    if (value.predicate == "a") then
      a_List.add(value.X})
    else
      b_List.add(value.Y)

  for each (a in a_List) do
    for each (b in b_List) do
      emit("p(a.X,b.Y)","");
```

The Map function will emit pairs of the form <Z,(a,X)> for predicate *a* and <Z,(b,Y)> for predicate *b*, namely the following pairs:

<2, (a,1)> <3, (a,1)> <2, (b,4)> <3, (b,4)>

MapReduce framework will perform grouping/sorting resulting in the following intermediate pairs:

<2, <(a,1), (b,4)>> <3, <(a,1), (b,4)>>

During the reduce phase we match predicates *a* and *b* on their common argument (which is the *key*) and use the values to emit new literals. Thus, the reducer with key:

2 will emit *p(1,4)*
3 will emit *p(1,4)*

As we see in our simple example, *p(1,4)* is inferred twice. We need to filter out duplicates as soon as possible because they will produce unnecessary duplicates as well, affecting the overall performance. For brevity, we do not provide pseudo-code for duplicate elimination as it is straightforward for readers that are familiar with the MapReduce framework.

Single Join in Map. In case of highly skewed data distribution, joins cannot be performed during $Reduce$ due to skewed workload, which affects severely parallelization. However, such joins can be performed efficiently during Map if at least one of the two relations fits in main memory. As a real-world example, [13,6] show that semantic web data are highly skewed following zipf distribution. Joins can be performed in Map according to the following pseudo-code:

```
// Create an in memory Map from a.Z to a.X
a_HashMap = load_literals_with_predicate_a();
map(Long key, String value):
   // key: position in document
   // value: document line (positive literal)
   if (value.predicate == "b") then
```

```
if (a_HashMap.contains(b.Z)) then
  for each (X in a_HashMap.key(b.Z).iterator()) do
    emit("p(X,b.Y)","");
```

First, we need to load in memory facts for predicate *a*, namely *a(1,2)* and *a(1,3)*, and create a HashMap from *a.Z* to *a.X* (load_literals_with_predicate_a()) in order to reassure quick lookups. During Map, we traverse through given data and for each literal with predicate *b* (value.predicate == "b"), we lookup the HashMap (a_HashMap.contains(b.Z)) for matching *Z* values. In case the two relations can be joined on their common argument (Z), we calculate and emit a new literal (p(X,b.Y)) for each *X* in the HashMap. During the map phase the Map function with value:

b(2,4) will emit *p(1,4)*
b(3,4) will emit *p(1,4)*

For joins that are performed in Map, we can use the reduce phase for duplicate elimination. Duplicates are grouped together during grouping/sorting. Thus, for each group ($Reduce$ function) we eliminate duplicates by emitting each literal (*key*) only once (in this case *values* are ignored). For brevity, we do not provide pseudo-code for the $Reduce$ as it is straightforward for readers that are familiar with the MapReduce framework.

Multiple Joins. We have already described how to perform a single join. However, we may need to perform multiple joins (multi-way join) in order to compute a positive goal. Consider the following program:

q(X,Y) ← a(X,Z), b(Z,W), c(W,Y).
p(X,Y) ← c(W,Y), b(Z,W), a(X,Z).

In order to compute *q(X,Y)*, we can apply our approach for single join twice, by first joining *a(X,Z)* and *b(Z,W)* on *Z*, producing a temporary literal (say *ab(X,W)*), and then join *ab(X,W)* and *c(W,Y)* on *W* (producing the final goal). However, we can optimize multi-way join by taking into consideration the whole program instead of computing each rule separately. Note that the body of both rules is practically identical. Both rules consist of the same three predicates (a,b,c), which have the same common arguments (a,b have Z and b,c have W). Thus, we may perform the required joins once and produce new literals for both *q(X,Y)* and *p(X,Y)*.

Multi-way join have been described in [8] and optimized in [2]. In order to achieve an efficient implementation, optimizations in [2] should be taken into consideration. However, [2] require better knowledge of the available data than our general assumptions on data distribution (uniform or skewed).

3.2 Final Goal Calculation

Once positive goals are calculated, we need to perform anti-joins with negative subgoals on their common arguments in order to retain as final goals literals whose values are found in the positive goal, but not in any of the negative subgoals. Consider the following rule:

a(X) :- b(X), not c(X).

we need to retain values of *X* that are found in *b*, but not in *c*. In order to perform an anti-join, we follow the aforementioned approach for a single join, however, emitting a final goal for each value of *X* that is supported by the predicate *b*, and not by the predicate *c*.

3.3 Special Cases

Goals with No Common Arguments. Once we have calculated a positive goal, prior to performing anti-joins, we need to take into consideration if the positive goal and the negative subgoals have common arguments. Consider the following rule:

a(X) :- b(X), not c(Y).

here *b(X)* is the positive goal and *c(Y)* is the negative subgoal. However, the positive goal and the negative subgoal have no common arguments. Thus, every positive goal will be included as final goal since there is no negative subgoal preventing conclusions. In this case we can optimize by omitting anti-joins since they do not affect the final result.

Cartesian Product. The calculation of a positive goal may depend on positive subgoals that have no common arguments. Such calculation results in a cartesian product. Consider the following rule:

a(X,Y) :- b(X), c(Y), not d(X).

In order to compute the positive goal (say *bc(X,Y)*) we need to compute the cartesian product of *b(X)* and *c(Y)*. To the best of our knowledge there is no efficient solution proposed in the literature for cartesian product computation for the MapReduce framework. However, if one of *b(X)* or *c(Y)* fits in memory (say *b(X)*), then cartesian product can be computed in the same fashion as a single join that is performed in Map. We load *b(X)* in main memory, while a Map function is applied on each *c(Y)*. Thus, cartesian product is produced by matching each *c(Y)* with every *b(X)* that is found in memory.

Nested Subgoals. For simplicity of presentation we focused on rules where **not** was applied only on literals. However, our approach can be generalized for programs containing rules where **not** applies to a conjunction of literals. Consider the following program:

p(X,Y) ← a(X,Z), b(Z,Y), **not** (c(X,Z), **not** d(Y)).

We can compute *p(X,Y)* by rewriting the program. We need to replace each **not** that applies to a conjunction of literals with a logically equivalent expression by transforming the body of the rule into *Disjunctive Normal Form*. Let us perform the following transformations:

a(X,Z) ∧ b(Z,Y) ∧ **not** (c(X,Z) ∧ **not** d(Y)). ≡
a(X,Z) ∧ b(Z,Y) ∧ (**not** c(X,Z) ∨ d(Y)). ≡
(a(X,Z) ∧ b(Z,Y) ∧ **not** c(X,Z)) ∨ (a(X,Z) ∧ b(Z,Y) ∧ d(Y)).

Since we have a disjunction of conjunctive clauses, we may rewrite the program by replacing the aforementioned rule with a set of new rules. Specifically, for each conjunctive clause, we introduce a new rule, where the head of the new rule is the head of the initial rule (p(X,Y)), while the body of the new rule is the conjunctive clause. Thus, for the aforementioned program the initial rule will be replaced by the following rules:

p(X,Y) ← a(X,Z), b(Z,Y), **not** c(X,Z).
p(X,Y) ← a(X,Z), b(Z,Y), d(Y).

Once we have generated the new program, we may proceed with computing ranks and then perform reasoning as described above (provided that the new program is stratified and for each rule of the new program, all arguments of the final goal belong to a positive subgoal, while the set of arguments of negative subgoals is a subset ($\subseteq$) of the set of arguments of positive subgoals).

3.4 Final Remarks

Consider the following program:

p(X,Y) ← a(X,Z), b(Z,Y), **not** c(X,Z), **not** d(Y).

For simplicity of presentation we proposed the computation of the positive goal *ab(X,Z,Y)*, followed by two anti-joins, first with *c(X,Z)* and then with *d(Y)*. However, one can follow a more optimal approach by mixing the application of joins and anti-joins. Specifically, for the aforementioned program, we can perform an anti-join on *a(X,Z)* and *c(X,Z)*, producing *ac(X,Z)*, and an anti-join on *b(Z,Y)* and *d(Y)*, producing *bd(Z,Y)*. Subsequently, we join *ac(X,Z)* and *bd(Z,Y)* in order to compute the final goal.

The second approach is more optimal since it generates less intermediate results, while calculating the required final goal (p(X,Y)). However, in order to reassure correct application of anti-joins, for each anti-join on a positive and a negative subgoal the following condition must hold: the set of arguments of the negative subgoal (NS) is a subset of the set of arguments of the positive subgoal (PS), while the two sets have at least one common argument, namely $NS \subseteq PS$ and $(NS \cap PS) \neq \emptyset$.

Now let us point out the necessity of the imposed restrictions. It is required that for each rule, all arguments of the final goal belong to a positive subgoal. Consider the following program:

p(X,Y) ← a(X,Z), **not** b(Z,Y).

here for each value of *X* in *a(X,Z)*, we need to compute the following subset $new\_Y = H\_U - Y\_In\_b(Z,Y)$, where H_U is the Herbrand universe and $Y\_In\_b(Z,Y)$ is the set of values of *Y* that are found in *b(Z,Y)* such that *a(X,Z)* and *b(Z,Y)* have common values on *Z*. Such computation will introduce a significant overhead for the computation of new_Y, and will require either a cartesian product of each value of *X* with its corresponding subset new_Y (which is not applicable in general, see Subsection 3.3) or storing the final goal (p(X,Y)) in the form *p(X,*new_Y*)* which will result in long new_Y sequences that will eventually affect parallelization.

We have posed an additional restriction, namely in order to perform an anti-join, the set of arguments of the negative subgoal must be a subset ($\subseteq$) of the set of arguments of the positive subgoal. Consider the following program:

$$p(X,Y) \leftarrow a(X,Y), \textbf{not } b(Y,Z).$$

here we cannot perform an anti-join on a(X,Y) and **not** b(Y,Z) in order to compute the final goal (p(X,Y)) since for a given value of Y we need to check whether all literals *b(Y, H_U)*, where H_U is the Herbrand universe, are classified as positive or negative. Thus, for given *X,Y* we may infer the final goal (p(X,Y)), if there is at least one *b(Y,Z)* that is classified as negative. However, for more complex rules such as:

$$p(X,Y) \leftarrow a(X,Z), b(Z,Y), \textbf{not } c(Z,W), \textbf{not } d(W,U).$$

an efficient implementation for MapReduce is yet to be defined since for a given Z we need to find a combination of *c(Z,W)* and *d(W,U)* that are both classified as negative, while avoiding the full materialization of negative literals. Full materialization of both positive and negative literals (Herbrand base) may easily become prohibiting even for small datasets, and thus, is not applicable to big data.

Finally, the approach for non-stratified programs is different and comes with certain challenges. An extension of our approach would be the computation of the well-founded semantics. However, the definition of unfounded sets (see Definition 4) affects the scalability of the whole process. In order to conclude that $p \in A$ we need to check every instantiated rule R of **P** whose head is p. Such evaluation has to be conducted by a single node (since the MapReduce framework does not allow communication between nodes during map or reduce phase), causing either main memory insufficiency or skewed load balancing, decreasing the overall parallelization.

4 Experimental Evaluation

Methodology. In order to evaluate our approach, we searched for proposed benchmarks in the literature. In [14], authors evaluate the performance of several rule engines on data that fit in main memory. However, our approach is targeted on data that exceed the capacity of the main memory. Thus, in order to perform evaluation, we adjusted certain parameters of the proposed methodology in [14]. We evaluate our approach considering *large join tests*, *default negation* and *datalog recursion*, while the rest of the proposed evaluation metrics in [14] are not applicable. Specifically, we do not perform evaluation for *indexing* since the MapReduce framework does not provide such an option. We have not yet developed a complete system that could perform reasoning based on our approach, thus, all optimizations and cost-based analysis were performed manually. Finally, in [14], authors separate *loading* and *inference* time, focusing on inference time. However, such a separation is difficult for our approach since loading and inference time may overlap.

Dataset. Experiments in [14] where based on datasets that consisted of up to several millions facts. We aim to evaluate our approach for up to 1 billion facts. The main goal of our approach is to evaluate performance in terms of execution time and reassure scalability. Thus, we perform experiments for several dataset sizes for both *uniform* and *zipf* (highly skewed) data distributions.

Large Join Tests. Consider the following program:

$$p_i(X,Y) \leftarrow a(X,Z), b(Z,Y).$$

for $1 \leq i \leq N$, where N is the total number of rules following the above rule pattern. Scalability of large joins is examined over several dataset sizes, number of rules, data distributions and number of nodes.

Default Negation. In this paper, we provide a solution for stratified programs. Thus, we cannot use the well known win-not-win example [17] (also used in [14]), which works for locally stratified and non-stratified programs. Consider the following program:

$$p(X,Y) \leftarrow a(X,Y), \textbf{not } b(X,Y).$$

Scalability of anti-joins is examined over several dataset sizes, data distributions and number of nodes. The number of rules has the same effect for anti-joins as for joins (since it affects only the total amount of final output), and thus, such an evaluation is omitted.

Datalog Recursion. In order to evaluate datalog recursion, we applied a different evaluation method. Instead of generating random data and calculating the transitive closure of a relation, we evaluated joins using the program described in *large join tests* (for $N = 1$). We performed joins, for uniform data distribution (zipf distribution would require more complex computation techniques, which are out of the scope of this paper), changing the percentage of the matched values for the argument *Z* from 0% to 100%. In such a way, we were able to estimate the required time for each matching percentage, as this percentage may vary significantly throughout the transitive closure calculation.

Platform. We have implemented our experiments using the Hadoop MapReduce framework[2], version 1.0.4. We have performed experiments on a cluster of the University of Huddersfield. The cluster consists of 9 nodes (one node was allocated as "master" node), using a Gigabit Ethernet interconnect. Each node was equipped with 2 cores running at 1.86GHz, 3GB RAM and 150GB of storage space.

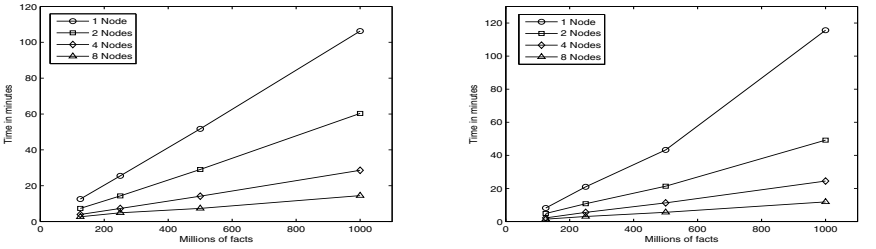
Results. We can identify four main factors that affect the performance of our approach:

1. **Number of facts**, affecting the input size.
2. **Number of rules**, affecting the output size.
3. **(Anti-)Join percentage**, affecting the output size.
4. **Compression ratio**, affecting the output size, when compression algorithm is used.

our results correspond to several combinations of the above four factors.

Figure 3 shows the runtimes of our system for join operations with input sizes up to 1 billion facts, while number of rules is set to 1 and compression ratio remains fairly stable. We see that for uniform distribution our approach scales linearly, as join percentage remains stable at 50%. However, for zipf distribution our approach exhibits several fluctuations as the number of facts increases, since the join percentage fluctuates as well.

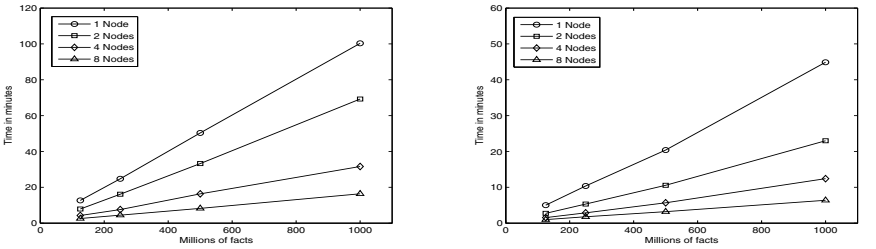
[2] `http://hadoop.apache.org/mapreduce/`



(a) Join for uniform distribution (b) Join for zipf distribution

Fig. 3. Runtime in minutes for join operations as a function of dataset size, for various numbers of nodes

Figure 4 presents the runtimes of our system for anti-join operations with input sizes up to 1 billion facts. For both uniform and zipf distribution, the number of rules is set to 1, while compression ratio and anti-join percentages remain stable. In this case we see that our system scales linearly for both data distributions.



(a) Anti-join for uniform distribution (b) Anti-join for zipf distribution

Fig. 4. Runtime in minutes for anti-join operations as a function of dataset size, for various numbers of nodes

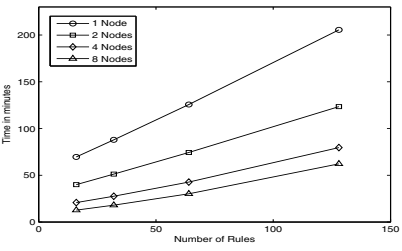


Fig. 5. Runtime in minutes for various numbers of rules and nodes

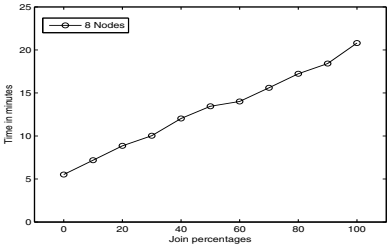


Fig. 6. Runtime in minutes for various matched values percentages

Figure 5 depicts the scaling properties of our system for 16, 32, 64 and 128 rules. We need to point out that the system scales linearly for up to 64 rules, while for 128 rules the runtime is higher than the expected (linear). This is attributed to the fact that compression for 128 rules is less effective, resulting in larger amounts of output.

Figure 6 illustrates the runtimes of our system for various join percentages, while all the other factors remain stable (500 million facts, 1 rule, 8 nodes and fairly stable compression ratio). As expected, while the join percentage increases, the runtime increases as well since larger amounts of output are being produced. In general, for the case of recursion, long chains of MapReduce jobs and low join percentages should be avoided because reading and sorting/grouping the input introduces a high overhead for the whole computation.

5 Conclusion and Future Work

In this paper we studied the feasibility of stratified semantics over large amounts of data. In particular, we considered stratified semantics, equivalent to the well-founded semantics for stratified programs, proposed a parallel approach based on the MapReduce framework, and ran experiments for various data sizes, rule sizes and data distributions. Our experimental results indicate that such reasoning can scale up to 1 billion facts even on a modest setup.

In future work, we intend to explore the potential of logic programming considering computation over big data through parallelization. Parallelization techniques such as OpenMP[3] and Message Passing Interface (MPI) may provide higher degree of flexibility than the MapReduce framework, giving the opportunity to overcome current limitations. Finally, we plan to study more complex knowledge representation methods, including the well-founded semantics [17], Answer-Set programming [1], RDF/S ontology evolution [11] and repair [18].

References

1. Answer sets. In: van Harmelen, F., Lifschitz, V., Porter, B. (eds.) Handbook of Knowledge Representation, ch. 7
2. Afrati, F.N., Ullman, J.D.: Optimizing joins in a mapreduce environment. In: EDBT (2010)
3. Baader, F., Kosters, R.: Nonstandard Inferences in Description Logics: The Story So Far. In: Mathematical Problems from Applied Logic I. International Mathematical Series, vol. 4 (2006)
4. Billington, D.: Defeasible Logic is Stable. J. Log. Comput. 3(4), 379–400 (1993)
5. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters
6. Duan, S., Kementsietsidis, A., Srinivas, K., Udrea, O.: Apples and oranges: a comparison of RDF benchmarks and real RDF datasets
7. Fensel, D., van Harmelen, F., Andersson, B., Brennan, P., Cunningham, H., Valle, E.D., Fischer, F., Huang, Z., Kiryakov, A., Il Lee, T.K., Schooler, L., Tresp, V., Wesner, S., Witbrock, M., Zhong, N.: Towards larkc: A platform for web-scale reasoning. In: ICSC, pp. 524–529 (2008)

[3] `http://openmp.org/wp/`

8. Fische, F.: Investigation & Design for Rule-based Reasoning. Tech. rep., LarKC (2010)
9. Goodman, E.L., Jimenez, E., Mizell, D., Al-Saffar, S., Adolf, B., Haglin, D.: High-Performance Computing Applied to Semantic Databases. In: Antoniou, G., Grobelnik, M., Simperl, E., Parsia, B., Plexousakis, D., De Leenheer, P., Pan, J. (eds.) ESWC 2011, Part II. LNCS, vol. 6644, pp. 31–45. Springer, Heidelberg (2011)
10. Haase, C., Lutz, C.: Complexity of Subsumption in the EL Family of Description Logics: Acyclic and Cyclic TBoxes. In: ECAI 2008, pp. 25–29 (2008)
11. Konstantinidis, G., Flouris, G., Antoniou, G., Christophides, V.: A Formal Approach for RDF/S Ontology Evolution. In: ECAI (2008)
12. Kotoulas, S., van Harmelen, F., Weaver, J.: KR and Reasoning on the Semantic Web: Web-Scale Reasoning (2011)
13. Kotoulas, S., Oren, E., van Harmelen, F.: Mind the data skew: distributed inferencing by speeddating in elastic regions. In: WWW, pp. 531–540 (2010)
14. Liang, S., Fodor, P., Wan, H., Kifer, M.: Openrulebench: an analysis of the performance of rule engines. In: Proceedings of the 18th International Conference on World Wide Web, WWW 2009, pp. 601–610. ACM, New York (2009), http://doi.acm.org/10.1145/1526709.1526790
15. Nebel, B.: Terminological Reasoning is Inherently Intractable. Artificial Intelligence 43, 235–249 (1990)
16. Oren, E., Kotoulas, S., Anadiotis, G., Siebes, R., ten Teije, A., van Harmelen, F.: Marvin: Distributed reasoning over large-scale Semantic Web data. J. Web Sem. 7(4), 305–316 (2009)
17. Ross, K.A.: The well-founded semantics for general logic programs. Journal of the ACM 38, 620–650 (1991)
18. Roussakis, Y., Flouris, G., Christophides, V.: Declarative Repairing Policies for Curated KBs. In: HDMS (2011)
19. Serfiotis, G., Koffina, I., Christophides, V., Tannen, V.: Containment and Minimization of RDF/S Query Patterns. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 607–623. Springer, Heidelberg (2005)
20. Tachmazidis, I., Antoniou, G., Flouris, G., Kotoulas, S.: Towards parallel nonmonotonic reasoning with billions of facts. In: KR (2012)
21. Tachmazidis, I., Antoniou, G., Flouris, G., Kotoulas, S., McCluskey, L.: Large-scale parallel stratified defeasible reasoning. In: ECAI, pp. 738–743 (2012)
22. Urbani, J., Kotoulas, S., Maassen, J., van Harmelen, F., Bal, H.: OWL reasoning with webPIE: Calculating the Closure of 100 Billion Triples. In: Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T. (eds.) ESWC 2010, Part I. LNCS, vol. 6088, pp. 213–227. Springer, Heidelberg (2010)
23. Urbani, J., Kotoulas, S., Oren, E., van Harmelen, F.: Scalable Distributed Reasoning Using MapReduce. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 634–649. Springer, Heidelberg (2009)

Distributed ECA Rules for Data Management Policies*

Hao Xu[1], Arcot Rajasekar[1], Reagan W. Moore[1], and Mike Wan[2]

[1] DICE Center, University of North Carolina at Chapel Hill, USA
xuh@email.unc.edu, {sekar|moore}@diceresearch.org
[2] Retired from DICE Center, University of California at San Diego, USA

Abstract. Data management policies for a distributed system that handles "big data" usually require a core feature set provided by a rule framework. In this paper, we describe a viable core feature set, partially extending the event-condition-action framework, based on our experience in developing and applying the integrated Rule-Oriented Data-management System [1,2,3]. We attempt to formalize some aspects of this feature set, focusing on location-aware and asynchronous execution of rules, based on the Calculus of Concurrent Systems [4].

1 Introduction

Modern data management systems manage collections ranging in size from moderate to a hundred million files or more totaling petabytes of data. The requirements for managing large collections of data include both a number of generic capabilities and diverse features that depend on the details of the data management application. A rule-based data management system provides the flexibility for implementing heterogeneous data management systems within a general framework, thereby improving the reuse of data management expertise and infrastructures, and speeding up the development and deployment of new types of data grids. In a data management system, data management policies usually involve keeping track of data objects and their provenance through their lifecycles, and validating properties such as authenticity and integrity. The advent of "big data" further requires data management systems to perform these operations in a scalable way. This imposes a unique challenge for implementing data management policies on such systems.

To address this challenge, we developed the integrated Rule-Oriented Data-management System (iRODS) [1,2,3], a community-driven open source data grid software solution that helps researchers, archivists and others manage large sets of computer files. In addition to a comprehensive set of generic features, including high-performance network data transfer, support for a wide range of

* This research is partially supported by NSF grant #0940841 "DataNet Federation Consortium" and NSF grant #1032732 "SDCI Data Improvement: Improvement and Sustainability of iRODS Data Grid Software for Multi-Disciplinary Community Driven Application".

L. Morgenstern et al. (Eds.): RuleML 2013, LNCS 8035, pp. 203–217, 2013.

physical storage systems, backup and replication, and metadata management, a key component that makes iRODS highly configurable and easily extensible for a wide range of use cases is policy enforcement points (PEPs) and the iRODS rule engine. PEPs are events generated from the data management system which trigger the execution of rules. The rule engine provides a rule language which can be used to define machine actionable rules which implement data management policies at PEPs. These features have been vetted through the application of the software across multiple scientific domains (astronomy, oceanography, seismology, hydrology, biology, social sciences, climate studies, neuroinformatics, patient health, genomics, etc.) and across multiple types of data management applications (data sharing, publication, archiving, analysis).

In this paper, we overview the iRODS rule engine, focusing on its extensions to the event-condition-action (ECA) framework, provide a formalism for remote and asynchronous execution of rules in our rule engine, and discuss some implementation issues. In the next section, we give a bird's-eye view of extensions to ECA in the iRODS rule language. In the third section, we formalize part of our extensions by extending Calculus of Concurrent Systems (CCS) [4] with the concepts of multiple sorts of localities and proximity to data. In Sect. 4, we discuss some of the general features of the iRODS rule engine. In Sect. 5, we discuss related work and in Sect. 6, we summarize the paper.

2 Extensions to ECA in the iRODS Rule Language

We have designed and implemented our rule engine from scratch, which has been a key component of iRODS since the the first release of the software. The design of the iRODS rule language starts with a simple ECA framework, where the basic structure of a rule in the iRODS rule language looks like [1]

```
RuleName {
     on(RuleCondition) {
          RuleActions
     }
}
```

As the software evolves, we added new features to the rule engine based on the requirements of our application domains. In the following subsections, we introduce the following features: location-aware execution, asynchronous execution, error handling, static and dynamic checking, and controlled vocabularies.

2.1 Location-Aware Execution

An execution of a computer program requires both data and code. Location-aware execution is useful in scenarios where we need to ask the question such as "should we move the data closer to the code or the code closer to the data?" In particular, the following scenarios are of interest:

[1] If we follow W3C's classification of rules [5], this rule naturally falls into the reactive category. However, iRODS rules can also be used to implement other types of rules.

- Users have terabyte files on a remote storage location. They want to execute some rule to extract the headers of those files. In this case, it is more efficient to run the rule at the remote storage location, without moving the files, and send the result back, than move the files and run the rule locally.
- Users have CPU intensive computations that operate on a set of relatively small files in a remote storage location. In this case, it is more efficient to run the rule at a server node where there is sufficient idle CPU resource, than on the remote storage location.

In our rule language, location-aware execution is implemented by the `remote` microservice. The iRODS rule language supports two ways to specify where a rule should be run. Explicitly, a rule programmer can specific on which server node of a distributed data management system a rule should be run. For example, suppose Node A receives an event which would trigger Rule R to execute. Here the rule can specify that Node A should send Rule R to be run on Node B. A concrete example could be a dedicated server for virus scan:

```
acPostProcForPut {
    remote("virusScanServer") {
        scanVirus($objPath);
    }
}
```

This rule is triggered at PEP `acPostProcForPut`. This PEP is invoked when a data object has been ingested. We assume that `virusScanServer` is the host name for our virus scan server, and `scanVirus` is a microservice that scans for a virus. `$objPath` is the logical path of the data object that has been ingested. This rule invokes the `scanVirus` microservice on the `virusScanServer` server. In more complex rules, a rule can be sent among multiple nodes, which means that we can have a chain of responsibility pattern, in which each node processes the rule and passes it on to the next node in the chain.

In addition, one can implicitly specify that a rule should be executed where the data are located, when the cost of moving large chunks of data across the network can be much larger than the cost of moving rules.

2.2 Asynchronous Execution

While synchronous execution may be a more straightforward programming model, asynchronous execution is crucial for scalability in a distributed system, as it ensures that one node is not necessarily blocked while waiting for the response to a request sent to another node.

In the iRODS rule language, asynchronous execution is specified by the `delay` microservice. As an example, we can make the rule from the previous subsection asynchronous, so that it returns immediately to the server which initiates the remote execution. We can write it like this

```
acPostProcForPut {
    remote("virusScanServer") {
        delay("<PLUSET>0s</PLUSET>") {
            scanVirus($objPath);
        }
    }
}
```

The `delay` microservice makes the following code block asynchronous. The parameter `<PLUSET>0s</PLUSET>` means that we want the delay to be 0 seconds, which means the code block is scheduled without delay. Note that the variables are preserved throughout the remote and delayed execution, so that we can access them inside remote and delayed blocks.

There is a list of other parameters that can be used to schedule other execution modes. Examples are: run repeatedly or once, run at a fixed time point or at a time point relative to the time it is received, and repeat at fixed intervals or repeat at variable intervals for repeated rule.[2]

These execution modes are useful in a data management system. For example,

- Users have a large number of files to ingest. They want to finish the initial ingest as fast as possible, and delay the virus check to later.
- Users want to do a virus check periodically at a fixed internal.

The execution of an asynchronous rule is implemented through node-local prioritized queues. The node-local scheduler makes a best-effort attempt to execute each rule in the queue. When the load exceeds the computational capabilities of a node, the rules stay in the queue until they are scheduled.

2.3 Error Handling

Because of the complexity of the underlying data management system, the rule language cannot impose a specific error handling scheme. In iRODS, we need to deal with both transactional and nontransactional parts of the underlying system. Therefore, instead of giving an error-handling-mechanism-specific semantics to the rule language, we provide a simple language construct where system-specific error handling can be implemented. For each action, the rule programmer can specify a recovery action, which traps the errors thrown by that action. In the recovery action, the rule can try to detect any conflicts or try to "undo" exactly what the action did. Recovery actions provide a mechanism for implementing rollback of changes to the system when a rule fails. However, it is not always possible to "undo" a change consistently in a distributed and concurrent environment that is only partially transactional, as the change made by one process may depend on that made by another. In this case, the user

[2] Here, we assume that we are dealing with a best-effort type of scheduling system, i.e. there are no real-time requirements. We also assume that the clocks are synchronized and the drift in clocks among the server nodes are negligible in our applications.

can either handle the error by simply notifying the administrator and waiting for a manual solution, or by implementing application-specific schemes such as multi-version concurrency control (MVCC) [6].

The recovery action to an action is specified by the `:::` operator. The recovery actions for an action are triggered when that action or any subsequent action fails and are executed in the reverse order of the actions. For example, we can modify our virus scan rule as follows, where we use the `msiSendEmail` microservice to send an email for a recovery action:

```
acPostProcForPut {
    remote("virusScanServer") {
        delay("<PLUSET>0s</PLUSET>") {
            scanVirus($objPath) :::
            msiSendEmail("admin", $objPath);
        }
    }
}
```

2.4 Static and Dynamic Checking

As is known in software engineering, the cost of fixing a software defect increases with the length of time until it is discovered. Type checking is used in mainstream programming language such as Java and Python to ensure that a class of common errors – type errors – are discovered as soon as possible.

- Static typing and dynamic typing: In languages like Java, most type checking is done at compile time. This type of type checking is called static. In languages such as Python, type checking is done at runtime, which is called dynamic. The advantage of static type checking is that type errors are discovered earlier, while the advantage of dynamic typing is reduced requirements for type annotation, and increased flexibility such as "duck typing" [7]. Gradual typing [8,9] tries to create a middle ground between static typing and dynamic typing. In gradual typing, one can have both statically typed elements and dynamically typed elements.
- Strong typing and weak typing: In both Java and Python, things are strongly typed, meaning that every value has a type, and values of one type cannot be converted to values of arbitrary types. In contrast, C is a language with weak typing, for example, one can convert an unsigned long integer to a pointer of any type, effectively allowing the interpretation of any memory block as any type of object, which could cause computer security problems. In Java, this is almost impossible.

Our rule language has strong gradual typing. This makes it easier for people with less programming experience to start writing rules, as they are not required to annotate anything (functions, variables, rules) with types. As they develop rules, they may find it helpful to have certain static guarantees, which they can achieve by annotating their program with types. For example, if we want to ensure that a parameter of `scanVirus` is a path literal, we could write `scanVirus : path -> integer`,

which specifies that the parameter of `scanVirus` is of type `path` and the return value of the microservice is of type `integer`.

2.5 Controlled Vocabularies

Domain specific rules are supported through a set of controlled vocabularies. On the one hand, controlled vocabularies are a mechanism for limiting how the state of a system can be modified, thereby simplifying the task of reasoning about a system. This is especially important for systems that manage lots of data, where it is crucial to prevent users from inadvertently changing the state of the system. On the other hand, controlled vocabularies provide a mechanism to extend the capability of the system to meet the need of specific domains. This is important for the adoption of data management by researchers from different fields. Examples are data grids used by scientists, which would benefit from the expertise of digital preservation policies in managing their research data.

In iRODS, vocabularies are implemented through the module system. A module usually contains a set of microservices implementing a vocabulary. It can be turned on or off, providing or removing access to the vocabulary. As vocabularies evolve, a mapping mechanism is provided so that the older terms can be mapped to the newer ones.

3 Formalism

In this section, we attempt to provide a formalism for the location-aware execution and asynchronous execution aspects of the iRODS rule engine, based on CCS. Since CCS itself already provides a mechanism for modeling asynchronous events, we are going to focus on location-aware execution in this section first, and revisit asynchronous execution later. In a distributed environment, there is a need for distinguishing between actions performed at different locations. Many formal languages [10,11,12,13,14,15,16,17,18,19] have been studied for representing locality in a process algebra. Locality is a core feature which is used to provide a way to distinguish processes by execution location, which is not possible with the basic process algebra. For example, we cannot distinguish between $a.b + b.a$ and $a|b$ by observational congruence on CCS, but they are distinguishable under location semantics.

Although location semantics help distinguish processes that are more sequential from processes that are more parallel, in practice for a data management system, we need to further distinguish processes running on

- Heterogeneous nodes: difference of the type of location, such as a tape resource or a computation node, and
- Different proximity to data: how far away a process is from the data that it processes.

In this section, we introduce two new concepts that capture these two types of differences. We start with a basic version of CCS with location prefix, and extend it with our new concepts. Our use of the location prefix is closely related to the static approach [18,12] and we follow the notations in [12]. We use CCS instead of the π-calculus [20] for simplicity.

In our formalism, we have a set $\mathcal{P}$ of possible agents,[3] ranged over by P. To construct agents we need actions. Let $\mathcal{N}$ be a set of names of channels, ranged over by $a, b, c, \ldots$. For each name a of a channel, we define two actions a and $\overline{a}$ which correspond to inputing and outputing from channel a. We follow the convention that $\overline{\overline{a}} = a$. We model an event by a channel e. An event is triggered by an input action e. The output action $\overline{e}$ corresponds to listening to the event. In data grids, we also allow actions that model operations on data objects. We model these actions by input channels. Their corresponding output channels should not appear in any agent. There is a special action τ which corresponds to a "handshake" between two agents on a channel. We denote the set of all possible actions by $\mathcal{A}$, ranged over by $\alpha, \beta, \gamma, \ldots$. We have a set $\mathcal{L}$ of locations, ranged over by l. Let $u, v, w, \ldots$ range over sequences of locations $l_1 :: l_2 :: \ldots :: l_n$, which we call localities. If $n = 0$, the locality is denoted by ϵ. Following the standard location semantics, there is a partial prefix order $\prec$ on sequences of locations. To define recursive agents, we use $X, Y, Z, \ldots$ to denote an agent variable that is bound to some agent. We denote the agent obtained by replacing a sequence $\widetilde{X}$ of agent variables with a sequence $\widetilde{P}$ of agents in P by $P(\widetilde{P}/\widetilde{X})$. There is an agent 0 which does nothing, indicating inaction.

We have a set of locality sorts $\mathcal{S}$ and function $t : \mathcal{L}^* \to \mathcal{S}$ which maps a locality to its sort. We use sorts to distinguish different types of localities. For example, to model a data grid with different types of storage nodes, each type of storage nodes can be modeled by a locality sort, and t can map the locality of each node to the sort that models the type of that node. We have a function $loc : \mathcal{A} \to \mathcal{P}(\mathcal{L}^*)$, which maps each action to a set of localities of the data that it operates on. For example, if an action α operates on data from locality u, then we have $u \in loc(\alpha)$. We allow one action to be mapped to more than one locality to model the situation where a data object is replicated to multiple nodes, and the action may choose one of the replicas to operate on.[4] We denote the set of all localities by $\mathbb{U}$ and use it to model actions that do not operate on any data. We have a set of proximities $\mathcal{R}$ with well-founded total order $\leq$ and a maximum element ∞. We use proximities to model the unit cost of data transfer between two nodes in a data grid. Given a subset R of $\mathcal{R}$, we denote by $min_{\leq} R$ the minimum element of R and $min_{\leq} \emptyset = \infty$. We have a proximity function $prx : \mathcal{L}^* \times \mathcal{L}^* \to \mathcal{R}$. In a data grid, we use this function to model the unit cost of transferring data from where the data are stored to where the action that operates on the data is performed. Since our *loc* function allows multiple localities of data, we lift prx to sets of localities: for all $v \in \mathcal{L}^*$, $U \subset \mathcal{P}(\mathcal{L}^*)$,

[3] In our paper, we use the term "agent" in the context of process algebra.

[4] In another type of applications, we could also give the mapping different semantics, where an action has to operate on data from all localities that it is mapped to.

$$P ::= 0 \mid X \mid A.P \mid P|P \mid P\backslash L \mid fix(\widetilde{X} = \widetilde{P}) \mid P + P \mid l :: P$$
$$A ::= \tau \mid a \mid \overline{a} \mid r(l)$$

Fig. 1. Syntax

$$prx(U, v) = min_{\leq}\{prx(u, v) \mid u \in U\} \tag{1}$$

The syntax of an agent is given in Fig. 1. We define a transition relation

$$P \xrightarrow[u]{A} P' \tag{2}$$

between agents. For succinctness, when $u = \epsilon$, we write simply

$$P \xrightarrow{A} P' \tag{3}$$

In each transition, we specify the action taken over the arrow and the locality where the action is taken under the arrow.

Remote Execution. We allow explicit specification of static location by the $r(l)$ action. We now explain the $r(l)$ action. Intuitively,$r(l)$ designates that the subsequent actions should be moved to location l. It forces the following transition

$$u :: r(l).P \xrightarrow[u::l]{r(l)} u :: l :: P \tag{4}$$

Transitions to any other location are not allowed. For example, suppose that we have an agent $r(l).P$. This agent initiates a remote agent $l :: P$ at location l through the transition $r(l).P \xrightarrow[l]{r(l)} l :: P$. The difference between an $r(l)$ action and directly adding a location prefix $l ::$ is that they model different intents. A location prefix only specifies where an action should take place, while an $r(l)$ action specifies a causal relation between an action at the current locality and the subsequent actions that take place at location l. For example, the following agents should be different: $a.0 \mid l :: b.0$ and $a.0 \mid r(l).b.0$. The former specifies that action b takes place at location l but does not specify anything related to the current locality, i.e., agent $b.0$ may very well be at location l to begin with. The latter specifies that agent $b.0$ at location l is a result of a remote execution request from the current locality. We will see that our strong bisimulation relation is able to distinguish between these two agents. This feature is essential in modeling the system behavior when a rule is sent from one server to another.

Asynchronous Execution. Asynchronous rules are modeled by events. An output action $\overline{e}$ delays the execution of subsequent actions until event e is triggered by an input action. There are two types of events that we can model here: Logical events, such as file create, file access, file update, etc., which are triggered by changes of system state, and time-based events, which are triggered by changes

$$\frac{\alpha \neq r(l) \text{ for any } l}{\alpha.P \xrightarrow{\alpha} P} \quad (prefix) \qquad \frac{P \xrightarrow[u]{\alpha} P'}{l :: P \xrightarrow[u::l]{\alpha} l :: P'} \quad (loc)$$

$$\frac{P \xrightarrow{\alpha} P'}{P \mid Q \xrightarrow{\alpha} P' \mid Q} \quad (com1) \qquad \frac{Q \xrightarrow{\alpha} Q'}{P \mid Q \xrightarrow{\alpha} P \mid Q'} \quad (com2)$$

$$\frac{P \xrightarrow{\alpha} P' \quad Q \xrightarrow{\overline{\alpha}} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'} \quad (com3) \qquad \frac{P_j\{\widetilde{fix}(\widetilde{X} = \widetilde{P})/\widetilde{X}\} \xrightarrow{\alpha} P'}{\widetilde{fix}(\widetilde{X} = \widetilde{P}) \xrightarrow{\alpha} P'} \quad (fix_j)$$

$$\frac{P \xrightarrow{\alpha} P' \qquad \alpha \notin L \cup \overline{L}}{P\backslash L \xrightarrow{\alpha} P'\backslash L} \quad (restrict) \qquad \frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'} \quad (sum1)$$

$$\frac{Q \xrightarrow{\alpha} Q'}{P + Q \xrightarrow{\alpha} Q'} \quad (sum2) \qquad r(l).P \xrightarrow[l]{r(l)} l :: P \quad (rem)$$

Fig. 2. Semantics

in the physical clock at some location.[5] To model events from the underlying system, we introduce the notion of event generators. An event generator is a process which only contains input actions to channels that model events: $e_{n_1}.e_{n_2}.\ldots.0$. An event generator models a specific system behavior. For example, suppose that we have events based on a given data object o as follows: e_1: pre data object o put; e_2: post data object o put; e_3: pre data object o get; e_4: post data object o get. A system behavior where data object o is put into the data grid and then retrieved back from the data grid can be modeled by event generator $e_1.e_2.e_3.e_4.0$. We may have multiple event generators. For example, in a data grid with both physical-clock-based events and logical-clock-based events, it is conceptually clearer to group the events into two event generators. For another example, if we want to model a concurrent access to the data grid by two separate users, we can model events generated by those two users in two separate event generators.

The semantics of our calculus is summarized in Fig. 2. Most of the rules are a variant of the standard rules in CCS, except the rule for remote execution which we have discussed above.

Bisimulation. We define a bisimulation relation which

- distinguishes between agents which are not distinguishable in the standard CCS, based on whether they process on heterogeneous nodes and their proximity to data, and

[5] In some applications, there are multiple listeners for a single event. Since events are modeled as channels, we can encode the scenario by using extra auxiliary processes, or directly model the behavior of one source interacting with multiple channels.

– relates agents that are syntactically different but process on the same sort of node and have the same proximity to data.

We define a relation $E \subset \mathcal{A} \times \mathcal{A}$ which identifies all actions of the form $r(l)$: $\alpha E \beta$ if and only if $\alpha = \beta$ or α is of the form $r(l)$ and β is of the form $r(l')$ for some $l, l' \in \mathcal{L}$. A strong bisimulation relation $\sim$, parametrized by t, loc, and prx, should satisfy the following property:

– If $P \sim Q$ and $P \xrightarrow[u]{\alpha} P'$, then there exists a Q' and a v such that $Q \xrightarrow[v]{\beta} Q'$, $P' \sim Q'$, $\alpha E \beta$, $t(u) = t(v)$, and $prx(loc(\alpha), u) = prx(loc(\alpha), v)$.
– If $P \sim Q$ and $Q \xrightarrow[u]{\alpha} Q'$, then there exists a P' and a v such that $P \xrightarrow[v]{\beta} P'$, $P' \sim Q'$, $\alpha E \beta$, $t(u) = t(v)$, and $prx(loc(\alpha), u) = prx(loc(\alpha), v)$.

Compared to the strong bisimulation relation of CCS: We do not require that α and β are identical but equivalent under E. We require $t(u) = t(v)$ and $prx(loc(\alpha), u) = prx(loc(\alpha), v)$, i.e., the sort of the localities of the two transitions and the proximity to data should be the same.

We can define weak bisimulation and observational congruence in a similar manner to their definition in the standard CCS [12]. In our observational congruence, we can further distinguish between a strong observational congruence and a weak observational congruence by whether we choose to ignore actions of the form $r(l)$.

As an example, suppose our data grid is set up as follows. There are two disk storage nodes at location d_1 and location d_2, one tape storage node d_3 and two computation nodes at location c_1 and c_2. Before defining the functions and relations, we first establish a equivalence relation on localities. We have the following equivalent classes:

$$\{\epsilon\}, \{u :: d_1 \mid u \in \mathcal{L}^*\}, \{u :: d_2 \mid u \in \mathcal{L}^*\},$$
$$\{u :: d_3 \mid u \in \mathcal{L}^*\}, \{u :: c_1 \mid u \in \mathcal{L}^*\}, \{u :: c_2 \mid u \in \mathcal{L}^*\}$$

For each equivalent class we choose a representative:

$$\epsilon, d_1, d_2, d_3, c_1, c_2$$

We will define all functions and relations on these representatives. We extend the definition by identifying a locality with its representative.

The set $\mathcal{S}$ of locality sorts is $\{\mathsf{d}, \mathsf{t}, \mathsf{c}, \mathsf{u}\}$, where d corresponds to disk storage, t corresponds to tape storage, c corresponds to computation, and u corresponds to unspecified. The function t is defined as follows: for all $i \in \{1, 2\}$,

$$t(\epsilon) = \mathsf{u} \tag{5}$$
$$t(d_i) = \mathsf{d} \tag{6}$$
$$t(d_3) = \mathsf{t} \tag{7}$$
$$t(c_i) = \mathsf{c} \tag{8}$$

The set $\mathcal{R}$ is defined as $\{0, 1\}$. We define $\leq$ on $\mathcal{R}$ as $\{(0, 0), (0, 1), (1, 1)\}$ and let $\infty = 1$. We have a binary discrete measure here. The proximity prx function is defined as follows: for all $u, v \in \{d_1, d_2, d_3, c_1, c_2\}$,

$$prx(u, v) = \begin{cases} 0, & u = v \neq \epsilon \\ 1, & \text{otherwise} \end{cases} \tag{9}$$

This proximity function allows us to distinguish between a process which is run on a server where the data object being processed are stored and a process which is run on a server where the data object being processed are stored somewhere else. Now suppose that we have two families of actions e_i and f_i for $i \in \{1, 2, 3\}$, such that e_i requires more data accesses and f_i requires more CPU resources. We define loc as for all $i \in \{1, 2, 3\}$,

$$loc(e_i) = \{d_i\} \tag{10}$$

$$loc(f_i) = \{d_i\} \tag{11}$$

i.e., the actions are indexed by the indices of the data storage node where the data they process are stored.

Next, let us take a look at the following agents:

1. $e_1 \mid f_2$: run e_1 and f_2 at an unspecified node.
2. $e_1.f_2 + f_2.e_1$: run either $e_1.f_2$ or $f_2.e_1$ at an unspecified node.
3. $r(d_1).e_1 \mid f_2$: run e_1 at d_1 and f_2 at an unspecified node.
4. $r(d_1).e_1 \mid r(d_2).f_2$: run e_1 at d_1 and f_2 at d_2.
5. $r(d_3).e_1 \mid r(c_2).f_2$: run e_1 at d_3 and f_2 at c_2.
6. $r(d_2).e_1 \mid r(c_2).f_2$: run e_1 at d_2 and f_2 at c_2.
7. $r(d_1).e_1 \mid r(c_1).f_2$: run e_1 at d_1 and f_2 at c_1.
8. $r(d_1).e_1 \mid r(c_2).f_2$: run e_1 at $d_1$and f_2 at c_2.

The most optimal agent is 7 or 8, which in our setting is bisimilar. Even though they execute f_2 on different nodes, these nodes are of the same sort and have the same proximity to the data. However, neither of them is bisimilar with any other agent because of either different locality sorts or different proximities to the data.

4 Implementation

Rules for distributed data management systems usually are written based on an asynchronous, distributed execution model, which requires the rule engine implementation to work closely with the data management system, i.e., the rule engine should be able to tap into the distributed infrastructure provided by the underlying data management system, and complement it with rule specific components that can also be managed by the underlying data management system. This is a tighter integration than merely adding a non-distributed rule engine engine to a distributed system. In this section, we briefly discuss some general features of the iRODS rule engine: debugging, extensibility, and interoperability.

4.1 Rule Debugger

The task of debugging a distributed rule system is challenging. We facilitate this task by providing a rule debugger that provides the capability of online debugging through a distributed debugger, and supports offline debugging through an auditing mechanism. The rule debugger is based on the iRODS messaging system called XMessage. During online debugging, one or more iRODS agents can be attached to or detached from the debugger through XMessage. When an agent is attached to the debugger, commands and updates are issued through a debugging console which provides basic functionality for a general debugger and transferred through XMessage. Offline debugging is supported by a rule auditing framework which logs all events generated by the distributed rule engine. The iRODS rule engine supports rule metadata which tell the rule debugger whether or not to audit a specific rule, creating an effective separation of system-level rules and user-level rules.

4.2 Extensibility

Domain specific languages (DSLs) help improve the efficiency of rule development, reduce errors, and improve readability of rules. A classic example is Structured Query Language (SQL). When SQL is passed around as a generic string in a programming language, the following problems could occur:

- Syntax errors are detected at runtime, as opposed to compile time. This increases the roundtrip time for fixing errors in software development.
- Parametrized SQL statements generated through simple string concatenation without robust validation are prone to SQL injection attacks.
- The syntax usually involves making calls to APIs which construct, execute, and iterate over a SQL statement, which is not as readable as the SQL statement itself.

The iRODS rule language provides support to embedded DSLs through an extensible compiler architecture. The rule parser is written in a C macro based embedded DSL which implements a variant of the parsing expression grammar (PEG). New productions can be added to the parser using this DSL. The DSL makes use of the C macro facilities to translate DSL code to C code, eliminating the need for any external tools, ensuring cross platform support.[6] The semantic extension can be done by adding new microservices. Each microservice provides a way to interpret certain types of language features.

An example is language integrated general query. In iRODS, we have a database, called iCAT, which can be queried through a general query language that is similar to SQL. The rule language provides support to this specialized query language through the extensible compiler feature discussed above.

[6] Including Linux, Solaris, Mac OS and AIX. [21]

4.3 Interoperability with Underlying Data Management System Components

There are two requirements for interoperability. First, the rule engine should be able to access the underlying data management system. This is usually accomplished by providing a programming API (microservice modules). Second, the rule engine should be able to make use of the underlying data management system to improve its own scalability, robustness, and managibility.

The iRODS data management system can be configured to manage rules as data objects, which enables several important features. First, the rules are backed up. The iRODS backup module puts all the rule engine related files into an iRODS collection or the iCAT, which can be backed up using iRODS's own facilities. In case of a local file corruption, the rules can be retrieved from an iRODS collection or the iCAT. Second, the rules can be distributed to multiple server nodes through the iCAT, using a rule, without having to manually copy the rules to other server nodes. Third, the rules can be assigned metadata, which improves the manageability of the rules.

5 Related Work

There are many rule languages which implement ECA. RuleML [22,23,24] consolidates many different types of rules within a single framework. Drools [25] is a rule engine which provides a rule language that can be used to implement ECA rules and their variants. Jess [26] is a rule engine that provides a rule based programming framework; the programming language is a superset of the CLIPS [27] programming language. Web Services Business Process Execution Language (BEPL) [28] is an OASIS standard which is designed for orchestration of business processes, which provides features for programming rules that react on message. Closely related to the iRODS rule engine are, Taverna [29], which is a workflow engine that provides an XML based language workflow language, and Kepler [30], a scientific workflow engine with a graphical workflow language. Both of these two systems provide extensive graphical user interface support for editing and viewing the workflow through various related tools. OASIS XACML separately developed a similar concept to our PEP, which is also called policy enforcement points [31]. We based our formalism on CCS. Other process algebras [32] include Communicating Sequential Processes [33], Algebra of Communicating Processes [34], and the π-calculus [20]. Most of these calculi share common constructs such as sequential composition, parallel composition, and communication. We choose to use CCS as it is simple and its location semantics have been extensively studied. Many formal languages [10,11,12,13,14,15,16,17,18,19] have been studied for representing locality in a process algebra, failures [35,10,17], and reversibility [36]. Our extension addresses the problem that are important in implementing data management policies on a distributed, heterogeneous data grid. We do not address complex event processing (CEP) [37] or event calculus (EC) [38,39] in this paper. Finding out how to incorporate these concepts in our formalism is part of future work.

6 Summary

In this paper, we described the following extensions to ECA: asynchronous and delayed execution, remote execution, recovery actions, typing, and vocabulary extension. These extensions have been implemented in the iRODS data management system, and have been vetted through the application of the software across multiple scientific domains and across multiple types of data management applications. With these extensions, we enable the tuning of generic infrastructure to meet the requirements of a specific application, which is essential for building generic data management infrastructures. We formalize part of these extensions by extending CCS with the concepts of multiple sorts of localities and proximity to data. We define a bisimulation relation that can distinguish between agents that run at different sorts of localities and at different proximities to data.

References

1. http://irods.org
2. Ward, J.H., Wan, M., Schroeder, W., Rajasekar, A., de Torcy, A., Russell, T., Xu, H., Moore, R.W.: The integrated Rule-Oriented Data System (iRODS) Microservice Workbook (2012)
3. Rajasekar, A., Moore, R., Hou, C.Y., Lee, C.A., Marciano, R., de Torcy, A., Wan, M., Schroeder, W., Chen, S.Y., Gilbert, L., Tooby, P., Zhu, B.: iRODS Primer: integrated Rule-Oriented Data System. Synthesis Lectures on Information Concepts, Retrieval, and Services 2(1), 1–143 (2010)
4. Milner, R.: Communication and Concurrency. Prentice-Hall, Inc., Upper Saddle River (1989)
5. http://www.w3.org/2005/rules/wg/wiki/Classification_of_Rules.html
6. Bernstein, P.A., Goodman, N.: Concurrency Control in Distributed Database Systems. ACM Computing Surveys (CSUR) 13(2), 185–221 (1981)
7. http://en.wikipedia.org/wiki/Duck_typing
8. Siek, J.G., Taha, W.: Gradual Typing for Functional Languages. In: Scheme and Functional Programming Workshop, vol. 6, pp. 81–92 (2006)
9. Siek, J., Taha, W.: Gradual Typing for Objects. In: Ernst, E. (ed.) ECOOP 2007. LNCS, vol. 4609, pp. 2–27. Springer, Heidelberg (2007)
10. Amadio, R.M.: An Asynchronous Model of Locality, Failure, and Process Mobility. In: Garlan, D., Le Métayer, D. (eds.) COORDINATION 1997. LNCS, vol. 1282, pp. 374–391. Springer, Heidelberg (1997)
11. Boudol, G., Castellani, I., Hennessy, M., Kiehn, A.: Observing Localities. Theoretical Computer Science 114(1), 31–61 (1993)
12. Castellani, I.: Observing Distribution in Processes: Static and Dynamic Localities. Technical Report RR-2276, INRIA (May 1994)
13. Corradini, F., De Nicola, R.: Locality Based Semantics for Process Algebras. Acta Informatica 34(4), 291–324 (1997)
14. De Nicola, R., Ferrari, G.L., Pugliese, R.: KLAIM: A Kernel Language for Agents Interaction and Mobility. IEEE Transactions on Software Engineering 24(5), 315–330 (1998)
15. Montanari, U., Yankelevich, D.: Location Equivalence in a Parametric Setting. Theoretical Computer Science 149(2), 299–332 (1995)

16. Murphy, D.: Observing Located Concurrency. In: Borzyszkowski, A.M., Sokolowski, S. (eds.) MFCS 1993. LNCS, vol. 711, pp. 566–576. Springer, Heidelberg (1993)
17. Riely, J., Hennessy, M.: Distributed Processes and Location Failures. In: Degano, P., Gorrieri, R., Marchetti-Spaccamela, A. (eds.) ICALP 1997. LNCS, vol. 1256, pp. 471–481. Springer, Heidelberg (1997)
18. Aceto, L.: A Static View of Localities. Formal Aspects of Computing 6, 201–222 (1994)
19. Boudol, G., Castellani, I., Hennessy, M., Kiehn, A.: A Theory of Processes with Localities. Formal Aspects of Computing 6, 165–200 (1994)
20. Milner, R., Parrow, J., Walker, D.: A Calculus of Mobile Processes, I. Information and Computation 100(1), 1–40 (1992)
21. `https://www.irods.org/index.php/Testing`
22. Paschke, A., Boley, H., Zhao, Z., Teymourian, K., Athan, T.: Reaction RuleML 1.0: Standardized Semantic Reaction Rules. In: Bikakis, A., Giurca, A. (eds.) RuleML 2012. LNCS, vol. 7438, pp. 100–119. Springer, Heidelberg (2012)
23. `http://ruleml.org`
24. Boley, H., Paschke, A., Shafiq, O.: RuleML 1.0: The Overarching Specification of Web Rules. In: Dean, M., Hall, J., Rotolo, A., Tabet, S. (eds.) RuleML 2010. LNCS, vol. 6403, pp. 162–178. Springer, Heidelberg (2010)
25. `http://www.jboss.org/drools/`
26. `http://herzberg.ca.sandia.gov/`
27. `http://clipsrules.sourceforge.net`
28. Organization for the Advancement of Structured Information Standards (OASIS): Web Services Business Process Execution Language (WS-BPEL) Version 2.0 (April 2007)
29. `http://www.taverna.org.uk/`
30. `http://kepler-project.org`
31. eXtensible Access Control Markup Language (XACML) Version 3.0
32. Baeten, J.C.: A Brief History of Process Algebra. Theoretical Computer Science 335(2), 131–146 (2005)
33. Hoare, C.A.R.: Communicating Sequential Processes. Communications of the ACM 21(8), 666–677 (1978)
34. Bergstra, J.A., Klop, J.W.: The Algebra of Recursively Defined Processes and the Algebra of Regular Processes. In: Paredaens, J. (ed.) ICALP 1984. LNCS, vol. 172, pp. 82–94. Springer, Heidelberg (1984)
35. Amadio, R.M., Prasad, S.: Localities and Failures (extended summary). In: Thiagarajan, P.S. (ed.) FSTTCS 1994. LNCS, vol. 880, pp. 205–216. Springer, Heidelberg (1994)
36. Danos, V., Krivine, J.: Reversible Communicating Systems. In: Gardner, P., Yoshida, N. (eds.) CONCUR 2004. LNCS, vol. 3170, pp. 292–307. Springer, Heidelberg (2004)
37. Paschke, A., Vincent, P., Springer, F.: Standards for Complex Event Processing and Reaction Rules. In: Olken, F., Palmirani, M., Sottara, D. (eds.) RuleML - America 2011. LNCS, vol. 7018, pp. 128–139. Springer, Heidelberg (2011)
38. Shanahan, M.: The Event Calculus Explained. In: Veloso, M.M., Wooldridge, M.J. (eds.) Artificial Intelligence Today. LNCS (LNAI), vol. 1600, pp. 409–430. Springer, Heidelberg (1999)
39. Bragaglia, S., Chesani, F., Mello, P., Sottara, D.: A Rule-Based Calculus and Processing of Complex Events. In: Bikakis, A., Giurca, A. (eds.) RuleML 2012. LNCS, vol. 7438, pp. 151–166. Springer, Heidelberg (2012)

Semantic Relation Extraction from Legislative Text Using Generalized Syntactic Dependencies and Support Vector Machines

Guido Boella, Luigi Di Caro, and Livio Robaldo

Department of Computer Science, University of Turin
Corso Svizzera 185, Turin, Italy

Abstract. In this paper we present a technique to automatically extract semantic knowledge from legislative text. Instead of using pattern matching methods relying on lexico-syntactic patterns, we propose a technique which uses syntactic dependencies between terms extracted with a syntactic parser. The idea is that syntactic information are more robust than pattern matching approaches when facing length and complexity of the sentences. Relying on a manually annotated legislative corpus, we transform all the surrounding syntax of the semantic information into abstract textual representations, which are then used to create a classification model by means of a standard Support Vector Machine system. In this work, we initially focus on three different semantic tags, achieving very high accuracy levels on two of them, demonstrating both the limits and the validity of the approach.

Keywords: Automatic Semantic Annotation, Semantic Information Extraction, Dependency Parsing, Support Vector Machines.

1 Introduction and Motivations

At present, there is a huge amount of legal data coming from different sources of information. In the light of this, one of the most interesting challenge is how to semantically analyse such data in order to access, reuse, and create knowledge from what already exists, addressing the problem of the so-called "knowledge acquisition bottleneck", i.e., the prohibitive cost of building semantic resources. In [8], we investigated the plausibility of the best performing method of classification in general on legislative text, improving the performance of the classifier by using both syntactic and statistical analyses.

In this paper we propose a novel technique that is able to automatically identify semantic relations in legal text relying on a Machine Learning-based use of the syntactic dependencies extracted with the parser TULE [18], one of the best-performing syntactic parsers for Italian and English. Most of the existing work in this field uses manual rather than automatic generation of sequential patterns that induce semantic information. Although this approach can achieve good results, it is limited in the sense that it exclusively relies on the sequentiality

L. Morgenstern et al. (Eds.): RuleML 2013, LNCS 8035, pp. 218–225, 2013.

of the expressions. Natural language offers potentially infinite ways of expressing concepts, without necessary posing any limit on the length and complexity of the sentences. Our assumption is that syntax is less dependent than learned patterns on the length and the complexity of textual expressions. In some way, patterns grasp syntactic relationships, but without any linguistic knowledge.

This research is part of a wider effort to use intelligent technologies for analysing legal and legislative data in the Eunomos legal document and knowledge management system [5]. Our overall vision is to make texts more meaningful and clear for professional users who need to know how law affects their domain of interest. To make this effort sustainable and economically profitable, we need to use intelligent technologies as much as possible, from NLP to semantic search, as explained in [7].

2 Related Work

According to well-known surveys on Ontology Learning like [3] and [9], the problem of extracting ontologies from text can be faced at different levels of granuarity. Most of the existing approaches uses symbolic methods that are based on lexico-syntactic patterns, which are manually crafted or deduced automatically [1]. The seminal work of [16] represents the main approach based on fixed patterns to identify IS-A relationships. The main drawback of such technique is that it does not face the high variability of how a relation can be expressed in natural language. Still, it generally extracts single-word terms rather than well-formed and compound concepts. [2] proposed similar lexico-syntactic patterns to extract *part-whole* relationships.

Finally, pure statistical approaches present techniques for the extraction of hierarchies of terms based on words frequency as well as co-occurrence values, relying on clustering procedures [10,12,22]. The central hypothesis is that similar words tend to occur together in similar contexts [15]. In [21], the authors presented an approach based on term co-occurrences to infer relationships between different concepts. Such techniques are defined by [3] as *prototype-based ontologies* rather than formal ontologies, and they usually suffer from the problem of data sparsity in case of small corpora. In [13], the authors proposed a system that used a Machine Learning classifier to associate specific category codes to an input text, for different categories like "Acting agent" and "Emergency type", although without using any advanced Natural Language Processing technique on the sentences.

Since our approach relies on a standard supervised classification strategy, we also mention the existence of other algorithms like Naive Bayes Classifiers, Decision Trees, Neural Networks, and several others; despite this, Support Vector Machines are usually employed when working with textual data. While the common use of these algorithms is to label texts with topics or categories, as in [4], in this paper we present a way to label short chunks of texts with semantic relations.

3 Approach

In this section we present our approach to identify semantic relations between entire legal texts and some entities. Our methodology consists in seeing the problem in the following way: given a set of semantic annotations $sem(x)$ between chunks x and the semantic tag *sem*, the task becomes to fed a SVM-classifier with their syntactic context. All the nouns y that are not associated to the semantic tag *sem* are used as negative examples. This way, the classifier is asked to learn a syntactic model of the chunks that underlies the semantic annotation *sem*. Then, when parsing a new text, all its chunks are passed to the *sem*-based classifier that decides if they can be annotated with *sem*. In the next sections we provide the details on how we organized the classification process.

3.1 Local Syntactic Information

The problem of finding a relation between a term and a semantic label can be faced by using the term's local syntactic information. Dependency parsing is a procedure that extracts syntactic dependencies among the terms contained in a sentence, like modifiers of nouns, arguments of verbs, and so forth. The idea is that a semantic tag may be characterized by limited sets of syntactic contexts. According to this assumption, the task can be seen as a classification problem where each term in a sentence has to be associated with a specific semantic label given its syntactic dependencies.

The process starts as follows: the extracted dependencies are transformed into abstract textual representation in the form of triples. In particular, for each syntactic dependency $dep(a, b)$ (or $dep(b, a)$) of a considered noun a, we create an abstract term *dep-target-*$\hat{b}$ (or *dep-*$\hat{b}$*-target*), where $\hat{b}$ becomes the generic string "*noun*" in case it is a noun (different from a); otherwise it is equal to b. This way, the nouns are transformed into tectual abstractions. This procedure creates a level of generalization of the feature set that collapses the variability of the nouns involved in the syntactic dependencies. For instance, let us consider the sentence below:

> *A pena di una ammenda da 2500 a 6400 euro o dell'arresto da tre a sei mesi, il datore di lavoro deve mantenere in efficienza i dispositivi di protezione individuale e assicurare le condizioni d'igiene, mediante la manutenzione, le riparazioni e le sostituzioni necessarie e secondo le eventuali indicazioni fornite dal fabbricante.*
>
> *[Under penalty of 2500 to 6400 euros or a three to six months detention, the employer must maintain the personal protective equipment and ensure the hygiene conditions through maintenance, repairs and replacements necessary and in accordance with any instructions provided by the manufacturer.]*

The first rows of the result of the dependency parsing (the numbers are unique identifiers)[18] are shown below:

ARG(pena-2,a-1)
RMOD(dovere-24,pena-2)
ARG(ammenda-5,di-3)
ARG(ammenda-5,un-4)
RMOD(pena-2,ammenda-5)
ARG(2500-7,da-6)
RMOD(ammenda-5,2500-7)
ARG(euro-10,a-8)
ARG(euro-10,6400-9)
RMOD(dovere-24,euro-10)
COORD(arresto-13,o-11)
ARG(arresto-13,di-12)
RMOD(pena-2,arresto-13)
ARG(tre-15,da-14)
RMOD(arresto-13,tre-15)
ARG(mese-18,a-16)
ARG(mese-18,sei-17)
RMOD(dovere-24,mese-18)
RMOD(mantenere-25,efficienza-27)
ARG(datore-21,il-20)
SUBJ(dovere-24,datore-21)
ARG(lavoro-23,di-22)
RMOD(datore-21,lavoro-23)
TOP(ROOT-0,dovere-24)
INDCOMPL(dovere-24,mantenere-25)
ARG(efficienza-27,in-26)
ARG(dispositivo-29,il-28)
OBJ(mantenere-25,dispositivo-29)

where SUBJ stands for subject relations, OBJs are themes, ARGs are mandatory arguments, COORDs are cordinations, INDCOMPLs are indirect complements, and RMODs are modifiers. Then, the following terms are identified as nouns by the POS-tagger[18]: *pena, ammenda, euro, arresto, mesi, datore, lavoro, efficienza, dispositivi, protezione condizioni, igiene, manutenzione, riparazioni, sostituzioni, indicazioni, fabbricante.*

At this point, the system creates one instance for each identified noun. For example, for the noun phrase "*datore di lavoro*" (work supervisor), the instance will be represented by three abstract terms, as shown in Table 1. In the instance, the noun under evaluation is replaced by the generic term *target*, while all the other nouns are replaced with *noun*. It is important to note that only the term "datore" (i.e., "supervisor") is taken into account, since "di lavoro" (i.e., "of work") is one of its modifiers.

Table 1. The instance created for the noun "*datore*" is composed by three items (one for each syntactic dependency related to "*datore*"). Note that the considered noun "*datore*" is replaced by the generic term "**target**", while the other nouns are replaced with "*noun*".

Dependency	Instance Item
ARG(datore, il)	ARG-**target**-il
SUBJ(dovere, datore)	SUBJ-dovere-**target**
RMOD(datore, lavoro)	RMOD-**target**-noun

3.2 Learning Phase

Given a legal text T, the system produces as many input instances as the number of nouns contained in T. In particular, for each noun n in T, and for each semantic tag *sem*, we produce an instance T^n_{sem} associated with the label *positive* if n has been annotated with *sem* in the training corpus (*negative*, otherwise). At the end

of this process, all the instances are transformed into numeric vectors according to the Vector Space Model [20], and they are finally used as input training set for a Support Vector Machine classifier [11]. This is done for all the semantic information that we tested; this means that we build three classifiers, one for each semantic tag. Once the classifiers are built, we can classify all the nouns of a text as belonging to one of the three semantic labels (or none of them) by passing their syntactic dependencies to such classifiers. According to the example described in the previous section, for all its 17 nouns contained in the text, the system builds 17 textual instances, as partially shown in Table 2.

Table 2. Some of the instances created for the sentence of the example (one for each noun). Note that the nouns "*datore*" and "*manutenzione*" are labeled as *active role* and *involved object* respectively, as in the annotated dataset. The noun "*dispositivo*" is a negative example for all of the three semantic tags.

noun	instance	active role	passive role	inv. object
datore	arg-target-il subj-dovere-target rmod-target-noun	+	-	-
manutenzione	arg-target-mediante arg-target-il rmod-assicurare-target arg-target-noun arg-target-noun	-	-	+
dispositivo	arg-target-il obj-mantenere-target rmod-target-noun	-	-	-
...	...	...	...	...

4 Evaluation

4.1 Data

The data used for evaluating our approach contain 156 legal texts annotated with various semantic information, with a total of 2253 nouns. In particular, the data include an extended structure for prescriptions, which has been described in [6] as individual legal obligations derived from legislation. For our experiments we initially put the focus on three types of semantic tags:

Active Role. The active role indicates an active agent involved within the situation described in the text. Examples of common entites related to active roles are directors of banks, doctors, security managers.

Passive Role. The passive role indicates an agent that is the beneficiary of the described norm. Examples of agents associated with passive roles are workers and work supervisors.

Involved Object. An involved object represents an entity that is central for the situation being described. Examples are types of risk for a worker, the location of a specific work, and so on.

In the corpus there are 188 annotated active roles, 41 passive roles, and 270 involved objects out of a total of 2253 nouns (about the 22%).

4.2 Algorithms and Tools

For the implementation of our approach we used a Support Vector Machine (SVM) binary classifier, since it usually achieves state-of-the-art accuracy levels [11,17] with textual data. This algorithm makes use of vectorial representations of text documents [20] and works by calculating the hyperplane having the maximun distance with respect to the nearest data examples. More in detail, we used the Sequential Minimal Optimization algorithm (SMO) [19] with a polynomial kernel. The system relies on an external application based on the WEKA toolkit [14], a framework that supports several algorithms and validation schemes allowing an efficient and centralized way to conduct experiments and evaluate the results.

4.3 Results

In this section we present the evaluation of our technique on the task of automatically associate semantic labels to the nouns contained in the corpus. Notice that our approach is susceptible from the errors given by the POS-tagger and the syntactic parser. In fact, in case the POS-tagger does not recognize that a target term is actually a noun, then the latter is not considered as a possible item that deserves a semantic label. The same thing happens for the parsing procedure, that can produce errors that can avoid the correct classification of a target noun.

The result of our evaluation is threefold: first, we evaluate the ability of the proposed approach to identify and annotate active roles; then we focus on the passive roles; finally, we face the more challenging recognition of involved objects, given their high level of semantic abstraction. Table 3 shows the accuracy levels reached by the approach using the 10-folds cross validation scheme.

As can be noticed, the approach works almost perfectly with the *active role* semantic tag. This means that the syntactic context of the active roles are well circumscribed, thus it is easy for the classifier to build the model. Regarding the *passive role* tag, even if the approach is precise when identifying the right semantic label (100% of Precision), it returns many false negative (27% of Recall). In a semi-supervised context of an ontology learning process, this can be anyway a good support, since all of what has been automatically identified is likely to be correct. Finally, the *involved object* semantic tag gave quite low results in terms of Precision and Recall. On average, only six to ten nouns classified as involved objects were actually annotated with the right semantic label. This is due to the very wide semantic coverage of this specific tag, and its consequently broad syntactic context.

Table 3. Precision, Recall and F-Measure values for the identification of active roles, passive roles, and involved objects, using 10-folds cross validation

Active Role	*Precision*	*Recall*	*F-Measure*
yes	97.2%	92.6%	94.8%
no	99.3%	99.8%	99.5%
Passive Role	*Precision*	*Recall*	*F-Measure*
yes	100.0%	26.8%	42.3%
no	98.7%	100.0%	99.3%
Involved Object	*Precision*	*Recall*	*F-Measure*
yes	59.3%	31.9%	41.4%
no	91.3%	97.0%	94.1%

5 Conclusion and Future Work

We presented an approach to automatically annotate legislative text with semantic information like active and passive roles, and involved objects. In particular, we used local syntactic information that are transformed into textual representation to work with a standard Support Vector Machine classifier. The aim of this work was to revisit these tasks as classical supervised learning problems that usually lead to high accuracy levels with high performance when faced with standard Machine Learning techniques. Our first results on this method highlight the validity of the approach with two out of the three tested semantic tags.

References

1. Auger, A., Barriere, C.: Pattern-based approaches to semantic relation extraction: A state-of-the-art. Terminology 14(1), 1–19 (2008)
2. Berland, M., Charniak, E.: Finding parts in very large corpora. In: Annual Meeting Association for Computational Linguistics, vol. 37, pp. 57–64. Association for Computational Linguistics (1999)
3. Biemann, C.: Ontology learning from text: A survey of methods. In: LDV Forum, vol. 20, pp. 75–93 (2005)
4. Boella, G., di Caro, L., Humphreys, L., Robaldo, L.: Using legal ontology to improve classification in the eunomos legal document and knowledge management system. In: Semantic Processing of Legal Texts Workshop (SPLeT 2012) at LREC 2012 (2012)
5. Boella, G., Humphreys, L., Martin, M., Rossi, P., van der Torre, L.: Eunomos, a legal document and knowledge management system to build legal services. In: Palmirani, M., Pagallo, U., Casanovas, P., Sartor, G. (eds.) AICOL Workshops 2011. LNCS (LNAI), vol. 7639, pp. 131–146. Springer, Heidelberg (2012)
6. Boella, G., Martin, M., Rossi, P., van der Torre, L., Violato, A.: Eunomos, a legal document and knowledge management system for regulatory compliance. In: Proceedings of Information Systems: A Crossroads for Organization, Management, Accounting and Engineering (ITAIS) Conference. Springer, Berlin (2012)

7. Boella, G., di Caro, L., Humphreys, L., Robaldo, L., van der Torre, L.: Nlp challenges for eunomos, a tool to build and manage legal knowledge. In: Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC) (2012)
8. Boella, G., Di Caro, L., Humphreys, L.: Using classification to support legal knowledge engineers in the eunomos legal document management system. In: Fifth International Workshop on Juris-informatics (JURISIN) (2011)
9. Buitelaar, P., Cimiano, P., Magnini, B.: Ontology learning from text: An overview. In: Ontology Learning from Text: Methods, Evaluation and Applications, vol. 123, pp. 3–12 (2005)
10. Candan, K., Di Caro, L., Sapino, M.: Creating tag hierarchies for effective navigation in social media. In: Proceedings of the 2008 ACM Workshop on Search in Social Media, pp. 75–82. ACM (2008)
11. Cortes, C., Vapnik, V.: Support-vector networks. Machine Learning 20(3), 273–297 (1995)
12. Fortuna, B., Mladenič, D., Grobelnik, M.: Semi-automatic construction of topic ontologies. In: Ackermann, M., et al. (eds.) EWMF/KDO 2005. LNCS (LNAI), vol. 4289, pp. 121–131. Springer, Heidelberg (2006)
13. Grabmair, M., Ashley, K.D., Hwa, R., Sweeney, P.M.: Toward extracting information from public health statutes using text classification machine learning. In: JURIX, pp. 73–82 (2011)
14. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The weka data mining software: an update. SIGKDD Explor. Newsl. 11, 10–18 (2009), http://doi.acm.org/10.1145/1656274.1656278
15. Harris, Z.: Distributional structure. Word 10(23), 146–162 (1954)
16. Hearst, M.: Automatic acquisition of hyponyms from large text corpora. In: Proceedings of the 14th Conference on Computational linguistics, vol. 2, pp. 539–545. Association for Computational Linguistics (1992)
17. Joachims, T.: Text categorization with support vector machines: Learning with many relevant features. In: Nédellec, C., Rouveirol, C. (eds.) ECML 1998. LNCS, vol. 1398, pp. 137–142. Springer, Heidelberg (1998)
18. Lesmo, L.: The Turin University Parser at Evalita 2009. Proceedings of EVALITA 9 (2009)
19. Platt, J., et al.: Sequential minimal optimization: A fast algorithm for training support vector machines. In: Advances in Kernel Methods-Support Vector Learning, vol. 208, pp. 98–112 (1999)
20. Salton, G., Wong, A., Yang, C.S.: A vector space model for automatic indexing. Commun. ACM 18(11), 613–620 (1975), http://doi.acm.org/10.1145/361219.361220
21. Srinivasan, P., Rindflesch, T.: Exploring text mining from medline. In: Proceedings of the AMIA Symposium, p. 722. American Medical Informatics Association (2002)
22. Yang, H., Callan, J.: Ontology generation for large email collections. In: Proceedings of the 2008 International Conference on Digital Government Research, pp. 254–261. Digital Government Society of North America (2008)

Interpreting Spatiotemporal Expressions from English to Fuzzy Logic

William R. Murray, Philip Harrison, and Tomas Singliar

Boeing Research and Technology, PO Box 3707, Seattle, WA 98124, USA
{william.r.murray,philip.harrison,tomas.singliar}@boeing.com

Abstract. We discuss extensions to a controlled natural language allowing spatiotemporal expressions to be interpreted as fuzzy logic functions. The extensions first required new sentence templates. Next, changes to a GPSG parser modified its lexicon, and then extended its parsing and logical form rules to allow user-defined spatial and temporal constraints to be extracted. The sentence templates ground user-defined culturally-specific times and places to boundaries surrounding prototypical ideals. Query points, defined by location and time, are compared to these definitions using Gaussians centered at prototypical 'ideal' times or places. The Gaussians provide soft fall-off at the boundaries. Fuzzy logic operators allow larger expressions to be interpreted, analogous to Boolean combinations of terms.

The mathematically-interpreted spatiotemporal terms act as domain features for a machine learning algorithm. They allow easy specification (compared to programming) of basis functions for an inverse reinforcement learning algorithm that detects anomalous vehicle tracks or suspicious agent behavior.

Keywords: controlled natural language, CPL, spatiotemporal reasoning, GPSG parsing, inverse reinforcement learning, fuzzy logic, anomaly detection.

1 Introduction

In this paper, we discuss extensions to a controlled natural language to allow spatiotemporal expressions to be interpreted using fuzzy logic. Semantically, we interpret spatial and temporal expressions with fuzzy logic or Gaussian expressions. Fuzzy logic is used for noun-noun expressions (e.g., “winter weekend”) and coordinate expressions (e.g., “neither in the market place nor the city center”), while Gaussian expressions are used to provide 0 to 1 membership values for user-defined temporal concepts (e.g., “Ramadan”) and spatial concepts (e.g., “Kirkuk airport”).

In [1], we introduced new sentence templates for the controlled natural language CPL to allow user definitions of culturally-specific temporal and spatial names as background knowledge. First, we allow users to define new spatial or temporal concepts, grounding them with UTC times or geographic coordinates. Here are four examples:

L. Morgenstern et al. (Eds.): RuleML 2013, LNCS 8035, pp. 226–233, 2013.

Examples of Spatial Concept Definitions:

a. Seattle is a city at 47°36′35″N 122°19′59″W.
b. Alamo Square is a park at 37.776384°N 122.434709°W with radius 0.6 km.

Examples of Temporal Concept Definitions:

c. Ramadan starts 20 July 2012 and ends 18 August 2012.
d. Lunch is from 11:00 to 13:00.

Each definition defines a most prototypical point for the center of a time interval or a radial spatial region. A Gaussian is centered at that point, clamped at 1.0 between the boundaries, and then falls off smoothly beyond the boundaries. The soft fall-off provides a means of expressing the vagueness of a temporal concept, such as "lunch," or a geographic concept, such as "town square." In Fig. 1, for example, lunch is modeled as a time that is most prototypically centered at noontime, includes times from 11 AM to 1 PM, and then falls off outside those boundaries.

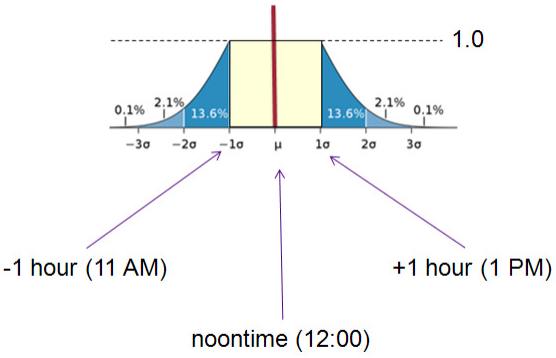


Fig. 1. A membership function approximating the vague temporal concept "lunch." The most prototypical time is μ=12 and the boundaries are ±σ, with σ=1. There is a rapid, but smooth fall-off for values of time outside of 11 AM and 1 PM. In essence, we have superimposed a Gaussian, scaled it to have value 1 at μ ±σ, and clamped its value to 1.0 from μ-σ to μ+σ.

User-defined concepts, grounded in this way, can then be compared to query "track points," defined as events in time and space. Each track point corresponds to an observation of a vehicle traveling with a given UTC time and geographic coordinates. Vague spatial expressions, such as "near a bridge," and temporal expressions, such as "work hours," can be composed as expressions that model aspects of the flow of normal traffic. For example, we would consider traffic to be lower "after work hours in a city" and higher "in the afternoon near a bridge," or "near the market place."

1.1 Motivation

We need to represent this imprecision, as our machine learning algorithm needs to know how "close" or "near" a point in time and space is to one of the concepts (e.g., "town market," a place; and "market day," a time). The machine learning algorithm feature values should only vary from 0.0 to 1.0 and provide a soft fall-off as a track point moves away from satisfying a feature. Essentially, the degree of user concept proximity is interpreted as a 0.0 to 1.0 fuzzy set membership [2].

The motivation is for ISR (intelligence, surveillance, and reconnaissance) detection of anomalous vehicle tracks, currently, or suspicious agent behavior, in the future. Each track comprises a large series of track points for a vehicle. Each track point has a time stamp, expressed as a UTC time, and location, expressed as latitude and longitude. A suspicious track may indicate individuals taking circuitous routes to evade checkpoints, or following convoy routes for reconnaissance. Normal civilian traffic, in aggregate, would be expected to take the most convenient routes. Similarly, mercantile ship tracks can be analyzed to monitor possible smuggling. For more details, see [3].

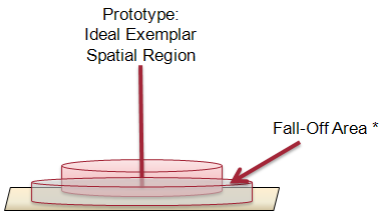


Fig. 2. Similarly, we "soften" our binary predicate for spatial locations to have a soft fall-off. Now, rather than having a predicate such as **mosque**(lat,long) that returns True or False, we have a membership function that returns 0.0 (far from the mosque) to 1.0 (inside the mosque, at its center), with a fall-off (e.g., 0.8 could be outside the mosque but close to it).

The machine learning algorithm is based on inverse reinforcement learning [4]. It learns normal aggregate behavior by inferring a utility function over many vehicle tracks. More accurate domain modeling leads to more accurate detection of anomalous vehicle tracks and fewer false alarms. The traffic analysis depends crucially on spatial features (e.g., checkpoints, bridges, markets, border crossings, etc.) and temporal features (e.g., religious holidays, work hours, special events, etc.). Consequently, users, such as intelligence analysts, can more easily use the application by expressing relevant spatiotemporal features in controlled English. The alternative, and prior approach, was to require coding for each change in features. This approach is unsuitable for subject matter experts unfamiliar with programming or harried by real-time exigencies.

1.2 Culturally Dependent Concepts

Ideas of time and space are relative to a culture [5-6]. If we wish to model traffic flow in Spain, compared to the U.S. or Iraq, we need to take into account cultural definitions of work times, meal times, and holidays, as we expect increased or decreased traffic at these times. We also need to model important places in a town or city that affect traffic, e.g., checkpoints, bridges, construction, traffic barriers, markets, etc.

Most culturally defined spatiotemporal concepts are not sharply demarcated. For example, is 1:40 PM still "lunch time"? Where, exactly, are the city center boundaries? Instead, human conceptual representation relies on prototypes (also called schemas, cognitive frames, and idealized cognitive models) characterized by family resemblances, and examples that are typical, ideal, and most memorable [5].

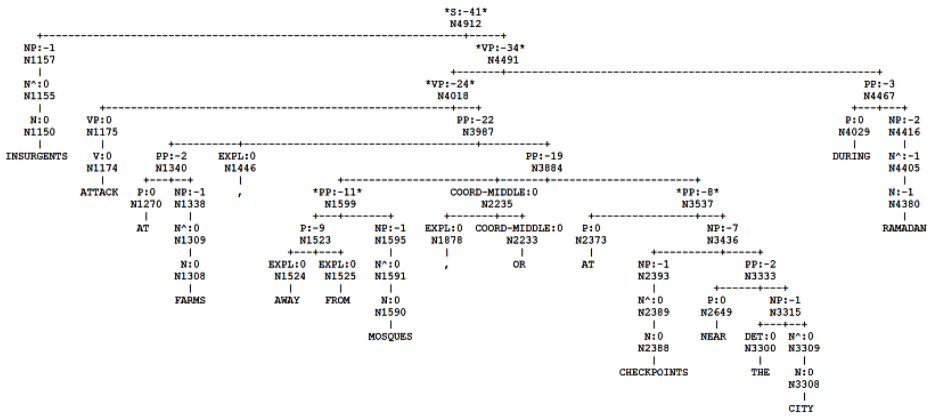


Fig. 3. Parse tree for "Insurgents attack at farms, away from mosques, or at checkpoints near the city during Ramadan"

We model places (spatial regions) as having a central point (the most prototypical point, e.g., 'downtown' for a city, or a town square off the high street for a village) with a circle around it, as shown in Fig. 2. This modeling is very poor for large, irregularly shaped regions, such as city boundaries, compared to polygonal boundary representations, but is adequate for modeling proximity to urban-scale structures, such as mosques or gas stations that we envision as most typical of our traffic modeling application. A more precise definition of spatial concepts would be obtained from geographic information systems (GIS), but a default definition independent of such data sources is both necessary for graceful performance degradation, and convenient for conceptual development and experimentation.

The radius of a circle modeling a spatial region models the size of the physical location, encompassing the bulk of its area, and intuitively how "close" or "near" a track point may be to the place. Thus, a car may be "close" to a major bridge if it is 1 km from it, but not "close" to a café until it is within the same block. In both examples, the spatial descriptor is relative to the size of the object being discussed. We use a

similar intuition here, and allow categories of places (e.g., churches) to have a default radius that we override for exceptionally large or small instances.

2 Extending the Parser

CPL's GPSG parser, called SAPIR [7], was extended to better account for sentences with spatiotemporal content, such as the following:

Examples of Spatiotemporal Domain-Modeling Sentences:

a. Insurgents attack in the day, in Kirkuk, but not in the market place.
b. Insurgents attack farms close to the city.
c. Civilians drive away from checkpoints that are outside the city.
d. The Sunnis watch the Shiites of Karbala during Ashura.
e. Insurgents attack at farms, away from mosques, or at checkpoints near the city during Ramadan

We will use the last sentence as an example. Its parse tree is shown in Fig. 3. Note that the prepositional phrase "near the city" modifies the noun "checkpoints," but the prepositional phrase "during Ramadan" and the entire disjunction "at farms, away from mosques, or at checkpoints near the city" modifies the main verb, "attack."

The parse tree can be traversed to extract a logical form, and from that just the spatial and temporal constraints that refer to user-defined concepts (e.g., 'night,' 'market place,' Kirkuk, or Ramadan), or absolute (e.g., dates, times, and coordinates) temporal and spatial entities. This form, which we call the *spatiotemporal constraint form,* is shown below:

```
(AND (OR (:SPATIAL-CONSTRAINTS ("at" NIL "farm"))
         (:SPATIAL-CONSTRAINTS ("away from" NIL "mosque"))
         (AND (:SPATIAL-CONSTRAINTS ("at" NIL "checkpoint"))
              (:SPATIAL-CONSTRAINTS ("near" "the" "city"))))
     (:TEMPORAL-CONSTRAINTS ("during" NIL "Ramadan")))
```

The spatiotemporal constraint form specifies the following:

1. User-defined spatial and temporal concepts to be compared to the track point;
2. Means, using fuzzy logic operators, to compute overall degree of membership;
3. Prepositions, that may tighten the σ used to compare track points to user concepts (e.g., "inside" adjusts σ to be less than "near").

Each spatial and temporal primitive is interpreted as a lambda function. The function is applied to track point location and UTC time parameters and compared to the μ and $\pm\sigma$ boundaries defined with the user concept definition and corresponding Gaussian.

The term values for user-defined concepts (e.g., "Ramadan") are combined with fuzzy logic operators analogous to AND, OR, and NOT Boolean operators. We use MIN (or multiplication) for AND, MAX for OR, and $\lambda x.\ (1.0 - x)$ for NOT.

Various extensions to the parser were required to enable the translations:

1. Existing words, or word combinations (multiwords), had to be marked as prepositions to be recognized as such. Although the lexicon included many prepositions, the following were added for convenience:
    a. Stative verbs that can be interpreted as prepositions: 'containing,' 'enclosing,' 'overlapping,' 'bordering,' 'touching,' 'following,' and 'preceding.'
    b. Adjectives that can modify prepositions: by adding these phrases 'near by,' 'near to,' 'opposite from,' 'opposite to,' 'away from,' 'far from,' 'close by,' 'right by,' and 'right at.'
2. New word features **spatial** and **temporal** were added. They mark user-defined concepts for special treatment in SAPIR's grammatical rules for attachment, and in the generation of the logical and spatiotemporal constraint forms.
3. Verb and preposition attachment rules were modified so that prepositions:
    a. With words marked with the **temporal** feature preferentially attach to verbs.
    b. With words marked with the **spatial** feature also preferentially attach to verbs, *unless*:
        i. They follow a common noun marked with the **spatial** feature (e.g., '…at checkpoints near the city').
        ii. They follow the prepositions 'of' or 'from' (e.g., '…the mayor of Karbala').

Where neither of these new rules apply, the current parser heuristics apply as before. Some of these attachment heuristics are domain-specific and were originally tuned for physics, chemistry, and biology as used in the HALO project [8].

We describe a work in progress and expect these rules, the logical form, and the spatiotemporal constraint form to evolve over time. For example, other than the attachment rule above, there is currently no interpretation or use of the preposition 'from' in a spatial context, as that requires determining the initial origin of a vehicle for a track point. We also anticipate extending the domain modeling, e.g., we may model different kinds of events (travelling, stopping), different kinds of track vehicles (e.g., truck versus car), and eventually add track vehicle origins or destinations. Finally, we may consider anomaly detection in other domains where events and tracks take on new meanings (e.g., in cyberwarfare, a track may be a sequence of user actions such as file accesses).

3 Related Work

Most controlled natural languages (e.g., ACE [9], PENG [10], Rabbit [11], CELT [12], and our earlier versions of CPL [13]) have made little special provision for time and space, other than distinguishing between times and places for answering *when* or *where* questions, or relating named times and places to hypernyms. For example, CPL treats WordNet as an ontology, and WordNet [14] describes Baghdad as an instance

of city and Ramadan as an Islamic calendar month (a hyponym of time_period#n1). In addition, PENG and CPL have used versions of the situation or event calculus for a limited handling of time. Other than our work in [1], none of these provided fuzzy logic interpretations for spatial and temporal prepositions or concepts.

In contrast to controlled natural languages, TimeML [15] and SpaceML [16] provide markup languages for free-text natural language. TimeML marks events and words signaling temporal orderings in Allen's interval algebra [17]. Additionally, it marks aspectual and subordination links between events, states, and explicitly provided dates and times (also expressed as partially specified UTC, as in our work). SpaceML similarly provides a qualitative representation of spatial regions, in this case suitable for RCC-8 [18], but also tied into quantitative values.

In both cases these markup languages differ from the current work by focusing on qualitative relational reasoning, i.e., how one time or place is qualitatively related to another, although both also provide tie-ins to absolute values in time (UTC) or place (geographic coordinates). The current work, in contrast, provides a different kind of reasoning: rather than trying to determine the ordering in time and space of two events, we provide a measure of how close one event is to a target category. The target category is not a single event, but a composite description of multiple categories, each of which is a temporal or spatial concept with vague boundaries. Furthermore, users may define new spatiotemporal concepts grounded appropriately to their culture and situation, a capability not supported by the markup languages.

4 Summary

We describe work in progress to handle vague spatial and temporal terms in English, that are culturally and situationally specific, and that are interpreted in terms of fuzzy logic. We have extended the SAPIR parser for the CPL controlled natural language to allow interpretation of sentences incorporating spatiotemporal constraints. A spatiotemporal constraint form is extracted from the logical form of a sentence parse. That constraint form, in turn, is interpreted in terms of fuzzy logic operators applied to more basic set membership evaluations. These more basic concept definitions define measures of set membership (e.g., "To what degree does this track point belong to the class of points "near to the market place" or "during Ramadan"?). They are interpreted as scaled and clamped Gaussians, based on user-defined boundaries and prototypical central points. The interpretation from English to fuzzy logic facilitates the application of machine learning algorithms for anomaly detection, by allowing non-programmer subject matter experts to more rapidly model new or changing domains.

The research contribution is to extend controlled natural language into the realm of vague and approximate natural language expressions, and to provide one means of translating the natural language into formal models amenable to machine reasoning. In this case we use fuzzy logic to handle vague spatiotemporal terms and prepositions, and Gaussians around prototypical central points to formalize the natural language descriptions.

References

1. Murray, W.R., Singliar, T.: Spatiotemporal Extensions to a Controlled Natural Language. In: Kuhn, T., Fuchs, N.E. (eds.) CNL 2012. LNCS, vol. 7427, pp. 61–78. Springer, Heidelberg (2012)
2. Zadeh, L.A.: Fuzzy logic= computing with words. IEEE Transactions on Fuzzy Systems (1996)
3. Singliar, T., Marginenantu, D.: Scaling up Inverse Reinforcement Learning through Instructed Feature Construction. In: Snowbird Learning Workshop 2011 (2011), http://snowbird.djvuzone.org/2011/abstracts/132.pdf
4. Ng, A., Russell, S.: Algorithms for inverse reinforcement learning. In: Proceedings of ICML (2000)
5. Lakoff, G.: Women, Fire, and Dangerous Things. University of Chicago Press (1990)
6. Kövecses, Z.: Metaphor in Culture: Universality and Variation. Cambridge University Press (2005)
7. Harrison, P.: A New Algorithm for Parsing Generalized Phrase Structure Grammar, Ph.D. dissertation, University of Washington, Seattle (1988)
8. Barker, K., Porter, B., Clark, P.: A Library of Generic Concepts for Composing Knowledge Bases. In: Proc. 1st Int Conf. on Knowledge Capture, K-Cap 2001 (2001)
9. Fuchs, N.E., Schwertel, U., Schwitter, R.: Attempto Controlled English – Not Just Another Logic Specification Language. In: Flener, P. (ed.) LOPSTR 1998. LNCS, vol. 1559, pp. 1–20. Springer, Heidelberg (1999)
10. Schwitter, R., Tilbrook, M.: PENG: Processable ENGlish. Technical report, Macquarie University, Australia (2004)
11. Engelbrecht, P., Hart, G., Dolbear, C.: Talking Rabbit: A User Evaluation of Sentence Production. In: Fuchs, N.E. (ed.) CNL 2009. LNCS (LNAI), vol. 5972, pp. 56–64. Springer, Heidelberg (2010)
12. Pease, A., Murray, W.R.: An English to Logic Translator for Ontology-Based Knowledge Representation Languages. In: Proceedings of 2003 IEEE International Conference on Natural Language Processing and Knowledge Engineering, NLPKE 2003 (2003)
13. Clark, P., Murray, W.R., Harrison, P., Thompson, J.: Naturalness vs. Predictability: A Key Debate in Controlled Languages. In: Fuchs, N.E. (ed.) CNL 2009. LNCS (LNAI), vol. 5972, pp. 65–81. Springer, Heidelberg (2010)
14. Fellbaum, C.: WordNet: An Electronic Lexical Database (Language, Speech, and Communication), Hardcover, May 15 (1998)
15. Pustejovsky, J., Castaño, J., Ingria, R., Saurí, R., Gaizauskas, R., Setzer, A., Katz, G.: TimeML: Robust Specification of Event and Temporal Expressions in Text. In: Fifth International Workshop on Computational Semantics, IWCS-5 (2003)
16. Cristani, M., Cohn, A.G.: Mark-up Language for Spatial Knowledge. Journal of Visual Languages and Computing 13, 97–116 (2002)
17. Allen, J.F.: Towards a general model of action and time. Artificial Intelligence 23(2) (1984)
18. Cohn, A.G., Bennett, B., Gooday, J., Gotts, N.M.: Qualitative spatial representation and reasoning with the region connection calculus. Geoinformatica (1997)

Combining Acquisition and Debugging of Business Rule Models*

Adeline Nazarenko and François Lévy

LIPN, Université Paris 13 – Sorbonne Paris Cité & CNRS (UMR 7030), France
firstname.lastname@lipn.univ-paris13.fr

Abstract. Business rules (BR) can be acquired from complex texts such as laws, regulations or contracts. However knowledge extraction and formalization is a complex task that involves business experts as well as Information Technology engineers and that is error-prone. Instead of waiting until the rule base is completed or the BR decision system is put into production to detect problems, we propose to detect inconsistencies and errors at an early stage, before the formalization work is completed. This paper presents the quality procedures that can be implemented in the process of BR acquisition from NL regulations. We show that the documented business rule models under construction are useful to detect potential anomalies at a semi-formal level of the BR base, where the rules exploit a formal vocabulary but are simply structured into premises and conclusions. Even at the prior and textual level, these documented models give the business experts a global and structured view over the NL regulation, which helps the formalization process.

1 Introduction

Business rule acquisition is acknowledged as a serious bottleneck for the development of Business Rules Management Systems (BRMS). In many industrial cases, documents are used to produce structured information about the business and the resulting business models are exploited in BRMS to assist decision making. Actually, since source documents, such as regulatory texts, often drive and constrain business processes, their content must be caught in the operational business models on which BRMS rely on.

Acquiring Business Rules (BR) models from text cannot be automated, however. We argued that this process is a complex one, which must be decomposed into several phases (elicitation, normalization and formalization) [11], possibly handled by different people, but nothing warranties that the resulting BR base is error-prone. Many problems can occur. The original text may not be perfect: there may be implicit and thus missing information, ambiguous phrases and more or less apparent contradictions. Errors can also arise in the acquisition

* This work has been initiated in the context of the FP7 231875 ONTORULE project (http://ontorule-project.eu). We thank to our partners for the fruitful discussions.

L. Morgenstern et al. (Eds.): RuleML 2013, LNCS 8035, pp. 234–248, 2013.

process itself, because it is difficult to have a global view of the source text and the rule base under construction.

In most cases, however, you must wait until the rule base is completed or the BRMS is put into production to detect problems. In software development, it is well known that poor quality of development leads to greater efforts in detecting bugs and non-compliance. According to experts[1], repairing a bug costs 1 at specification time, 20 at development time and 150 at operation time. The same holds for the BR development. It is much better to detect inconsistencies and errors before the formalization work is completed. This paper presents the quality procedures that can be implemented in the process of BR acquisition from NL regulations.

After a review of related works (Sec. 2), this paper presents the documented BR model, the structure on which our debugging approach relies (Sec. 3). This approach is based on some *a priori* knowledge of inconsistency (Sec. 4) and it operates both at the semi-formal and at the textual levels (Sec. 5 and 6) of the BR base. The whole approach is mainly illustrated on the analysis of a vehicle safety-belt testing regulation edited by the UNO (regulation number 16). Car manufacturers have to conform to this text.

2 Related Works

The acquisition of business rules from texts raises several important challenges: the formalization of the knowledge that is extracted from texts, the control of the quality of resulting knowledge base and the traceability of the formal knowledge. These challenges have been addressed in the state of the art of the BR or, more generally, of the semantic web community.

2.1 Knowledge Formalization

Text-based acquisition naturally addresses the problem of informal to formal transformation of knowledge. Texts encodes pieces of informal language but BRMS requires that knowledge be formalized and support operation. Text and operational rule base are located at the two extremes of the formalization continuum [3] and the main question deals with the translation of textual information into formal knowledge. The problem has been addressed in different ways.

Controlled languages have deserved a lot of attention, in the academic field (Attempto Controlled English (ACE) [7]) in the industrial domain (RuleSpeaks and Oracle Policy Automation (OPA)) or at the OMG standardization level (Semantics of Business Vocabulary and Business Rules (SBVR)[2]). Most works have focused on the controlled to formal language translation (see for instance, the partial translation recently proposed by [18] on the SBVR side). Less effort has been devoted to the natural to controlled language transformation. [6] and [1] give only partial solutions, covering only the syntactic transformations or simple sentences.

[1] Barry Boehms, in his keynote to Equity 2007. The figures can be found in `http://fr.scribd.com/doc/20893084/Advanced-OOP-and-Design-Patterns`, slide 27.

[2] `http://www.omg.org/spec/SBVR/1.0/`

Exploiting NL regulations and documents is nevertheless an important issue for the development of BRMS, whenever regulations are written in policy documents. We proposed [10,13] an interactive approach for transforming NL sentences that have been extracted from source texts into formal rules. Since this transformation cannot be realized all at once or by a single expert, we proposed to decompose the formalization process into two main phases: the knowledge elicitation and normalization on the one hand, and its formalization on the other hand. This departs the roles of the business expert in charge of the elicitation, from that of the IT specialist who takes the normalized and semi-formal form rather than the source text as input and formalizes it, taking the final application and a specific target rule engine into account.

2.2 Rule Base Maintenance

A second important issue is related to the quality of the rule bases. It concerns the correctness of the rules as well as their maintenance when the source documentation or the applicative goals evolve. The logical works focus on the first point. To the best of our knowledge, systematic testing methods have not been developed, due to the difficulty of defining a significant subset of the input space. Discovering errors in the rules rather relies on a typology of errors and problems that frequently occur in rule bases.

Different kinds of anomalies are known and described by engineers maintaining rule systems. In details, these forms depend on the syntax of rules and the underlying reasoning mechanism. *Inconsistencies* are sets of rules which lead to contradictions, *i.e.* a and $\neg a$ for some a. When a complete inference mechanism is available, it allows to decide if a given set of rules is inconsistent. Most rule systems apply their rules to different input sets, and name *conflict* the case where an input set leads to a contradiction. Other anomalies manifest that something has been inadequate in the formalization process. They are classified by [16,17,2,5] in *redundancy* (a rule can be omitted without affecting the inferences), *circularity* (an inference depends on itself) and *deficiency* (some conclusions are lacking for a valid input set). In [16,17], a pure rule system is considered, while [2] and [5] deal with ontology and rules combination. The difference is not in the expressive power but in the interaction of two knowledge bases[3]. Note that the notion of valid input set is hard to define *a priori* and generally results from an expert judgment.

In the following, we show that some of these error types can be detected before formalization, thanks to the semantic annotation on which documented business models are based.

2.3 Semantic Annotation

Traceability has been considered more widely in the Semantic Web context. Semantic annotation has first been proposed as a way to enrich or enlighten texts

[3] Except for the ontology + Logic Programming combination, which mixes two different views of negation.

with some kind of "light interpretation" [15]. The texts are actually annotated with respect to a semantic model, which is often a formerly designed ontology [12]. The important point is that an annotated document supports semantic querying while traditional search mainly relies on plain text. A similar approach is actually embodied in semantic media wikis, where the semantic level is used to help text browsing or reading, and supports semantic functionalities [9,4]. This last approach is close to ours apart the fact that the semantic model supporting the text annotation is richer in BRMS.

We argue in the following that this semantic annotation allows to implement some quality procedures early in the design of BR: at the semi-formal level and even at the textual level.

3 Documented Business Rule Models

Our approach of business rule model acquisition strongly relies on the semantic annotation trend. The overall methodology is to provide an interactive environment that reduces the expert's effort, but remains under his/her control for the quality of the result. It has been described in [10,13,14] and is recalled here.

When they are developed, the rules are gradually linked to the source ontology and to the fragments of text that gave them birth. This defines an *indexing* network that supports traversals in and between the semantic model (ontology and rule base) under construction and the text. Queries on this complex graph help the expert collecting relevant fragments that would be otherwise uneasy to gather. We call the whole structure – document, semantic model and links – a Documented Business Rule Model (DBRM) (Figure 1).

The annotation scheme describes the data structure in which the source document, the underlying ontology and the business rules that derive from them

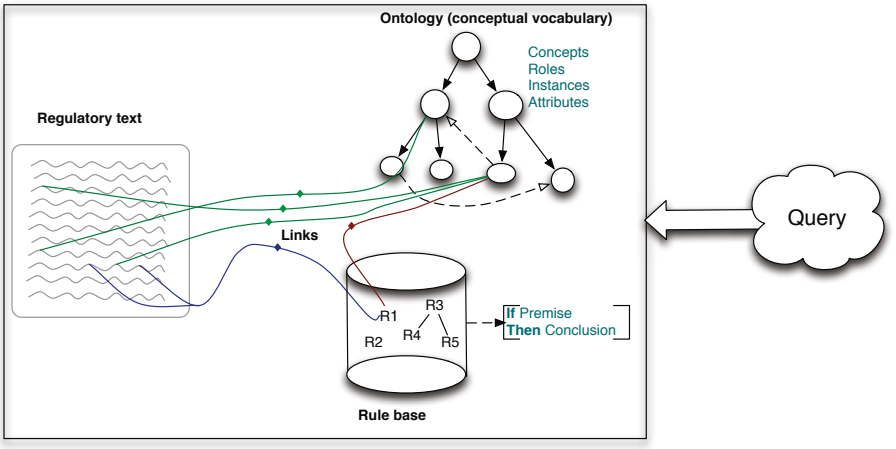


Fig. 1. Index structure with ontologies, rules and texts

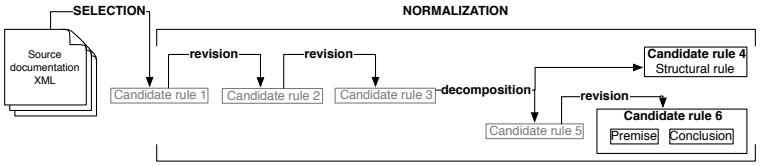


Fig. 2. Structure of the rule base. The selection, revision and decomposition operations are encoded as as many annotation relations between the source candidate rules and the resulting ones.

are encoded. The basic elements of this scheme are (1) the textual units that are either tokens or sequences of tokens (elementary character strings resulting from a segmentation process), (2) the ontological entities (concepts, instances, properties) that are used for annotating the text [14], and (3) the candidate rules that compose the rule base [8].

The rule base is actually composed of candidate rules that are partially ordered by a derivation relation (see Figure 2). The initial candidate rules are simply sentences that have been extracted from the acquisition text. They are stored as annotations of the source text. They can be progressively reformulated and decomposed, which gives rise to additional candidate rules that are stored in turn as annotations of the previous ones. At the beginning of this derivation process, the candidate rules are traditional NL sentences. At this *textual level*, a candidate rule is a text fragment as in

If a test has a duration less than 6 hours, the test is invalid

At some point, the business expert in charge of the rule elicitation can decompose them into a premise and a conclusion. At this *semi-formal level*, a candidate rule is a pair of text fragments, which are delimited by **IF** and **THEN** markers in the linearized form:

IF a test has a duration less than 6 hours, **THEN** the test is invalid

In addition to these high-level annotation relations, low-level ones mark the textual units that compose the sentences and the candidate rules: *full words* (which have a proper meanning) are linked to elements of the conceptual (also called lexical) vocabulary, *i.e.* the ontological elements (concepts, properties or individuals) that compose the domain ontology chosen to interpret the source document ; *grammatical words* are either absorbed in a higher level entity (*e.g. of* in the concept "attachment of the belt") or have a role of logical connectives, and are reduced to the keywords that serve as grammatical words in the chosen controlled language. The full annotation scheme is described in Fig. 3.

For sake of readability, these low-level annotations are encoded in an SBVR-style, with underlined and colored fonts: Concept, Relation, Individual, Attribute[4] and key word.

[4] Resp. class, object property, instance and data property in OWL.

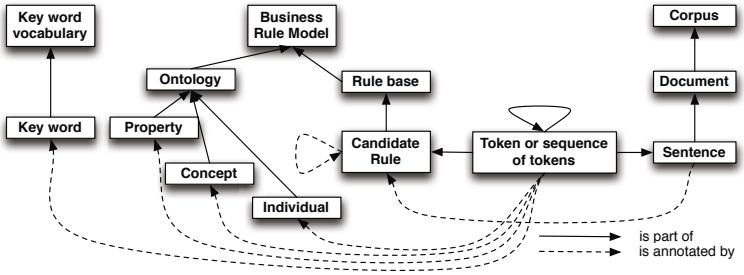


Fig. 3. Full annotation scheme

IF a test has a duration less than 6 hours, **THEN** the test is InvalidTest[5]

Roughly, lexical annotations are built from a lexicalized domain ontology[6] which either preexists or is built on a sample of the text. With the help of search algorithms, the expert marks some initial candidate rules. This parallels the Model Driven Architecture (MDA) in which an Object Model is generally built before the rule acquisition. In technical terms, the whole index data structure is encoded as an RDF graph. Annotation links are encoded in RDFa: the RDF annotations are anchored in textual units of XML documents and refer to resources that are OWL entities or candidate rules. The documents can be visualized with HTML browsers, where low-level annotations are usually represented in a SBVR-style. A SPARQL engine (CORESE) performs queries on the index structure, supporting also regular expression search on the text.

4 *A priori* Knowledge on Inconsistency Problems

Our approach to debugging is to be used as early as possible, during the elicitation of knowledge, because it is better to detect problems before working on the formalization of rules, especially as different persons may support elicitation and formalization. Of course, the detection of inconsistencies requires reasoning based on formal rules, but knowing in advance the type of problems that may arise and the traditional sources of errors helps detect potential problems in advance and define a prudent acquisition strategy.

Several types of formal problems have been listed in the state of the art (Section2.2):

[5] Here is a simplified underlying XML encoding of the same semi-formal rule: *<Premise>*If a *<Concept reference="onto:#Test">* test *</concept>* has a *<Attribute reference="onto#duration">* duration *</Attribute>* less than 6 hours *</Premise> <Conclusion>* the *<Concept reference="onto:#Test">* test *</Concept>* is *<Concept reference="onto:#InvalidTest">* invalid *</Concept></Conclusion>*

[6] An ontology where each ontological element has lexical information attached ; we use SKOS for attaching lexical information.

- *Contradictions* certainly reveal a modeling error, since a and $\texttt{not}\,a$ cannot both be the case. Either one or several rule literals are mistaken, or a condition is missing. Basic contradiction patterns involve one or two rules.
- *Redundancy* refers to heterogeneous causes: erroneous translations can lead to unsatisfiable rule conditions; a subcase may be considered and described apart from the general case. In this latter event, redundancy may have unexpected effects if the related rules are modified independently of each other.
- *Circularity* may be normal when some rules involve recursivity but it may also show that synonymies have not been detected, or that unclear and probably useless distinctions have been made in the model.
- *Deficiency* results from an incomplete modeling. It occurs occasionally at the formal level, when a rule is missing. It may also appear at the textual level, if some significant part of text remains unexploited.

The expertise in regulation and legal text analysis gives another source of information for understanding text-based acquisition errors. The main problems come from the fact that acquisition processes sequentially, one sentence or one paragraphs after the other, whereas understanding rule information would require to have a global view of the regulation, which is presented in one or several large documents:

- It is difficult for the analyst to focus on the precise semantics of a sentence so as to rewrite it as a simple and normalized statement and to take a large context into account. However, extracted from their context, some words cannot be interpreted (*e.g.* pronouns without their antecedent) or can be misunderstood ("a test" can actually refer to a very specific test if the title of the section is clear enough).
- It is also difficult to combine local analysis with long-distance dependencies. However, the relevant rule information can be spread into different sections of the acquisition document (for instance between the definition and the cases sections) and a general rule may not be explicitly related to all its relevant specific cases, which are described in a different part and are nevertheless exceptions to the general rule.

Based on this *a priori* knowledge, we propose in the following some usual patterns of errors that can be matched on the documented business rule model and more specifically on the base of textual or semi-formal BR, using SPARQL queries and more generally semantic technologies.

5 Debugging at the Semi-formal Level

At the semi-formal level, no formal reasoning can be performed but the syntactic patterns of errors listed by researchers for the systematic checking of anomalies at the formal level (see section 2.2) can be transposed in an approximate way to identify potential sources of errors.

At that level, one has to rely on the mentions of concepts, relations and attributes in the premise and conclusion parts of the rules, as well as on concept

disjointness and subsomption relations. We assume that the preexisting ontology and semantic annotation are error-prone and we focus on the detection of anomalies in the rule base under construction.

Most examples are taken from the vehicle safety belt testing domain and are based on a UNO regulation (# 16). This text describes the various tests that the car manufacturers must perform on safety belts to conform to international safety regulations. Since our data do not present all kinds of anomalies, some examples have been manually designed for illustration, but all these examples refer to the same basic knowledge: if the required conditions are not satisfied, the individual performance test is considered as invalid and cannot be carried out ; if the test is valid, it may be passed or failed according to the performance output. In the ontology, we therefore have two disjoint subconcepts or subclasses (ValidTest and InvalidTest) of the Test concept and two disjoint subconcepts (PassedTest and FailedTest) of ValidTest.

5.1 Models of Contradiction

As enumerating all the contradictory subsets of a logical set of formulas is not tractable, the formal checks only consider a limited list of simple cases.

Self-contradicting Rule. This involves a single rule in which the conclusion contradicts the premise or one of its conditions, as in:

IF a test is PassedTest and the test has a duration less than 6 hours
THEN the test is InvalidTest

Such a contradiction can be easily detected through the presence of disjoint concepts in the premise and conclusion of the rule. Of course, this is not a formal proof of contradiction but a useful hint for the expert in charge of the acquisition and debugging of the rule base.

Asymetric Rule Pair. This pattern involves two rules with inconsistent conclusions but with one premise subsuming the other.

IF the breaking strength test has a breaking strength load greater than 4 kilos
THEN the breaking strength test is passed
IF the breaking strength test has a breaking strength load greater than 4 kilos and the breaking position is less than 1cm distant from the attachment of the belt **THEN** the breaking strength test is invalid

At the semi-formal level of BR, one can detect that one premise in included in the other and that the conclusions contain disjoint concepts. In the above example, the two premises are similar but with one more condition in the second rule. This means that the second rule is dominated. To solve the contradiction, a new condition can be added to the first rule (*e.g.* "if the breaking position is at least 1cm distant from the attachment").

Ambivalent Rule Pair. This pattern involves two rules in which the premises are consistent but neither dominates the other and the conclusions are inconsistent.

IF the breaking strength test has a cold exposure phase the duration of which is shorter than 6 hours THEN the test is invalid
IF the breaking strength test has a breaking strength load greater than 4 kilos THEN the test is passed

The disjointness of the conclusions can be checked as in the previous pattern. The consistency of premises cannot be proven at the semi-formal level but one can at least check that neither includes the other. In the above example, the expert should ask "What if the breaking-strength test has a cold-exposure phase the duration of which is shorter than 6 hours and has a breaking-strength load greater than 4 kilos ?". A duration condition could be added in the second rule.

5.2 Models of Circularity

As the OWL ontology is preexisting, only rule circularities are considered here.

Circularity between the Rules and the Taxonomy. This anomaly occurs when the conclusion of a rule is also a sub-concept (or a sub-relation) of one of its conditions. For instance:

IF t is a Test and t has a cold exposure phase and t has a breaking strength load THEN t is a breaking strength test
A breaking strength test is a Test

In this case, the rule includes the definition of the sub-concept: BreakingStrength-Test should be defined as a subconcept of Test which has a ColdExposurePhase and a BreakingStrengthLoad. Checking this pattern only requires comparing for subsumption the ontological descriptions of the concepts (and relations) involved in the premise and in the conclusion.

Circular Rule Chains. This anomaly extends the previous one, by involving more than two rules. The circularity occurs if each rule conclusion is also present as a premise in the following rule in the chain. It some cases, this may reveal an underlying problem. It is very often due to an inadequate modeling, as several classes can be merged.

Once the rules marked, detecting the circular rule chains amounts to building a directed graph between the litterals (an edge from each premise of each rule to its conclusion), and then searching for cycles of this graph[7].

5.3 Models of Redundancy

Unsatisfiable Rule Condition. The pattern corresponds to a rule in which two premise conditions are inconsistent.

[7] This can be done with Tarjan algorithm in $\mathcal{O}(E+V)$, with E=nbr of edges, V=nbr of vertices.

IF the test is passed and the breaking strength load is more than 3 kilos and the test is invalid **THEN** the test is a breaking strength test

Of course, since the condition cannot be satisfied, the rule is useless. Such an anomaly shows a probable misunderstanding of the text, unless the text itself is ill formed. In the DBRM, premises can be searched for disjoint concepts. Searching for twice used attributes is also useful and allows the expert to either simplify the set of values, or detect conflicts.

Redundant Subconcept. Another anomaly occurs when the conclusion is subsumed (subconcept or subrelation) by the premise of the rule, as in

IF a breaking strength test has a cold exposure phase **THEN** it is a Test

The resulting rule is trivial, if one assumes that a BreakingStrengthTest is a Test. One must probably reconsider either the rule or the subsomption relation. In the example above, the misunderstanding could come from the fact that BreakingStrengthTest is not a subconcept of Test, but BreakingStrengthTestWithColdExposurePhase would be. Comparing the entities of the premise and conclusion helps checking this type of anomalies.

Redundant Premises. An inappropriate translation from the NL text can yield a rule with one premise condition implying the other. In the following example, the second form of the rule is derived from the first by resolving the reference of *it*:

IF a Test is a breaking strength test **THEN** it has a cold exposure phase
IF t is a Test and t is a breaking strength test **THEN** t has a cold exposure phase

Unfortunately, this pattern is complex to check on a large text, for computational reasons.

Subsumed Rule. A rule subsumes another one if their conclusions are identical and one premise is subsumed by the other.

R_1: **IF** t is a Test and t has a BreakingStrengthLoad **THEN** t is a breaking strength test
R_2: **IF** t is a Test and t has a cold exposure phase and t has a BreakingStrengthLoad **THEN** t is a breaking strength test

This configuration may show an erroneous interpretation of one of the rules. As stated in section 4, it can also results from an unseen relation between a particular case and a general one.

5.4 Models of Deficiency

Incomplete Attribute Range in the Premise. Deficiency is not easy to characterize in a general way. The only formal hint is given by the attributes

and more specifically by consideration of their range. Bellow is an example from the steel industry domain. The severity of steel coils defects is defined by the following rules:

> **IF** the surface of defect on a coil is less than 7 mm^2 **THEN** the defect has severity low.
> **IF** the surface of defect on a coil is more than 12mm^2 **THEN** the defect has severity medium.

As severity is undefined for surfaces between 7 and 12 mm^2, one can suspect that a rule is missing. The argument relies on the convexity of the attribute's range, which is the case of a great majority of numeric ranges. An analogous analysis can be made for enumerated ranges, suspecting an anomaly if only some values are dealt with.

In concrete terms, a SPARQL query can easily gather all the rules of the rule base that have a given attribute in their premise. It is also possible to narrow the query to rules having the same given entities in their premise – here, surface, defect and coil. A further narrowing can check for similar conclusions. In the above example, such a query would return the rules that also have defect and severity in their conclusions, which provides the expert with a more manageable set of rules.

6 Debugging at the Textual Level

Fewer tests can be made at the textual level, where candidate rules are simple text fragments, possibly with annotation, but without decomposition into premises and conclusions. However, precautions must be taken at this level. Quite often, the problems stem from ignorance of the text structure. Most text types have a specific *style* and an internal organization that bear part of the text meaning or at least guide its interpretation. Neglecting these stylistic rules often lead to misinterpret the text content, which is not properly analyzed in context.

6.1 Discrepancies between Different Parts of the Text

In many regulatory texts, the basic terms and general categories (conditions, purposes, objectives, etc.) are defined before the statement of rules, but the vocabulary of these different parts is expected to be consistent. The presence of undefined basic terms or categories in the rule part of the text is a possible anomaly.

For instant, the EU regulation about safety belts has

- a Definitions part where all the physical elements of the safety belt system are defined,
- a Specifications part describing the requirements to check (each test is normally mentioned by at least one specification in this part),
- a Description part where it is described how the tests must be performed.

In the Description part, an "exposure test" is mentioned twice but there is no such mention in the specification part: this reveals an anomaly. A search for the word "exposure" shows that it occurs in the subtitles of the paragraph "Conditioning of straps for the breaking-strength test". A human expert can conclude that exposure test does not denote a test but rather some form of conditioning, *i.e.* the first part of a complex test.

6.2 Undetected Rules

It is often difficult to identify the rule statements in the source text because they can have various surface forms. There are many ways to express a rule prescription. If the rule detection cannot be fully automated, some characteristic rule markers may nevertheless be searched for. For instance, in the UNO regulation, the future tense almost always has a prescriptive meaning, as in the following sentences:

> The dynamic test shall be performed on two belt assemblies which have not previously been under load
> If any of these items and parts do not comply with the above conditions, they shall be subjected to the cold impact test
> The abrasion conditioning will be performed on every device in which the strap is in contact with a rigid part of the belt
> The mating parts of the buckle shall be coupled together manually immediately after being removed from the refrigerated cabinet.

Of course, this test is not always fully reliable. According to our experiments on the UNO regulation corpus, searching for "shall" returned 91 rule statements and 5 additional sentences which were not interpreted as rules. This means that a sentence containing "shall" but not marked as a rule is worthy to be presented as a potential anomaly. This is not specific to the UNO example, since "shall" can be found in many legal regulations. Such anomaly detection patterns should be designed separately for each type of texts.

6.3 Rules without Annotation

Once a sentence is recognized as a rule, it is highly improbable that this rule does not involve any domain specific notion. A rule without any conceptual annotation is suspect. It may not be a rule, the domain description may be incomplete, or the rule may fall out of the scope of the application. Whatever the answer, checking this point early in the acquisition process can help building an adequate domain description. The following example from the UNO regulation falls in the third category:

> The apparatus shall not be constructed of materials that will affect the corrosiveness of the mist.

The absence of annotation shows that the annotation focuses on properties of the *objects* submitted to the tests, rather than on the *devices* used for conducting

the tests. Those devices, accurately described in the annexes, have been left out of the domain description. As this makes sense with regard to the target application, the selection of the above rule should be cancelled.

6.4 Large Text Fragments with no Rule

Texts stating business rules and regulatory texts avoid irrelevant considerations. They are generally divided into parts, which have different functional roles. Some parts may be peripheral with respect to the target application but the relevant ones often state a large set of business rules. In those parts, a large chunk of text without any detected rule can be considered as an anomaly: the expert is expected to look for missing rules.

For instance, the seven first parts of the UNO regulation # 16 are the following:

- Part 1 lists the types of vehicles under concern,
- Parts 3, 4 and 5 describe how to apply for approval of a vehicle type's safety belts, how an approved belt is marked and how the approval is delivered.
- Part 2 defines 51 technical terms used in the following, *e.g.* "Lap belt: A two-point belt which passes across the front of the wearer's pelvic region".
- Part 6 specifies the required values for different properties of the safety belt systems
- Part 7 describes in detail the tests to be performed and how the measures are taken during the test

The parts 1, 3, 4 and 5 are not relevant for the target application but the other are. Each term of part 2 has to be checked against the ontology. Parts 6 and 7 have to be implemented: each paragraph contains at least one rule. And, as already mentioned in section 6.1, each test in part 7 must be related to at least one specification in part 6.

Simple strategies for grasping the sentences involving a particular notion may also provide some degree of checking, *e.g.* every sentence mentioning a salt solution refers to the conditioning phase of the corrosion test.

7 Conclusion

As the acquisition of business rules from text is an important challenge, we have considered improving the quality of the result by debugging the rules as early as during the NL acquisition phase. Two complementary approaches have been explored. The first one relies on patterns of anomalies attested at the formal level by people working on the maintenance of rule systems. Some important patterns can be searched for at a semi-formal level of description of the business rules, based on the indexing structure of the Documented Business Rules Model, and with the help of web semantic and search techniques. The second approach relies on the structure of the text to find hints of mis-translation. These hints can help discovering problems that are out of the scope of formal methods. Both approaches are exemplified on real texts. They provide an interesting tool to improve the quality of acquired rules.

References

1. Bajwa, I.S., Lee, M.G., Bordbar, B.: Sbvr business rules generation from natural language specification. In: AAAI Spring Symposium 2011 Artificial Intelligence 4 Business Agility, pp. 541–545. AAAI, San Francisco (2011)
2. Baumeister, J., Kleemann, T., Seipel, D.: Towards the verification of ontologies with rules. In: Wilson, D., Sutcliffe, G. (eds.) FLAIRS Conference, pp. 524–529. AAAI Press (2007)
3. Baumeister, J., Reutelshoefer, J., Puppe, F.: Engineering intelligent systems on the knowledge formalization continuum. International Journal of Applied Mathematics and Computer Science (AMCS) 21(1) (2011), http://ki.informatik.uni-wuerzburg.de/papers/baumeister/2011/2011-Baumeister-KFC-AMCS.pdf
4. Baumeister, J., Reutelshoefer, J., Puppe, F.: Knowwe: A semantic wiki for knowledge engineering. Applied Intelligence 35(3), 323–344 (2011)
5. Baumeister, J., Seipel, D.: Anomalies in ontologies with rules. J. Web Sem. 8(1), 55–68 (2010)
6. Dinesh, N., Joshi, A., Lee, I., Sokolsky, O.: Reasoning about conditions and exceptions to laws in regulatory conformance checking. In: van der Meyden, R., van der Torre, L. (eds.) DEON 2008. LNCS (LNAI), vol. 5076, pp. 110–124. Springer, Heidelberg (2008)
7. Fuchs, N.E., Schwertel, U., Torge, S.: A natural language front-end to automatic verification and validation of specifications. Tech. Rep. PMS-FB-1999-5, LMU München, Munich (1999)
8. Guissé, A., Lévy, F., Nazarenko, A.: From regulatory texts to BRMS: How to guide the acquisition of business rules? In: Bikakis, A., Giurca, A. (eds.) RuleML 2012. LNCS, vol. 7438, pp. 77–91. Springer, Heidelberg (2012)
9. Kuhn, T.: AceWiki: Collaborative Ontology Management in Controlled Natural Language. In: Lange, C., Schaffert, S., Skaf-Molli, H., Vlkel, M. (eds.) Proc. of the 3rd Semantic Wiki Workshop. CEUR Workshop Proceedings, vol. 360 (2008)
10. Lévy, F., Nazarenko, A., Guissé, A., Omrane, N., Szulman, S.: An environment for the joint management of written policies and business rules. In: Proceedings of the International Conference on Tools with Artificial Intelligence (IEEE-ICTAI 2010), pp. 142–149 (2010)
11. Lévy, F., Nazarenko, A.: Formalization of natural language regulations through SBVR structured English (Tutorial). In: Morgenstern, L., Stefaneas, P., Lévy, F., Wyner, A., Paschke, A. (eds.) RuleML 2013. LNCS, vol. 8035, pp. 19–33. Springer, Heidelberg (2013)
12. Ma, Y., Nazarenko, A., Audibert, L.: Formal description of resources for ontology-based semantic annotation. In: Proceedings of the International Conference on Language Resources and Evaluation (LREC 2010). ELRA, Malta (2010)
13. Nazarenko, A., Guissé, A., Lévy, F., Omrane, N., Szulman, S.: Integrating written policies in business rule management systems. In: Bassiliades, N., Governatori, G., Paschke, A. (eds.) RuleML 2011 - Europe. LNCS, vol. 6826, pp. 99–113. Springer, Heidelberg (2011)
14. Omrane, N., Nazarenko, A., Rosina, P., Szulman, S., Westphal, C.: Lexicalized ontology for a business rules management platform: An automotive use case. In: Olken, F., Palmirani, M., Sottara, D. (eds.) RuleML - America 2011. LNCS, vol. 7018, pp. 179–192. Springer, Heidelberg (2011)

15. Popov, B., Kiryakov, A., Ognyanoff, D., Manov, D., Kirilov, A.: Kim – a semantic platform for information extraction and retrieval. Nat. Lang. Eng. 10(3-4), 375–392 (2004)
16. Preece, A., Shinghal, R.: Foundation and application of knowledge base verification. Intl. Journal of Intelligent Systems 9(8), 683–701 (1994)
17. Preece, A., Shinghal, R., Batarekh, A.: Principles and practice in verifying rule-based systems. Knowledge Engineering Review 7(2), 115–141 (1992)
18. Solomakhin, D., Franconi, E., Mosca, A.: Logic-based reasoning support for sbvr. In: CILC (Italian Conference on Computational Logic), pp. 311–325 (August 2011)

Author Index

Antoniou, Grigoris 188
Athan, Tara 13

Batsakis, Sotiris 37
Bauer, Bernhard 84
Bikakis, Antonis 158
Boella, Guido 218
Boley, Harold 13, 52
Brunstein, Svenja 129

Caire, Patrice 158
Cristani, Matteo 99

Dasiopoulou, Stamatia 144
De Clercq, Sofie 68
De Cock, Martine 68
Di Caro, Luigi 218

Efstathiou, Vasiliki 144

Frenzel, Christoph 84
Fusco, Mario 173

Getoor, Lise 1
Governatori, Guido 13, 99, 114
Grosof, Benjamin N. 2
Grüninger, Michael 12

Harrison, Philip 226
Herzig, Philipp 129

Kompatsiaris, Ioannis 144

Le Traon, Yves 158
Lévy, François 19, 234

Meditskos, Georgios 144
Moawad, Assaad 158
Moore, Reagan W. 203
Murray, William R. 226

Nain, Grégory 158
Nazarenko, Adeline 19, 234
Nowé, Ann 68

Olivieri, Francesco 99

Palmirani, Monica 13
Paschke, Adrian 13
Proctor, Mark 173

Rajasekar, Arcot 203
Robaldo, Livio 218
Rotolo, Antonino 99, 114

Sanneck, Henning 84
Scannapieco, Simone 99
Schill, Alexander 129
Schockaert, Steven 68
Sierhuis, Maarten 34
Singliar, Tomas 226
Sottara, Davide 173

Tachmazidis, Ilias 188

Wan, Mike 203
Webber, David 36
Wolf, Bernhard 129
Wyner, Adam 13

Xu, Hao 203