
RuleML2013@Challenge, Human Language Technology and Doctoral Consortium

**Proceedings of the 7th International Rule Challenge, the HLT and the
DC at RuleML2013, the 8th International Symposium on Rules**

Seattle, Washington, July 11th-13th, 2013.

Edited by

Paul Fodor *

Dumitru Roman **

Darko Anicic ***

Adam Wyner ****

Monica Palmirani *****

Davide Sottara *****

François Lévy *****

* State University of New York at Stony Brook, USA

** SINTEF, Norway

*** FZI / KIT, Germany

**** University of Aberdeen, UK

***** CIRSFID-University of Bologna, Italy

***** Arizona State University, USA

***** LIPN, Univ. Paris 13, France

Table of Contents

- Preface

Part 1: RuleML2013@Doctoral Consortium

Doctoral Consortium Papers

1. Proposal for Using NLP Interchange Format for Question Answering in Organizations
Majid Latifi
2. Process Representation Using Transaction Logic
Reza Basseda
3. First Approaches on Knowledge Representation of Elementary (Patent) Pragmatics
Shashishekhar Ramakrishna

Part 2: RuleML2013@Challenge

Invited Demo Papers

4. RAWE: An Editor for Rule Markup of Legal Texts
Monica Palmirani, Luca Cervone, Octavian Bujor, Marco Chiappetta

Challenge Demo Papers

5. R-CoRe: A Rule-based Contextual Reasoning Platform for Aml
Assaad Moawad, Antonis Bikakis, Patrice Caire, Grégory Nain, Yves Le Traon
6. Interpreting Regulations with SBVR
Elie Abi-Lahoud, Tom Butler, Donald Chapin, John Hall
7. Graph-based Editor for SWRL Rule Bases
Jaroslaw Bak, Maciej Nowak, Czesław Jędrzejek
8. Advanced Knowledge Base Debugging for Rulelog
Carl Andersen, Brett Benyo, Miguel Calejo, Mike Dean, Paul Fodory, Benjamin N. Grosofz, Michael Kifery, Senlin Liangy, Terrance Swiftx
9. Knowledge-based Highly-specialized Terrorist Event Extraction
Jakub Dutkiewicz, Czesław Jędrzejek, Jolanta Cybulka, Maciej Falkowski
10. SBVR as a Semantic Hub for Integration of Heterogeneous Systems
Ling Shi, Dumitru Roman, Arne J. Berre
11. Grailog KS Viz:A Grailog Visualizer for Datalog RuleML Using an XSLT Translator to SVG
Martin Koch, Sven Schmidt, Harold Boley, Rainer Herpers
12. Importation Closure that is Robust to Circular Dependencies
Tara Athan

13. [Transforming Association Rules to Business Rules: EasyMiner meets Drools](#)
Stanislav Vojíř, Tomáš Kliegr, Andrej Hazucha, Radek Škrabal, Milan Šimůnek

Part 3: RuleML2013@Human Language Technology

Human Language Technology Papers

14. [Constructing Controlled English for Both Human Usage and Machine Processing](#)
Ping Xue, Steve Poteet, Anne Kao, David Mott, Dave Braines
15. [RECON - A Controlled English for Business Rules](#)
Ed Barkmeyer, Fabian Neuhaus
16. [A Study on Translating Regulatory Rules from Natural Language to Defeasible Logics](#)
Adam Wyner, Guido Governatori

The whole proceedings can also be downloaded as a single file ([pdf](#)).

10-July-2013: submitted by Monica Palmirani
10-July-2013: published on CEUR-WS.org

Preface

This volume collects the ten demo papers accepted for presentation at the RuleML2013 International Rule Challenge (the 7th International Web Rule Challenge), three papers accepted at the Human Language Technology special track of RuleML2013 and three selected papers accepted for the RuleML2013 Doctoral Consortium.

The 7th International Web Rule Challenge is one of the highlights at RuleML2013 Conference, providing a competition among innovative rule-oriented applications that are aimed at both the research and industrial side. The International Web Rule Challenge is a forum where new ways of the use of rule-based systems are presented and practical experiences about implementing these systems are reported. The Challenge is devoted to disseminating the most advanced practical experiences with rule-based applications. These papers include rule-based implementations/tools/applications, editing environments and IDEs for Web rules, demonstrations of engineering methods, implementations of rule standards, demos, case studies, use cases, experience reports, best practice solutions, rule benchmarks and evaluations. The Challenge session also features an invited demo paper by Monica Palmirani on RAWE, an editor for rule markup of legal texts and conversion to LegalRuleML based on Akoma Ntoso markup. This year, the major topics of the Challenge papers were: extensions and implementations of rule-related standards (W3C RIF, RuleML, SBVR, BPMN, BPEL), defeasible reasoning, editing environments and IDEs for Web rules, distributed rule bases and rule services, and e-reports on industrial experience about rule systems.

The RuleML Doctoral Consortium is part of the RuleML International Symposium on Rules since 2011. It attracts Ph.D. researchers in the area of Rules and Markup Languages from different backgrounds (e.g. theoretical, application, vertical domain-specific) and encourages a constructive and fruitful interdisciplinary approach. At the doctoral symposium, students present their ideas in a dynamic and friendly setting as well as interact with academics and commercial experts in the field, who evaluate their research projects from both theoretical and application points of view.

The Human Language Technology Track addresses the knowledge acquisition bottleneck that arises when converting the vast amount of regulatory text on the Web expressed in natural language to formal, machine-processable rules. Six papers in total were accepted to the Track, three of which appear in the associated LNCS volume and three of which appear in this volume. Topics represented in the Track include using controlled languages, extracting semantic information from legislative text, and mapping English onto fuzzy logic. There are six talks and one tutorial.

We warmly thank all authors, students, supervisors, referees, co-chairs, members of the program committee and the organizing team that made the RuleML2013 Symposium, International Web Rule Challenge, and Doctoral Consortium a great success.

July 2013

Paul Fodor, Dumitru Roman, Darko Anicic, Adam Wyner, Monica Palmirani, Davide Sottara, François Lévy

Proposal for Using NLP Interchange Format for Question Answering in Organizations

Majid Latifi

Department of Software, Universitat Politècnica de Catalunya – BarcelonaTech(UPC), Barcelona, Spain
mlatifi@lsi.upc.edu

Abstract. The growth of technology and sciences has greatly influenced the area of management and decision-making procedures, and has dramatically changed the decision-making processes in different levels, both quantitatively and qualitatively. Knowledge management plays a vital role in supporting enterprise learning, since it facilitates the effective collective intellect of the enterprise. Different methods for user-friendly knowledge access have been developed previously. The most sophisticated ones provide a simple text box for a query which takes Natural Language (NL) queries as input. Question Answering (QA) system is playing an important role in current search engine optimization. Natural language processing technique is mostly implemented in QA system for asking user's question and several steps are also followed for conversion of questions to query form for getting an exact answer. Query languages have complex syntax, requiring a good understanding of the representation schema, including knowledge of details like namespaces, class and property names. In this research we proposed a model to implement Conceptual Question Answering and Automatic Information Inferences for the enterprise's operational knowledge management in ontology-based learning organization.

Keywords: Enterprise Ontology, Learning Organization, Question Answering(QA), Information Inference, NLP.

1 Introduction

Retrieval and extraction processes - for enterprise management and decision-making - have gained an excessive importance as the mass of data and information stored in various resources increases. Knowledge is considered a key factor for enterprise prosperity at present and future. Knowledge management is an integrated, systematic process that applies a suitable combination of information technologies and human cooperation in order to identify, manage and share the information capitals. In addition, it both includes the explicit and implicit knowledge of the staff and it applies various and extensive methods to retrieve, store and share knowledge in a certain enterprise.

The application of “Semantic Web” technologies to learning processes is receiving an increasing attention from the perspective of facilitating the selection, delivery and tailoring of learning experiences. But most of the current approaches are centered on

the final interaction of the learner with the “learning objects” provided for him/her, neglecting the organizational perspective. From the viewpoint of an organization, the application of Semantic Web technologies should be motivated by the improvement of learning-oriented mechanisms, including both cultural and structural aspects, and considering the ideal of achieving a state of continuous improvement in learning behavior. Such an approach to achieving a “semantic learning organization” gives a complementary perspective to existing “educational Semantic Web” propositions [2]. A main need for the semantic enterprise model is one which extracts and displays the enterprise semantics.

Most knowledge bases provide facilities for querying through the use of some formal language such as SPARQL or SeRQL. However, these have a fairly complex syntax, requiring a good understanding of the data schema and being prone to errors due to the need to type long and complicated URIs. These languages are homologous to the use of SQL for interrogating traditional relational databases and should not be seen as an end user tool[13].

The obvious solution to these problems is to create some additional abstraction level that provides a user friendly way of generating formal queries. It may be possible to infer from this information for the machine so that we can carry out the decision-making and planning procedures in enterprise processes through automatic inference.

2 Statement of the Problem and Related Work

A basic method to transform an organization into a learning organization is to apply knowledge management within the organization. By facilitating the process of creating and sharing knowledge, and through providing positive working environments and effective rewarding systems, knowledge management accelerates enterprise learning and helps the enterprise adjust itself to today's rapid changes and hence survive in pace with these changes[9]. By using ontology, we can identify the meanings related to a domain, an enterprise or a society or even determine these meanings within different societies in details as desired [3]. In Ontology-based QA system, the knowledge based data, where the answers are sought, has a structured organization. The question-answer retrieval of ontology knowledge base provides a convenient way to obtain knowledge for use, but the natural language need to be mapped to the query statement of ontology. Accessing structured data such as that encoded in ontologies and knowledge bases can be done using either syntactically complex formal query languages or complicated form interfaces that require expensive customization to each particular application domain.

Probably due to the extraordinary popularity of search engines such as Google, people have come to prefer search interfaces which offer a single text input field where they describe their information need and the system does the required work to find relevant results. While employing this kind of interface is straightforward for full text search systems, using it for conceptual search requires an extra step that converts

the user's query into semantic restrictions like those expressed in formal search languages. Following are discussed some examples of such query interfaces.

CLOnE[9], presents a controlled language for ontology editing and a software implementation, based partly on standard NLP tools, for processing that language and manipulating an ontology. The input sentences are analyzed deterministically and compositionally, which the software consults in order to interpret the input's semantics; this allows the user to learn fewer syntactic structures since some of them can be used to refer to either classes or instances, for example. A repeated-measures, task-based evaluation has been carried out in comparison with a well-known ontology editor.

The Controlled Language for Ontology Editing (CLOnE) allows users to design, create, and manage information spaces without knowledge of complicated standards (such as XML¹, RDF² and OWL³) or ontology engineering tools. It was implemented as a simplified natural language processor that allows the specification of logical data for semantic knowledge technology purposes in normal language. CLOnE is designed either to accept input as valid or to reject it and warn the user of his errors; because the parsing process is deterministic, the usual IE performance measures (precision and recall) are not relevant.

QACID [10] is based on collection of queries from a given domain which are analyzed and grouped as clusters and those are manually annotated using SPARQL queries. Each query is considered as bag of words, mapping between words in NL queries into KB by using string distance metrics. SPARQL generator replaces the ontology with instances mapped for original NL query. It is domain specific and the performance depends on the types of questions collected in domain.

ONLI (Ontology Natural Language Interaction) [11] is a natural language question answering system used as front-end to the RACER reasoner and to nRQL, RACER's query language. ONLI assumes that the user is familiar with the ontology domain and works by transforming the user's natural language queries into nRQL. No details are provided regarding the effort required for re-purposing the system.

QAAL [12] surveys different types of question answering system based on ontology and semantic web model with different query format. For comparison, the types of input, query processing method, input and output format of each system and the performance metrics with its limitations was analyzed and discussed. There are basically three types of question classification methods available. Those are machine learning approaches, knowledge based approach and template based approach. In QAAL system is used template based approach for fast retrieval of answer. If the question is already asked in that system, the retrieval takes place within question template table, otherwise matching is performed using Graph Matching Algorithm and uses Spread Activation Algorithm for query matching with the ontology.

¹ eXtensible Markup Language

² Resource Description Framework

³ Web Ontology Language

QuestIO [13] system has a natural language interface for accessing structured information, that is domain independent and easy to use without training. It brings the simplicity of Google's search interface to conceptual retrieval by automatically converting short conceptual queries into formal ones, which can then be executed against any semantic repository. The QuestIO application is open-domain (or customizable to new domains with very little cost), with the vocabulary not being predefined but rather automatically derived from the data existing in the knowledge base. The system works by converting NL queries into formal queries in SeRQL. It was developed especially to be robust with regard to language ambiguities, incomplete or syntactically ill-formed queries, by harnessing the structure of ontologies, fuzzy string matching, and ontology-motivated similarity metrics. It works by leveraging the lexical information already present in the existing ontologies in the form of labels, comment and property values.

PANTO [14] model a Portable nAtural laNguage inTerface to Ontologies which accepts input as natural language form and the output is in SPARQL query. It is based on triple model in which parse tree is constructed for the data model using the off-the-shelf Standford parser. Logic rules are applied for natural language queries as negation, comparative and superlative form. For mapping WordNet and String metric algorithms are used. The parse tree forms the intermediate representation as Query Triples Form. Then PANTO converts Query Triples form into OntoTriples form which are represented as entities in ontology.

OntoTriples are finally interpreted as SPARQL form. The performance of PANTO is analyzed by using FMeasure type. At the maximum 88.05% Precision is achieved for Geography domain with tested queries. So this system helps bridge the gap between the real world users with the semantic web based on logic model.

AquaLog [15] is capable of learning the user's jargon in order to improve his experience by the time. Their learning mechanism is good in a way that it uses ontology reasoning to learn more generic patterns, which could then be reused for the questions with similar context. In this system two major models are used as Linguistic Component which is used to convert the NL questions into Query-triple format and Relation Similarity Service (RSS) which takes Query Triple form into Onto-Triple form. The data model is triple like {Subject, Predicate, Object} type. The Performance is based on Precision, Recall and also failure types are referred separately. At average 63.5 % of successive answers are retrieved from ontology with closed domain environment.

QASYO [16] is a sentence level question-answering system that integrates natural language processing, ontologies and information retrieval technologies in a unified framework. It accepts queries expressed in natural language and YAGO [18] ontology as inputs and provides answers drawn from the available semantic markup which combining several powerful techniques in a novel way to make sense of NL queries and to map them to semantic markup. Semantic analysis of questions is performed in order to extract keywords used in the retrieval queries and to detect the expected answer type. In the QASYO model there are 4 phases: question classifier, linguistic component, query generator and query processor which characterizing it's architec-

ture as a waterfall model. One NL query gets translated into a set of intermediate, triple-based representations, query-triples, and then these are translated into ontology-compatible triples.

The whole QA process is composed of two consecutive phases: question analysis and answer retrieval. This model requires both an evaluation of its query answering ability. Another extension is to provide information about the nature and complexity of the possible changes required for the ontology and the linguistic component.

Knowledge management system includes methods for obtaining or gathering information, organizing, distributing and sharing information among the staff in an organization. In this research, the potential role of the Semantic Web Technology as a driver for advanced learning organizations and Question Answering system is focused on providing access to the information stored in a KB by means of natural language queries.

3 Research Objectives

The current research is aimed to show that using standard NLP tools, ontology and informal to formal semantic query model proposed in the current research can establish a relationship among various sectors including duties, activities, resources and information structure of a certain enterprise so that managerial requirements can be desirably met through semantic modeling. As a result, we may have a better chance of using this information for the managers and the users through conceptual queries on the information system of the enterprise. In attention to the actual state of semantic web technology and NLP, the recommended path for organizations that are committed to the view of a learning organization is that of first addressing infrastructural elements. Such infrastructures can be considered as the study and provision of the ontologies for each aspect of the semantic learning organization. Therefore, how can we improve knowledge management in enterprises through an appropriate selection based on ontology?. Also, how can we respond to the managerial requirements of the enterprises from simple decisions to strategic ones and how can we perform automatic extraction of the information?. Consequently, the following objectives are followed in parallel with works carried out previously:

1. Conceptual framework for the notion of a semantic learning organization with using semantic search model instead of using normal keyword search model is provided.
2. Designing and presenting a method to translate user's semantic queries into well-defined queries using the results of NLP Interchange Format (NIF) to answer the semantic questions.
3. The necessity to be robust and ability to deal with all kinds of input including ungrammatical text, sentence fragments, short queries, etc.

4 Scope of Activity

4.1 Learning Organization Ontology

The existing organizational architecture is faced with a semantic shortage between humans and systems for having a precise and general understanding of them, which in turn causes communication problems between humans and systems or vice versa. These problems prohibit the materialization of the organizations in an assembled and concordant form with other organizations [7]. Our goal is not only to design a ‘conceptual’ ontology model but also to implement it as an operational ontology. This approach, mainly favored by the research community, may be beneficial for integrating the domain ontology model with an inference engine for the language. Trying to match the users’ requests by providing appropriate formal commands is faced with restrictions, and thus making such semantic query by programmers is demanding, time consuming and inefficient.

4.2 Translating Natural Language Questions into Well-defined Queries

There is technically too complicated to represent and comprehend the domain for a domain expert who has little knowledge in the well-defined queries. More importantly, from a practical point of view, there is no publicly known robust engine to manage a large KB with practical performance. On the other hand, we should increase the machines' capability in understanding the organizational structure (Intelligent-making). To this end, having analyzed the existing concepts in the scope of knowledge management of the learning organizations, we reckon the significance of the information capitals of an enterprise through an ontology-based method. Answering to semantic questions will help increase the capability of learning organizations.

The growing interest in Semantic Web applications and need to translate natural language question into a machine-readable format create many uses for such applications. It is implemented as a natural language processor that allows the specification of logical data for semantic knowledge technology purposes in normal language, but with high accuracy and reliability. The components are based on NLP Interchange Format(NIF) with using statistical machine translation method.

5 Modelling of Conceptual Question Answering Method in Learning Organizations

We designed an initial model to implement Conceptual Question Answering and Automatic Information Inferences for the enterprise's operational knowledge management in learning organization. To achieve this goal, we evaluate the SPARQL and SeRQL languages for semantic search. In [5] is shown an application of SPARQL-DL query language to natural language processing, more especially as a rule engine to use within a semantic parser. As shown, the use of such formalism for this task has several advantages including the straightforward conversion of a typed dependency graph

in an ontology. In Fig. 1, the general model of our proposed system is represented. It has the following modules.

- **Query Parsing and Analysis:** In this phase, the analytical operation of the question is found out. This Analysis is responsible for Natural Language Processing (NLP). It is a technique to identify the type of a question, type of an answer, subject, verb, noun, phrases and adjectives from the question. Tokens are separated from the question and the meaning is analyzed and the reformulation of question is sent to the next stage. The input is converted into Natural Language and is implemented using word segmentation algorithm. In word segmentation algorithm the input query from the user is divided as keywords which is further subdivided and searched in knowledge base to get correct answers.

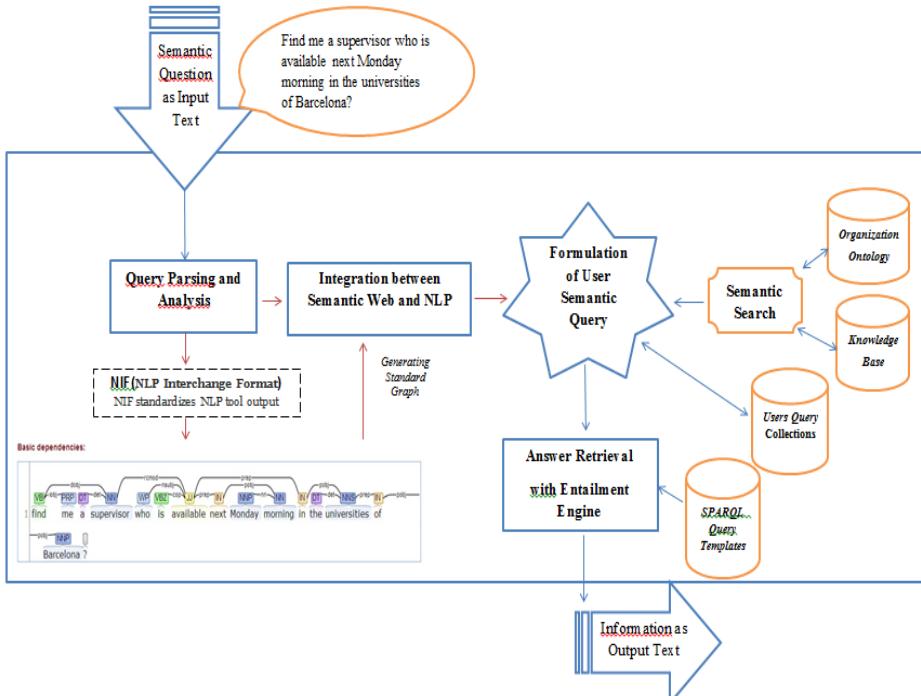


Fig. 1. Suggested Model for Semantic Question Answering

- **Integration between Semantic Web and NLP:** The tools available nowadays for Natural Language Processing can achieve very good results on many complex tasks such as the parsing of a sentence. An NLP Interchange Format for integrating NLP applications is presented by [6]. NIF addresses weaknesses of centralized integration approaches by defining an ontology-based and linked-data aware text annotation scheme. The NLP Interchange Format (NIF) is an RDF/OWL-based format that aims to achieve interoperability between NLP tools, language resources and annotations. The core of NIF consists of a vocabulary, which allows to represent strings as RDF resources. By being directly based on RDF, Linked Data and

ontologies, NIF also comprises crucial features such as annotation type inheritance and alternative annotations, which are cumbersome to implement or not available in other NLP frameworks [17].

- **Regenerating of Semantic Query:** According to the user's choice, the formulation of query is generated with the help of YAGO[18] and WordNet [19] which are implemented as semantic matching model.
- **Semantic Search:** At next stage, the Search is carried out using Conceptual Graph Matching algorithm which is the best technique. All the sentences in repository are framed as conceptual graph and the given question is also framed as conceptual graph. The matching of question CG with given CG are checked out using CG matching algorithms and the result is displayed at front-end of the our system. Graph patterns are important concepts in semantic search. RDF model is organized and graph patterns are used to formulate and encode constraint queries for locating sub graph in RDF network.
- **Graph Matching in Ontology:** Conceptual Graph acts as an intermediate language for mapping natural language questions and assertions to a relational database. Conceptual Graph (CG) contains concept, concept relation and argument. It is a graph which represents logic based on semantic model of artificial intelligence and existential graphs. Resource Description Framework (RDF) is a framework which contains triple syntax to express annotations as subject, predicate and object. Information resources are commonly represented as uniform Resource Identifiers (URIs). URIs are described by RDF. RDF triples are visualized as directed labeled graph in which subject; objects are represented as nodes and predicates as arcs.
- **Searching Ontology Nodes:** Semantic Search Algorithm is based on Conceptual Graph form of user query and domain ontology. In [8] Spread Activation is a method for searching the nodes in ontology as in semantic manner. It exploits relations between nodes in ontology. Nodes may be terms, class, object etc. Relations are labeled directed or weighted manner. SA algorithm creates initial nodes that are related to the content of the user's query and assign weights to them. After that, nodes will activate with different nodes on ontology by some rules.
- **Template based Approach:** There are basically three types of question classification methods are available. Those are machine learning approaches, knowledge based approach and template based approach. In this research we use template based approach for fast retrieval of answer. If the question is already asked in that system, the retrieval get from question template table form, otherwise matching is performed using matching algorithm.
- **Answer Retrieval with Entailment Engine:** This part of the system is based on an entailment engine. This module uses entailment techniques to infer semantic deductions between a users'query collections and the SPARQL query collections included in the formulation of user semantic query previously obtained. This process allows the system to associate new incoming queries with their corresponding SPARQL expressions in order to retrieve the answer sought from the RDF database.

6 Conclusions

The main undertaking of the current contribution is to present ongoing work in facilitating learning organizations and their use of ontology-based tools by striving to translate natural language queries into well-defined queries and retrieving exact answers, which in turn can be executed in the framework presented here . A model was introduced to automatically convert semantic query to formal query in a bid to provide answers for conceptual question and to infer information from organizational knowledge base.

Answers are retrieved from ontology using semantic search approach interoperability for NIF components, web services and question-to-query algorithm is evaluated in our system for analyzing performance evaluation. Finally performance of question answering system of getting exact result can be improved by using semantic search methodology to retrieve optimum answers from organizational ontology model.

Acknowledgments

I would like to thank our software department (LSI) from KEMLG research group in Polytechnic University of Catalonia (UPC). Especially, I would like to thank Dr. Miquel Sàncchez-Marrè for his helpful comments and guidance. I acknowledge the financial support of the Generalitat de Catalunya through the AGAUR agency for Consolidated Research Groups. This support (2009SGR 1365) was granted to the Knowledge Engineering & Machine Learning group (KEMLG).

References

1. Latifi, M., Khotanlou, H., Latifi, H.: An Efficient Approach Based On Ontology to Optimize the Organizational Knowledge Base Management for Advanced Queries Service. In: 3rd IEEE International Conference on Communication Software Networks(ICCSN), ISBN: 978-1-61284-485-5, pp. 269 – 273 (2011)
2. Sicilia, M., Lytras, M.: The Semantic Learning Organization. In: The Learning Organization, Vol. 12 Iss: 5, pp.402 – 410 (2005)
3. Daconta, M., C., Smith, K., T., Obrst, L., J.: The Semantic Web: A Guide to the Future of XML, Web Services, and Knowledge Management. John Wiley & Sons, USA (2003)
4. Aggestam, L.: Learning Organization Or Knowledge Management: Which Came First, The Chicken Or The Egg?. In: Information Technology and Control, vol 35, No.3 (2006)
5. Vitucci, N., Arrigoni Neri, M., Tedesco, R., Gini, G.: Semanticizing syntactic patterns in NLP processing using SPARQL-DL queries. Politecnico di Milano, Dipartimento di Elettronica e Informazione Via Ponzio 34/5, 20133 Milano, Italy (2012)
6. Hellmann, S., Lehmann, J., Auer, S.: NIF: An ontology-based and linked-data-aware NLP Interchange Format. <http://svn.aksw.org> (2012)
7. Kang, D., Lee, J., Choi, S., Kim, K.: An ontology-based Enterprise Architecture, Expert Systems with Applications, pp.1456-1464 (2010)
8. Suchal, J., Caching spreading activation search. Slovak University of Technology(2007)

9. Funk, Adam, et al.: CLOnE: Controlled language for ontology editing. *The Semantic Web*, Springer Berlin Heidelberg, pp.142-155 (2007)
10. Fernandez, O., R. Izquierdo, S. Ferrandez and J.L. Vicedo, Addressing ontology-based question answering with collections of user queries. *Inform. Proces. Manage.*, 45: 175-188. DOI: 10.1016/j.ipm. (2008)
11. Shamima Mithun, Leila Kosseim, V.H.: Resolving quantifier and number restriction to question owl ontologies. In: Proceedings of The First International Workshop on Question Answering (QA2007), Xian, China (2007)
12. Kalaivani, S., and K. Duraiswamy, Comparison of Question Answering Systems Based on Ontology and Semantic Web in Different Environment. *Journal of Computer Science* 8.9, pp: 1407-1413 (2012)
13. Tablan, V., Damjanovic, D, Bontcheva, K, A Natural Language Query Interface to Structured Information, Springer-Verlog Berlin Heidelberg, ESWC 2008, pp. 361-375 (2008)
14. Wang, C., M. Xiong, Q. Zhou and Y. Yu, PANTO: A portable natural language interface to ontologies. Proceedings of the 4th European Semantic Web Conference, (ESWC' 07), Publication post of DBLP, pp: 473-487(2007)
15. Lopez, V., Motta, E.: Ontology driven question answering in AquaLog. In: NLDB 2004 (9th International Conference on Applications of Natural Language to Information Systems), Manchester, UK (2004)
16. Moussa, Abdullah M., and Rehab F. Abdel-Kader,: QASYO: A Question Answering System for YAGO Ontology. *International Journal of Database Theory and Application* 4.2 (2011)
17. Schierle, M.: Language Engineering for Information Extraction. Phd thesis, University at Leipzig, Leipzig (2011)
18. F. M. Suchanek, G. Kasneci and G.Weikum,: YAGO: A Core of Semantic Knowledge Unifying WordNet and Wikipedia. In Proceedings of 16th International World Wide Web Conference (IW3C2), pp. 697-706 (2007)
19. Miller, George A.: WordNet: a lexical database for English. *Communications of the ACM* 38.11, pp. 39-41 (1995)

Process Representation Using Transaction Logic

Reza Basseda

Stony Brook University, Stony Brook, NY, 11794, USA

Abstract. Representing and answering the queries about the dynamic behavior of processes in knowledge base systems has become a challenging research area in the field of logic programming and knowledge representation systems. In this report, we are going to show how transaction logic can be used to efficiently represent dynamic behavior embedded in different domains. The ability of properly representing state changes in transaction logic enables us to express dynamic behavior of processes in different domains. The use of transaction logic to represent dynamic behavior decreases the size of knowledge bases and the query response time in comparison with other existing approaches. The efficiency of our method along with other features of transaction logic and its theoretical basis makes it an appropriate approach to represent dynamic behavior of processes in various domains.

Keywords: Process Representation, Transaction Logic

1 Introduction

In many real world applications of knowledge representations systems, effective representation of processes embedded in the domain knowledge enables the knowledge base system to answer a wide range of queries about those processes. For example, in the medical domain, physiology explains different processes by showing how different organs and parts of a human body interacts with each other while anatomy discusses the structure of the human body and its organs. A medical knowledge base system needs to represent both of the physiological and anatomical knowledge in order to be able to answer the queries about diseases and medical experiments.

Let us illustrate this concept via an example. Consider the process of myocardial infarction (MI) or acute myocardial infarction (AMI) in medical science, which is commonly known as a heart attack. Basically, myocardial infarction results from the interruption of blood supply to a part of the heart, causing heart cells to die. This is most commonly due to occlusion (blockage) of a coronary artery following the rupture of a vulnerable atherosclerotic plaque, which is an unstable collection of lipids (cholesterol and fatty acids) and white blood cells (especially macrophages) in the wall of an artery. The resulting ischemia (restriction in blood supply) and ensuing oxygen shortage, if left untreated for a sufficient period of time, can cause damage or death (infarction) of heart muscle tissue (myocardium) [?]. The process starts with the step of increasing cholesterol and other lipids in the blood. This step is followed by the step of lipid

dysregulation. After the step of lipid dysregulation, the formation of atherosclerotic plaque happens. The formation of atherosclerotic plaque causes narrowing of the coronary arteries and narrow coronary arteries leads to have insufficient blood supply for myocardial muscles. Finally, insufficient blood supply for myocardial muscles results in myocardial infarction. Representation of such process in a knowledge base system needs various features to exist in the system. The system needs to represent a process in terms of different steps. Each of those steps can be defined as an abstract process as well. Each process defines a set of potential dynamic changes in the system over the set of knowledge base facts. The execution of each step also depends on the various logical formulas evaluated at the different states of the knowledge base which are created during the course of the execution. It is apparent that those dynamic and static definitions of changes and terms are tightly connected to each other.

This example shows that we need to explicitly represent processes in the knowledge base systems as they are associated with some features which may be involved in query answering. For example, time duration of execution of a process or the name of a process may be queried. However, explicit representation of processes may raise other issues. For example, treating processes as first class entities of a knowledge base system may require us to express different relationships between those entities.

There are several logic programming frameworks which can be used to address the process representation problem in knowledge base systems. Situation calculus [1] provides a representation for state changes in logic. The basic concepts in the situation calculus are situations, actions, and fluents. To describe a dynamic domain in the situation calculus, one specifies a set of actions describing what changes the situations. A set of fluents is also required to describe the changing situations. Like the situation calculus, the event calculus [2] has actions, which are called events. It also has changing properties or fluents. But unlike the standard situation calculus in which an exact sequence of hypothetical actions is represented, the event calculus is based on possibly incomplete specification of a set of actual event occurrences. Different event calculus extensions addressed the frame problem in different ways [3].

A class of action languages has been developed that is independent of a specific axiomatization [4] [5] [6]. These languages try to provide high expressiveness, natural-language-like syntax, and clear formal semantics, which are important in procedural knowledge representation. [7] uses a modular action language, \mathcal{ALM} in order to represent procedural knowledge. It was used to formalize of biological processes, including cell division, in \mathcal{ALM} . [8] also uses an action modeling scripting language to represent and reason about signaling networks. [9] is also a variation of action language \mathcal{A} [10] to represent procedural knowledge in biological networks. [11] also can be used to represent dynamic behavior in domain knowledge base systems.

Both of the above mentioned approaches are facing difficulties when it comes to process representation. Since situation calculus is using monotonic reasoning and scientific knowledge representation which usually involves non-monotonic

reasoning, situation calculus is not suitable for process representation in scientific domain. Process representation in event calculus has several problems. This formalization of events is intended as a formal analysis of the concepts rather than as a program or even as a program specification [2]. As updates in event calculus are additive and do not delete information about events, execution of a large number of process steps may be impractical. Explicit declaration of relation between processes also requires a large number of auxiliary predicates and rules. For example, to represent a containment relation between two processes, several rules and facts may be required. Although action modeling languages can represent processes in terms of action execution sequences, they are not scalable knowledge representation languages. Since they don't support features required for efficient knowledge representation such as object orientation and higher orderness, scientific knowledge representation using this type of languages is harder and less reusable. Action and process definition syntax in action modeling languages is usually different than regular logic programming syntax. That difference makes the integration of dynamic behavior and static specification of domain knowledge difficult using action modeling languages.

Transaction Logic is an extension of classical predicate logic that accounts in a clean and declarative fashion for the phenomenon of state changes in logic programs and databases [12]. Our case study shows that \mathcal{TR} eases the expression of dynamic behavior of the processes embedded in different domains. The logic of state changes provided by \mathcal{TR} facilitates the inference about processes represented in \mathcal{TR} . That representation of state changes within the logic formulas provides non-monotonic reasoning for procedural knowledge in scientific domains. Since \mathcal{TR} is a declarative formalism for specifying and executing procedures that update a logical theory, it can naturally express both the static knowledge and the dynamic behaviors in different scientific domains. We can also combine \mathcal{TR} with other logical formalism such as F-logic and HiLog in order to have object oriented and higher order formalisms. Those logical formalisms simplify the representation between processes. As dynamic behavior representation in \mathcal{TR} does not need to have any axiomatization in order to address the frame problem, the process representation in \mathcal{TR} is more scalable in comparison with other logical formulations of processes.

\mathcal{TR} includes a Horn-like fragment which supports logic programming. This logic programming framework simplifies the integration of dynamic behavior with other components of knowledge base systems. Specification of processes in the language used for specification of logical terms and rules makes the expressiveness of logical formulas and terms available for process representation. This logic programming framework also helps us to easily express a wide range of queries about the dynamic behavior of processes. \mathcal{TR} is also implemented in Flora [13], which is a perfect system for knowledge representation and reasoning.

Our process representation approach using \mathcal{TR} shows that other features of \mathcal{TR} can also help to have a very expressive and robust process specification in a knowledge base systems. For example, we took advantage of hypothetical queries to represent the concept of fault tolerant processes in the knowledge

base systems. Incremental tabling and other developments in our implementation framework also may help us to improve query answering time.

We will explain our process representation technique in the next section. Section 3 will describe our case study experiments. We will also have a brief analysis of our results in section 3, and section 4 will conclude our study.

2 Methodology

The over all representation of processes in \mathcal{TR} is simple and natural. We classify processes into two groups: complex processes and primitive processes. A complex process is a sequence of complex or primitive processes and a primitive process is a single step of execution. The relationships between processes can be represented by simple logical predicates. For example, suppose process p^1 is a sequence of processes p_1, p_2, p_3 . We use *complex_process/1* and *primitive_process/1* to indicate the type of process. *first_step(p, p₁)* says that process p_1 is the first step of process p . *next_step(p, p₁, p₂)* and *next_step(p, p₂, p₃)* show that p_1 in p is followed by p_2 and p_2 in p is followed by p_3 . We do not provide the formal explanation of these concepts due to space limit.

To keep track of the execution of complex processes, we need a structure maintaining the execution status of the complex process. The current step of a process, *current_step(P, SP)*, is an example of such a structure. A primitive process does not have internal structure.

Sequential execution of subprocesses can be defined recursively as shown below.

$$\begin{aligned} execute(P) \leftarrow & \neg complex_process(P) \wedge current_step(P, CS) \otimes \\ & execute(CS) \otimes advance(P, CS) \otimes execute(P). \end{aligned} \quad (1)$$

A complex process will be successfully executed if all of its subprocesses successfully complete their execution.

$$\begin{aligned} execute(P) \leftarrow & \neg complex_process(P) \wedge \\ & current_step(P, CS) \wedge \sim next_step(P, CS, _). \end{aligned} \quad (2)$$

advance(P, CS) in (1) above refers to changing the execution status of process P . For example, it can represent the current step change as follows. Note that elementary transactions of *insert* and *delete* are defined in our transition oracle as shown in [12].

¹ In this section, capital letters denote logical variable and lower case is used to denote constant and predicate symbols

$$\begin{aligned}
\text{advance}(P, CS) \leftarrow & \text{complex_process}(P) \wedge \text{current_step}(P, CS) \\
& \wedge \text{next_step}(P, CS, NS) \\
& \otimes \text{current_step.delete}(P, CS) \otimes \text{current_step.insert}(P, NS).
\end{aligned} \tag{3}$$

Execution of primitive processes can be defined in terms of elementary transactions *insert* and *delete*. We also can extend the transition oracle and define a specific primitive process execution as a elementary transaction. For example, assume the transaction *doit*(*P*) executes the elementary transaction associated to the primitive process *P*. We can show the *successful* and *failed* execution of *P* as in (4) and (5). Note that no matter *doit*(*P*) finishes successfully or not, *execute*(*P*) will finish successfully. However the value of *result*(*P, R*) in the final state of knowledge base will depend on the execution of *doit*(*P*).

$$\begin{aligned}
\text{execute}(P) \leftarrow & \text{primitive_process}(P) \otimes \text{doit}(P) \otimes \\
& \text{result.insert}(P, \text{success}).
\end{aligned} \tag{4}$$

$$\begin{aligned}
\text{execute}(P) \leftarrow & \text{primitive_process}(P) \otimes \sim \text{doit}(P) \otimes \\
& \text{result.insert}(P, \text{failure}).
\end{aligned} \tag{5}$$

Execution of primitive process may also include some conditional statements. We can use such precondition and postcondition statements to guard the execution of a primitive process. *precondition*(*P*) and *postcondition*(*P*) predicates can simply express those postcondition and precondition statements for a primitive process *P*. Now, we can show the *successful* and *failed* execution of *P* as in (6) and (7). In this formulation of *execute*(*P*), the evaluation of this predicate will depend on the evaluation of *precondition*(*P*) and *postcondition*(*P*). For example, assume that the execution of process *p*₃ is guarded with the conjunction of *g* and the successful execution of process *p*₁ and it does not have any postcondition. This can be represented as (8) and (9).

$$\begin{aligned}
\text{execute}(P) \leftarrow & \text{primitive_process}(P) \wedge \text{precondition}(P) \otimes \text{doit}(P) \otimes \\
& \text{result.insert}(P, \text{success}) \wedge \text{postcondition}(P).
\end{aligned} \tag{6}$$

$$\begin{aligned}
\text{execute}(P) \leftarrow & \text{primitive_process}(P) \wedge \text{precondition}(P) \otimes \sim \text{doit}(P) \otimes \\
& \text{result.insert}(P, \text{failure}) \wedge \text{postcondition}(P).
\end{aligned} \tag{7}$$

$$\text{precondition}(p_3) \leftarrow g \wedge \text{result}(p_1, \text{success}). \tag{8}$$

$$\text{postcondition}(p_3) \leftarrow \text{true}. \quad (9)$$

Serial conjunctions used in our formulas allow sequential execution of subprocesses. Note that in (1), if transaction $\text{execute}(CS)$ fails and returns false, the transaction $\text{execute}(P)$ also fails and returns false. One can use hypothetical reasoning to have more fault-tolerant processes. For example, (10) redefines (1) such that if $\text{execute}(CS)$ fails, $\text{failed}(CS)$ will be executed instead and $\text{execute}(P)$ will be completed and return true. \sim in (10) denotes default negation and term $\sim \diamond \text{execute}(CS)$ draws if the hypothetical execution of $\text{execute}(CS)$ fails. Similarly we can redefine (5) as (11). This kind of reasoning can be useful in *exception handling*.

$$\begin{aligned} \text{execute}(P) \leftarrow & \text{complex_process}(P) \wedge \text{current_step}(P, CS) \otimes \\ & \sim \diamond \text{execute}(CS) \otimes \text{failed}(CS) \otimes \\ & \text{advance}(P, CS) \otimes \text{execute}(P). \end{aligned} \quad (10)$$

$$\begin{aligned} \text{execute}(P) \leftarrow & \text{primitive_process}(P) \otimes \sim \diamond \text{doit}(P) \otimes \\ & \text{result.insert}(P, \text{failure}). \end{aligned} \quad (11)$$

A sample implementation of this approach is available in our demo.

3 Example: A cell mitosis division process

Through a simple implementation of mitosis cell division process, we compared our \mathcal{TR} process representation technique with action modeling languages. We used Flora-2 [13], an object oriented knowledge base reasoning system, to develop an abstract biological knowledge base including mitosis cell division process. We compared our \mathcal{TR} -based system with those obtained by a manual translation of the same knowledge base to the \mathcal{AL}_d action modeling language [7]. We also compared our implementation with an implementation based on the event calculus concepts in Flora-2.

As shown in Figure 1, the comparison of systems in terms of lines of code shows that \mathcal{TR} provides a more succinct representation by far. Generating auxiliary rules for inertia axioms, completeness of states, and execution possibility, complicates \mathcal{AL}_d programs. We should also mention that the \mathcal{AL}_d program is including just one test query but \mathcal{TR} and the event calculus based solutions responds to 7 queries. This means that for the equal test conditions, the size of \mathcal{AL}_d program would be much more than 707 lines.

As shown in Figures 2 and 3, via a set of sample queries, we considered the response time of above mentioned implementations. The execution time also shows that, \mathcal{TR} is much faster than \mathcal{AL}_d . Moving to \mathcal{TR} from event calculus,

Method	Lines of Code
event calculus	1196
\mathcal{AL}_d	707
\mathcal{TR}	490

Fig. 1. Length of sample knowledge base system implemented using different methods

the overhead of transactional updates leads to decrease in the response time to test queries, which we are yet to understand. Our solution in \mathcal{TR} also suffered from a bug in XSB which prevented us from taking advantages of incremental tabling. Because of that we had to refresh several tables after each transactional update. Fixing that bug will improve \mathcal{TR} 's response time.

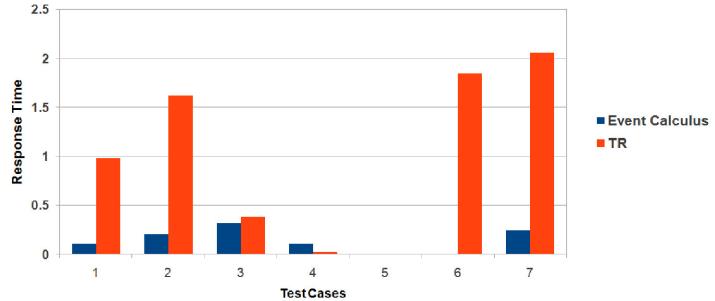


Fig. 2. Comparison of response time in event calculus and \mathcal{TR} for different test cases

This example apparently shows that \mathcal{TR} is promising candidate for representing processes in knowledge base systems.

There are other features in Flora-2, which we used in our development. Object orientated syntax and higher order rules are examples of these features. As those features are beyond the scope of this report, we do not consider them here.

4 Conclusion

In this paper, we discussed several methods for representing processes, which are included in knowledge representation systems as part of domain knowledge. As mentioned before, dynamic domain languages require a large number of auxiliary rules and axioms, which complicates knowledge representation. They also lack many features that facilitate knowledge representation and process specification such as higher order rules and object orientation. \mathcal{TR} allows definitions of processes as first class entities. Through an experiment, we showed that, it also simplifies programs and makes them more extensible and reusable. It also apparently improves the response time in comparison with methods based on action modeling languages.

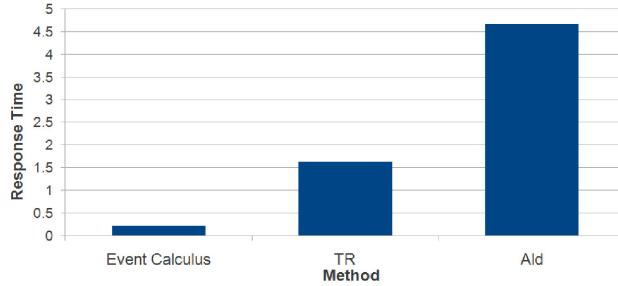


Fig. 3. Comparison of response time in different methods

We are planning to investigate \mathcal{TR} 's scalability in terms of size and complexity of process descriptions. Expansion of elementary updates to domain specific updates are useful. In addition, we are planning to consider other capabilities of \mathcal{TR} as a process representation tool. For example, we can study how \mathcal{TR} can represent concurrent behaviors. In this way, we should consider how \mathcal{TR} can encode other process specification conventions such as process algebra. For example, encoding process algebra's concepts and operational structural semantics in \mathcal{TR} would enable it to act as a theorem prover engine in the process algebra's domain.

References

1. McCarthy, J., Hayes, P.J.: Some Philosophical Problems from the Standpoint of Artificial Intelligence. In: Machine Intelligence. Volume 4. (1969) 463–502
2. Kowalski, R., Sergot, M.J.: A logic-based calculus of events. New Gen. Comput. **4** (January 1986) 67–95
3. F. van Harmelen, V.L., Porter, B.: Event Calculus. In: Handbook of Knowledge Representation. Elsevier (2007)
4. Baral, C., Gelfond, M. In: Reasoning agents in dynamic domains. Kluwer Academic Publishers, Norwell, MA, USA (2000) 257–279
5. Lin, F.: Embracing causality in specifying the indirect effects of actions. In: Proceedings of the 14th international joint conference on Artificial intelligence - Volume 2. IJCAI'95, San Francisco, CA, USA, Morgan Kaufmann Publishers Inc. (1995) 1985–1991
6. Gelfond, M., Inclezan, D.: Yet Another Modular Action Language. In: Proceedings of SEA-09, University of Bath Opus: Online Publications Store (2009) 64–78
7. Inclezan, D., Gelfond, M.: Representing Biological Processes in Modular Action Language ALM. In: Proceedings of the 2011 AAAI Spring Symposium on Formalizing Commonsense, AAAI Press (2011) 49–55
8. Baral, C., Chancellor, K., Tran, N., Tran, N., Joy, A., Berens, M.: A knowledge-based approach for representing and reasoning about signaling networks. Bioinformatics **20**(1) (January 2004) 15–22

9. Tran, N., Baral, C.: Reasoning about non-immediate triggers in biological networks. *Annals of Mathematics and Artificial Intelligence* **51**(2-4) (December 2007) 267–293
10. Gelfond, M., Lifschitz, V.: Representing action and change by logic programs. *Journal of Logic Programming* **17** (1993) 301–322
11. Lesprance, Y., Kelley, T.G., Mylopoulos, J., Yu, E.S.K.: Modeling dynamic domains with congolog. In: In Proceedings of the Eleventh Conference on Advanced Information Systems Engineering (CAiSE99) (Lecture Notes in Computer Science, Springer) (1999)
12. Bonner, A.J., Kifer, M.: An overview of transaction logic. *Theoretical Computer Science* **133** (1994)
13. : Flora-2 : Users Manual

First Approaches on Knowledge Representation of Elementary (Patent) Pragmatics

Shashishekhar Ramakrishna^{1,2}

¹ Freie Universität Berlin, Königin-Luise-Str. 24-26, 14195 Berlin, Germany,

² Teles PRI GmbH, Ernst-Reuter-Platz 8, 10587 Berlin, Germany,

s.ramakrishna@teles.de

Abstract. The focus of this article is to provide first approaches to a possible key solution representation and construction of legal norms, especially the national patent law norms. A semantic-system based on these approaches, complementary to the FSTP/IES-Expert system, would aim at (semi)-automatically translating the parts of the notion legal certainty from its natural language non procedural presentation to a declarative logical presentation by formal modeling through interpreting the pragmatics facts based within a National Legal Systems. This paper covers the initial abstract solutions and possible outcomes as gathered during the first year of PhD research³.

Keywords: Facts Screening and Transformation Processor (FSTP), Innovation Test, 35 U.S.C (§§ 112, 102/103, and 101)

1 Motivation

Current emerging technologies are mostly 'Model' based inventions i.e. intangible subject matter based. In general, an innovation claimed through its patent application, can be seen as a pair <claim, its claimed invention>, wherein the specification (including drawings) forms the second part of the pair. An inventive property/statement of an invention, disaggregated on levels of abstraction or on grains of mental resolution into elementarily properties henceforth referred as binary inventive concept. It provides the required degrees of separation of concerns for evaluating such properties independently in the light of its subject matter. Next to trivial elementary inventive concepts are logically error resistant as they represent a single/separated concern. The same holds for a non-inventive concept of a claimed inventions element, describing one of its non-inventive properties.

(Semi)-/Automatic evaluation by means of applying elementary pragmatics, 'EP' and National Patent Laws on such binary (non-)inventive concepts requires a semanticsystem for reasoning against the considered concepts, capable of the acquisition and processing of enormous amounts of background knowledge in a machine understandable format, keeping in mind its interdependence to

³ This Ph.D thesis is being supervised by Prof. Adrian Paschke, Freie Universität Berlin, Germany.

each other. Such a (sub)-system working in conjunction with the existing Facts Screening and Transformation Processor, FSTP [1]/Innovation Expert System, IES [2], enables a person of pertinent skill, who is needed for recognizing non-elementary pragmatics, to recognize automatically and/or guided interactively by the FSTP/-IES to consider whether such elementary properties of an innovation at issue (after its disaggregation) can be considered as Anticipate (A), Not-Anticipate (N) to its prior arts/considered reference set (RS).

2 Background - The Fact Screening and Transformation

As described in [1], [2] an innovation/creation over existing knowledge, provided as a reference set RS of prior art documents, is representable by a technique teaching, TT.0 which goes beyond the knowledge of the RS - just as in a patent/application. This compound of knowledge, representing an innovation, is called “PTR”, standing for a “pair of TT.0 and RS”.

The Innovation Expert System (IES) thus is the PTR Expert System, defined by the epistemological and practical requirements it meets: For any PTR to which it is applied, it is supporting its user in

1. deriving from it all technical and legal facts alias relations between TT.0 and a given RS respectively a given context, such as a given legal system (in the U.S e.g. to 35 U.S.C §§ 112, 102/103, and 101) and then
2. leveraging on this analysis instantly recognizing and answering any reasonable query for any such relation

The PTR Expert System (ES) is built around the PTRs “FSTP Test” [3], hence is also called FSTP ES. The FSTP Test of a PTR supports structuring of its PTR. This PTR-DS is disaggregated into three levels of knowledge representations (KR), “**o/AD/BID**”-KR. Wherein, **o** refers to “original”, **AD** to “Aggregated \wedge Disclosed”, and **BID** to “Binary \wedge Independent \wedge Disclosed”. IES supports, initially screening its documents/technical teachings for elementary building blocks of its creativity/inventivity, i.e. for its inventive and non-inventive concepts. Technical informal inventive and non inventive concepts/properties are then transformed into technical formal inventive concepts/facts, then transforming those into the technical primary facts, and finally transforming them into the technical secondary facts, called **basic** resp. **semantic** (alias **creative**) resp. **textbfpragmatic** (alias **innovative**) facts. These technical secondary facts use metrics induced by the Highest Courts precedents on creativity/innovation by their numbers of BID-inventive concepts embodied by TT.0. From these BID-inventive concepts, the classical yes/no answer to the question, whether TT.0 is indicated obvious over RS, can be derived by this metric. The semantic/creative and pragmatic/innovative facts extend this metric much further by first defining a PTR plcs specific (plcs = patent law carrying semantic) innovation geometry, which depicts the plcs-height/-creativity of its TT.0 over its RS. Based on plcs-height/-creativity, TT.0s pragmat-ic/innovative height over RS additionally takes into account the PTRs pmgp (pmgp= patent monopoly

granting pragmatics) in any National Patent System (NPS) which represents the national/socio/economic principles underlying the idea of rewarding an innovation by granting a 20 years monopoly to its TT.0.

3 Goals/Aim

The object of our concern in this thesis is to create a semantic-system, capable of (semi-) automatically translating the parts of the notion “legal certainty” such as patent laws (e.g. in U.S, 35 U.S.C §§ 112, 102/103, and 101) from its natural language non procedural presentation to a declarative logical presentation by formal modeling through interpreting pmgp based on NLS/ (NNI = National Normative judicial Interpretation of facts).

Figure 1, shows few (10+) basic tests as proposed in [4] enabled by its inventive concepts, automatically prompting their user through exploratively checking its meeting the requirements as stated by few NPS'es (e.g.: 35 USC 112, 102/103, and 101). Applying these tests to inventive concepts requires the requirements of the NPS'es to be modeled into declarative rules, due to their modular feature and their capability to use the same knowledge in many different ways. Modeled rules are used in deductive (non-monotonic) reasoning for legal interpretations. NPS'es, like complex computer systems, constantly face questions that aim to ascertain the state of things or the correctness of a certain contention, like these modeled rules/tests. Hence, the legal questions regarding which of a number of modeled/competing legal rules could apply in a given situation amount to some error and inconsistencies, thereby leading to inconsistent reasoned output/legal interpretations. One such non-trivial approach would be that such modeled rulebases are updated manually/guided by the system using inductive learning techniques (applying rules on case laws). Such an approach would be a long term goal and is not considered for the current use-cases shown in this thesis. The list below re- expresses the intended aim, providing possible approaches/solutions to the research question stated in Section 4 of this thesis in detail:

1. To manually(and/guided-by-system) analyze and extract the rules and ontological concepts described in the natural language descriptions of NPS'es.
2. To identify the required semantics and inference rules needed for legal reasoning with NPS'es and for the legal interpretation enabling the separating of novel innovations from obvious steps.
3. Logic-based declarative representation of these chains of complex rules for legal reasoning on top of structured formal ontology domains representing the conceptualization of the NPS'es and the underlying domains of skill and elementary pragmatics.
4. Developing a legal reasoning sub-system to the FSTP ES which allows pmgp dependent information to be derived from the NPS knowledge bases and to be used in the FSTP for semi-automated legal decision support and compliance checks with the applicable NPS for a PTR. This includes
 - (a) Address the trade-off between required expressiveness of the knowledge representation and its computational complexity of the legal reasoning in FSTP.

- (b) Provide support for the different roles involved, such as inventor, person of pertinent skill, examiner, patent agent etc. This requires different representation languages from natural language format for expressing questions, answers, proofs and explanations to platform-independent serializations in XML and Semantic Web formats to platform-specific executable formats on the logical reasoning layer.
- (c) Provide support for life cycle management of knowledge. This addresses e.g., collaborative knowledge engineering and management (versioning, different roles such as author, maintenance), updates in the NPSes by new decisions which lead to corresponding isomorphic updates in the NPSes knowledge bases, integration of internal and external (semantic) background knowledge e.g. about skill, elementary pragmatics, usage data (annotations, proofs, etc.).

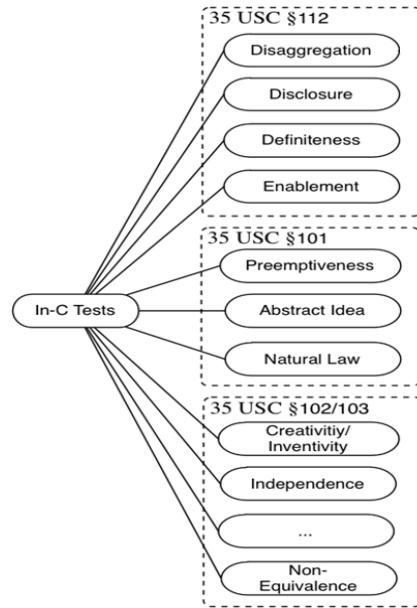


Fig. 1. 10+ In-C tests to be applied on an inventive concept for patent eligibility, in accordance to US patent law

4 Research Questions

The research question will be refined and detailed after the literature review and baseline study, from the following general problem domains of a knowledge representation.

1. Syntax:
 - (a) Which representation and interchange format for the representation of the knowledge on different representation layers? (human-oriented computational independent, platform-independent supporting integration and interchange, platform-specific logical reasoning).
2. Semantics:
 - (a) Which inference and interpretation semantics (non-monotonic vs. monotonic, expressiveness vs. computational complexity, closed-world vs. open world, “ontologies vs./and rules”,)
3. Association problem:
 - (a) How to connect the formal representation with the real-world resources and norms?

Requirements derived from these knowledge representation problem domains can be distinguished according to functional requirements for the concrete knowledge representation and non-functional requirements during design time (development / engineering of the knowledge) and run time (use of the knowledge).

1. Functional Requirements:
 - (a) e.g., expressiveness, ...
2. Non functional requirements at design time:
 - (a) e.g., composability and extensibility, interoperability, declarative implementability, modifiability and evolvability, reusability and interchangeability,...
3. Non functional properties at runtime:
 - (a) e.g., usability, understandability and explanation, correctness and quality, scalability and efficiency, safety and information hiding (need-to-know principle),...

5 Proposed Approach

An abstract model of the system envisioned as a solution to the problem can be seen in Figure 2. An existing state-of-the-art prior art search module, using a semantic search engines like, Cognition [5], DeepDyve, etc retrieves patents through large databases which forms the required RS (if previously not specified by the jury) for the TT.0. Thus formed PTR-DS will be transformed from their natural language texts into some standard representation formats like XML, using text-mining, semantic recognition and annotation techniques supporting human knowledge engineers in the fact screening and transformation process. Similar to the PTR-DS, the existing patent rules from NPS have to be transformed from their natural language format to more standardized rule representation formats. We propose to use LegalRuleML [6], an XML standard for legal knowledge representation based on RuleML [7] which supports the modeling of norms.

Parallelly, we map patent norms as used in landmark case law decisions to a workflow using some configurable workflow model. Where, each node on the

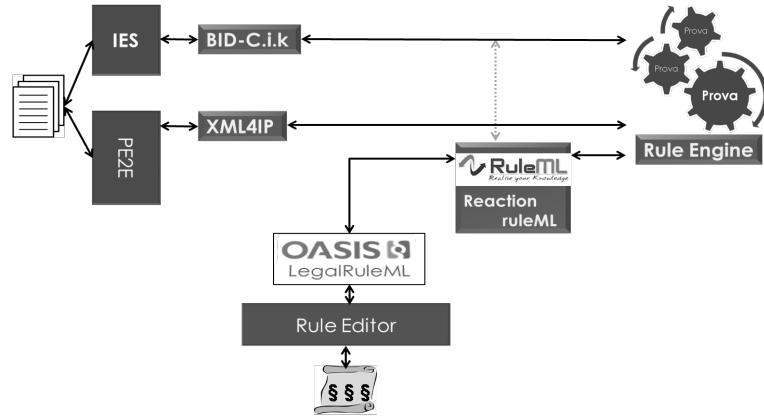


Fig. 2. Cognitive system (abstract model)

workflow (B-tree) represents a complex legal rule represented using the ReactionRuleML [8] representation format. This resolves complex legal questions and automates the analysis of a large number of patent norms with respect to their logical coherence in a given NPS. The workflow itself is represented using LegalRuleML, which provides the functionalities like reusability, lifecycle management of nodes or the entire workflow to capture the changes over time of the rules when the legal binding text changes.

LegalRuleML is also used to point out logical inconsistencies in current case law decisions and can also be used to evaluate the compliance of semantic facts with case law and positive law. Thereby, providing a powerful and declarative way to control and reuse such semantically linked meanings with the help of independent micro-ontologies about the NPSs and domain specific pragmatic contexts (skill ontologies, elementary pragmatics, standards etc.) for flexible processing and legal reasoning. The required (patent) rules/constraints are built by the rule creator module, which uses a distributed rule inference services network like Prova [9], a java based open source rule language for reactive agents and event processing.

Figure 3 shows the process of generating a generic workflow pertaining legal rule from landmark decisions' specific workflow. This allows capturing the different interpretations of the same law on different use cases and, thereby, arriving at a generalized workflow.

6 Elementary Pragmatics

Elementary Pragmatics are disclosures (explicit/implicit) of certain art which can be easily understood by a person of pertinent skill. According to certain National Patent Systems, an EP must not be just claimed to exist, but must be documented in an enabling way.

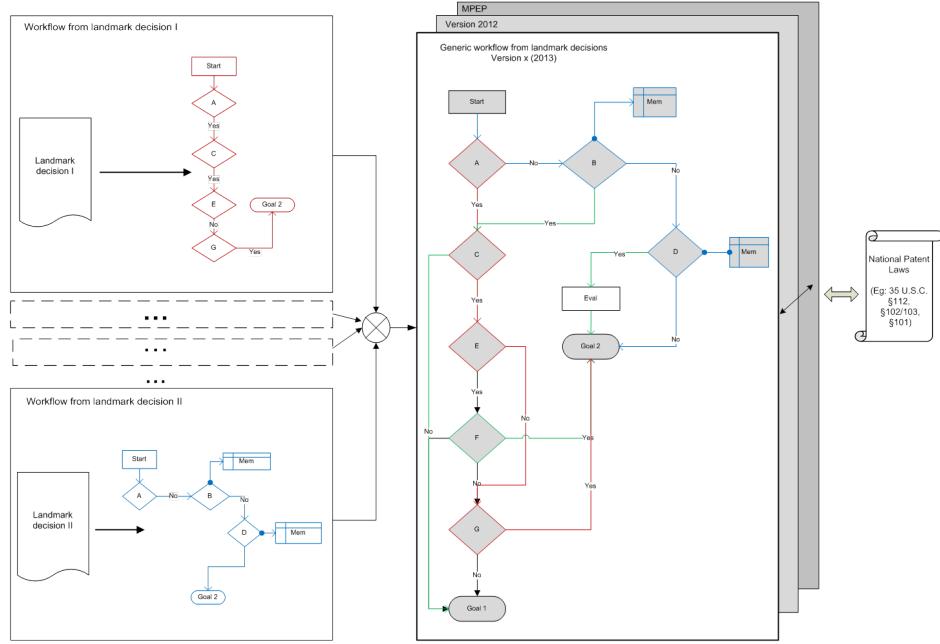


Fig. 3. specific workflow to generic workflow

EP can be divided into 4 types as shown in Figure 4 :

1. EP from Formal Rules for Filing, EP-PFP
2. EP from Patentability Conditions, EP-P
3. EP from Post Grant Procedures, EP-PGP
4. EP from Litigation, EP-L

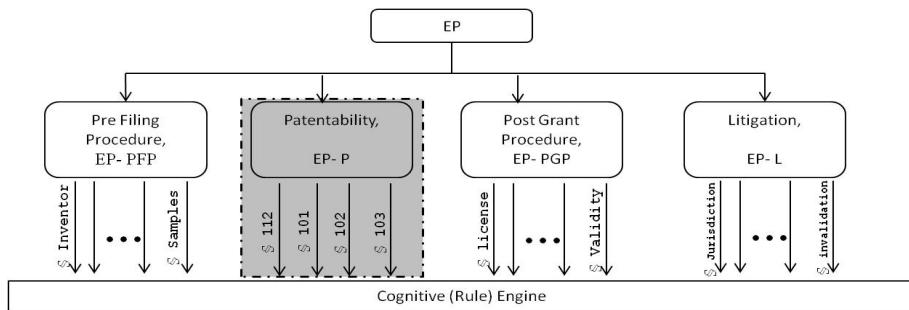


Fig. 4. Classification of EP in a National Patent System.

We narrow down our focus on Elementary Pragmatics from Patentability Conditions, EP-P. Specifically on four paragraphs (35 USC §§ 112, 102/103, and 101) of the U.S patent system [10]. Figure 5 shows the general evaluation procedure for an inventive concept under patentability. A set of inventive concepts is patent eligible, 'pe' if and only if it satisfies all the patentability criterias or EP- P's.

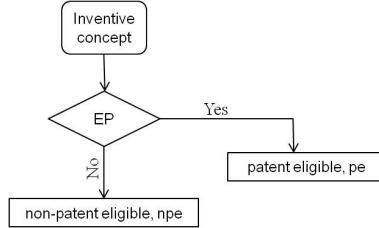


Fig. 5. Evaluation procedure.

7 Proposed Framework

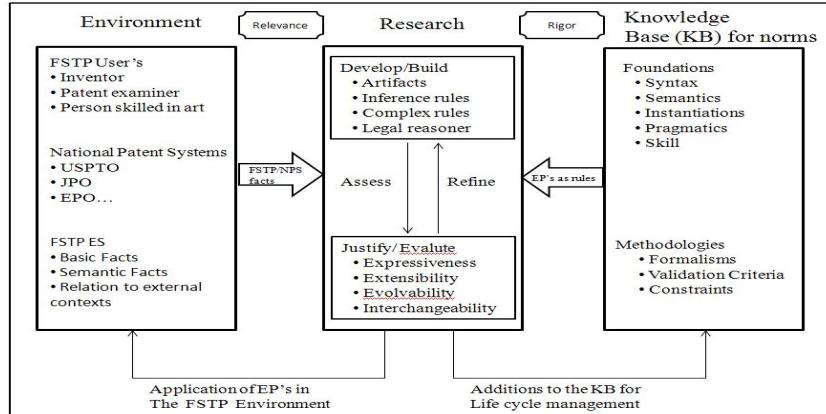


Fig. 6. Proposed Framework.

We propose a legal information system framework [11] as shown in Figure 6. The proposed framework is based on a general information system research framework [12]. The central 'Research' module is fed with information from both 'Environment' and 'Knowledge Base (KB)' modules. Information/raw material such as FSTP facts, which include the norms from various NPS'es, are fed by the 'Environment' module and the syntax, semantics, pragmatics and instantiations

encompassing a norm are fed by the 'Knowledge Base' module. The central research module works towards building the inference rules required for the legal reasoner. The 'develop/build' sub-module including legal reasoner is evaluated for the norms expressiveness, extensibility and interoperability criteria's. Based on the results, the rules and the reasoner are refined again. This iterative process of (re-)assessing and refining is completed when all criteria are effectively evaluated. Processed information is fed back to the environment module for its actual usage within the FSTP ES. Additional information for the lifecycle management of a norm and its contexts is sent back to KB module.

7.1 Methodologies

Elaborating on the process of mapping patent norms to workflow models we start with the representation of all landmark case-law decisions concerning a specific norm onto a workflow. Wherein, LegalRuleML is used for such representation. We reiterate the formal template of LegalRuleML showing the most general representation syntax as defined in [13].

```
<lrml:LegalRuleML>
  <!-- Referencing the textual provisions-->
  <lrml:LegalSource> ... </lrml:LegalSource>

  <!-- Capturing the ex-ternal temporal dimensions of the rules are represented -->
  <lrml:Timeinstants> ...</lrml:Timeinstants>
  <lrml:TemporalCharacteristics> ...</lrml:TemporalCharacteristics>

  <!-- Agent and the authority of the rules for provenance-->
  <lrml:Agents> ... </lrml:Agents>
  <lrml:Authorities> ... </lrml:Authorities>

  <!-- Associates property values to rules and also adds metadata such as jurisdiction, role, and strength-->
  <lrml:Context>
    <lrml:appliesRole>
      <lrml:appliesStrength>
        <lrml:appliesAuthority>...</lrml:appliesAuthority>
        <lrml:appliesJurisdiction>..</lrml:appliesJurisdiction>
      </lrml:appliesStrength>
    </lrml:appliesRole>
  </lrml:Context>

  <!-- Rules (constitutive and prescriptive) are modelled-->
  <lrml:Statements>
    <Rule>
      .
      <!-- ReactionRuleML-->
      ..
    </Rule>
  </lrml:Statements>
</lrml:LegalRuleML>
```

Listing 1.1. General LegalRuleML syntax

Each node on the workflow represents a disaggregated rule/norm. Such rule or set of rules are embedded inside **< lrml: Statements >** using ReactionRuleML. Listing 1.2 shows a formal template of ReactionRuleML, showing the most general representation syntax as defined in [8].

```
<Rule>
  <!--rule info and life cycle management, modularization-->
  <!--(semantic) metadata of the rule-->
  <meta> ... </meta>
  <!--scope of the rule e.g. a rule module-->
  <scope> ...</scope>
```

```

<!--rule interface description-->
<!--intended semantic profiles-->
<evaluation> ... </evaluation>
<!--rule interface signature and modes-->
<signature> ... </signature>

<!--rule implementation-->
<!--qualifying declarations, e.g. priorities, validity-->
<qualification> ... </qualification>
<!--quantifying declarations, e.g. variable bindings -->
<quantification> ... </quantification>
<!--event part-->
<on> ... </on>
<!--condition part-->
<if> ... </if>
<!--(logical) conclusion part-->
<then> ... </then>
<!-- action part -->
<do> ... </do>
<!--postcondition e.g. to check effects of execution-->
<after> ... </after>
<!--(logical) else conclusion-->
<else> ... </else>
<!--alternative action, e.g. exception handling-->
<elsedo> ... </elsedo>

</Rule>

```

Listing 1.2. General ReactionRuleML syntax

7.2 Develop/Build (Legal Reasoner)

Disaggregated patent rules are wrapped to form a reactive agent as shown in Figure 7. A generic module “semantic- interface” here is used to depict the need for hybrid reasoning due to the fuzzy nature of patent rules. ReactionRuleML messaging is used for distributed reasoning. Listing 1.3 shows general message syntax.

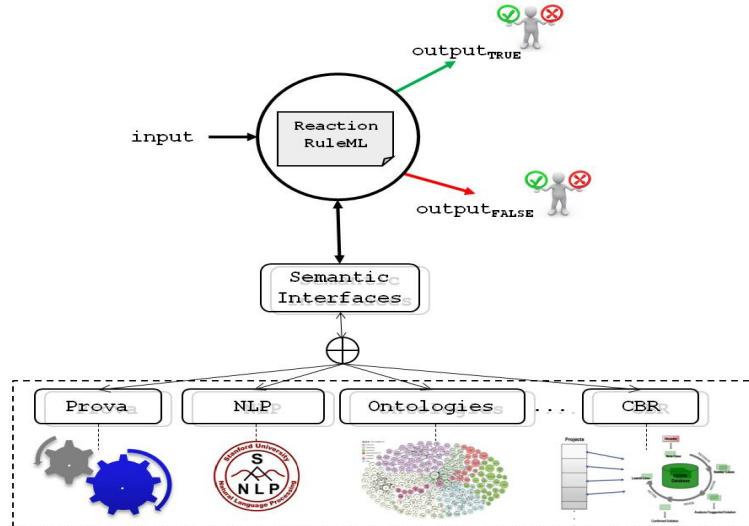


Fig. 7. Reactive agent with hybrid reasoning.

```

<Message directive="PRAGMATIC_CONTEXT" >
  <oid> <!-- conversation ID--> </oid>
  <protocol> <!-- transport protocol --> </protocol>
  <sender> <!-- sender agent/service --> </sender>
  <receiver> <!-- receiver agent/service --> </receiver>
  <content> <!-- message payload --> </content>
</Message>

```

Listing 1.3. ReactionRuleML Messaging syntax

8 Examples

8.1 35 U.S.C § 112 6th paragraph

Consider the latest Court of Appeals of Federal Circuit (CAFC) decision on § 112 6th paragraph in Lighting Ballast Control LLC v. Philips Electronics and Universal Lighting Technologies, Inc' [14]. Under this decision the court re-explained the norms within the 6th paragraph of § 112 (35. U.S.C Patent law). For our analysis, we map the decision and its citations into workflow. Figure 8[a] shows some excerpts from the decision itself. Figure 8[b] shows the workflow mapped from the decision for the analysis of "Means-Plus-Functions-Claiming" In its decision, the CAFC with the help of citations explains how other factors influencing this decision have to be handled.

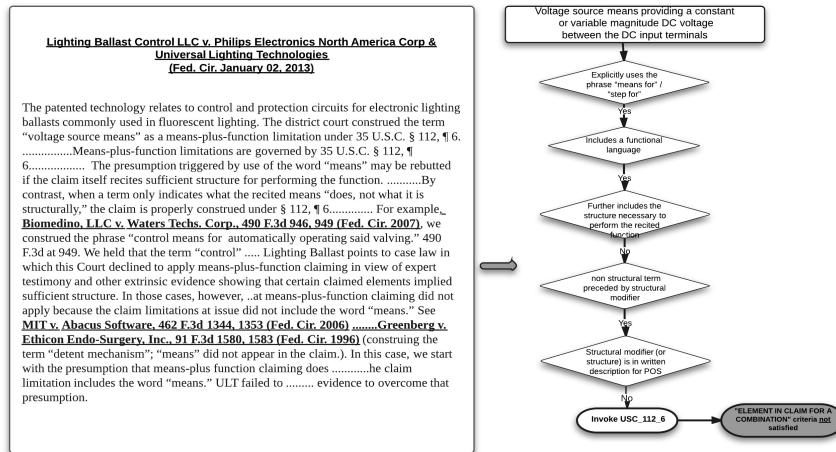


Fig. 8. CAFC decision 'Lighting Ballast v Philips Electron' [14] mapped to a workflow.

We propose to use inductive approaches by populating the workflow with related decisions ('Biomedino LLC v. Water Techs Corp' [15], 'MIT v. Abacus Software' [16], 'Greenberg v. Ethicon Endo Surgery, Inc' [17] etc) to obtain a

generic workflow for 6th paragraph of § 112 as shown in Figure 9. Where, the norm in workflow format is represented using LegalruleML representation format described before and the nodes/rules are represented using ReactionRuleML. For this example we use Stanford parser, SentiwordNet, Prova and Pre defined legal lexicons, PUBPAT as semantic interface for reasoning the norms. A person of pertinent ordinary skill and creativity (posc) as defined by USSC affirms every reasoned result.

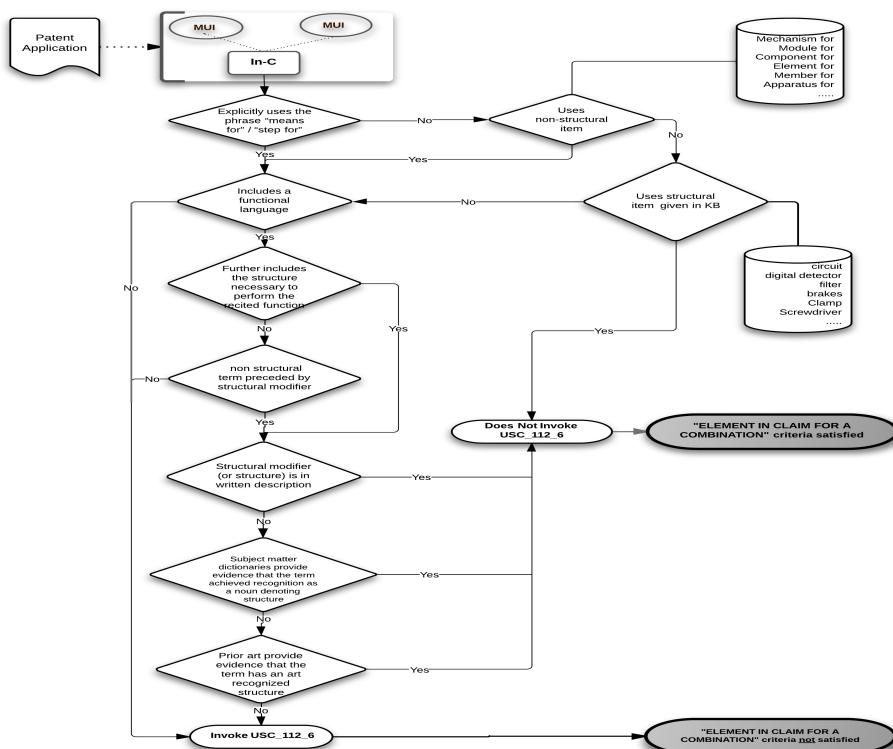


Fig. 9. 6th paragraph of § 112 (35. U.S.C Patent law)..

9 Conclusion and Future Steps

The solution to have a sub-system, based on configurable EP which connects the FSTP ES, thus making it full-/semi automatized in handling queries pertaining to EP and NLS thereby, providing a uniform platform for standardizing the generation and representation of complex rules (built using fewer NPS goal clauses/(patent) rules. Such a system would serve as a ready reckoner in drawing legal conclusions on top of scientific fact determined during FSTP analysis.

This would then help in applying the (elementary) cognitive norms required for interpretation and evaluation of such identified facts.

10 Acknowledgements

The author would like to thank Prof. Adrian Paschke and Prof. Sigram Schindler for their constructive comments and suggestions. This work has been partially supported by the Fact Screening and Transformation Project (FSTP) funded by the TelesPRI GmbH: www.fstp-expert-system.com.

References

1. Schindler S.: *THE FSTP EXPERT SYSTEM*, WO 2012/022612 A1 (to Sigram Schindler Beteiligungsgesellschaft mbh [DE/DE]; Ernst-Reuter-Platz 8 10587 Berlin (DE)), 23 February 2012.
2. Schindler S.: *INNOVATION EXPERT SYSTEM, IES, AND ITS PTR DATA STRUCTURE, PTR-DS*(to Sigram Schindler Beteiligungsgesellschaft mbh [DE/DE]; Ernst-Reuter-Platz 8 10587 Berlin (DE)), May 2013.
3. Ramakrishna S., Karam N., Paschke A.: *The FSTP Test: a novel approach for an invention's non-obviousness analysis*, JURIX 2012, pp 129-132, 2012.
4. *Brief of Amicus Curiae Sigram Schindler Beteiligungsgesellschaft mbH in support of neither party*, LIGHTING BALLAST v. PHILIPS ELECTRON (2008), No. 2012-1014 (USSC, pending), 28 May 2013.
5. Dahlgren K.: *Technical Overview of Cognition s Semantic NLP (as Applied to Search)*, ReCALL, pp. 120, 2007.
6. Palmirani M., Governatori G., Rotolo A., Tabet S., Boley H., and Paschke A.:*LegalRuleML: XML-Based Rules and Norms*, in Rule - Based Modeling and Computing on the Semantic Web, vol. 7018, Springer Berlin Heidelberg, 2011, pp. 298-312.
7. Boley H., Paschke A., and Shafiq O.:*RuleML 1.0: The Overarching Specification of Web Rules*, in RuleML, 2010, pp. 162-178.
8. Paschke A., Boley H., Zhao Z., Teymourian K., and Athan T.:*Reaction RuleML 1.0: Standardized Semantic Reaction Rules*, in RuleML 2012, 2012.
9. Kozlenkov A.:*Prova Rule Language Version 3.0 User's Guide*, Internet: <http://prova.ws/index.html>, 2010.
10. Title 35 of the United States Code. 1952.
11. Ramakrishna S.: *Cognitive System for Knowledge Representation of Elementary Pragmatics*, in Proceedings of the RuleML2012 Challenge, at the 6th International Symposium on Rules, 2012, online ceur-ws.org/Vol-874/paper2.pdf.
12. Hevner A R., March S T., Park J., and Ram S.:*Design science in information systems research*, MIS Quarterly, 2004.
13. Athan T., Boley H., Governatori G., Palmirani M., Paschke A., and Wyner A.:*OASIS LegalRuleML*, in Proceedings of 14th International Conference on Artificial Intelligence and Law (ICAIL 2013), 2013.
14. Lighting Ballast Control LLC v. Philips Electronics and Universal Lighting Technologies, Inc No. 2012-1014 (Fed. Cir. January 2, 2013).
15. Biomedino, LLC v. Waters Techs. Corp., 490 F.3d 946, 949 (Fed. Cir. 2007).
16. MIT v. Abacus Software, 462 F.3d 1344, 1353 (Fed. Cir. 2006).
17. Greenberg v. Ethicon Endo-Surgery, Inc., 91 F.3d 1580, 1583 (Fed. Cir. 1996).

RAWE: A Web Editor for Rule Markup in LegalRuleML

Monica Palmirani¹, Luca Cervone¹ Octavian Bujor¹, Marco Chiappetta¹

¹CIRSFID, University of Bologna.
{monica.palmirani, luca.cervone, octavian.bujor, marco.chiappetta}@unibo.it

Abstract. This paper presents a Web editor (RAWE: Rules Advanced Web Editor) for marking up legal rules starting from legally binding texts. The Web editor exploits the legal information embedded in the Akoma Ntoso markup, in combination with XML techniques, so as to help the legal-knowledge engineer model legal rules and convert them into LegalRuleML, an OASIS XML standard candidate.

Keywords: Legal Reasoning, Akoma Ntoso, LKIF-core, LegalRuleML.

1. Introduction

This paper presents a Web editor for marking up legal texts in a legal document's main structure, normative references, and legal metadata using the Akoma Ntoso [2] [13] [25] XML standard, now undergoing the OASIS standardization process. The same Web editor exploits the legal information embedded in legal markup, in combination with XML techniques, to help the legal-knowledge engineer model legal rules using a logic formalism and convert them into LegalRuleML [1] [23][24], another OASIS XML standard candidate. The two standards—Akoma Ntoso and LegalRuleML—are complementary in implementing the legal-knowledge modelling and representation of legal documents. The main goal of the RAWE Web editor is to provide a tool capable of managing in an integrated way the advantages of the Akoma Ntoso and of LegalRuleML, applying the isomorphism principle [3][9][22] to connect, as far as possible, legally binding textual provisions with the logic formalism expressed using rules. Usually, AI&Law experts are too focused on the task of applying a logic formalism to achieve isomorphism, but the legal experts (judges, lawyers, and administrators) are interested in verifying the results of the legal reasoning engine and in finding evidence in the legally binding text.

Secondly, a legal text changes over time, and so the rules need to be updated accordingly. If the isomorphism principle is not applied, it is quite difficult to determine whether those rules need to be updated. The RAWE editor helps to maintain text and rules aligned and to minimize manual markup activity.

Thirdly, the aim of the RAWE is to show how it is possible to export LegalRuleML in RDF serialization to favour Linked Open Data interoperability. Finally, in the future, the same editor will export LegalRuleML files in other

proprietary languages, like SPINdle [15] or Carneades [8][11], so as to permit legal reasoning.

2. From Open Text to Open Rules

The first point to be made in clarifying the goals RAWE would like to achieve is to draw a distinction among three conceptual layers: norms (abstract mandatory commands concerning rights or duties), textual provisions (sequences of texts), and rules (rendering of the text into logical rules).

A **norm**, following Kelsen's definition [14], is an abstract mandatory command concerning rights or duties. A norm is usually expressed in writing using legal texts or in an oral way (e.g., a social norm, an oral contract) or in other representations (e.g., symbolic road signs).

Textual provisions (or simply *provisions*) are the instantiation of general norms in one possible textual representation (a sentence, article, or paragraph).

Legal rules are interpretations of one or more provisions formalized using logical rules in the form of antecedent and consequent. Sometimes several provisions will form a single rule, or a single provision may include multiple rules.

Usually, in the state of the art, AI&Law scholars focus their attention only on the rule modelling and on the foundational logical theory, and apart from the isomorphism principle [3], the connection with the text over time and the ontology aspects have been neglected. There is an important theoretical debate in the AI&Law community on the interpretation of the legal textual provisions expressed in natural language and on the canonization of rules using logical formalisms [4]. The prevalent theory is now oriented towards *hybrid interpretation* [27] (rather than pure *textualism*, or pure *interpretation*). We want to make visible in the text the “evidence” that there is a minimal but reasonable interconnection, following the legal theory of interpretation, with a logical rule in a formal representation. This exercise sometimes forces the legal-knowledge expert to split the original provision into two or more rules, or to duplicate the rules, or to compress several sentences into a single rule. In this scenario, we have to manage an N:M relationship among norms, textual provisions, and the ontology that we want to capture and represent maintaining a strong separation among these three levels.

Nevertheless, it is obvious that the isomorphism approach alone presents some exceptions and limitations that need to be balanced in a reasonable way. We have at least three cases where the legal rules have no textual link: (i) when we have implicit rules deriving from the general principles of the legal system (e.g., *lex superior*, *lex specialis*, *lex posterior*); (ii) when the legal-knowledge engineer includes a personal interpretation as a summary of his/her expertise; and (iii) when the legal-reasoning engine produces rules. In these cases the Web editor provides metadata to distinguish those rules deriving from the legal text from those that are a free interpretation of the rules' author. The RAWE editor permits multiple interpretations of the same legal text and makes it possible to follow the isomorphism principle, but also to derogate from it if need be.

Finally, the Web editor exports all the metadata in RDF format to favour the interconnection of Legal Open Data with Linked Open Data. The goal is to release RDF triples about the rule knowledge base, in such a way as to connect that with other datasets available in the Linked Open Data Cloud. This permits more-effective filters of the legal resources in the Semantic Web domain (e.g., geo-localizing legal resources on the map using the jurisdiction and the temporal metadata filter to find the legal rules relevant to a given context, such as environment law or construction law).

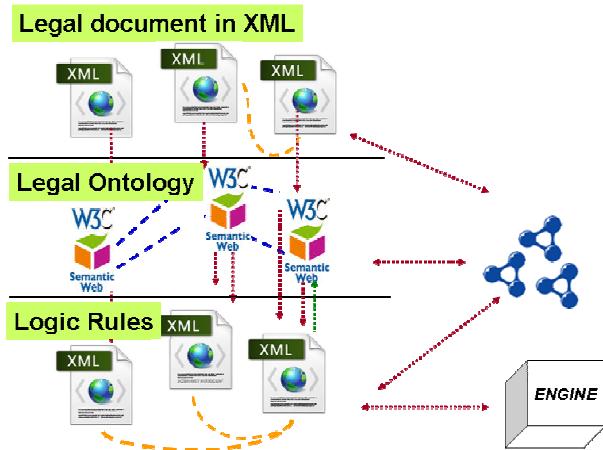


Fig. 1 – Scenario of relationships among different layers in legal knowledge modelling

3. Related Work

The AI&Law community itself [27] has spent the last two decades modelling legal norms using different logics and formalisms, usually fed manually to a legal-reasoning engine. Some visual tools [11] or editors [15][19] in the past have been developed to model rules, but the methodology used starts from a reinterpretation of the legal source text by a legal-knowledge engineer who extracts the norms, applies models and theory using a logic representation, and finally represents them with a particular formalism. The RAWE approach is different: it starts from the legal text marked up in some Legal XML standard and, exploiting the text's regularity, detects some metadata that is also useful for modelling rules.

Over the last decade, several Legal XML standards have arisen for describing legal texts (Akoma Ntoso, CEN Metalex [5] [16]) and rules (RuleML, RIF, SWRL, etc.), but the two communities are mostly separated, and they pursue their goal separately. In the meantime, the Semantic Web, and in particular legal ontology research, combined with the NLP extraction of semantics, has given a great impulse to the modelling of legal concepts [17][18][20][7][6][12][26]. In this scenario there is an urgent need to close the gap between the text description, represented using XML

techniques, and the norms formalized with logical rules, this in order to realize an integrated and self-contained representation. There are three main reasons:

- Legal knowledge is currently presented in a disjointed way in the original text that inspired the logical modelling. This disconnection between legal-document management and the logical representation of the embedded rules strongly affects the real usage of the legal-document knowledge in favour of citizens, public administrations, and businesses (e.g., contracts, insurance regulation, banking soft law).
- Management of changes undergone over time by legal documents—especially acts, regulations, and contracts—that by nature are variable and subject to frequent modifications, significantly affecting the coordination between the text and the rules that should be remodelled.
- The legal validity of the text as authentically approved by the competent entities (e.g., contractors) should be preserved across all manipulations. On the other hand, it is important to connect legal document resources, which themselves include many legality values (e.g., authenticity, integrity, evidence in trial, written form, etc.), with the multiple interpretations coming from legal-knowledge modelling.

Certainly, one of the main challenges over the last five years has been to acquire the ability to capture, with the help of NLP techniques, all the relevant legal knowledge embedded in a legal document and to represent it in an appropriate formal model. However, there hasn't been significant progress on the state of the art in this respect, especially in languages other than English. So it is important to improve the user interface technique to help the legal-knowledge expert to easily model legal rules and prepare an environment for a future NLP integration. RAWE is the only Web editor in the state of the art that can model legal texts and rules in a coordinated and consistent way using a WYSIWYG interface exploiting two important legal XML standards: Akoma Ntoso and LegalRuleML.

4. Akoma Ntoso and LegalRuleML Synergy

As mentioned before, Akoma Ntoso and LegalRuleML are two XML standards for modelling and representing legal documents. RAWE can coordinate the knowledge captured with these two standards so as to help the end user mark up the legal rules using a logic formalism enriched with temporal parameters.

Akoma Ntoso is specifically designed to model a legal document's structure and legal metadata, like the preface, preamble, sections, conclusions, normative references, dates, and signatures. The Akoma Ntoso metadata block additionally defines the conditions under which the legal textual fragment is valid, effective and in force, while also defining jurisdiction, the document's authority, and other relevant legal metadata, like modifications. All those metadata are also significant in defining the context of a legal rule, helping the legal reasoning engine filter the rules pertinent to a particular case (e.g., infringement of the rule at a given date in 1999).

The following example displays as `<temporalGroup id="t5">` the block that defines the interval of efficacy and enforceability of Section 504 of the US Code.

```

<akomaNtoso>
  <act name="act">
    <meta>
      ... meta data about the legal document ...
    </meta>
    <coverPage>
      Cover page content
    </coverPage>
    <preface>
      ... the preface of the document ...
    </preface>
    <preamble>
      ... the preamble of the document ...
    </preamble>
    <body>
      <clause id="tit17-chp5-sec504-clsc" period="#t5">
        ... the normative part of the document ...
      </clause>
    </body>
  </act>
</akomaNtoso>

```

LegalRuleML is designed to model in logical formalism the norms expressed in a legal text. It does so especially using deontic operators: obligation, right, permission, prohibition. LegalRuleML is also intended to define the context for each rule by providing a set of metadata like the temporal parameters, the original textual sources, the jurisdiction, the author, and the authority of the rules. The fragment below shows the main structure of a LegalRuleML document composed of different metadata blocks defining the author who modelled the text into rules (<lrml:Agents>), recording the original legal resources IRI (<lrml:References>), and providing the temporal parameters (<lrml: TemporalCharacteristics>), the context, and each rule's date of creation (<lrml:Context key="ruleInfo1" hasCreationDate ="#t8">). The <lrml:Contex> provides the environment in which the rules are valid (time, author, jurisdiction, etc.).

<pre> <lrml:LegalRuleML> <lrml:Agents> <lrml:Agent key="aut1" sameAs="&unibo;/person.owl#m.palmirani"/> </lrml:Agents> <lrml:References> <lrml:Reference refersTo="ref2" refID="/us/USCode/eng@/main#tit17-sec504-clsc-pnt1" refIDSysName="AkomaNtoso2.0-2012-10"/> </lrml:References> <lrml:TimeInstants> <ruleml:Time key="t6"> <ruleml:Data xsi:type="xs:dateTime">1999-12- 09T00:00:00.0Z</ruleml:Data> </ruleml:Time> </lrml:TimeInstants> <lrml:TemporalCharacteristics key="tblock1"> <lrml:TemporalCharacteristic key="e2-e"> </pre>	general definition of Agent with value an uri to m.palmirani
	definition of legal text fragment
	definition of instant time
	definition of intervals and

<pre> <lrml:forRuleStatus iri="#&lrmlv;#Efficacious"/> <lrml:hasStatusDevelopment iri="#&lrmlv;#Ends"/> <lrml:atTimeInstant keyref="#t6"/> </lrml:TemporalCharacteristic> </lrml:TemporalCharacteristics> <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:appliesSource keyref="#ref2"/> <lrml:toStatement keyref="#rule1"/> </lrml:Context> <lrml:hasStatements key="rulebase-v2"> <lrml:ConstitutiveStatement key="rule1"> <ruleml:if> ...</ruleml:if> <ruleml:then>... </ruleml:then> </lrml:ConstitutiveStatement> </lrml:hasStatements>... </lrml:LegalRuleML></pre>	situations definition of the rule context rule base block
---	--

Entering all the `<lrml:Context>` information manually for each rule is a really time-consuming task, especially when the legal text has gone through several modifications over time. Moreover, it is difficult to maintain consistency between legal textual provisions and rules in the dynamicity of the legal system. For this reason the RAWE Web editor exploits the information embedded in the Akoma Ntoso text proposition (e.g., section, article), and it reuses those data to define the context of the rules when accurately connected to the legal provision.

The following example presents a fragment of Section 504 of the US Code concerning copyright infringement and the related rules. Section 504 is presented in the version updated at time t5, which in Akoma Ntoso is defined in the `<temporalGroup id="t5">` block.

When the end-user selects a portion of the legal text with the mouse in the Web editor window, all the related metadata recorded in Akoma Ntoso are detected and exported in LegalRuleML to model the rules.

The following example shows in the two standards (i) the correspondence among temporal events; (ii) the correspondence among temporal intervals; and (iii) how it is possible to reuse the Akoma Ntoso information in LegalRuleML (compact form).

i) Event definition in the Akoma Ntoso metadata block	Event definition in LegalRuleML, automatically extracted from the Akoma Ntoso text using mouse-over
	<code><lrml:TimeInstants></code>

<pre><eventRef source="#rp5" id="e6" type="amendment" date="1999-12- 09"/></pre>	<pre><ruleml:Time key="t6"> <ruleml:Data xsi:type="xs:dateTime">1999-12- 09T00:00:00.0Z</ruleml:Data> </ruleml:Time> </lrml:TimeInstants></pre>
Intervals definition in Akoma Ntoso in the metadata block <pre> <temporalData source="#palmirani"> <temporalGroup id="t5"> <timeInterval refersTo="#inforce" start="e6"/> <timeInterval refersTo="#efficacy" start="e6"/> </temporalGroup> </temporalData></pre>	Intervals definition in LegalRuleML <pre> <lrml:TemporalCharacteristics key="tblock1"> <lrml:TemporalCharacteristic key="e2-e"> <lrml:forRuleStatus iri="#lrmlv;#Efficacious"/> <lrml:hasStatusDevelopment iri="#lrmlv;#Ends"/> <lrml:atTimeInstant keyref="#t6"/> </lrml:TemporalCharacteristic> </lrml:TemporalCharacteristics></pre>
	Context of rule1 in LegalRuleML, automatically built using Akoma Ntoso information <pre> <lrml:Context key="ruleInfo1" > <lrml:appliesTemporalCharacteristics keyref="#tblock1"/> <lrml:appliesStrength iri="#lrmlv;defeasible"/> <lrml:appliesAssociations> <lrml:Associations key="sourceBlock1"> <lrml:Association> <lrml:appliesSource keyref="#sec504- clsc-lst1-pnt2"/> <lrml:toTarget keyref="#rule1"/> </lrml:Association> </lrml:Associations> </lrml:appliesAssociations> <lrml:toRuleText keyref="#rule1"/> </lrml:Context></pre>
Text in Akoma Ntoso <pre> <clause id="tit17-chp5-sec504- clsc"> <num>(c)</num> <heading>Statutory Damages.</heading> <list id="tit17-chp5-sec504- clsc-lst1"></pre>	Rule definition in LegalRuleML connected to the textual provision selected by mouse-over <pre> <lrml:Penalty key="rule3-penalty1"> <lrml:Obligation key="rule3- penalty1-ob1"> <ruleml:And> <ruleml:Atom key="rule3- penalty1-ob1-axml"> <ruleml:Rel</pre>

<pre> <point id="tit17-chp5- sec504-clsc-lst1-pnt1"> <num>(1)</num> <content> <p>-Except as provided by clause (2) of this subsection, the copyright owner may elect, at any time before final judgment is rendered, to recover, instead of actual damages and profits, an award of statutory damages for all infringements involved in the action, with respect to any one work, for which any one infringer is liable individually, or for which any two or more infringers are liable jointly and severally, in a sum of not less than \$750 or more than \$30,000 as the court considers just. For the purposes of this subsection, all the parts of a compilation or derivative work constitute one work.</p> </content> </point> </pre>	<pre> iri="&lrmv;payFine" min Pay </ruleml:Rel> <ruleml:Var>X</ruleml:Var> <ruleml:Ind>750 </ruleml:Ind> </ruleml:Atom> <ruleml:Atom key="rule3- penalty1-ob11-axml"> <ruleml:Rel iri="&lrmv;payFine" Pay max </ruleml:Rel> <ruleml:Var>X</ruleml:Var> <ruleml:Ind>30,000 </ruleml:Ind> </ruleml:Atom> </ruleml:And> </lrm1:Obligation> </lrm1:Penalty> </pre>
---	---

5. From LegalRuleML Meta-model to RDF Serialization

LegalRuleML was designed based on a meta-model¹ that defines relationships among different classes of the elements in the XML-schema. For helping this approach the technical author of the XML-schema (Tara Athan) implemented also several rdfs schemas. The following fragment of rdfs schema shows the relationship among the element `<lrm1:Role>` and the property `<lrm1:appliesRole>`. Following this approach all the elements that start with lower case are edges and the elements that start with upper case are nodes of a graph.

```

<rdfs:Class rdf:about="#Role">
    <rdfs:isDefinedBy rdf:resource="&lrm1mm;#" />
    <rdfs:label>Role</rdfs:label>
    <rdfs:comment>The class of roles played by agents relative to
LegalRuleML things.</rdfs:comment>
    <rdfs:subClassOf rdf:resource="#Thing"/>
</rdfs:Class>
<rdf:Property rdf:about="#appliesRole">
    <rdfs:isDefinedBy rdf:resource="&lrm1mm;#" />
    <rdfs:label>appliesRole</rdfs:label>

```

¹ Meta model is now under revision and the authors take this version from the OASIS repository: <https://tools.oasis-open.org/version-control/browse/wsvn/legalruleml/trunk/schemas/?rev=71&sc=1>

```

<rdfs:comment>A role applied to the targets by
the subject association or rule context.
</rdfs:comment>
<rdfs:domain rdf:resource="#AssociationOrContext"/>
<rdfs:range rdf:resource="#Role"/>

```

Using this meta-model it is possible to extract some relationships among elements. Some assertions in RDF format about the knowledge base rules are possible especially from the <lrml:Context>. These assertions build a set of RDF triples useful for improving information retrieval of the legal rules, and related legal textual sources, in the Semantic Web. The contextualization of the legal rules (e.g. Jurisdiction, Author, Authority, etc.) permits to create enriched connection with the Linked Open Data Cloud (e.g. geo-localization of the legal rules on the maps):

```

<rdf:RDF xmlns:rdf="&rdf;" xmlns:rdfs="&rdfs;" xmlns:xs="&xs;" 
xmlns:rulelmm="&rulelmm;" xmlns:lrmlmm="&lrmlmm;">
  <rdf:Description rdf:about="www.example.2.1.1.xml#rule1">
    <lrmlmm:appliesRole>
      <lrmlmm:Role rdf:about="&lrmlv;#Author">
        <lrmlmm:filledBy
        rdf:resource="http://monica.palmirani.cirsfid.unibo.it"/>
      </lrmlmm:Role>
    </lrmlmm:appliesRole>
    <lrmlmm:appliesSource rdf:resource="&akn;#sec504-clsc-pnt1"/>
    <lrmlmm:appliesSource rdf:resource="&akn;#sec504-clsc-pnt1"/>
    <lrmlmm:appliesStrength rdf:resource="&lrmlv;defeasible"/>
    <lrmlmm:appliesJurisdiction rdf:resource="&jurisdictions;us"/>
    <lrmlmm:appliesAuthority rdf:resource="&authorities;congress"/>
  </rdf:Description>
</rdf:RDF>

```

The same mechanism should be applied to the other assertions included in the <lrml:Context>.

6. RAWE Functionality

RAWE permits the following functionalities:

- Authentication of the end-user and customization of the environment according with the personal profile (e.g., legal system, legal tradition, legal guidelines);
- Multilanguage interface and environment;
- Customized interface and buttons on the basis of the user profile;
- Mark-up of a legal text with Akoma Ntoso standard using parsers to automatically detect the normative references, dates, metadata, and structure of legal documents;
- Record of the XML files in the eXist repository [21];
- Tree of the marked-up elements;
- On-the-fly view in Akoma Ntoso and in LegalRuleML;
- Conversion and export in PDF, XML, ePub, or RDF format;
- Web editor environment with WYSIWIG interface;
- Undo function;

- Contextual functionalities based on the XML tree and XML-schemas;
- Mouse-over for detecting the metadata of a portion of legal text and reuse for modelling legal rules;
- Toolbar for marking up the document's structure;
- Toolbar for marking up legal rules.

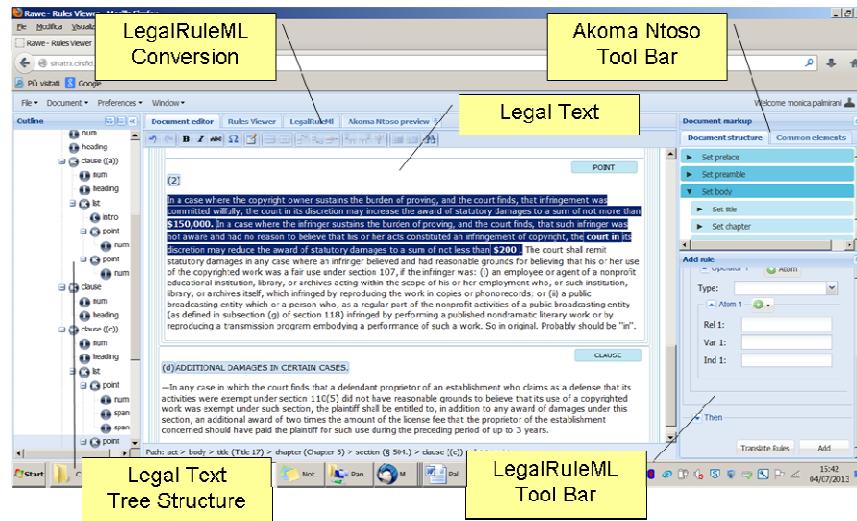


Fig. 2 – RAWE Web editor for marking up legal texts and normative rules

There are some critical points that we have faced in the RAWE implementation using HCI techniques:

- **Contextual Composition of the Rule.** In LegalRuleML we have three groups of rules: Prescriptive, Constitutive and Behaviors. Each group permits some particular modeling following the legal theory (e.g. Prescriptive rule is a sequence of deontic operators, Penalty needs a separate regime, Constitutive rule doesn't include deontic operators, etc.). For this reason RAWE needs to take in consideration the LegalRuleML prescriptive grammar constraints and lead the end user to compose the rules correctly.
- **Reparation** is a binary relationship between a penalty and a prescriptive rule or violation. So we found a smart interface way to select the two parts of the relationship and to connect them to each other.
- **Metadata in Context.** If we need to refine or readjust the context and the related metadata, we need a new toolbar and panel. RAWE permits to readjust the metadata imported by Akoma Ntoso and to add new ones.
- **Extra isomorphism rules.** Sometimes we need to include extra rules not directly linked to the legal text. RAWE permits to model this particular situation. However other some critical issues need to be addressed in the future:
- **Ontology.** Some elements of the rule modeling need to be enriched with the definitions of an external vocabulary or ontology (e.g. LKIF[10]).
- **Key.** We need to create a naming convention to harmonize the ID definition.

- **Meta-Rules.** In the future LegalRuleML will be also be able to manage meta-rules (rules about other rules), and we need to find a mechanism for linking rules as antecedents and consequents.
- **Multiple interpretation.** In this version of the editor is not possible to have multiple interpretations of the same legal textual document fragment.
- **Granularity.** For now the granularity of the isomorphism is on the rule. In the future we will be able to also manage the same functionality on the body, head, and atom.

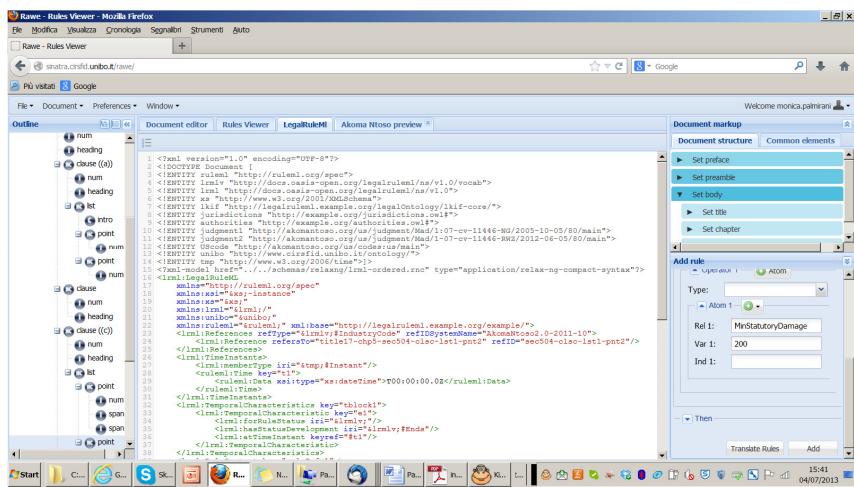


Fig. 3 – RAWE conversion of a rule in the LegalRuleML standard

7. The RAWE Architecture

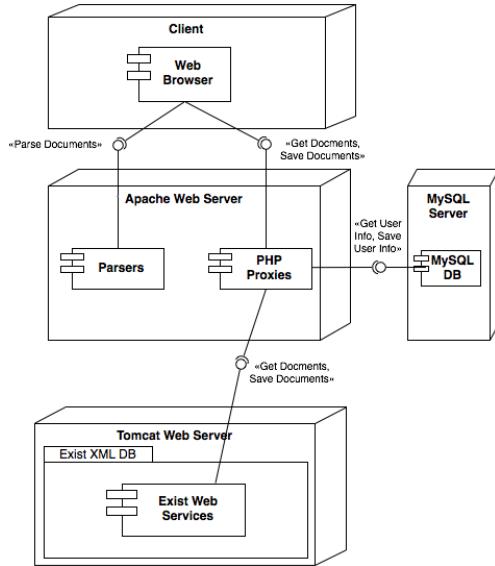
RAWE is a specialized Web editor developed using several open-source technologies, such as Sencha ExtJS 4.1 and TinyMCE.

Sencha ExtJS is an MVC framework that makes it possible to build an extraordinarily rich Web application. It supplies the instruments with which to easily develop the core of the application based on the Model View Controller pattern, and, moreover, it comes with a big range of user interface widgets. The other core strength of ExtJS lies in its component design. If the developer needs a new component that is not yet developed, the default components can be extended and the result is encapsulated in the default components. ExtJS is also completely cross-browser, so it is possible to deliver the application on a wide range of browser and operating systems. The latest smart phone and tablet browser are also covered, so it is possible to use ExtJS-based applications with touch screens and gestures.

TinyMCE is a platform-independent Web-based Javascript HTML WYSIWYG editor control. It can convert HTML text area fields or other HTML elements into editor instances. We integrated it into ExtJS, developing a new component for the

framework. The component retains all the functionality of the TinyMCE editor, but the effects of those functionalities are intercepted by the core of the ExtJS application. With this strategy, each event handled by the editor simply fires other events handled by the other components of the application. This means that there is no specific semantic on which TinyMCE itself relies, and TinyMCE can be substituted on demand with other open-source WYSIWYGs.

The editor uses the HTML5 standard in order to mark up documents. When an element is marked up, it is wrapped by a generic HTML element (such as *span* or *div*), and various classes are assigned to it in order to give to it semantic meaning for the editor itself and for the tool in charge of translating it into the desired document format. This means that there is not a meta markup language in the middle of a translation from HTML to another document format, and this carries the benefit of preventing data loss and having immediate access to the HTML version of the document without further conversions.



8. Conclusion

We have presented RAWE, a Web editor for marking up legal rules exploiting the previous markup of legal texts in Akoma Ntoso. RAWE is developed to enable application of the isomorphism principle; nevertheless, it is also open to the addition of rules not properly linked with the legal textual provisions, this in order to permit multiple interpretations or the inclusion of implicit rules. RAWE transforms all the rules in LegalRuleML and it saves them in a native XML repository, *eXist*. It is also possible to export the outcomes to a XML file. Finally, RAWE can convert in RDF the `<lrml:Context>` for creating a repository capable, in the future, of implementing an endpoint SPARQL for managing a better filter of legal resources in the Linked Open Data Cloud. Future work will be focused on the critical points stressed in the paper for managing advanced features.

References

- [1] Athan T., Boley H., Governatori G., Palmirani M., Paschke A., Wyner A.: OASIS LegalRuleML. In Bart Verheij, ed, Proceedings of 14th International Conference on Artificial Intelligence and Law (ICAIL 2013). ACM, 2013.

- [2] Barabucci G., Cervone L., Palmirani M., Peroni S., Vitali F.: Multi-layer Markup and Ontological Structures in Akoma Ntoso. In: LNCS 6237/2010, pp. 133-149, Springer, 2010.
- [3] Bench-Capon T. and Coenen F.: Isomorphism and legal knowledge based systems. Artificial Intelligence and Law, 1(1):65–86, 1992.
- [4] Boella G., Governatori G., Rotolo A., Torre L.V.D.: A Formal Study on Legal Compliance and Interpretation. ;In AICOL Workshops(2009), Springer, 162-183, 2011.
- [5] Boer A., Hoekstra R., de Maat E., Hupkes E., Vitali F., Palmirani M., Rátai B.: CEN Metalex Workshop Agreement (2009-08-28 proposal). <http://www.metalex.eu/WA/proposal>.
- [6] Breuker J., Boer A., Hoekstra R., Van Den Berg C.: Developing Content for LKIF: Ontologies and Framework for Legal Reasoning, in Legal Knowledge and Information Systems, JURIX 2006, pp.41-50, ISO Press, Amsterdam, 2006.
- [7] Brighi R., Lesmo L., Mazzei A., Palmirani M., Radicioni D.: Towards Semantic Interpretation of Legal Modifications through Deep Syntactic Analysis. JURIX 2008: 202-206, 2008.
- [8] Ceci, M., and Gordon, T. F.: Browsing case-law: An application of the carneades argumentation system. In Proceedings of the RuleML2012@ECAI Challenge, H. Aït-Kaci, Y.-J. Hu, G. J. Nalepa, M. Palmirani, and D. Roman, Eds., vol. 874, pp. 79-95.
- [9] Gordon T. F., Governatori G., Rotolo A.: Rules and Norms: Requirements for Rule Interchange Languages in the Legal Domain. RuleML 2009: pp. 282-296, Springer, 2009.
- [10] Gordon T. F.: Constructing Legal Arguments with Rules in the Legal Knowledge Interchange Format (LKIF). In: Computable Models of the Law, Languages, Dialogues, Games, Ontologies (2008), pp. 162-184, Springer, 2008.
- [11] Gordon, T. F.: The Carneades web service. In Computational Models of Argument – Proceedings of COMMA 2012, B. Verheij, S. Szeider, and S. Woltran, Eds., IOS Press, pp. 517-518.
- [12] Hoekstra R., Breuker J., Di Bello M., Boer A.: The LKIF Core Ontology of Basic Legal Concepts. In: Casanovas P., Biasiotti M.A., Francesconi E., Sagri M.T. (eds.), Proceedings of LOAIT 2007, 2007.
- [13] <http://www.akomantoso.org/> naming convention of the URI
- [14] Kelsen H.: Reine Rechtslehre, 2d. ed., Wien, 1960.
- [15] Lam H., Governatori G.: The Making of SPINdle. RuleML 2009 proceeding, pp. 315-322, 2009.
- [16] Lupo C., Vitali F., Francesconi E., Palmirani M., Winkels R., de Maat E., Boer A., and Mascellani P.: General xml format(s) for legal sources - Estrella European Project IST-2004-027655. Deliverable 3.1, Faculty of Law, University of Amsterdam, Amsterdam, The Netherlands, 2007.
- [17] Mazzei A., Radicioni D., Brighi R.: NLP-based extraction of modificatory provisions semantics. ICAIL 2009: pp. 50-57, ACM, 2009.
- [18] Mommers L.: Ontologies in the Legal Domain. In: Poli R., Seibt J. (eds.), Theory and Applications of Ontology: Philosophical Perspectives, Springer 2010, pp. 265-276, 2010.
- [19] Palmirani M., Brighi R.: An XML Editor for Legal Information Management. Proceeding of the DEXA 2003, Workshop on E-Government, Praga, 1-5 September, pp. 421-429. Springer-Verlag Berlin Heidelberg, 2003.
- [20] Palmirani M., Brighi R.: Model Regularity of Legal Language in Active Modifications. AICOL Workshops 2009: pp. 54-73, Springer, 2009.
- [21] Palmirani M., Cervone L.: Legal Change Management with a Native XML Repository. A cura di G. Governatori. Legal Knowledge and Information Systems. JURIX 2009. The Twenty-Second Annual Conference. Rotterdam. 16th-18th December 2009,pp. 146-156, Amsterdam: ISO press, 2009.
- [22] Palmirani M., Contissa G., Rubino R: Fill the Gap in the Legal Knowledge Modelling. In Proceedings of RuleML 2009, pp. 305-314, Springer, 2009.
- [23] Palmirani M., Governatori G., Rotolo A., Tabet S., Boley H., Paschke A.: LegalRuleML: XML-Based Rules and Norms. RuleML America 2011: 298-312, Springer, 2011.
- [24] Palmirani M., Governatori G., Rotolo A., Tabet S., Boley H., Paschke A.: Legal-RuleML: XML-Based Rules and Norms. RuleML America, Springer, 2011, pp.298-312.
- [25] Palmirani M.: Legislative Change Management with Akoma-Ntoso, in Legislative XML for the Semantic Web, Springer, Law, Governance and Technology Series Volume 4, 2011, pp 101-130.
- [26] Sartor G.: Legal Concepts as Inferential Nodes and Ontological Categories. In Artif. Intell. Law 17(3) 2009, pp. 217-251, 2009.
- [27] Sartor G.: Legal Reasoning: A Cognitive Approach to the Law. Vol. 5. Treatise on Legal Philosophy and General Jurisprudence. Berlin: Springer, 2005.
- [28] Vitali F., Palmirani M.: Akoma Ntoso Release Notes. [<http://www.akomantoso.org>]. Accessed 5 July 2013.

R-CoRe: A Rule-based Contextual Reasoning Platform for AmI *

Assaad Moawad¹, Antonis Bikakis², Patrice Caire¹, Grégory Nain¹ and Yves Le Traon¹

¹ University of Luxembourg, SnT
`firstname.lastname@uni.lu`

² Department of Information Studies, University College London
`a.bikakis@ucl.ac.uk`

Abstract. In this paper we present R-CoRe; a rule-based contextual reasoning platform for Ambient Intelligence environments. R-CoRe integrates Contextual Defeasible Logic (CDL) and Kevoree, a component-based software platform for Dynamically Adaptive Systems. Previously, we explained how this integration enables to overcome several reasoning and technical issues that arise from the imperfect nature of context knowledge, the open and dynamic nature of Ambient Intelligence environments, and the restrictions of wireless communications. Here, we focus more on technical aspects related to the architecture of R-Core, and demonstrate its use in Ambient Assisted Living.

Keywords: contextual reasoning, distributed reasoning, Ambient Intelligence, system development

1 Introduction

Ambient Intelligence (AmI) is 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 [6]. Therefore, they must be able to reason about their *contexts*, i.e. with any information relevant to the interactions between the users and system.

Reasoning models and methods are therefore essential to: interpret and integrate context data from various information sources; infer useful conclusions from the raw context data; and enable systems to make correct context-aware decisions in order to adapt to changes in the environment and to their users' needs, intentions and desires. The challenges in these tasks are primarily caused by the

* The present research is supported by the National Research Fund, Luxembourg, CoPAInS project (code: CO11/IS/1239572).

imperfection of context data, the open and dynamic nature of AmI environments and the heterogeneity of participating devices. According to [12], context data in AmI environments may be unknown, imprecise, ambiguous or erroneous. It is typically distributed among devices with different computing capabilities and representation models, which may join or leave the environment at random times and without prior notice. Moreover, devices communicate using wireless networks, which are unreliable and restricted by the range of transmitters.

In previous works we classified existing contextual reasoning approaches into three main categories [3]: *ontological* approaches, which use Description Logics to derive implicit knowledge from the existing context data; *rule-based* approaches, which are based on more expressive logics; and *probabilistic* approaches, which use Bayesian networks or other probabilistic models to explicitly model uncertainty in the context data. In the same paper we argued that, compared to others, rule-based approaches offer significant advantages, such as *simplicity* and *flexibility*, *formality*, *expressivity*, *modularity*, *high-level abstraction* and *information hiding*. In [2] we introduced *Contextual Defeasible Logic* (CDL): a new distributed, non-monotonic approach for contextual reasoning, which enables heterogeneous entities to collectively reason with uncertain and ambiguous information. In order to enable the deployment of CDL in real environments, in [15] we proposed the integration of CDL in Kevoree [8] - a software framework that facilitates the development of Distributed Dynamically Adaptive Systems.

In this paper, we present R-CoRe, a Rule-based Contextual Reasoning Platform of Ambient Intelligence, which is the outcome of this integration. The main features of R-CoRe are:

1. It is totally distributed. Entities are represented as Kevoree nodes and communicate through dedicated communication channels.
2. It is rule-based. The local knowledge of each entity is modeled as a CDL theory and knowledge exchange is enabled by mapping rules.
3. It enables handling inconsistencies that arise from the integration of knowledge from different sources using preferences, which reflect the confidence that each node has in the quality of knowledge imported by other nodes.
4. It is dynamic and adaptive. Using the auto-discovery capabilities of Kevoree, it can handle cases of devices that join or leave the system at any time.

We developed R-CoRe for the needs of the CoPAInS project¹ (Conviviality and Privacy in Ambient Intelligence Systems). CoPAInS focuses on the tradeoffs to be made in Ambient Assisted Living systems [16], particularly as they pertain to conviviality, privacy and security [7]. In this framework, R-CoRe is used for the simulation of AAL scenarios, with which we test and validate our methods.

The remaining of the paper is structured as follows: Section 2 briefly presents the theoretical background of this work. Section 3 describes our running AAL example. Section 4 presents the main features of Kevoree. Section 5 presents in detail the R-CoRe architecture, while Section 6 demonstrates its use in Ambient Assisted Living. Section 7 concludes and presents our plans for future work.

¹ <http://wwwen.uni.lu/snt/research/serval/projects/copains>

2 Background

The underlying reasoning model of R-CoRe is Contextual Defeasible Logic (CDL [2,5]). CDL is a non-monotonic extension of Multi-Context Systems specifically designed for the requirements of Ambient Intelligence systems. Multi-Context Systems (MCS [11,10]) are 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. CDL extends the original MCS with defeasible mapping rules to capture the uncertainty of the knowledge that an agent imports from external sources; and with a preference ordering on the system contexts, which is used to resolve the potential inconsistencies that may arise from the information exchange between mutually inconsistent 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 (a set of positive and negative literals of the form $(c_i : a_i)$), R_i is a set of rules, and T_i is a preference ordering on C . R_i consists of a set of *local rules*, which represent the local knowledge of an agent, and a set of *mapping rules*, through which agents may share parts of their local knowledge. The body of a local rule is a conjunction of *local* literals (literals that are contained in V_i), and its head is labeled by a local literal too. A mapping rule, on the other hand, contains both local and foreign literals (literals from the vocabularies of other contexts) in its body, while its head is labeled by a local literal:

$$r_i^m : (c_j : a^1), \dots, (c_k : a^{n-1}) \Rightarrow (c_i : a^n)$$

By representing mappings as defeasible rules and by ordering contexts in terms of preference, CDL enables handling inconsistencies that arise when importing conflicting information from different contexts.

We have obtained the following results for CDL: a proof theory [4]; an argumentation semantics [2]; four algorithms for distributed query evaluation and a complexity analysis [5]. The algorithms proceed roughly as follows: when a context C_i receives a query about one of its local literals $(c_i : a_i)$, it first attempts to evaluate its truth value using its local rules only. If this is not possible, it generates the proof trees for $(c_i : a_i)$ and its negation $\neg(c_i : a_i)$, using the local and mapping rules of C_i . For each of the foreign literals $(c_j : b_j)$ contained in one of the two proof trees, it generates a similar query and sends it to the appropriate context (C_j). In case of conflict, i.e.. the truth values of all literals in both proof trees are true, C_i compares the proof trees using preference information from T_i .

We applied CDL in real scenarios of Mobile Social Networks [1] and Ambient Intelligence [5], and showed how it addresses issues that arise from the imperfection of context data. In [15], we discussed some of its limitations with regard to its deployment in real environments, and explained how its integration with Kevoree enabled us to overcome several technical issues related to communication, detection, adaptability and dynamicity. Here we focus more on technical aspects of this integration, and demonstrate the use of the integrated framework, R-CoRe, in an example scenario from Ambient Assisted Living.

3 An Ambient Assisted Living Example

Below 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 the CoPAInS project.

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.

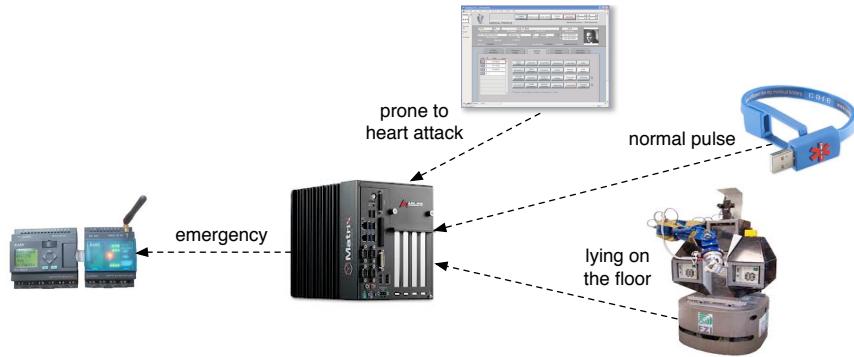


Fig. 1. Context Information flow in the scenario.

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.

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 [8] is an open-source environment that provides means to facilitate the design and deployment of Distributed Dynamically Adaptive Systems, taking advantage of Models@Runtime [17] mechanisms throughout the development process.

This development platform is made of several tools, among which the Kevoree Modeling Framework (KMF) [9], 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.

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

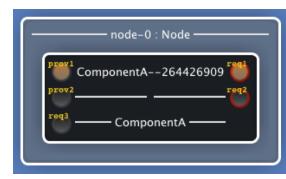


Fig. 3. A component instance, inside the node instance on which it executes

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.

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.



Fig. 4. An instance of Group on top, of Channel on the bottom

5 R-CoRe Architecture

In this section, we describe how CDL and Kevoree are integrated in the R-CoRe architecture. We should note that the parts of CDL, which were not directly mapped to existing elements Kevoree, were implemented in Java.

5.1 Java Library

Our Java implementation is composed of a main rcore package containing 4 sub-packages: agencies, interceptor, knowledge, logic packages and the main Query component class.

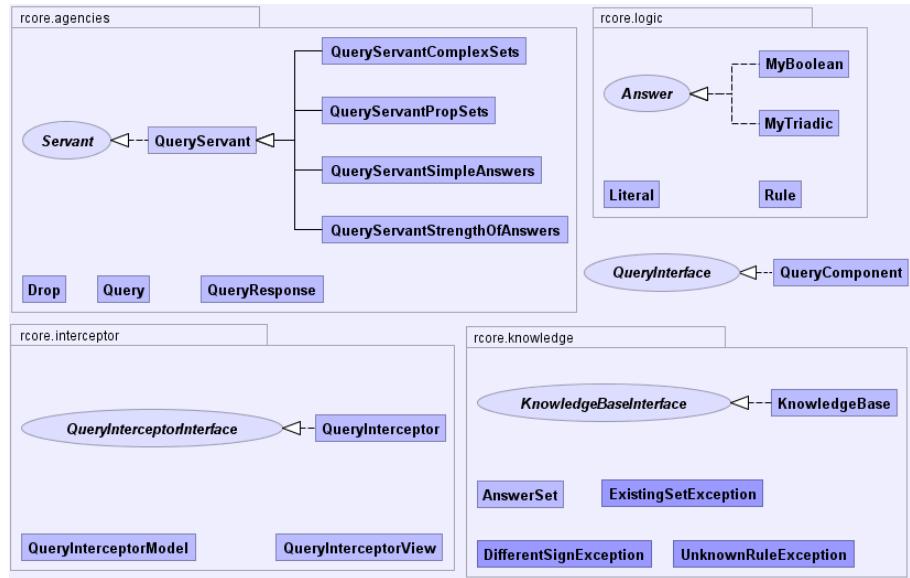


Fig. 5. General overview of R-CoRe.

- The agencies package contains the classes used by the query servant thread, which implements the reasoning algorithms of CDL [5]. This package contains also the `Query` class and the `QueryResponse` class. These classes represent the messages that will be exchanged between `QueryComponents`.
- The interceptor package contains the model, the view and the controller of the interceptor component. The controller is a component developed to run on the Kevoree platform.
- The knowledge package includes the `KnowledgeBase` class, which stores the local rule theory, the mapping rules and the preference order of each node. This class contains also the methods used to load and save the knowledge base and preference order from and to files.

- The logic package contains the classes that represent (in memory) the literals and rules.
- Finally the query package contains the QueryComponent class, which is the main component developed to run on the Kevoree platform.

Figure 5 shows a UML overview of our implementation.

5.2 Query Component

In our platform, the notion of context, 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 R-CoRe, 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.

5.3 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. 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 remote components that were queried for the two literals using the context preference order. The result is reported back to the query component.

5.4 Query Interceptor

In order to monitor and control all the exchanges happening between the Query components, we created a Query Interceptor component. Its main job is to capture all the queries transiting on the exchange channel, display them on a graphical interface to the users, and allow the users to forward them manually afterwards. This component serves two purposes: It enables demonstrating the reasoning process using a single graph that is created in a step by step fashion; it also facilitates debugging the system by centralizing all the exchanges in one component.

The Graphical interface of the Query Interceptor has two parts: the graph on the left that is used to visualize the information exchange; and the user controls on the right. When a query is sent from a component a to a component b through the interceptor, two vertices, a and b , and an edge from a to b are added to the graph. If the Interceptor is set to the *demo-mode* (by selecting a check box called ”demo mode“ on the Graphical User Interface GUI), the query is paused at the interceptor, and the user has to click the *Next* button in order to actually forward the query from the interceptor to component b . The same happens when a response is sent from b to a . If the *demo-mode* is not selected, all the queries and responses are forwarded automatically without any intervention of the user as if the Interceptor is transparent or turned off. The Interceptor also contains *Reset* button, which clears the graph and restarts the monitoring.

On the Kevoree platform, the Query Interceptor component has two ports: *Query In* port that receives all the queries sent from the Query components, and a *Query out* port to forward the query back to the components after being displayed on the Interceptor GUI.

5.5 Query class and loop detection mechanism

The query Java class that we developed for R-CoRe has the following attributes: the queried literal, the name of the component that initiated the 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 evaluating the query, a set of supportive sets (set of foreign literals that are used for the evaluation of a query), 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 the query evaluation algorithm.

As the query evaluation algorithm is distributed, 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.

6 Demonstrating R-Core

6.1 Setup

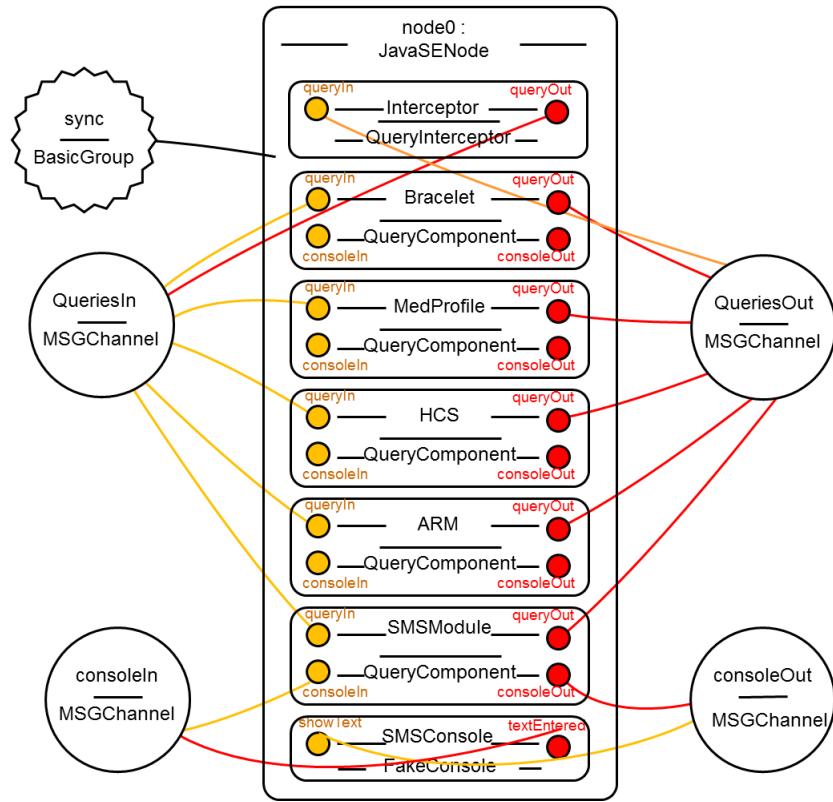


Fig. 6. The running example implemented, a snapshot of the Kevoree Editor.

Applying the above methodology on the running example described in section 3, 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 are able now to initiate queries on the sms module.

Figure 6 shows our experimental setup, which involves the 5 query components, the console connected to the sms module (*FakeConsole*) and the Query Interceptor component. Note that all query input and output ports of the query components are connected to the Interceptor in order to allow us to capture all the exchanges for our demo session.

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

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. 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.

6.2 Execution

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*) on the console by typing ***dispatchSMS*** on the console of the SMSModule.²

The SMS module starts a new query servant which initiates in its turn a new query about (*hcs : emergency*). This query is captured by the interceptor and it

² You can run the demo and access its source code and all other necessary files at: <https://github.com/securityandtrust/ruleml13>.

is displayed on the graph GUI; in fact two nodes representing the SMSModule and HCS are added to the graph. If the *demo mode* of the interceptor is selected, the user has to click on *Next* button each time to forward a query from one component to another. This step-by-step mode is very useful to understand the actual interactions and to slow down the exchanges.

The knowledge base of *hcs* contains one rule supporting (*hcs : emergency*), *M2*, and another one supporting its negation, *M1*. *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*). Each time a query or a query response is generated from any component and dispatched to any other, the interceptor captures it, adds it to the graph, and waits for the user to click *Next* before forwarding the query or the response to the appropriate component.

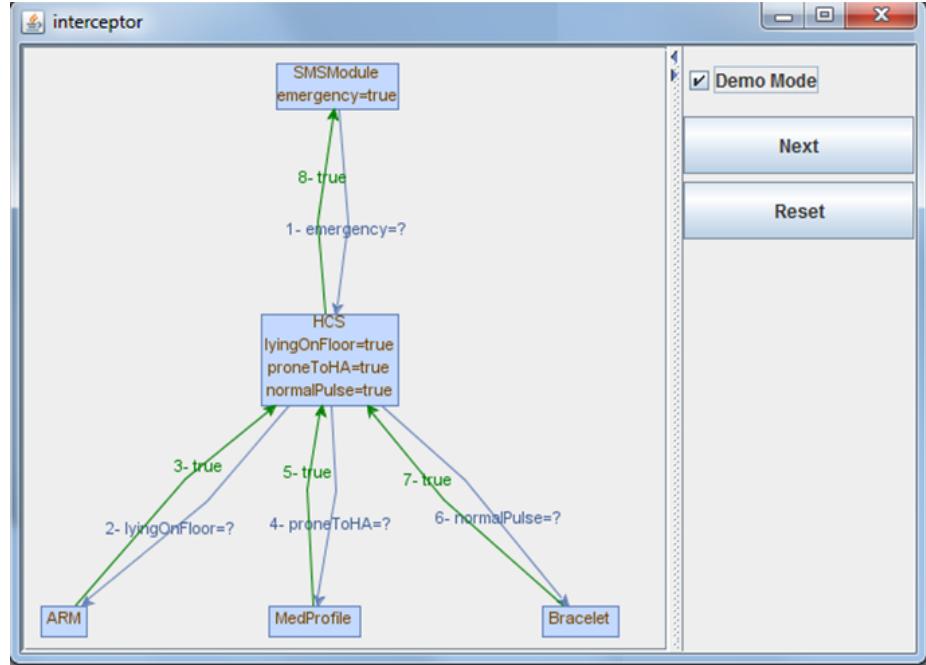


Fig. 7. Different steps of execution of the running example.

Screen-shots from the demo during different steps of execution are displayed in figure 7. Following this, the SMS module will display, on the console connected to smsModule, the answer for *dispatchSMS*, which in this case is *true*.

7 Conclusion and Future Work

In this paper, we presented R-CoRe: a Rule-based Contextual Reasoning framework for Ambient Intelligence and demonstrated its use Ambient Assisted Living. Being based on Contextual Defeasible Logic, R-CoRe enables reasoning with imperfect context data in a distributed fashion. The capabilities of the underlying software platform, Kevoree, further enables R-CoRe to overcome several technical issues related to communication, information exchange and detection.

R-CoRe 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 are able to adapt between different strategies depending on the available resources.

Another issue is with the development of the rule theories. In this version of R-CoRe, users have to use the syntax of CDL to create the rule and preference bases of each node. A future plan is to develop or integrate appropriate rule-editing tools that will enable plain users to create and configure the rule and preference bases using simple natural-language-based constructs. A tool that we can use for such purposes is S²DDREd [13]: an authoring tool for Defeasible Logic, which provides the users with semantic assistance during the development of rule theories.

In the future, we also plan to extend CDL to support shared pieces of knowledge, which are directly accessible by all system contexts, and implement this extension in R-CoRe using the *groups* feature of Kevoree (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 R-CoRe, 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 [14], which combines data from multiple sources to infer higher-level conclusions,

and study previous works on the integration of CEP and reaction rules [18]. We will test and evaluate all our deployments and extensions to R-CoRe 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. Antoniou, G., Papatheodorou, C., Bikakis, A.: Reasoning about Context in Ambient Intelligence Environments: A Report from the Field. In: KR. pp. 557–559. AAAI Press (2010)
2. Bikakis, A., Antoniou, G.: Defeasible Contextual Reasoning with Arguments in Ambient Intelligence. IEEE Trans. on Knowledge and Data Engineering 22(11), 1492–1506 (2010)
3. Bikakis, A., Antoniou, G.: Rule-based contextual reasoning in ambient intelligence. In: RuleML. pp. 74–88 (2010)
4. Bikakis, A., Antoniou, G.: Contextual Defeasible Logic and Its Application to Ambient Intelligence. IEEE Transactions on Systems, Man, and Cybernetics, Part A 41(4), 705–716 (2011)
5. Bikakis, A., Antoniou, G., Hassapis, P.: Strategies for contextual reasoning with conflicts in Ambient Intelligence. Knowledge and Information Systems 27(1), 45–84 (2011)
6. Cook, D.J., Augusto, J.C., Jakkula, V.R.: Ambient intelligence: Technologies, applications, and opportunities. Pervasive and Mobile Computing pp. 277–298 (2009)
7. 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)
8. 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 (Jul 2012), <http://hal.inria.fr/hal-00713769>
9. 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: Models 2012. Innsbruck, Autriche (Oct 2012), <http://hal.inria.fr/hal-00714558>
10. Ghidini, C., Giunchiglia, F.: Local Models Semantics, or contextual reasoning=locality+compatibility. Artificial Intelligence 127(2), 221–259 (2001)
11. Giunchiglia, F., Serafini, L.: Multilanguage hierarchical logics, or: how we can do without modal logics. Artificial Intelligence 65(1) (1994)
12. Henricksen, K., Indulska, J.: Modelling and Using Imperfect Context Information. In: Proceedings of PERCOMW '04. pp. 33–37. IEEE Computer Society, Washington, DC, USA (2004)
13. Kontopoulos, E., Zetta, T., Bassiliades, N.: Semantically-enhanced authoring of defeasible logic rule bases in the semantic web. In: WIMS. p. 56 (2012)
14. Luckham, D.C.: The power of events - an introduction to complex event processing in distributed enterprise systems. ACM (2005)

15. Moawad, A., Bikakis, A., Caire, P., Nain, G., Traon, Y.L.: A Rule-based Contextual Reasoning Platform for Ambient Intelligence environments. In: RuleML. LNCS, Springer (2013)
16. 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)
17. 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. pp. 122–132. ICSE '09, IEEE Computer Society, Washington, DC, USA (2009), <http://dx.doi.org/10.1109/ICSE.2009.5070514>
18. Paschke, A., Vincent, P., Springer, F.: Standards for complex event processing and reaction rules. In: RuleML America. pp. 128–139 (2011)

Interpreting Regulations with SBVR

Elie Abi-Lahoud¹, Tom Butler¹, Donald Chapin², John Hall³

¹University College Cork

e.abilahoud@ucc.ie, tbutler@afis.ucc.ie

²Business Semantics Ltd

donald.chapin@businesssemantics.com

³Model Systems

john.hall@modelsystems.co.uk

Abstract. The wide and complex spectrum of regulations, especially in the financial services industry, calls for machine assistance in making sense of, and in consuming, regulatory text. This paper describes an approach to interpreting regulations with SBVR. The purpose is to clarify ambiguity in regulations by developing a shared vocabulary and shared guidance based on the regulatory text. The on-going work presented in this paper is part of the Governance, Risk and Compliance Technology Centre's (Ireland) current research activities that include the development of policy advice on compliance with US Anti-Money Laundering (AML) regulations for companies that are governed by these regulations. The approach is based on the navigation of US public databases – Federal Register, Code of Federal Regulations and US Code – to identify subsets of AML regulation relevant to companies based outside the USA. These subsets are imported into an SBVR toolset, where they are analysed and, if necessary, interpreted by the legal and financial experts on the team. A standardized vocabulary for AML is being developed in SBVR, together with advice on regulatory intent and formal expression of rules with which regulated companies must comply.

Keywords: case study, compliance, human language, regulation, SBVR.

1 Introduction

1.1 Overview

The work described here is a proof of concept undertaken by Ireland's Governance, Risk and Compliance Technology Centre (GRCTC) - an application of the Semantics of Business Vocabulary and Business Rules (SBVR) [11]. It consists of:

- Analysis of US regulations that are relevant to companies based in Ireland;
- Interpretation of the regulation source text and editing it into more formal representations using SBVR tools;
- Development of policies and rules for enforcement of the regulations, and advice to regulated companies on compliance.

The concept definitions developed in SBVR will eventually become part of two OWL2 ontologies being developed by GRCTC for the financial industry.

1.2 GRCTC

The GRCTC is an industry led, collaborative programme of research and innovation into Governance, Risk and Compliance (GRC) of the financial services industry. It has a consortium of academic partners and an industry steering consortium. Membership is open to companies and research providers with an office in Ireland and an interest in developing competence, outputs and technologies related to GRC in the financial services industry.

The programme is led by a multidisciplinary team of computer scientists and legal and financial subject matter experts (SMEs) at University College Cork. Its mission is to research

and develop industry-ready GRC solutions for the financial services industry to help industry stakeholders commercialise related GRC technologies.

1.3 Context

A major strand of GRCTC's activity is development of the Financial Industry Regulatory Ontology (FIRO) and the Financial Industry Governance Risk and Compliance Ontology (FIGO). FIRO will enable efficient access to the wide and complex spectrum of financial industry regulations, relying on formal semantics and regulatory rules. FIGO will provide formal semantics, a GRC knowledge base, and data models to inform and integrate GRC practices and data in the financial industry.

GRCTC has been invited by the Finance Domain Task Force of the Object Management Group (OMG) to submit FIRO for progression to an international standard.

1.4 SBVR

In 2013, GRCTC adopted the OMG's SBVR as its basis for FIRO and FIGO. SBVR is a vocabulary – more precisely, an ISO terminological dictionary – for defining business concepts and rules, represented in simplified natural language. It is based on ISO terminology standards and practice.

SBVR is business-oriented. It was developed for definition of things in a business (rather than the data that would represent them in information systems) and policies and rules that constrain the relationships between them and govern the activities in which they play roles.

An SBVR terminological dictionary + rulebook (“business vocabulary + rules”) comprises:

- Noun concepts, which correspond to things in a business
- Verb concepts, which correspond to relationships between defined things
- Definitional rules, which constrain these relationships
- Behavioral rules, which govern business activities in which defined things play roles

SBVR contains the noun concepts, verb concepts and definitional rules needed to define the noun concepts, verb concepts, definitional rules and behavioral rules for a specific business or business domain, such as Anti-Money-Laundering. SBVR is itself defined in SBVR, and any domain-specific terminological dictionary + rulebook is an extension of SBVR.

SBVR itself does not include behavioral rules. SBVR is a terminological dictionary that defines what SBVR is, including what behavioural rules are and how to specify them. But behavioral rules govern business activities and SBVR contains no business activities. The rules that govern how SBVR should be used are a matter for tool developers, methodologists, trainers and quality auditors.

Behavioral rules are typically defined for operational business activities. For GRCTC's work they are the rules that govern what regulated companies must do in order to comply with the AML regulations.

A terminological dictionary + rulebook defined with SBVR should be complete and consistent:

- Each noun concept must be explicitly defined, or adopted from an authoritative source, or acknowledged as ‘implicitly understood’ (the everyday natural language meaning of the term used).
- Only recognized noun concepts may play roles in verb concepts
- Rules may be built only from defined verb concepts and a defined set of structure elements (obligation, necessity, if ... then, that, at least...).

SBVR does not have a normative syntax; any syntax that has adequate expressive power is acceptable. SBVR is specified in SBVR Structured English (SE), a simplified version of natural English, and SBVR SE is probably the most widely used syntax for domain-specific SBVR models. The conceptual model is separated from the external representation, and any (suitable simplified) natural language may be used.

SBVR definitions and rules are intended for people in the business. They can be transformed to machine-readable ontologies, or to data models and rules for information systems that would support a business defined using SBVR.

1.5 Proof of Concept (PoC)

The GRCTC is developing a number of proofs of concept for FIRO and FIGO. One is the work described here. This focuses on the application of US Anti-Money-Laundering (AML) regulations. Some financial companies based in Ireland, including Irish companies that trade in the USA and Irish subsidiaries of US companies, are governed by a subset of these regulations. The scope of this PoC, however, is broader because the US Anti-Money-Laundering regulations apply to all Financial Services companies, in Europe or elsewhere, doing business in the US.

The purpose of the proof of concept is to demonstrate the capture of relevant US AML regulations and the formalisation of their vocabulary and rules as a basis for guidance on compliance. The results are:

- The relevant subsets of US AML regulations
- Interpretation of regulatory intent
- Behavioral rules with which industry partners must comply, and the vocabulary that defines their meaning, expressed in SBVR SE
- AML content for FIRO and FIGO

1.6 Related Work

The basis of the approach used by GRCTC originated in OntoRule [<http://ontorule-project.eu>], an EU Framework 7 project that ran from 2009 – 2011. The OntoRule case study for interpretation of regulation was undertaken by Laboratoire d'Informatique de Paris Nord (LIPN) and Audi AG, using a subset of EU regulations for car safety systems (seatbelts, airbags, brakes). The approach for the case study used SBVR constructs as patterns for analysis of regulation source text, using LIPN's Terminae software; it is described in [12]. LIPN has continued to develop the approach, as described in [9], [10] and [11].

2 The US Regulatory Framework

The approach taken for the proof of concept is enabled by the framework of US law and regulation.

US laws are created by Congress and most are codified in the United States Code (USC) [2]. Congress delegates authority for rules and regulations to departments and executive agencies, such as the US Treasury Department, which publish enforceable regulations that implement the laws. The division of authority is summarised in Figure 1.

Congress passes Laws	Executive Agencies Issue Rules/Regulations
Publish in Slip Law/Statutes at Large: codified in <i>US Code (USC)</i>	Publish in <i>Federal Register (FR)</i> : codified in <i>Code of Federal Regulations (CFR)</i>
Power is determined by Constitution Courts review for: <ul style="list-style-type: none">• Constitutionality	Power is delegated by Congress Courts review for: <ul style="list-style-type: none">• constitutionality & limits of delegated authority• arbitrary and capricious actions• Administrative Procedure Act requirements
Congress acts collectively to represent the will of the people	Agencies must seek and consider public comment on benefits of rules vs. burdens and costs
Set broad social and economic goals and legal requirements	Prescribe specific legal requirements to meet goals

Figure 1: US Regulatory Authority

Regulations ('rule' and 'regulation' are synonyms in the US regulatory domain) are published in the Federal Register (FR) [3], a daily journal that includes all proposed and final rules.

The Code of Federal Regulations (CFR) [4] is the codification of the final rules published in FR, showing the aggregated effect of related rules. Rules published in FR are defined as changes to be made to CFR.

Entries in CFR refer to laws in USC for authorizations and definitions. The relationship between FR, CFR and USC is illustrated in Figure 2.

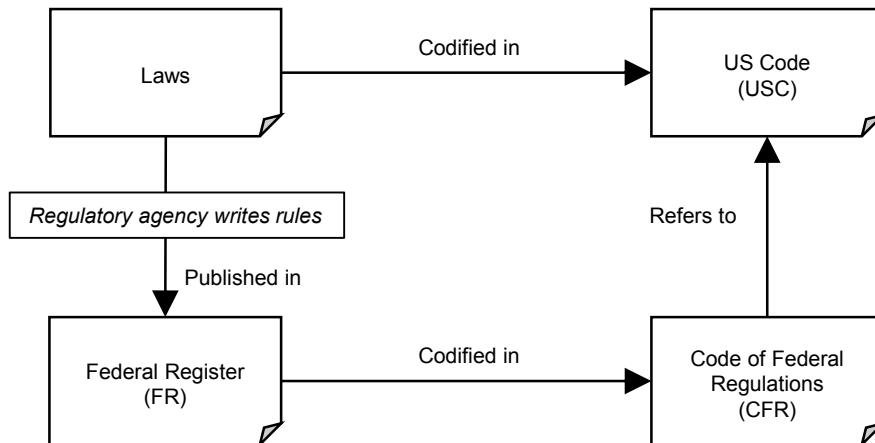


Figure 2: US Regulatory Framework

There is a statutory obligation to publish rules in FR, first as proposed rules, open for public discussion, and then as final rules, which define explicit changes to CFR. Final rule documents include the public discussion and responses from the regulatory agency, which is helpful for SMEs in understanding the intent of the rules.

The Code of Federal Regulations is organized as:

- 50 Titles, each naming a broad subject area for regulation (e.g. 12: Banks and Banking)
- Chapter: the rules of a single agency
- Part: rules on a single program or function
- Section: one provision of program/function rules

Sections are composed of paragraphs, numbered to up to 6 levels of subparagraph. A rule published in FR is normally within the remit of a single agency, and usually specifies CFR changes as addition, deletion or replacement of individually-identified paragraphs or subparagraphs.

3 Proof of Concept

3.1 Scope

The base document for the proof of concept was the FR final rule (76 FR 45403) "Bank Secrecy Act Regulations - Definitions and Other Regulations Related to Prepaid Access" [5], an 18-page document from the Financial Crimes Enforcement Network of the Treasury Department. Its scope is illustrated in Figure 3.

Like many FR rules, the direct scope was fairly narrow; it changed only 5 sections of 31 CFR Chapter X (namely, §1010.100, §1022.210, §1022.320, §1022.380, §1022.420). The open-ended aspect was the dependence on definitions in USC and other sections of CFR, which were needed for full understanding of the business impact of the changes.

The solution adopted was to introduce 'stubs', determined by the SMEs who were interpreting regulatory text. If a reference is encountered to a concept that would be familiar to people working in the domain (e.g. "investment company as defined in section 3 of the Investment Company Act of 1940 (15 U.S.C. 80a-3)"), an SME can declare it as a 'stub' and no further referencing will be followed. Over time, this truncation will be corrected. When the scope of

the work extends into areas that affect the concept, the stub will be replaced by the full definition, together with any further referencing needed.

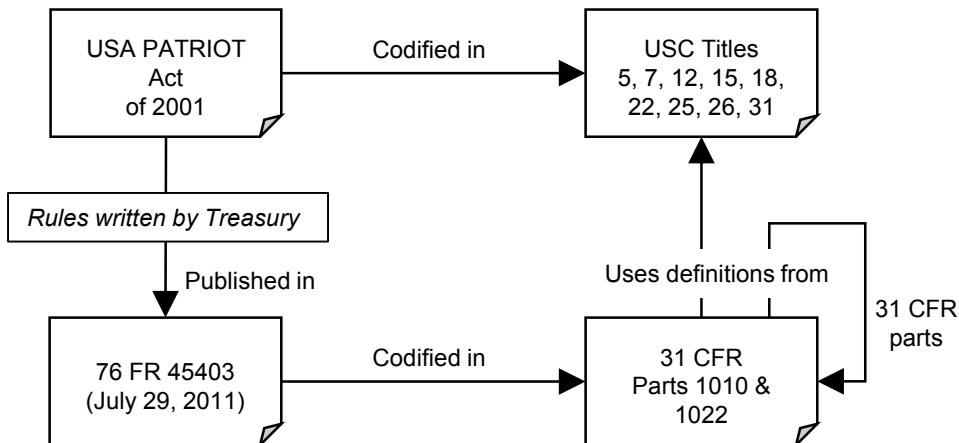


Figure 3: Scope of Proof of Concept

3.2 Selection of Regulatory Content

The regulations are presented in CFR. Two approaches were considered for selection of relevant subsets:

- Analysing the full content of 31 CFR X. This at first seemed the simpler approach, but would require consideration of over 100,000 words of source text, before following references to USC.
- Starting with the FR rule and following its changes through 31 CFR X. This resulted in a text of less than 26,000 words (equivalent to a little over 38 pages of 10-point type), including 28 references to USC, 16 to parts of 31 CFR X not directly changed by the rule, and 6 stubs.

The second approach was adopted. It provided a more ‘digestible’ text size for SMEs, and focused on the regulatory intent of the published rule. Also, it provides an approach that can be used directly with new rules as they are published.

3.3 Analysis and Formalization

The sections of 31 CFR 1022 amended by 76 FR 45403 use terms defined in section §1010.100 “General Definitions”. Before capturing in SBVR the interpretation of sections amended by 76 FR 45403, the SMEs captured, in SBVR SE, the definitions of §1010.100. These definitions are reused, in their SBVR format, while interpreting 76 FR 45403. The SMEs were able to capture in SBVR SE the definitions of 66 terms/concepts directly from section §1010.100. They also identified another set of 180 external terms/concepts. Those terms are not directly defined in §1010.100. Domain practitioners implicitly understand more than half, while the rest could be defined using business dictionaries or standardised industry vocabularies. In this proof-of-concept, terms defined in the Financial Industry Business Ontology¹ (FIBO [6]) were identified.

Having the “general definitions” in SBVR, the next step is to capture the interpretations of the four sections of 31 CFR X amended by 76 FR 45403. The task was assigned to a team of four SMEs. A straightforward division of labour could be done by attributing a section to each SME. However, a quick read of the amended sections reveals overlaps between them. To minimise having two or more SMEs modelling the same ‘Things’ (i.e. defining the same SBVR elements for the same Thing - *noun concepts*, *verb concepts*, etc.), a theme-based

¹ FIBO is an Industry standard being developed by the Enterprise Data Management Council and the Object Management Group. As a common language, it bridges the language gap between business and technology. As a machine-readable knowledge model, it facilitates the development of semantic applications for the financial industry.

division of labour was adopted. First, the content of each section is broken down into themes or categories. For example, §1022.380 Registration of money service business could be broken down in three themes or categories as follows:

1. Agency related provisions: 1022.380 (a) (3) and (4) (d)(1)(2) (E) (F) (G)(H)(ii)
2. Registration related provisions: (a)(1)(b)(1)(i)-(iii) (a)(2)(3)(4)(c)
3. Compliance related provisions: (e) (f)

Second, the identified themes are cross-referenced between sections and consolidated under theme headers. The result is eleven consolidated themes, which are then regrouped in four work streams, one for each SME. Appendix B details the division of labour aiming to attribute a (cross-sections) theme to each SME.

Each SME starts interpreting the rules assigned to him/her following the protocol described in Figure 4. The aim is to identify modified *verb concepts*, supporting *verb concepts* and *noun concepts*.

- ```

1- Read the text
2- Identify modalities (Obligations, prohibitions, etc.)
3- For each modality
 a. Add the relevant modality keyword
 b. Identify the English verb on which the modality is applied
 c. Stylise this verb using the SBVR verb style
 d. Identify the noun concepts (general, individual, etc.) or the
 verbal phrase(s) playing the roles in this verb
 e. If the verb roles are played by noun concepts, complete the
 SBVR modified verb concept by stylising the identified the
 noun concepts
 i. Add all the stylised noun concepts the
 noun_concepts_list
 f. If the verb roles are played by verbal phrases, stylise each
 verbal phrase by identifying English verbs, SBVR noun
 concepts and keywords
 i. Add each verbal phrase to the
 supporting_verb_concepts_list
4- For each noun concept in noun_concepts_list
 i. Start enriching by identifying the characteristic of
 each noun concept (if any)
 ii. Identify other definition elements

```

**Figure 4: SBVR-based Rule Interpretation Protocol**

To illustrate the previously described protocol, take for example, §1022.210 (d)(iv):

“iv) Money services business [...] must establish procedures to verify the identity of a person who obtains prepaid access under a prepaid program and [...]”

The first part of this rule expresses the obligation to “establish procedures to verify [...]. The obligation is on the verb “establish”. It will be stylised in the SBVR *verb* style (blue italic in SBVR SE). The first role is played by “Money services business” which is a *general noun concept* in SBVR (styled as green underlined in SBVR SE). According to Figure 5, Money services business is stylised as a *noun concept* and added to the noun concepts list to be further defined. The second role is played by “procedures to verify [...]. Procedures is styled as a *noun concept*, “to verify” as a *verb*, “the” as a *keyword* (styled orange in SBVR SE), etc. Below is a suggested formalisation of the rule based on the protocol described in Figure 5:

It is obligatory that each money services business *establishes* procedures to *verify* the *identity of* the person *obtaining* prepaid access *under* a *prepaid programme*.

The modified verb concept is ‘money services business *establishes* procedures’. One supporting verb concept is ‘person obtaining prepaid access *under* a *prepaid programme*’. Note each of the noun concepts are added to the noun concepts list and defined in a later phase (if the definition is not given by the current section or the general definitions section). Appendix C describes a more detailed example of SBVR interpretation of § 1022.210 (a).

## 4 Implementation and Future Work

Two software tools assist the SMEs in their SBVR interpretation along with a rich text editor (MS Word in this case). First, Confluence by Atlassian is a shared wiki with commenting functionalities that serves as a collaborative editing platform. Second, Designs for Management™ by Business Semantics is an SBVR editing suite that validates SBVR interpretations and generates machine-readable vocabularies and rules in the XML Metadata Interchange format based on the SBVR metamodel.

### 4.1 Phase 1: Collaborative Interpretation

The first phase consists of collaborative interpretation of the regulatory text limited in scope as described in section 3.1. An MS-Word template is used to capture the interpretations following the protocol described in Figure 4, whereas a classic collaborative wiki is mainly used to capture definitions of *noun concepts*. Its commenting functionalities allow the SMEs to interact on a given concept definition and discuss potential semantic precisions. They could vote on a definition or on a revision leaving an audit trail of the Vocabulary development. Appendix D illustrates how the SMEs used this wiki in the context of this proof of concept.

This phase resulted in identifying more than 300 noun concepts. Those concepts were (i) defined within the scoped regulation or (ii) commonly understood or (iii) defined by SMEs using domain authoritative sources. It also resulted in formalizing around 200 behavioral rules based on 76 FR 45403. This number along with the number of definitional rules is expected to evolve after the completion of phase 2 described hereafter.

### 4.2 Phase 2: Validation and Generation of a Machine-readable Vocabulary

Having a candidate vocabulary and a set of candidate behavioral rules, the second phase consists of validating and presenting them in a machine-readable format using Designs for Management™ ([DesignsForManagement.com](http://DesignsForManagement.com)). The latter is an SBVR-based software suite. It is used to ensure that the SBVR regulatory business vocabulary, and the regulatory guidance rules content are complete, consistent and compliant with the SBVR standard.

Designs for Management provides three ways to capture SBVR content:

1. Import from MS Word documents that use an SBVR SE template. This capability was used by the Object Management Group to import the SBVR and Date-Time Vocabulary standards as well as the SBVR EU-Rent Example.
2. Convert text in existing documents into SBVR terms and definitions with a right mouse menu option in the module: Smart Documentation™ editor.
3. Add new SBVR business vocabulary and business policy & rule entries using a forms interface that is supported by an AutoComplete function that inserts defined terms; definitions can be viewed by moving the mouse over the terms.

Once SBVR interpretations are entered in Designs for Management™, it validates them against the SBVR specification using several techniques. SBVR Terminological Dictionaries and Rulebooks, whether validated or under construction, can be displayed, printed and saved in HTML, MS Word and PDF formats. Moreover, SBVR model content can be exchanged in conformance with clause 2 of the SBVR specification [11] in the XML Metadata Interchange format. Appendix E, further describes the software architecture of Designs for Management™ and briefly presents its major modules, namely, Smart Glossary™, Smart Documentation™ and Clear Guidance™.

On-going work currently consists of importing the templates populated in phase 1 into Designs for Management™. This import/validation exercise helped identifying inconsistencies and/or omissions to be addressed by the SMEs. At the time of writing this paper, the curation and the consolidation of the vocabulary and the rules from phase 1 is not complete. However, a demonstration of a consistent SBVR vocabulary and rulebook for the EU Rent example 1.2 from the SBVR specification is publicly available in Designs for Management™ (at [DesignsForManagement.com](http://DesignsForManagement.com)).

#### 4.3 Future work

Part of the GRCTC research roadmap (cf. section 1) consists of creating a set of ontologies and data models for the financial industry using the developed SBVR vocabularies and rules as a starting point.

Currently, there is limited support for automated transformation. The SBVR-based vocabularies in the work done by LIPN, described above, were transformed to OWL ontologies, but this was done with LIPN's own software rather than SBVR-specific tools.

In general, transformation from SBVR to machine-readable ontologies involves manual intervention by SMEs. The process is well understood, but the transformation requires business decisions about business content that is not easily represented in formal information structures such as logical data models (other than as carried-forward text, perhaps styled in SBVR SE). Chapin and Hall [2] present a tutorial on transformations from an SBVR terminological dictionary to one or more logical data models. Tool support is currently experimental.

The OMG's Date-Time Vocabulary, developed as a foundation vocabulary to extend SBVR, has been transformed to an OWL2 Ontology [7]. Aspects of SBVR that have no OWL equivalents are carried forward as OWL annotations. This approach is one option raised in [8], which suggests several possibilities for separating SBVR content that is not easily represented in OWL.

There is less experience in transforming rules. SBVR has two kinds of rules: definitional (alethic) and behavioral (deontic). Definitional rules are the basis of constraints on associations in data models and ontologies and are addressed in the work referenced above. Behavioral rules govern or support activities. In SBVR they are declarative – they define states the business must be in (e.g. “a customer’s debt must not exceed his credit limit”). They can be directly supported in relational database systems by stored procedures and data base triggers, but many rules-based applications use production rules, which are procedural (e.g. “if the price of a new order would take the customer’s debt over his credit limit, then reject the order”). There is a fairly simple tutorial for transforming SBVR behavioral rules into production rules compliant with the OMG’s Production Rule Representation (PRR) standard, referencing the data model derived from the corresponding SBVR terminological dictionary.

Future work will, therefore, consist of leveraging the techniques previously mentioned to transform the SBVR interpretations to machine-readable ontologies, data models and rules.

### 5 Conclusions

This paper described a proof-of-concept on interpreting regulations using Semantics of Business Vocabulary and Business Rules (SBVR). This work was carried out as part of the research program of the Governance, Risk and Compliance Technology Centre in Ireland (GRCTC). After a brief description of the research context and the US Regulatory Framework, the proof of concept was detailed. First, the approach taken to limit to scope of interpretation within a regulatory document is described. Second, the division of labour between Subject Matter Experts (SMEs) is discussed. Third, the SBVR-based rule interpretation protocol was described and illustrative examples were provided. Finally, the software tools assisting the SMEs in their SBVR interpretation were presented.

The described approach is a step towards rendering the wide and complex spectrum of regulations more accessible. It tackles uncertainty and imprecision in regulations by combining Subject Matter Expertise and SBVR precision in representing domain knowledge. The produced vocabulary and guidance rules allow several practitioners to share their respective views on, and understanding of, the regulatory requirements while broadening their perception of the regulations. Capturing regulations in SBVR could also play a role in providing the regulators with national/ international view on the way the regulated perceive the regulations. However, the impact of the produced vocabulary and guidance rules is subject to their accessibility in terms of size and coherence. A large vocabulary including a high

number of fine grained and redundant concepts is likely more precise than a smaller, less expressive one, but might be less accessible in terms of complexity (harder navigation due to its size, contains more definitions, etc.). We believe that seeking an appropriate trade-off between accessibility and expressiveness is the key to a successful adoption of an SBVR based vocabulary and guidance rules.

### Acknowledgements

This work was supported by Enterprise Ireland and the Irish Development Authority (IDA) under the Government of Ireland Technology Centre Programme. The authors would like to acknowledge the major role played by the Subject Matter Experts, Leona O'Brien, John Lombard, Patrick O'Sullivan and Peter O'Sullivan, in building the described proof of concept, and thank them for their contribution.

### References

- 1 Bennett, Mike 2011. "Semantics standardization for financial industry integration," in *Collaboration Technologies and Systems* (CTS), IEEE, 23-27 May 2011, pp. 439-445, doi: 10.1109/CTS.2011.5928722.
- 2 Chapin Donald and Hall John: From SBVR to Logical Data Models. Data Management & Information Quality Europe Conference, London, Nov 3-6, 2008
- 3 Code of Federal Regulations (<http://www.gpo.gov/fdsys/browse/collectionCfr.action?selectedYearFrom=1&go=Go>)
- 4 Date Time Vocabulary Specification V1.0, Object Management Group (<http://www.omg.org/spec/DTV/1.0/Beta3>)
- 5 Federal Register (<https://www.federalregister.gov>)
- 6 Federal Register Final Rule "Bank Secrecy Act Regulations - Definitions and Other Regulations Related to Prepaid Access" (<http://www.gpo.gov/fdsys/pkg/FR-2011-07-29/pdf/2011-19116.pdf>)
- 7 Karpovic Jaroslav, Nemuraite Lina and Stankeviciene Milda: Requirements for Semantic Business Vocabularies and Rules for Transforming Them into Consistent OWL2 Ontologies. In proc. 18th International Conference, ICIST 2012. Springer Berlin Heidelberg, 2012.
- 8 Kendall Elisa and Linehan Mark: Mapping SBVR to OWL2. IBM Research Paper 2013. (<http://domino.research.ibm.com/library/cyberdig.nsf/papers/A9777F4EDB2552AE85257B34004C4EB3>)
- 9 Lévy François, Guissé Abdoulaye, Nazarenko Adeline, Omrane Nouha, Szulman Sylvie: An Environment for the Joint Management of Written Policies and Business Rules. ICTAI (2) 2010: 142-149
- 10 Lévy François and Nazarenko Adeline - Formalization of Natural Language Regulations through SBVR Structured English RuleML 2013
- 11 Nazarenko Adeline, Guissé Abdoulaye, Lévy François, Omrane Nouha and Szulman Sylvie: Integrating Written Policies in Business Rule Management Systems. RuleML Europe 2011: 99-113
- 12 Omrane Nouha, Nazarenko Adeline, Rosina Peter, Szulman Sylvie and Westphal Christoph: Lexicalized Ontology for a Business Rules Management Platform: An Automotive Use Case; RuleML America 2011; 179-192
- 13 Semantics of Business Vocabulary and Business Rules (<http://www.omg.org/spec/SBVR/1.1>)
- 14 United States Code (<http://uscode.house.gov>)

## **Appendix A - Glossary of Abbreviations**

| Acronym or Term | Meaning or Definition                               |
|-----------------|-----------------------------------------------------|
| <b>AML</b>      | Anti-Money-Laundering                               |
| <b>CFR</b>      | Code of Federal Regulations                         |
| <b>FIBO</b>     | Financial Industry Business Ontology                |
| <b>FIGO</b>     | Financial Industry GRC Ontology                     |
| <b>FIRO</b>     | Financial Industry Regulatory Ontology              |
| <b>FR</b>       | Federal Register                                    |
| <b>GRC</b>      | Governance, Risk and Compliance                     |
| <b>GRCTC</b>    | Governance, Risk and Compliance Technology Centre   |
| <b>ISO</b>      | International Organization for Standardization      |
| <b>OMG</b>      | Object Management Group                             |
| <b>OWL</b>      | Ontology Web Language                               |
| <b>OWL2</b>     | Ontology Web Language 2                             |
| <b>RDF</b>      | Resource Description Framework                      |
| <b>SBVR</b>     | Semantics of Business Vocabulary and Business Rules |
| <b>SBVR SE</b>  | SBVR Structured English                             |
| <b>SME</b>      | Subject Matter Expert                               |
| <b>USC</b>      | United States Code                                  |

## **Appendix B - Attributing cross-sections themes to SMEs from 76 FR 45403**

### **Part One – each section broken down into different categories**

§1022.380 Registration of money service business could be broken down in three themes or categories as follows:

1. Agency related provisions: 1022.380 (a) (3) and (4) (d)(1)(2) (E) (F) (G)(H)(ii)
2. Registration related provisions: (a)(1)(b)(1)(i)-(iii) (a)(2)(3)(4)(c)
3. Compliance related provisions: (e) (f)

§1022.210-Anti money laundering program for money service businesses could be broken down in five themes or categories as follows:

1. Definition related provision-1022.210 (a)
2. Required Standards for AML MSB programs-(b), (c), (d), (1)(i)(A)-(D), (d) (ii), second part of (iii) and (2)(i)-(ii), ( e).
3. Identity related provisions-(d)(iv)
4. Educational\Training related provisions-(2)(iii), (3)
5. Compliance date related provisions-(4)

§1022.320 Reports by money services businesses of suspicious transactions could be broken down in five themes or categories as follows:

1. Reporting and Identification related provisions-§1022.320 (a)(2)(i)-(iv), (a) (3)-(4).
2. Filing related provisions- (b)(2)-(3)
3. Retention of Records related provisions-(c )
4. Confidentiality/Disclosure of SARs related provisions-(d)(1)(i)-(ii) (A) (1)-(2), (B)(2), (e).
5. Other areas in this provision-(a)(1), (f) and (g)

§1022.420 Additional records to be maintained by providers and sellers of prepaid appears to be a self-contained provision

## **Part Two - Cross-referencing and consolidating break down of provisions between sections**

1. Agency provisions
  - §1022.380 (a) (3) and (4) (d)(1)(2)(i)- (ii)
2. Identity and Reporting related provisions
  - §1022.210 (d)(1)(iv)
  - §1022.320 (a)(2)(i)-(iv), (a) (3)-(4)
3. Compliance related provisions
  - §1022.380 (e), (f), 1022.320 (f)
  - §1022.210 (d)(4)
4. Registration related provisions
  - §1022.380 (a)(1), (b)(1)(i)-(iii), (b)(2)(3)(4), (c)
5. Definition related provision
  - §1022.210 (a)
6. Required Standards for AML MSB programs
  - §1022.210 (b), (c), (d) (1)(i)-(ii), second part of (iii) and (d)(2)(i)-(ii), (e)
7. Educational\Training related provisions
  - §1022.210 (d)(2)(iii), (3)
8. Filing related provisions
  - §1022.320 (b)(1)-(3)
9. Retention of Records related provisions
  - §1022.320(c ), 1022.420
10. Confidentiality/Disclosure of SARs related provisions
  - §1022.320 (d)(1)(i)-(ii) (A) (1)-(2), (B)(2), (e).
11. Other areas in this provision
  - §1022.320 (a)(1), and (g).
  - §1022.380 (a)(2)

## **Part Three - Regrouping themes and allocating to four SMEs**

**SME 1:** 1) Agency Provisions, 2) Identity and Reporting related provisions and 3) Compliance related provisions.

**SME 2:** 4) Registration related provisions, 5) Definition related provision and 6) Required Standards for AML MSB programs.

**SME 3:** 7) Educational\Training related provisions, 8) Filing related provisions and 9) Retention of Records related provisions.

**SME 4:** 10) Confidentiality/Disclosure of SARs related provisions and 11) Other areas in this provision.

## **Appendix C - An Example of SBVR Interpretation of §1022.210 (a)**

### **§ 1022.210 Anti-money laundering programs for money services businesses.**

(a) Each money services business, as defined by § 1010.100(ff) of this chapter, shall develop, implement, and maintain an effective anti-money laundering program. An effective anti-money laundering program is one that is reasonably designed to prevent the money services business from being used to facilitate money laundering and the financing of terrorist activities.

### **Business Rules**

It is obligatory that each money services business *develops* an anti-money laundering programme

It is obligatory that each money services business **implements** an anti-money laundering programme

It is obligatory that each money services business **maintains** an anti-money laundering programme

It is obligatory that each anti-money laundering programme **is effective**

It is obligatory that each anti-money laundering programme **prevents** money services business being used to facilitate money laundering and terrorist activities

### Verb Concepts

#### Modified verb concepts

anti-money laundering programme **is developed by** money services business

anti-money laundering programme **is implemented by** money services business

anti-money laundering programme **is maintained by** money services business

anti-money laundering programme **is effective**

Necessity: Each anti-money laundering programme **is reasonably designed**

anti-money laundering programme **prevents** money laundering and terrorist activities

#### Supporting verb concepts

money services business **is defined by** § 1010.100(f)

anti-money laundering programme **prevents** money laundering and terrorist activities

#### Noun concepts

anti-money laundering programme

money services business

money laundering and terrorist activities

## Appendix D - Using Confluence for Collaboration

The screenshot shows a web browser window for the Atlassian Confluence Semantic Repository. The URL is [https://grcthree.atlassian.net/wiki/display/SR/monetary\\_instruments](https://grcthree.atlassian.net/wiki/display/SR/monetary_instruments). The page title is "monetary\_instruments". The content area displays the following text:

**monetary\_instruments**  
Added by John Lombard, last edited by Peter O'Sullivan on Apr 18, 2013 (view change)

• Definition: monetary Instruments **include** currency and traveler's checks **in any form** and all negotiable instruments **that are either in bearer form, endorsed without restriction, made out to a fictitious payee, or otherwise in such form that title thereto passes upon delivery, and incomplete instruments signed but with the payee's name omitted, and securities or stock in bearer form or otherwise in such form that title thereto passes upon delivery.**

• Source: 31 CFR X subpart A §1010.100 (dd) (2013) <http://www.ecfr.gov/cgi-bin/text-idx?c=ecfr&SID=5a48cbc5b975eb90abdbf2f058d97249&rgn=div8&view=text&node=31.3.1.6.1.2.1.3.1&idno=31>

• Dictionary Basis:  
• General Concept:  
• Concept Type:  
• Necessity: **It is necessary that monetary instruments do not include warehouse receipts or bills of lading.**  
• Possibility:  
• Reference Scheme:  
• Note:

This screenshot presents the wiki editing interface. Here the definition of monetary instruments is displayed along with a necessity identifying what should not be considered as a monetary instrument. The left frame presents a tree list of vocabulary entries with several search capabilities.



Patrick O'Sullivan

The above definition could actually be formalised in a clearer way instead of dividing certain segments into implicitly understood concepts. One example of this is the term "an entity defined as an insured institutions section 401 of the National Housing Act". This could be repeated in other areas of the definition but for now the current definition will remain as a draft. A discussion has been held and the decision has now been taken to leave legislative sections in unstyled text. The actual title of the legislation/Act will though remain defined as an individual noun concept.

Another issue arises in relation to the terms [BankL association](#) and [SandL association](#) these are this author's own short hand terms for these two entities. Will a separate entry have to be produced for each of these? In any case these short hand terms would need uniform agreement with my colleagues. The long hand term will now be used instead for now.

A question arises here in relation to what the exact "possibilities" are in relation to this term. Yet again it seems to be a case of "possible necessities" arising.

[Reply](#) • [Edit](#) • [Delete](#) • [Like](#) • Mar 13, 2013



Elie Abi-Lahoud

Needs validation + corrections (individuals to be stylized, etc.)

[Reply](#) • [Edit](#) • [Delete](#) • [Like](#) • Apr 14, 2013



Peter O'Sullivan

I attempted some indenting to stylize the definition for greater clarity - needs more work/discussion.

[Reply](#) • [Edit](#) • [Delete](#) • [Like](#) • Apr 15, 2013



Elie Abi-Lahoud

1- yes we need to check if the categories are conformant to the original text

This screenshot illustrates an example of interaction between SMEs working on detailing the definition of Bank in the context of 31 CFR X.

## Appendix E - Using Designs for Management to Capture and Validate SME Interpretations

The Designs for Management™ software suite is hosted as a MS Windows Azure cloud service and uses an Azure-hosted SQL Server database. On the user's computer, it runs entirely within an Internet Browser and requires nothing additional to be installed on the user's computer. The four major browsers, Internet Explorer, Firefox, Chrome and Safari, are supported in all the environments for which they are available. This software suite is developed as a Visual Studio .Net Web Forms application, supplemented with Telerik cross-browser components and the Kendo UI HTML5/CSS3 JavaScript framework. It uses .Net Framework Forms Authentication for user authentication and authorization.

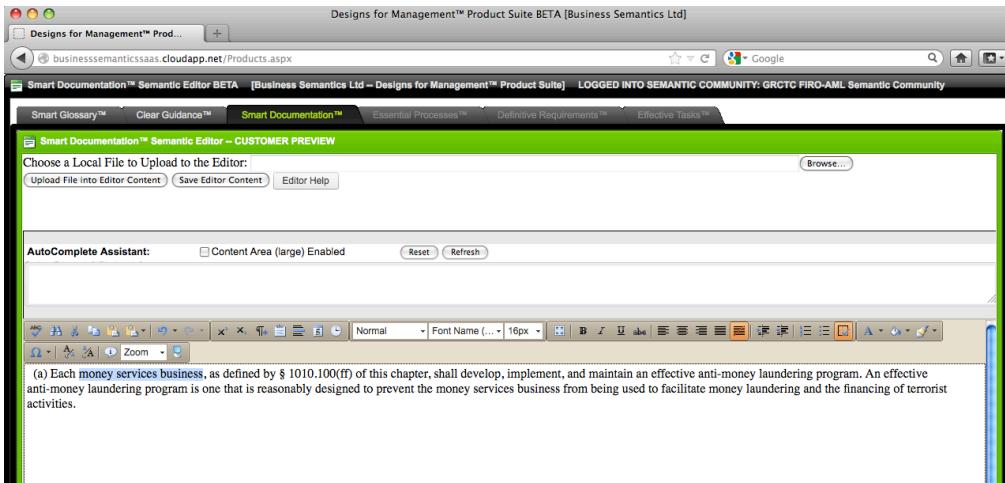
### Smart Glossary™

The screenshot shows the Smart Glossary™ interface. On the left, the 'MEANING and REPRESENTATION Explorer' pane lists various business terms such as 'Attorney General', 'bank', 'Bank Secrecy Act', 'beneficiary', 'beneficiary bank', 'broker or dealer in securities', 'business day', 'check casher', 'closed loop prepaid access', 'commodity', 'common carrier', 'contract of sale', 'Currency', 'dealer in foreign exchange', 'deposit account', 'domestic', 'established customer', 'execution date', and 'FinCEN'. On the right, the 'SBVR Basics - Core Language + Core Preferred Terms' editor pane displays a 'GENERAL NOUN CONCEPT -- in Core Language using Core Preferred Terms' screen. It shows a 'Select: originator' dropdown, a 'Conceptualized Within Subject Field:' dropdown, and a 'Primary Definition:' field containing the text 'The sender of the first payment order in a funds transfer.'. Below this, a note states 'Note: Changes in Styled Text Appear After Save'. At the bottom, there are fields for 'Preferred Term:', 'Community', 'Term is Not Easily Misunderstood by this Species', 'Community, and is Not a Synonym for any Concept belonging to this Semantic Community', 'Implicitly Understood:', and 'Term Context Subject Field:'.

This is a screenshot of *Smart Glossary*™. It is used to capture SBVR vocabulary elements. The terms identifying business concepts, their definitions, characteristics (if any) and other related elements are entered manually in the right frame. *Smart Glossary*™ provides the SME

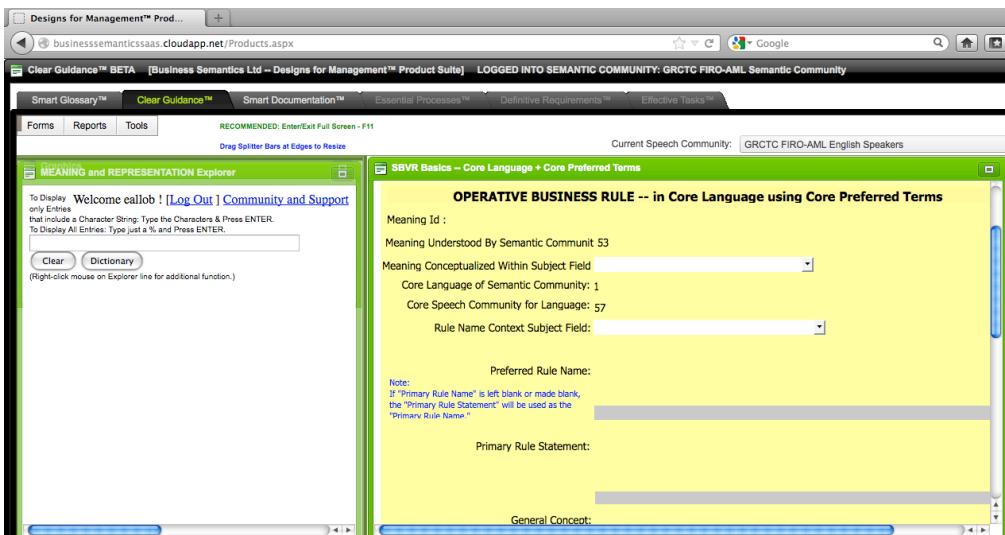
creating vocabulary entries with a list of SBVR predefined keywords and autocomplete functionalities taking into account previously entered vocabulary elements. The left frame displays an hierarchy of concepts in the vocabulary being edited.

### Smart Documentation™



This is a screenshot of *Smart Documentation™*. This module allows the SME to upload a regulatory text and start constructing the vocabulary by highlighting SBVR elements (general noun concepts, individual noun concepts, etc.) in the original text. Further refinement of the vocabulary entries identified in *Smart Documentation™* could be done in *Smart Glossary™*. This functionality was not used in the described proof of concept since the SMEs drafted their vocabularies in an MS Word template that was later imported into Designs for Management.

### Clear Guidance™



This is a screenshot of *Clear Guidance™*. The interface of this module resembles to *Smart Glossary™*. The left frame displays a list of operational rules (captured from the regulation in this case). The right frame allows the SME to build or edit the rule using built in SBVR keywords and elements from the previously created vocabulary.

### Seeing the Semantic Connections

A Meaning and Representation Explorer displays all or a selected part of the SBVR content and enables the browsing from any SBVR entry following the chain of semantic relations to anywhere in the SBVR content. The ability to choose at every point which kind of semantic relation(s) to browse next is available. In every context the full display of the entry, any styled term, other designation or verb concept wording can be seen

# Graph-based Editor for SWRL Rule Bases

Jaroslaw Bak, Maciej Nowak, Czeslaw Jedrzejek

Institute of Control and Information Engineering,  
Poznan University of Technology,  
M. Skłodowskiej-Curie Sqr. 5, 60-965 Poznan, Poland  
`{firstname.lastname}@put.poznan.pl`

**Abstract.** In this paper we present a prototypical implementation of a graphical tool for creating and editing (DL-safe) SWRL rules. The tool uses a graph-based approach to model rules expressed in the SWRL language. Rules are built from concepts and roles defined in an OWL ontology. Such a knowledge base can be visualised and edited in a user-friendly interface. Moreover, the presented tool provides methods for graphical representation of data and results of reasoning performed with the Pellet engine. We present a process of creating a knowledge base of family relationships as an example case. Perspectives of our future work are also presented.

**Keywords:** graphical rule representation, SWRL, ontology visualisation, reasoning

## 1 Introduction

The Semantic Web<sup>1</sup>, which is the extension of the World Wide Web, is still in active research and development. However, emerging technologies provide methods and standards for processing data according to the defined semantics. The semantics of data can be expressed by ontologies and rules which are of a special significance in the layered architecture of the Semantic Web. An ontology and a set of rules constitute a knowledge base of some particular domain. Using the knowledge base and data with an appropriate reasoner, we can perform reasoning tasks. Thus, additional knowledge can be inferred.

An ontology can be expressed using one of the OWL family of languages (OWL 1.1<sup>2</sup> and OWL 2 Profiles<sup>3</sup>), whereas a rule can be written in the Semantic Web Rule Language (SWRL) [1] or OWL 2 RL Profile<sup>4</sup>.

Despite the clear advantages and availability of semantics-based technologies, there are many software application areas where they do not occur or occur in a rela-

---

<sup>1</sup> <http://www.w3.org/standards/semanticweb/>

<sup>2</sup> <http://www.w3.org/Submission/owl11-overview/>

<sup>3</sup> <http://www.w3.org/TR/owl2-profiles/>

<sup>4</sup> Other appropriate languages also exist (e.g. RDF(S) for simple ontologies, RuleML for rules etc.) but currently we do not consider them in our work.

tively simple form (e.g. ontologies as vocabularies, rules as filters). The main reason for this is because ontologies and rules are too complex to handle by an ordinary user [2]. The process of acquiring ontology- and rule-based knowledge can be simplified with the use of a graphical representation and a user-friendly interface.

Since SWRL provides more powerful reasoning capabilities than OWL and some of the ontologies can be transformed into rules (e.g. Horn-SHIQ [3]) we focus on the development of a graph-based environment which will provide an easy way of creating and managing SWRL rule bases.

The main goal of this paper is to present a graph-based tool, in which an untrained user is able to construct a set of simple (DL-safe [4]) SWRL rules and to use them in order to obtain new (inferred) information according to the semantics defined in an OWL ontology. Both rules and the ontology constitute a knowledge base of a given domain. The ontology provides necessary concepts and roles, whereas the rules constitute additional knowledge mixing concepts and roles in a way which is not allowed in OWL. Additionally, a set of facts represents data. The constructed knowledge base and facts are expressed graphically in the form of directed graphs. The knowledge base can be applied to facts using a reasoning engine. After the inference process, a user gets the result, which is also represented graphically.

The paper is organized as follows. Section 2 presents the main overview of the proposed approach, a set of employed tools and related work. Section 3 describes a prototypical implementation with a demonstration of creating a simple knowledge base of family relationships. Section 4 contains concluding remarks and our future work.

## 2 Graph-based Representation of an Ontology and Rules

### 2.1 Existing Methods and Tools

Visualising data in the form of graphs is connected to a problem of knowledge representation (KR). Many investigators have created standardized notations for KR (Unified Modeling Language/Object Constraint Language (UML/OCL) [5], UML-based Rule Modeling Language (URML) [6], Object Role Modeling (ORM) [7], or SBVR<sup>5</sup> to name a few), however, so far many commercial tools tend to use their own standards. Other popular KR methods include: decision tables, decision trees and eXtended Tabular Trees [8]. Most commercial applications use those representations directly, or in a form of guided textual editors. In our approach, we aim to provide similar ways of representing both knowledge bases and rules. That is why the ORM approach, combined with a graph-based representation, seems to be sufficient to start with.

---

<sup>5</sup> Semantics of Business Vocabulary and Business Rules, <http://www.omg.org/spec/SBVR/1.0/>

There are a number of tools implementing graphical rules representations:

- Visual Rules<sup>6</sup> – it allows building of flow rules, decision tables and decision trees. It is focused on business logic and directs the flow of decision making by a defined life cycle. Events causing state changes are controlled by the rules. Both states and rules are converted and executed as Java code. Visual Rules lacks the ontology background, and focuses solely on business rules and decision flows.
- Drools Guvnor<sup>7</sup> – it provides many guided ways of creating rules: decision tables, rule flow and a single rule editor. It is a data repository for the Drools system. Guvnor offers many useful features: versioning and packaging of rules, models, functions and processes connected to knowledge bases and supervision of access to rule bases. We considered Drools as our reasoning module, but the Pellet reasoner is sufficient for the needs of SWRL rules.
- VisiRule<sup>8</sup> – it is an extension to Win-Prolog and it only allows creating decision flow models using a graphical paradigm. It offers a graphical representation of forward chaining rules with access to Prolog. VisiRule offers collaboration features; diagrams expressed in it may consist of nested parts. It is another platform designed for business flows rather than deductive rules.
- OntoStudio Graphical Rule Editor<sup>9</sup> – it is based on Object Logic [9], and operates on OL and SPARQL<sup>10</sup> queries. Diagrams here consist of concepts, their attributes and connections between them. It handles many known ontology formats (OWL, RDF, SPARQL, RIF) as well as UML 2.0. OntoStudio allows testing and debugging of rules. It does not allow the comparison of variables (comparisons between value and variable are allowed only). This approach is similar to ours, except that our tool visualises both ontology and reasoning results on a graph.
- CoGui<sup>11</sup> – it is a visualization tool for creating knowledge bases and conceptual graphs. It is based on the conceptual graph model introduced in [10]. The knowledge base of CoGui consists of hierarchies of concepts and relations, a set of individuals and a set of conceptual rules. It uses the CoGitan engine for inference tasks. The structure of graphs can be nested; relations are not restricted to unary or binary relations (n-ary relations are allowed). This tool does not support the OWL ontology format, nor does it operate on standardized rule notations.

---

<sup>6</sup> <http://www.bosch-si.com/technology/business-rules-management-brm/visual-rules-suite.html>

<sup>7</sup> <http://www.jboss.org/drools/drools-guvnor.html>

<sup>8</sup> <http://www.lpa.co.uk/vsr.htm>

<sup>9</sup> <http://www.semafora-systems.com/en/products/ontostudio/>

<sup>10</sup> <http://www.w3.org/TR/rdf-sparql-query/>

<sup>11</sup> <http://www2.lirmm.fr/cogui/>

- Protégé OWLViz<sup>12</sup> plugin – it creates a hierarchical view of the selected part of an ontology in the form of a directed graph. It does not allow manipulation of objects on the graph nor does it visualise SWRL rules.
- Protégé Axiomé<sup>13</sup> [11, 12] plugin – it supports visual rule base management, exploration, automated rule categorization, rule paraphrasing and rule elicitation functionality. It does not provide a way to create SWRL rules; instead it is designed to help users understand the meaning of rules. Axiomé can represent rules as a graph where each rule is represented as a node and direct edges between nodes indicate that SWRL atoms are shared by the rules.
- TopBraid Composer<sup>14</sup> – it is a visual modelling tool designed to create and manage ontologies in the Semantic Web standards. It is based on the Eclipse<sup>15</sup> platform and the Jena API<sup>16</sup>. TopBraid Composer offers drag-and-drop way of creating and editing OWL ontologies. It allows consistency checking and debugging of OWL Inference engine. Users are able to incorporate SPARQL rules (SPIN<sup>17</sup>) into the process of class definition to create some constraints.
- Snoggle<sup>18</sup> – a graphical SWRL-based ontology mapper. It creates directed graphs representing structures of source and destination ontologies and enables creation of mapping relations between concepts from both ontologies. Those mapping relations are then converted into SWRL rules.

## 2.2 Overview of the Approach

The main goal of this paper is to present a graph-based environment, in which a user can: load an ontology, create and edit SWRL rules, perform reasoning and obtain results. Moreover, an ontology, rules and data are represented graphically as directed graphs. Additionally, an ontology can be represented as simple (and calculated) taxonomies of concepts and both types of roles (datatype and object properties). As a result, we obtain a graphical representation of a knowledge base constructed from concepts, roles, rules and facts (data). The knowledge base can be easily understood by an ordinary user who tries to work with ontologies and rules. Our aim is to provide an easy-to-use and easy-to-understand tool which can be used in many domains where ontologies, rules and graphs can be employed to support a user's work.

The process of rule creation consists of creating two graphs which represents two parts of a rule: the body (left hand side) and the head (right hand side). In the presented approach, rules are of the following form: *if the body then the head*. Both the body

---

<sup>12</sup> <http://protegewiki.stanford.edu/wiki/OWLViz>

<sup>13</sup> <http://protegewiki.stanford.edu/wiki/Axiomé>

<sup>14</sup> [http://www.topquadrant.com/products/TB\\_Composer.html](http://www.topquadrant.com/products/TB_Composer.html)

<sup>15</sup> <http://www.eclipse.org/>

<sup>16</sup> <http://jena.apache.org/>

<sup>17</sup> <http://spinrdf.org/>

<sup>18</sup> <http://snoggle.semwebcentral.org/>

and the head consist of positive conjunctions of atoms that are defined in an ontology as classes (concepts), object properties (roles) and datatypes. Thus, the left hand side (LHS) of a rule should be perceived as conditional elements that need to be fulfilled in order to execute instructions written in the right hand side (RHS). The execution of a rule can add new statements to the given knowledge base in the form of new relations between objects and new classifications of them. For example, using rule (1) we can infer that a person which has a male child has a son.

$$Man(?y), Person(?x), hasChild(?x, ?y) \rightarrow hasSon(?x, ?y), Son(?y) \quad (1)$$

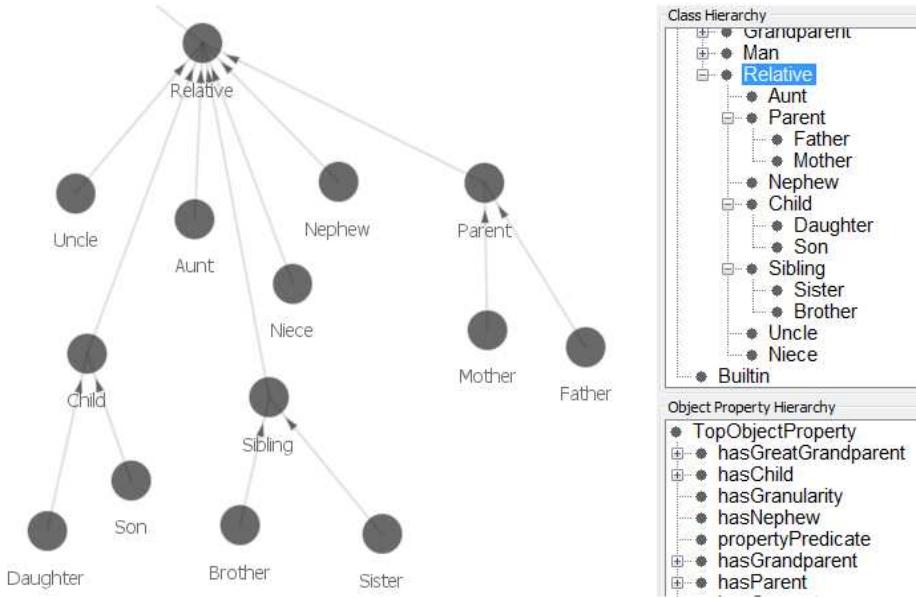
In rule (1) *Man*, *Person* and *Son* are OWL classes, *hasChild* and *hasSon* are object properties and  $?x$ ,  $?y$  are variables. By executing this rule we obtain a new relation between objects under both variables from rule (1) and a new classification of object under variable  $?y$ .

As mentioned before, we represent rules, an ontology and facts in a graphical form. Each of them is a different directed graph. Each graph consists of nodes and edges. The nodes are a graphical representation of OWL classes (or objects in data visualisation) whereas edges represent appropriate relations between classes (objects); or classes (objects) and datatypes. Usually, an object may belong to a number of OWL classes, for example an object of class *Son* belongs also to the following classes: *Man* and *Person*. In our method we decided to use the most detailed class, which is often represented as the most bottom concept in the taxonomy of OWL classes. The rest of the applicable classes are shown in a tooltip after moving the mouse above the object. Moreover, a user can choose which class she/he wants to see on a graph. The same approach is applied in the object and datatype property taxonomies. An example of choosing a visible class is presented in Figure 1.



**Figure 1.** Selecting a visible class is done by clicking on a class name from a popup menu.

When loading an ontology, we can obtain two kinds of visualization. The first one is a Protégé-like view of taxonomies as trees. We provide three trees: the taxonomy of classes, the taxonomy of object properties and the taxonomy of datatype properties. The second visualization type is a graph-based view in which taxonomies are represented as directed graphs. Since a (rooted) tree is a special kind of directed graph, the visualization in both types is very similar. The main difference between them is that, in the graph mode, we can manipulate the graph structure by using specialised layouts or by manual rearrangement. Both types of OWL classes visualisation are presented in Figure 2.



**Figure 2.** Visualisation methods of OWL classes.

Our graph-based editor supports the reasoning performed by the Pellet engine (see Section 2.3). Results are presented to a user as a new graph of objects or as a pair of graphs representing calculated taxonomies of classes and object properties. Moreover, a user can check the consistency of an ontology and verify results obtained from ontology- and rule-based reasoning.

### 2.3 Applied Tools

In the presented graph-based editor we apply the Semantic Web Rule Language with its syntax and semantics to read/write rules from/to an ontology. We employ the OWL Web Ontology Language version 1.1 as a way to express the semantics of a given domain. SWRL Built-ins [13] are used as comparison predicates between values of datatype properties or variables. Since we adapt SWRL as an OWL-based rule language we follow its semantics. As a result, negated atoms or disjunctions are not allowed. Moreover, we apply the *DL-safe rules* [4] approach, which considers decidable combinations of OWL DL and rule axioms. Decidability is preserved by forcing each rule to be DL-safe, which means that each variable is bound only to the individuals that explicitly occur in the assertional part (data) of the knowledge base. In other words, only facts that are explicitly stated can be used in the reasoning process.

We employed the OWL API<sup>19</sup> tool to parse and write OWL ontologies. The Pellet<sup>20</sup> engine is used as an OWL and SWRL reasoner. As a result of reasoning we can: check the consistency of an ontology and rules, calculate taxonomies, obtain potential

<sup>19</sup> <http://owlapi.sourceforge.net/>

<sup>20</sup> <http://clarkparsia.com/pellet/>

inconsistencies and infer new facts. In the editor, we can visualise an ontology before and after the calculation of taxonomies. Additionally, we can obtain a graph of facts before and after the reasoning process.

Visualization uses two Java libraries: Gephi<sup>21</sup> and Processing<sup>22</sup>. We use Gephi to manipulate graph structures. It is also responsible for managing the layout of the nodes on the graph. Nodes can be rearranged manually or placed according to their graph-based parameters (centrality, modularity, PageRank, etc.). We use Processing as a software sketchbook to create the views of an ontology structure and a set of facts, as well as to create the rule editor.

Our graph-based rule editor for SWRL rule bases is fully implemented in the Java language.

### 3 Graph-based Editor

#### 3.1 Rule Creation and Edition Method

Our graph-based editor consists of three tabs: ontology view, rule creation graph and instances view. The ontology view tab shows a visualisation of an ontology. The user can select a hierarchical structure of classes, datatypes or object properties to be visualised. Every edge in this view represents a *subClassOf* or *subPropertyOf* relation from the ontology. Our system proposes a calculated layout of classes, however this graph can be manually rearranged in order to improve the user impression and understanding. The hierarchy of classes is represented as grey circle nodes connected with edges. Structure of object properties is represented as blue square nodes connected with edges, whereas data properties are represented as green triangle nodes also connected with edges. All edges in the ontology view tab reflect the subsumption relation between two nodes.

The rule creation tab consists of 3 parts: conditions side, which represents the body of a rule; conclusions side, which reflects the head; and the class hierarchy presented in a tree structure. In order to create a rule, a user drags a class from the Class Hierarchy tree and drops it onto one of the rule sides (conditions or conclusions side). She/he is asked for a variable name or a value, which indicates the added object. Class concepts are presented as circles on the graph, with their class name and variable (value) as their labels. Both datatype properties and comparisons of variables can be added by right clicking on an object on the graph and selecting an appropriate option. The system limits datatype properties to those which can be linked with the selected object type (the selected class is in the domain of that datatype property or the domain constraint does not occur). Datatype properties are displayed in the form of a triangle connected by an edge with the corresponding object. The name of datatype is shown on the edge, and its value as a label of the triangle.

---

<sup>21</sup> <https://gephi.org/>

<sup>22</sup> <http://processing.org/>

**Table 1.** Representations of main elements in our Graph-based Editor for SWRL rules.

| Element                                         | Graph-based representation                                                                                          |
|-------------------------------------------------|---------------------------------------------------------------------------------------------------------------------|
| OWL Class                                       |  Son                               |
| OWL Class instance                              |  Sister(F31)                       |
| Object property between two OWL instances       |  Sister(F31) hasChild Mother(F21) |
| Datatype property between an object and a value |  Person(?x) hasAge 18             |

Relation between objects (object properties) can be added in a similar manner. After selecting a node and right clicking the other node, a list of available object properties is presented. After selecting one of them, it is displayed as an edge between selected nodes. In order to save a rule, the user needs to select an option from the top menu (*File*, then *Save as...*). A user can choose to save the ontology combined with the created rules.

The instances tab visualizes individuals (facts) stored in a knowledge base. They are represented as purple rhombs connected with each other by edges (roles from the ontology). Individuals can have datatype properties, which are visualized in the same way as in the rule creation panel, by triangles. After the reasoning, objects can belong to many OWL classes. This fact is impossible to represent on a static graph, however we present a method to solve this problem. After moving the mouse above an individual, a tooltip with all inferred classes appears. User can select which class should be visible on the graph as a default one.

The graphical representation of particular elements, which is applied in our editor, is presented in Table 1.

### 3.2 An Example Case

An example application of our tool is performed with an ontology describing family relationships. We slightly modified an ontology developed by Christine Golbreich presented in [14]. Her ontology is publicly available<sup>23</sup>. It contains the usual classes, e.g. *Person*, *Man*, *Woman*, *Child*, *Parent*, etc., and relationships within a family, e.g. *hasConsort*, *hasChild*, *hasParent*, etc.

---

<sup>23</sup> <http://protege.cim3.net/file/pub/ontologies/family.swrl.owl/family.swrl.owl>

The main difference between the original family ontology and our version of it, is the addition of:

- Classes: *Grandparent*, *Grandfather*, *Grandmother*, *GreatGrandparent*, *GreatGrandfather*, *GreatGrandmother*.
- Object properties: *hasCousin*, *hasGrandparent*, *hasGrandfather*, *hasGrandmother*, *hasGreatGrandparent*, *hasGreatGrandfather*, *hasGreatGrandmother*.
- Datatype property: *hasAge*.

Since the aforementioned new elements of the ontology are self-explanatory we do not provide more detailed descriptions. The modified version of the family ontology was then loaded into our editor. In the tool we created rules which are responsible to obtain instances of the following:

- Classes: *Grandfather*, *Grandmother*, *GreatGrandfather*, *GreatGrandmother*.
- Object properties: *hasCousin*, *hasGrandfather*, *hasGrandmother*, *hasGreatGrandfather*, *hasGreatGrandmother*.

In this paper, we present two rules created with our editor. Rule (2) asserts an instance of relation *hasCousin* which reflects that children of siblings are cousins of each other (*hasCousin* is defined as a symmetric property). Rule (2) is presented in Figure 3.

$$\begin{array}{l} \text{Person}(\text{?}x), \text{Person}(\text{?}y), \text{Person}(\text{?}w), \text{Person}(\text{?}z), \\ \text{hasParent}(\text{?}w, \text{?}z), \text{hasParent}(\text{?}x, \text{?}y), \text{hasSibling}(\text{?}y, \text{?}z) \\ \rightarrow \text{hasCousin}(\text{?}x, \text{?}w) \end{array} \quad (2)$$



**Figure 3.** Creation of rule (2).

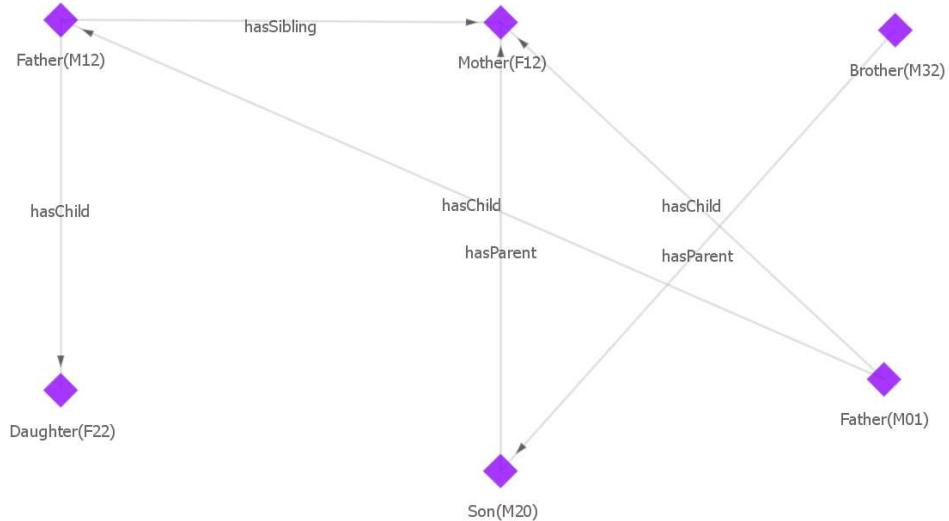
Rule (3) asserts an instance of class *GreatGrandfather* and an instance of a role *hasGreatGrandfather*. The rule expresses that a father of our grandparent is our great grandfather. Rule (3) is presented in Figure 4.

$$\begin{array}{l} \text{Person}(\text{?}x), \text{Person}(\text{?}y), \text{Person}(\text{?}w), \\ \text{hasGrandparent}(\text{?}x, \text{?}y), \text{hasFather}(\text{?}y, \text{?}w) \\ \rightarrow \text{hasGreatGrandfather}(\text{?}x, \text{?}w), \text{GreatGrandfather}(\text{?}w) \end{array} \quad (3)$$

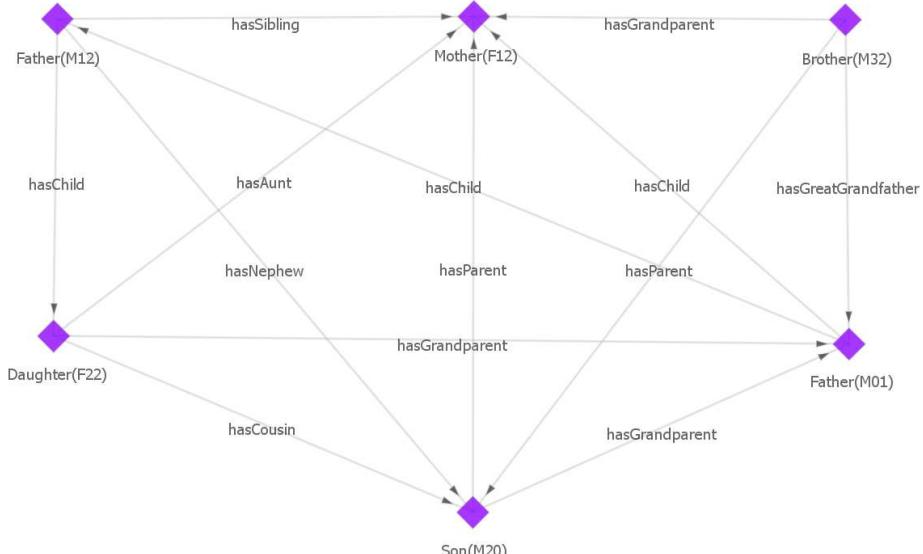


**Figure 4.** Creation of rule (3).

Created rules need to be applied to the set of facts in the ontology. After the reasoning process, executed by Pellet, a user obtains results presented in a new graph (in contrast to the graph before execution). Thus, new relations between objects and the classification of them are obtained. Figures 5 and 6 present two graphs: before reasoning (Figure 5) and after reasoning (Figure 6). These figures represent a part of the knowledge base to which rules (2) and (3) can be applied. Instances preceded by the letter ‘M’ represent men and instances preceded by ‘F’ represent women.



**Figure 5.** Graph of instances before reasoning.



**Figure 6.** Graph of instances after reasoning.

#### 4 Conclusions and Future Work

In this paper we have demonstrated a tool which supports graph-based creation and edition of SWRL rules. The tool provides a visualisation of an OWL ontology, SWRL rules and data. The graph-based representation is very convenient and intuitive. It is an initial implementation which supports the creation of SWRL rules in a graphical manner. The work presented in this paper is based on our previous experiences with graph-based representation of rules [15].

The developed graph-based editor can be used in many domains where ontologies, rules and graphs can be employed to support users in their work. Moreover, changes in a SWRL rule base can be made by business specialists without engaging an experienced programmer. As a result, the usual process of consultation between them is omitted or shortened in time. Thus, the tool can significantly increase their work's efficiency.

In further work, we will implement a query method for searching a knowledge base in a graphical manner. Moreover, we will provide a relational database interface. As a result, a semantic query will be posed into an integrated environment which will include a relational database, a set of rules and an ontology. In this case any graph containing nodes and edges could be entered as a search phrase. The reasoning engine will search the whole knowledge base for a given set of conditions, and return all objects that meet the specified requirements.

Another desired feature is to support OWL 2, which contains profiles designed for reasoning with rules and query answering, the RL and QL profiles respectively. A method of comparison between inferred and non-inferred knowledge bases is also planned.

Finally, we are going to make our tool available online for download and use with a free academic license (for non-commercial users) [16].

**Acknowledgement.** This work was supported by DS-MK 45-102/13 and 45-085/12 DS-PB grants.

## References

1. Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B., Dean, M.: Swrl: A semantic web rule language combining owl and ruleml. W3C Member Submission (May 21 2004), <http://www.w3.org/Submission/SWRL/>
2. Shotton D., Catton C., Klyne G., Ontologies for Sharing, Ontologies for Use, <http://ontogenesis.knowledgeblog.org/312?kblog-transclude=2>
3. Hustadt U., Motik B., Sattler U., *Data Complexity of Reasoning in Very Expressive Description Logics*, In IN PROC. IJCAI 2005, pages 466–471. Professional Book Center, 2005. (Cited on pages 5 and 37.)
4. Motik B., Sattler U., Studer R., *Query Answering for OWL-DL with Rules*. In Journal of Web Semantics, pages 549–563. Springer, 2004.
5. Object Constraint Language (OCL), v2.0. <http://www.omg.org/spec/OCL/2.0/>
6. UML-based Rule Modelling Language, <http://oxygen.informatik.tu-cottbus.de/reverse-i1/?q=URML>
7. Lukichev S., Jarrar M.: Graphical Notations for Rule Modeling. In: A. Giurca, D. Gasevic, and K. Taveter (Eds), Handbook of Research on Emerging Rule-based Languages and Technologies: Open Solutions and Approaches, IGI Publishing, 2009
8. Grzegorz J. Nalepa, Antoni Ligęza, and Krzysztof Kaczor. 2011. Overview of knowledge formalization with XTT2 rules. In Proceedings of the 5th international conference on Rule-based reasoning, programming, and applications (RuleML'2011), Nick Bassiliades, Guido Governatori, and Adrian Paschke (Eds.). Springer-Verlag, Berlin, Heidelberg, 329-336.
9. Michael Kifer, Georg Lausen, and James Wu. Logical foundations of object oriented and frame-based languages. *J. ACM*, 42(4):741–843, 1995
10. Sowa J. F., *Conceptual Structures: Information Processing in Mind and Machine*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1984.
11. Hassanpour S., O'Connor M. J., Das A. K., *A Rule Management and Elicitation Tool for SWRL Rule Bases*, 3rd International Rule Challenge at RuleML 2009, Las Vegas, NV.
12. Hassanpour S., O'Connor M. J., Das A. K., *Exploration of SWRL Rule Bases through Visualization, Paraphrasing, and Categorization of Rules*, International RuleML Symposium on Rule Interchange and Applications, Las Vegas, NV, 5858, 246-261, 2009.
13. SWRL Built-ins, <http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/>
14. Golbreich C., "Combining rule and ontology reasoners for the semantic web.", Rules and Rule Markup Languages for the Semantic Web. Springer Berlin Heidelberg, 2004. 6-22.
15. Nowak M., Bak J., Jedrzejek C., *Graph-based Rule Editor*, in Hassan Aït-Kaci, Yuh-Jong Hu, Grzegorz J. Nalepa, Monica Palmirani and Dumitru Roman, editors, RuleML2012@ECAI Challenge and Doctoral Consortium at the 6th International Symposium on Rules, Montpellier, France, August 27th-29th, 2012, volume 874 of CEUR Workshop Proceedings. CEUR-WS.org, 2012.
16. Demo site: [http://draco.kari.put.poznan.pl/ruleml2013\\_SWRLEditor/](http://draco.kari.put.poznan.pl/ruleml2013_SWRLEditor/)

# Advanced Knowledge Base Debugging for Rulelog<sup>\*</sup>

Carl Andersen<sup>\*\*</sup>, Brett Benyo<sup>\*\*</sup>, Miguel Calejo<sup>\*\*\*</sup>, Mike Dean<sup>\*\*</sup>, Paul Fodor<sup>†</sup>,  
Benjamin N. Grosof<sup>‡</sup>, Michael Kifer<sup>†</sup>, Senlin Liang<sup>†</sup>, and Terrance Swift<sup>§</sup>

**Abstract.** We present a novel approach to debugging expressively rich knowledge representation and reasoning (KRR) logic Rulelog. Rulelog is an extended form of declarative logic programs (LP) under the well-founded semantics, which allows higher-order logic formulas as axioms in combination with defeasibility mechanisms that include rule cancellation and priorities, along with default and explicit negation. Rulelog also supports strong knowledge interchange with all current major semantic web standards for logical KRR. Rulelog has been implemented in Flora-2 and Silk, both on top of XSB; and (less completely) in Cyc. The debugging approach described here is part of an integrated development environment, most fully implemented in Silk. The approach includes: reasoning trace analysis, based on tabled LP inferencing tables and forestlog; and justification graphs, which treat why-not and defeasibility as well as provenance. The reasoning trace analysis treats performance and runaway computations, including non-termination as well as classic subgoal-ordering issues that arise in database query optimization. Non-termination can be prevented entirely by leveraging the restraint (bounded rationality) feature of Rulelog. Revision/authoring of knowledge is interactive, based on a rapid edit-test-inspect loop and incremental truth maintenance.

## 1 Introduction

### 1.1 Rulelog

Rulelog is an expressively rich knowledge representation and reasoning (KRR) logic, based on a unique set of features that include:

1. defeasibility, based on argumentation theories (AT's) [21], i.e., *AT-defeasibility*. These theories provide features such as rule cancellation and priorities, along with default and explicit negation.

---

<sup>\*</sup> The order of the authors is alphabetical. Copyright © 2013 by the authors.

<sup>\*\*</sup> Raytheon BBN Technologies, USA

<sup>\*\*\*</sup> Declarativa, Portugal

<sup>†</sup> Stony Brook University, USA

<sup>‡</sup> Benjamin Grosof & Associates, LLC, USA

<sup>§</sup> CENTRIA, Universidade Nova de Lisboa, Portugal.

2. higher-order syntax, based on HiLog [1], and other meta-knowledge enabled by rule id's, i.e., *hidlog*;
3. classical-logic formula syntax, including existential as well as universal quantifiers, i.e., *omniformity* ([5] gives a compressed description); and
4. bounded rationality, based on *restraint* ([7] gives the basic *radial* form) which utilizes the undefined truth value of the well-founded semantics to represent “not bothering.”

Omniformity together with HiLog allows higher-order logic (HOL) formulas as axioms. The omniformity feature also includes and extends the Lloyd-Topor transformation [12] on rule bodies. Omniform rules are called “omni rules” or “omnis”, for short. The hidlog feature also includes reification, i.e., a formula can be treated as a term. The rule id's aspect of hidlog enables meta-info about axioms to be specified easily within the KB itself, e.g., meta-info about prioritization and about provenance. Other features include: object-based knowledge modeling (frame syntax), and aggregates (e.g., setof, sum, average, etc.).

Rulelog is the logic that was used in the Silk system [17] developed as part of Vulcan's Project Halo [8] advanced research effort, and grows out of earlier work on RuleML [15] and Semantic Web Services Framework [20]. A W3C Rule Interchange Format (RIF) dialect based on Rulelog is in draft [9], in cooperation also with RuleML.

The semantics of Rulelog is specified transformationally, into logic programs (LP) that are *normal*: those with logical functions and with default negation under the well-founded semantics. Using these transformations, Rulelog has been implemented most fully to date in Silk, which is architected as a Java layer that sits on top of Flora-2 [4]. Flora-2 sits in turn on top of XSB [22,19], which implements normal LP. Rulelog also has been implemented, less completely, in Cyc [2]. Both XSB and Flora-2 are available open source; Silk (i.e., the Java layer), purposed primarily as a scientific research effort, is proprietary.<sup>1</sup>

Rulelog supports strong semantic knowledge interchange with not only LP but also with first-order logic (FOL), and thus with all current major semantic technology standards for logical KRR, including RDF(S), SPARQL, SQL, XQuery, OWL-RL, OWL-DL, RIF-Core, and RIF-BLD, as well as with ISO Common Logic and thus SBVR.

Rulelog provides a good target for text-based authoring of knowledge [5], because of its ability to express defeasible HOL formulas as axioms.

Rulelog has been application-piloted in the domain of college-level biology for the task of question-answering in e-learning, in Project Halo. However, Rulelog is applicable to many other domains and tasks, e.g., that involve policies, contracts, law, and/or information integration.

---

<sup>1</sup> The Silk development effort, including maintenance, ended in April 2013.

## 1.2 Challenge of Debugging Knowledge in a Rulelog System

The expressivity of Rulelog raises a number of issues both in debugging and in understanding the behavior of Rulelog derivations.

The *justification problem* is a problem of explaining missing or unexpected (e.g., wrong or unintended) answers. This task is complicated not only by the types of inference used, but also by the transformations used to implement Rulelog reasoning. Answers to a query may be different than expected due to defeasibility or due to unexpected inferences made by the use of the higher-order reasoning provided by the HiLog component.

The *performance/termination problem* is a problem of indicating why a derivation has taken up more resources than expected — including non-termination as an extreme case. To explain the context of this problem, one of the major objectives of the Silk implementation of Rulelog was to be usable by knowledge engineers (KE’s) who are competent in logic, but who are not necessarily computer programmers. Such usage can give rise to knowledge bases constructed in a declarative manner, but with little attention to procedural aspects. Queries to such knowledge bases may lead to derivations that take longer than expected. In addition, as mentioned earlier, Rulelog uses logical functions both explicitly and implicitly (the later due to existential quantification, which is part of omniformity), and this use of logical functions can lead to non-termination.<sup>2</sup> While some performance issues can be addressed by optimizing compilers, users still need to understand what parts of a knowledge base give rise to poor performance or non-termination, so that these parts can be remedied.

Understanding Rulelog derivations is complicated by the semantics of Rulelog, which unlike first-order logic, is a fixed-point logic that supports recursive definitions. A Rulelog derivation, therefore, can be seen as a sequence of evaluations of recursive components in which the answers to a given subquery may be mutually dependent on answers to numerous other subqueries. Such a derivation can be partially modeled via a graph whose vertices are Rulelog atoms and whose edges are direct dependencies of the truth of one Rulelog atom on another. As will be shown later, such dependencies are implicit in our solution to the justification problem, but are explicit in our solutions to the performance/termination problem.

To partially address the justification and performance/termination problems, support is given by the tabled resolution of XSB, which serves as the computational underpinning of Rulelog in Silk. Although the details of tabling are quite complex, at a high level it handles recursive query evaluation by registering each tabled subgoal in a derivation. The first time a subgoal  $S$  is encountered in a derivation, a table is created for  $S$  and program clause resolution is used to derive answers for  $S$ , which are added to the table for  $S$  as they are derived. Subsequent calls to  $S$  need only resolve against answers in its table. In addition,

---

<sup>2</sup> FOL and normal LP also have this potential for non-termination in inferencing, for the same reason.

tabling keeps partial track of dependency information in order to determine the truth values of atoms in the 3-valued well-founded semantics. Although tables are central to the derivation strategy of Silk, they can also be examined by users to help understand features of a derivation.

A basic requirement in debugging is that the edit-test-inspect loop be rapid. This is addressed in Silk (and XSB and Flora-2) by the use of incremental methods for tabling in XSB and Flora-2. Such incremental tabling essentially constitutes truth maintenance.

The considerations so far indicate that a creative approach must be taken to understanding correctness, performance, and termination. Note that because of the complications of the transformations from Rulelog to normal logic programs, together with the technical details of tabled resolution, an interactive-debugger approach like that used in Prolog and other languages is impractical. Instead, we have developed a number of novel tools, each of which has an analytic component, which examines the internal structures of the engine and produces textual output, and a presentation component that makes the textual output more comprehensible to the user. The presentation components were incorporated into an overall graphical integrated development environment (IDE) for Silk, based on Eclipse, called Silklipse [6]. All of the tools described below have either been completed or are in the advanced stages of development.

## 2 Justification

Explanation of inferencing results, often called *justification*, has a long history in KRR, starting with the venerable *truth maintenance systems* [13]. The most practical previous approach to justification in LP is the method proposed for XSB’s tabled computations in [14]. Silk takes the previous ideas much further in several ways. First, it provides an attractive and easy-to-use visualization of the justification process through its Silklipse environment ([6] described an early version). Second, unlike XSB and other logic systems with explanation mechanisms, Silk supports defeasible reasoning through argumentation theories [21]. In the presence of defeasibility, a fact might be false or undefined because it is derived by the rules that are *defeated* by other rules. In those cases, it is necessary to explain how and why those rules were defeated. Silk provides such explanations. A key aspect is to explain why literals or rules have *false* (in the sense of NAF) truth value, i.e., *why-not*. Another key aspect is to explain how prioritization, or its lack, is involved. Third, unlike [14], justification is done not by transforming the original rules and blowing up the size of the knowledge base but through a separate small set of meta-rules, which is invoked on-demand when the user requests justification. Fourth, Silk supports rule-based transformation of the justification information: displaying it via automatically generated English text, and/or summarizing or otherwise reorganizing it.

Figure 1 shows a screenshot of a navigable justification in the Silk GUI. Some lines have been transformed into English text, while others have not been and

thus appear directly in Silk’s main logical syntax. E.g., the first line has been transformed into English text: “It is not the case that cell52 has a nucleus.” But lines 4 and 13 (among others) appear in the Silk logical syntax:

```
cell52 # red(blood(cell))
red(blood(cell))##eukaryotic(cell)
```

Here “#” means “is an instance of” and “##” means “is a subclass of.” Next we explain the icons that appear on the left in each line. “G” indicates a (sub)goal literal. “A” indicates an argument, i.e., a rule body supporting such a goal literal. Here, “argument” is in the sense of prioritized argumentation in defeasibility. Black bar (“—”) indicates a neg-argument, i.e., an argument for the neg (strong negation) of the goal literal. “F” indicates a fact, i.e., a literal that was directly asserted. “P” indicates prioritization info, i.e., that one rule’s tag has higher prioritization than another tag. Green indicates true, while red indicates false (in the naf sense). Green bang (“!”) indicates a undefeated (“live”) argument. Red down arrow (“↓”) indicates an argument that has been refuted, i.e., defeated by another conflicting argument that has higher priority. Plus (“+”) just to the right of “G” indicates that there are more arguments to see. When the “+” is black it indicates there are both pro (i.e., positive/for) and con (i.e., strong-negative/against) arguments to see; when green, it indicates there are more pro arguments but not more con arguments to see.

In this example, the relevant asserted logical rules in the KB can be described in English as follows:

```
cell52 is a red blood cell.
Eukaryotic cells have nuclei. (This rule has tag r1.)
Red blood cells are a subclass of eukaryotic cells.
Red blood cells do not have nuclei. (This rule has tag r2.)
r2 has higher priority than r1.
```

### 3 Trace-based Analysis

#### 3.1 Table dump: Examining Subqueries, Answers, and Rules

*Table dump* is a tool that produces a report on the subgoals that are among the most heavily called and the subgoals that have the most answers. This tool also lets the user know the *rules* that are the most heavily called ones. It thus helps to identify the bottlenecks in the knowledge base and then take measures such as to add appropriate guards to rules and to reorder subgoals within rules.

Figure 2 shows a screenshot of a navigable view of table dump info in the Silk GUI.

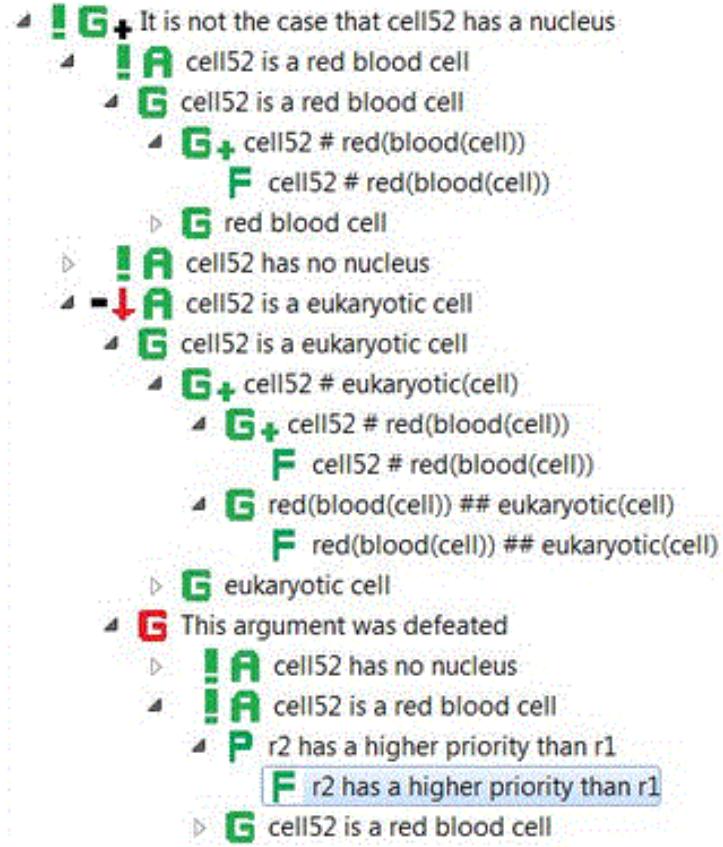


Fig. 1. Justification example

### 3.2 Forest logging

Although simple and powerful, the table dump approach lacks two main features needed to fully address the performance/termination problem. First, it does not provide an overview of how given subqueries in a derivation relate to one another through rules, and does not display information about the recursive components whose computation is central to a Rulelog derivation. Second, no information is provided about the order of events in a derivation, such as when subqueries were made, answers derived, and so on.

Within Silk, details of a Rulelog derivation can be reconstructed through another kind of trace-based analysis. XSB provides a mechanism to create a more dynamic trace or log of a derivation, called a *forest log* [18]. Using such a log, the structure of even very large recursive components can be analyzed, and

| Goal                        | Answers | Calls | Subgoal... | Subgoal ... | In Rule Body ID                                         |
|-----------------------------|---------|-------|------------|-------------|---------------------------------------------------------|
| ↳ \$(?A[?B->?C])            | 75      | 24392 | 343        | 24392       | "urn:uuid:750acf7e-3533-4a96-9457-1b1c3b0663f1"^^rifiri |
| > \$(?A[component(?C)->?B]) | 28      | 3852  | 70         | 0           | "urn:uuid:7c6c0510-9b35-4acf-b67d-99de5ddc52d3"^^rifiri |
| > \$(?A[next->0])           | 0       | 915   | 20         | 0           | "urn:uuid:7c6c0510-9b35-4acf-b67d-99de5ddc52d3"^^rifiri |
| > \$(?A[next->1])           | 2       | 915   | 20         | 0           | "urn:uuid:7c6c0510-9b35-4acf-b67d-99de5ddc52d3"^^rifiri |
| > \$(?A[next->2])           | 2       | 915   | 20         | 0           | "urn:uuid:7c6c0510-9b35-4acf-b67d-99de5ddc52d3"^^rifiri |
| > \$(?A[next->3])           | 2       | 915   | 20         | 0           | "urn:uuid:7c6c0510-9b35-4acf-b67d-99de5ddc52d3"^^rifiri |
| > \$(?A[next->4])           | 2       | 915   | 20         | 0           | "urn:uuid:7c6c0510-9b35-4acf-b67d-99de5ddc52d3"^^rifiri |
| > \$(?A[next->5])           | 2       | 915   | 20         | 0           | "urn:uuid:7c6c0510-9b35-4acf-b67d-99de5ddc52d3"^^rifiri |
| > \$(?A[next->6])           | 2       | 915   | 20         | 0           | "urn:uuid:7c6c0510-9b35-4acf-b67d-99de5ddc52d3"^^rifiri |
| > \$(?A[next->7])           | 2       | 915   | 20         | 0           | "urn:uuid:7c6c0510-9b35-4acf-b67d-99de5ddc52d3"^^rifiri |
| > \$(?A[next->8])           | 2       | 915   | 20         | 0           | "urn:uuid:7c6c0510-9b35-4acf-b67d-99de5ddc52d3"^^rifiri |
| > \$(?A[next->9])           | 2       | 915   | 20         | 0           | "urn:uuid:7c6c0510-9b35-4acf-b67d-99de5ddc52d3"^^rifiri |
| > \$(?A[next->10])          | 2       | 915   | 20         | 0           | "urn:uuid:7c6c0510-9b35-4acf-b67d-99de5ddc52d3"^^rifiri |
| > \$(?A[next->11])          | 2       | 915   | 20         | 0           | "urn:uuid:7c6c0510-9b35-4acf-b67d-99de5ddc52d3"^^rifiri |
| > \$(?A[next->12])          | 2       | 915   | 20         | 0           | "urn:uuid:7c6c0510-9b35-4acf-b67d-99de5ddc52d3"^^rifiri |
| > \$(?A[next->13])          | 2       | 915   | 20         | 0           | "urn:uuid:7c6c0510-9b35-4acf-b67d-99de5ddc52d3"^^rifiri |
| > \$(?A[next->14])          | 2       | 915   | 20         | 0           | "urn:uuid:7c6c0510-9b35-4acf-b67d-99de5ddc52d3"^^rifiri |
| > \$(?A[next->15])          | 2       | 915   | 20         | 0           | "urn:uuid:7c6c0510-9b35-4acf-b67d-99de5ddc52d3"^^rifiri |
| > \$(?A[next->16])          | 2       | 915   | 20         | 0           | "urn:uuid:7c6c0510-9b35-4acf-b67d-99de5ddc52d3"^^rifiri |
| > \$(?A[next->17])          | 2       | 915   | 20         | 0           | "urn:uuid:7c6c0510-9b35-4acf-b67d-99de5ddc52d3"^^rifiri |
| > \$(?A[next->18])          | 2       | 915   | 20         | 0           | "urn:uuid:7c6c0510-9b35-4acf-b67d-99de5ddc52d3"^^rifiri |
| > \$(?A[next->19])          | 2       | 913   | 20         | 0           | "urn:uuid:7c6c0510-9b35-4acf-b67d-99de5ddc52d3"^^rifiri |

**Fig. 2.** Table dump example

non-terminating derivations detected. This subsection first overviews forest logs, and afterwards discusses the analysis routines based on the logs.

The form of tabling used by XSB is called SLG resolution. The operational semantics of SLG evaluation (and hence a Rulelog derivation) can be modeled as a sequence of forests of trees, where each tree corresponds to a tabled subquery  $S$ , and represents the immediate subqueries that  $S$  produces along with any answers to  $S$ . In fact, each SLG operation is modeled as a function from forests to forests that creates a new tree, or adds a node or label to an existing tree.

Within XSB, SLG resolution is executed using a byte-code virtual machine analogous to that used by Java. An internal XSB flag can be set so that any byte-code instruction that corresponds to a tabling operation will log information about itself and its operands as a Prolog-readable term. For instance, if (tabled) subgoal  $S_1$  is called in the context of subgoal  $S_2$ , and it is the first time  $S_1$  is called in an evaluation, a fact of the form

$table\_call(S_1, S_2, new, ctr)$

is logged, where  $ctr$  (mnemonic for “counter”) is a sequence number for the fact. When a derivation ends or is interrupted, the log can be loaded into XSB and analyzed as a set of Prolog facts. Within XSB, the logging system is written at a very low level for efficiency. Turning on full logging usually does not slow down Flora-2 performance by more than 70-80%. XSB also provides routines to load

logs and index their facts on various arguments. Based on the logging libraries, logs containing hundreds of millions of facts have been loaded and analyzed.

### 3.3 Analyzing Recursive Components

Once a log has been loaded, a user may ask for an overview of a computation, which provides information on the total number of calls to tabled subgoals, the number of distinct tabled subgoals, the number of answers, and so on. In addition, the overview provides aggregate information on the number of mutually recursive components, and the number of subgoals in the components. Finally, the overview contains information indicating how stratified the negation (negation-as-failure, i.e., naf) was in a derivation by listing the total number of atoms whose truth value was undefined, along with a count of the various SLG operations used to evaluate well-founded negation.

Some derivations may give rise to very large recursive components—due to an unanticipated effect of higher-orderness, a knowledge base that is not sufficiently modularized, or other reasons. The analysis routines allow given recursive components to be examined, by listing the subqueries in the component, along with the pairs of calling and called subgoals within the components.

By examining this output, users can usually fix whatever problems gave rise to large recursive components. However for a very large component  $\mathcal{C}$ , the number of subqueries in  $\mathcal{C}$  may be on the order of  $10^5$  or more and the number of calling/caller pairs may be on the order of  $10^6$ . In such a case displaying every subquery or pair may be confusing at best. The analysis routines thus provide several abstraction routines that allow a user to coalesce similar atoms. For instance, if a component contained the subqueries  $p(a,X)$ ,  $p(b,X)$ ,  $p(c,X)$  ..., the analysis routines could use mode abstraction to coalesce all of these terms to  $p(\text{bound},\text{free})$ , or even predicate abstraction to coalesce all these terms to  $p/2$ . Recursive component analysis together with abstraction of atoms has been used to analyze the behavior of reasoning that was translated from Cycorp’s inference engine into the Silk implementation of Rulelog, for example.

### 3.4 Analyzing Runaway: Terminyzer

Runaway computation occurs when a query does not terminate or takes too long to come back with an answer. The first type of problem occurs typically due to the presence of function symbols and the second is largely due to computations that produce very large intermediate results most of which could be avoided with smarter evaluation strategies, such as subgoal reordering. The problem of determining whether a query is terminating or not has long been known to be undecidable, and the known sufficiency tests for them are weak for practical purposes. Cost-based optimization of LP via subgoal reordering has not been well studied for the case when recursion and logical functions are present.

The first tools we have developed for runaway give the user the means to interrupt the computation and inspect various statistics and the table dump, as described earlier. The user can also request the computation to stop after producing the desired number of answers.

One sophisticated diagnostic tool we have developed to tackle the non-termination problem is called the *Terminyzer* (short for “(non-)Termination Analyzer”) [11,10]. This tool relies on the previously described *forest logging* mechanism, which records the various tabling events that occur in the underlying inference engine XSB [19]. Among others, forest logging records when the different subgoals are called and when they receive answers. Terminyzer performs different kinds of analysis, such as *call-sequence analysis* and *answer-flow analysis*, and identifies the sequences of subgoals and rules that are being repeatedly called and in this way cause non-terminating computation.

Terminyzer also has a heuristic that may suggest the user to allow the system to reorder subgoals at run time and this avoid non-termination. For instance, in a composite subgoal  $p(?X,?Y)$ ,  $q(?X)$ , Terminyzer may detect that  $p(?X,?Y)$  is an infinite predicate. However, this infinity may be due to the infinite number of  $?X$ -values. If  $q(?X)$  binds  $?X$  to a concrete value first, non-termination will not occur. In such a case, Terminyzer may suggest the user to wrap the offending subgoal with a suitable delay quantifier—a novel facility supported by Flora-2 and Silk. For instance, if the above subgoal is rewritten as  $wish(ground(?X)) \sim p(?X,?Y)$ ,  $q(?X)$ , the system will not try to evaluate  $p(?X,?Y)$  unless  $?X$  is bound. If it is not bound, the evaluation of  $p(?X,?Y)$  is postponed and  $q(?X)$  will be evaluated next. If this binds  $?X$  then all is well and  $p(?X,?Y)$  can be evaluated next without a runaway. If  $?X$  is still unbound, some other subgoal may, perhaps, bind it, so  $p(?X,?Y)$  remains delayed. Only when the system determines that  $?X$  cannot be bound no matter what,  $p(?X,?Y)$  is submitted for evaluation. If this happens, the user would have to use the information provided by Terminyzer to decide whether the runaway is a mistake or is semantically justified. In the first case, this information will help the user fix the mistake; in the second, restraint could be used to prevent the runaway.

The presentation component of Terminyzer is integrated with Silkipse.

## 4 Restraint: Bounded Rationality and Prevention of Runaway

Another advanced way to control runaways is to use *restraint*, an approach to bounded rationality (and pragmatic incompleteness) that is semantically sound despite non-monotonicity [7]. With restraint, the semantics of inferencing—and thus corresponding computation—is limited in well-defined way; answers derived after the limits have been reached are given the truth value of *undefined*.

While Terminyzer is used for finding *mistakes* in user’s knowledge base, i.e., in situations when runaway computation is not intended, restraint is used when the knowledge base is *correct*. This typically occurs when the user query of

interest or one of its subqueries has an infinite number of answers, but only the first few need to be returned to the user.

One type of restraint is to limit a norm on subgoals, e.g., term size or depth, to be upper-bounded by a constant, which is called the *radius*. By setting the radius to a small enough value, radial restraint can be used to prevent runaways altogether.

There are several other useful types of restraint as well. In *skipping* restraint, conditions are specified (via rules) for when some other rules instances should be skipped, i.e., treated as having undefined truth value. In *unsafety* restraint: a literal that is (irremediably) unsafe with respect to NAF is treated as having undefined truth value. Likewise, an external-query (a.k.a. *sensor*) literal that is unsafe with respect to binding mode requirements is treated as having undefined truth value. In *unreturn* restraint, an external-query literal that does not return—e.g., due to network failure or server failure — is treated as having undefined truth value. In some situations, unsafety and unreturn restraints are preferable to throwing an error. Radial and skipping restraint are *voluntary* kinds of bounded rationality: the user specifies desired limits on reasoning via meta-rules knowledge. The limitation is cleanly semantic and specified as part of the knowledge base itself. By contrast, unsafety and unreturn are *involuntary*: limitations on reasoning are imposed by the circumstances of the inferencing mechanism and/or external environment.

All the above types of restraint straightforwardly combine with each other. They furthermore straightforwardly combine with the “anytime” approach to temporally bounded rationality [3,16]. In *anytime* restraint, a series of increasingly complete inferencing-result sets are computed and when a time limit is reached, the best one computed so far is returned. For instance a restraint radius is progressively incremented until the time limit is reached.

## 5 Overall Process of Knowledge Debugging

The tools we have described can be combined in a number of ways. The typical process of knowledge debugging goes as follows. A user runs a (test) query of interest. If the execution of the query does not take an unexpectedly/undesirably large amount of time or space, there is no performance/termination issue. The user looks at the answers to the query, and employs the justification tools to examine the explanation of those answers in terms of supporting conclusions and their associated assertions (rules knowledge). Along the way, the user looks for wrong or missing conclusions, and wrong or missing rules. The user may issue some other related queries as part of this investigation, and look at their explanations as well.

However, if execution of the query does take an unexpectedly/undesirably large amount of time or space, there is a performance/termination issue. At this point, the user needs to determine whether the runaway is due to non-termination or merely due to an inefficient computation. The first step in de-

terminating the culprit is to look at the table dump of the trace. If these show very large terms with deeply nested repeated function symbols, non-termination is the likely problem, and Terminyzer can be further employed to find the actual rule sequences that cause the problem. Otherwise, the user would use the table dump and the forest log tools to identify foci of computational effort, by looking for large tables (via table dump) or large recursive components (via forest log). The user next drills down progressively from the macroscopic (more aggregated and general) to the microscopic (more detailed and specific). Once sufficiently microscopic, the user then also employs the justification tools (as described above)—and/or employs restraint, especially in order to ensure termination (e.g., by limiting term size).

As usual in any kind of debugging, the above steps are iterated as needed.

## 6 Discussion: Scale, Skill

The debugging tools and process we have described have been used effectively for expressively rich Rulelog knowledge bases (KB’s) of substantial size, ranging up to tens of thousands of (non-fact) rules. “Expressively rich” here means with expressiveness beyond that of (normal) LP. Trace-based analysis has been used for forest logs ranging up to hundreds of millions of facts, as mentioned earlier.

An important direction for future work is how to empower Subject Matter Experts (SME’s), who lack skills in logic, to most effectively and efficiently debug knowledge, e.g., KB’s that they author via text-based techniques [5], including in collaboration or review with KE’s who do have skills in logic. This area requires considerable further research.

## 7 Acknowledgements

This work was supported by Vulcan, Inc., as part of the Halo Advanced Research project. Thanks to the rest of the Silk team, especially Paul Haley (Automata, Inc.) and Keith Goolsbey (Cycorp), for helpful discussions. Michael Kifer and Senlin Liang were also supported, in part, under the NSF grant 0964196.

## References

1. W. Chen, M. Kifer, and D.S. Warren. HiLog: A foundation for higher-order logic programming. *Journal of Logic Programming*, 15(3):187–230, February 1993.
2. Cyc. Cyc. <http://www.cyc.com> (project begun in approx. 1984), 2013.
3. T. Dean and M. Boddy. An Analysis of Time-dependent Planning. In *AAAI Conference on Artificial Intelligence*, pages 49–54, 1988.
4. Flora-2. Flora-2. <http://flora.sourceforge.net> (project begun in approx. 2000), 2013.

5. B. Grosof. Rapid Text-based Authoring of Defeasible Higher-Order Logic Formulas, via Textual Logic and Rulelog (Summary of Invited Talk). In *Proc. RuleML-2013, the 7th Intl. Web Rule Symposium*, 2013.
6. Benjamin Grosof, Mark Burstein, Mike Dean, Carl Andersen, Brett Benyo, William Ferguson, Daniela Inclezan, and Richard Shapiro. A SILK Graphical UI for Defeasible Reasoning, with a Biology Causal Process Example. In *Proc. of RuleML-2010, the 4th Intl. Web Rule Symp. (Demonstration and Poster)*, 2010.
7. Benjamin Grosof and Terrance Swift. Radial Restraint: A Semantically Clean Approach to Bounded Rationality for Logic Programs. In *Proc. AAAI-13, the 27th AAAI Conf. on Artificial Intelligence*, July 2013.
8. Halo. Project Halo. <http://projecthalo.com> (project begun in approx. 2002), 2013.
9. J. Sherman and M. Dean. RIF-SILK. <http://silk.semwebcentral.org/RIF-SILK.html> (project begun in approx. 2009), 2013.
10. Senlin Liang and Michael Kifer. A Practical Analysis of Non-Termination in Large Logic Programs. Technical report, Stony Brook University, 2013. <http://www.cs.stonybrook.edu/~sliang/iclp2013-tr.pdf>.
11. Senlin Liang and Michael Kifer. Terminez: An Automatic Non-Termination Analyzer for Large Logic Programs. In *PADL*, Berlin, Heidelberg, New York, 2013. Springer-Verlag.
12. J. W. Lloyd. *Foundations of Logic Programming*. Springer-Verlag, Berlin Germany, 1984.
13. D. McAllester. Truth maintenance. In Reid Smith and Tom Mitchell, editors, *Proceedings of the Eighth National Conference on Artificial Intelligence*, volume 2, pages 1109–1116, Menlo Park, California, 1990. AAAI Press.
14. G. Pemmasani, H.-F. Guo, Y. Dong, C.R. Ramakrishnan, and I.V. Ramakrishnan. Online Justification for Tabled Logic Programs. In *International Symposium on Functional and Logic Programming (FLOPS)*, number 2998 in Lecture Notes in Computer Science, pages 24–38, 2004.
15. RuleML. Rule Markup and Modeling Initiative. <http://www.ruleml.org> (project begun in approx. 2000), 2013.
16. S. Russell and E. Wefald. *Do the Right Thing: Studies in Limited Rationality*. MIT Press, 1991.
17. SILK. SILK: Semantic Inferencing on Large Knowledge. <http://silk.semwebcentral.org> (project begun in 2008), 2013.
18. T. Swift. Profiling Large Tabled Computations using Forest Logging. In *CICLOPS*, 2012. Available at <http://www.cs.sunysb.edu/~tswift>.
19. Terrance Swift and David Scott Warren. XSB: Extending Prolog with Tabled Logic Programming. *TPLP*, 12:157–187, January 2012.
20. SWSF. Semantic Web Services Framework. <http://www.w3.org/Submission/SWSF/>, 2005.
21. H. Wan, B. Grosof, M. Kifer, P. Fodor, and S. Liang. Logic Programming with Defaults and Argumentation Theories. In *Int'l Conference on Logic Programming*, July 2009.
22. XSB. XSB. <http://xsb.sourceforge.net> (project begun in approx. 1993), 2013.

# **Knowledge-based highly-specialized terrorist event extraction**

Jakub Dutkiewicz, Czesław Jędrzejek, Jolanta Cybulka, Maciej Falkowski

Institute of Control and Information Engineering,

Poznan University of Technology,

Pl. M. Skłodowskiej-Curie 5, 60-965 Poznań, Poland

{firstname.lastname}@put.poznan.pl

**Abstract.** In this paper we present a prototype of a system aimed at event extraction using linguistic patterns with semantic classes. The process is aided with an auxiliary tool for mapping verb statistics across messages. The sentence analyzer uses linguistic associations, based on VerbNet across the message and between messages' sentences to select semantic role fillers. We restrict ourselves to the coverage of one event type only – namely a kidnapping – and to two events template slots (semantic roles): a perpetrator and a person\_target (a human target). We designed rules involving semantic role filling using previous works on coreference. We used the Sundance parser and AutoSlog extraction patterns generator. Then we applied the semantic role filler and event resolution tool SRL Master. Our approach yields high performance on the MUC-4 data set.

**Keywords.** knowledge-based information extraction, semantic roles, terrorist event discovery

## **1 Introduction**

Event extraction is one of the most important tasks of knowledge discovery. It may be regarded as the core of knowledge-based systems that aim at providing the public (people, organizations, government agenda etc.) with condensed and filtered information concerning events. These events are described in texts written in natural language, thus the posted problem is related to the issue of information extraction (IE). Particularly, the task is to extract data concerning the described action (the event) and its arguments (called event roles). To implement the considered task different approaches are applied. They can be classified according to the provenance of the approach (pattern-based linguistic ones vs. classifier-based (statistical) methods) or to the ‘openness’ of it (fully open extraction vs. trained with the use of corpora one). The next important classification criterion is the nature of the context of the extraction, namely locality (one sentence only) or a larger context that takes into account consecutive sentences (a discourse). In many cases the hybrid methods are used that combine the different approaches. The open extraction systems (operating across one sentence context) scale well to the open corpora [1,3], especially that acquitted from the Web. But the most accurate IE systems are domain-specific, that use linguistic patterns and are somewhat trained with the aid of statistics. Our work follows the

latter approach in that we use a training domain-specific corpus. Let us characterize it briefly.

Due to a series of DARPA Message Understanding Conferences (MUCs), significant progress in pattern-based (NLP based) extraction technologies has been achieved. In this work we capitalise on the results of MUC-3 and MUC-4 ([10] that were held in 1991-1992) conferences, which used news reports corpus (MUC corpus) on terrorist activities in Latin America. MUC Conferences developed standards for evaluation, e.g. the adoption of metrics like precision and recall.

The goal of MUC was to extract from texts an information concerning 7 classes of terrorist events: Attack, Kidnapping, Hijacking, Bombing, Arson, Robbery and Forced Work Stoppage, plus several variations on each (for accomplished, threatened and attempted incidents). The process of extraction was augmented by the knowledge frames (event templates) generation. Every such template consisted of 24 attributes-slots. A document (a multi-sentenced message concerning an event) could be labeled with more than one template type. The MUC-4 corpus consists of 1700 documents, from which 1300 (DEV) were used in MUC-4 for training, 200 documents (TST1+TST2) were used as a tuning set, and the last 200 documents (TST3+TST4) were applied as the test set. The resulting knowledge base frames are called “key templates”. We filter out messages concerning one event type only, namely the kidnapping. Also, from among 24 slots we consider the two of them: a perpetrator and a person\_target.

The main contributions of the presented paper are:

- a method of comparing events to check whether a given two events are in fact identical or whether they are different, on the basis of semantic typing (semantic classes) of event’s arguments; it relies on using several types of rules, namely atomic, filling thematic role rules and whole events comparing rules; the method may be also used in coreference resolution
- an implementation of a corpus crawling tool that looks for words/phrases that lexicalize the kidnapping event
- additional lexical rules related to identification of victims and perpetrators.

The paper is organised as follows. Section 2 contains some notes concerning related works. In Section 3 our extraction method is presented. Section 4 describes a prototype implementation of the Word-statistics tool and its use. Section 5 demonstrates our information extraction results. In section 6 we give the concluding remarks and mention on our future work.

## 2 Related works

The main drawback of open information extraction [3] is that it uses the natural language features which do not classify (semantically type) arguments of an extracted relation. Additionally, in such methods the syntactic patterns (for example, regular expressions) do not match verb arguments that are distant from the verb phrase in a sentence. These are the features having the great negative impact on the ability to compare events (whether they are identical or not) described in the different sentences. In our work we avoid this drawback.

Authors of [4] use the language resources (dictionaries) to obtain sets of words that are relevant to the semantic class (a type of a verb argument). Having such extensionally defined types (semantic classes) they use them in the extraction process. In this work it is also shown how to apply such classes in the process of events comparison.

The method of event's comparison is also described in [5]. Here, the authors compare them (and extract their arguments) on the basis of head parts of noun phrases. For example, the events described in the following two sentences:

- 1) A customer in the store was shot by masked men.
- 2) The two men used 9mm semi-automatic pistols.

are in fact the same due to the fact that they use the same word “men”. In our approach the events may be unified (or differentiated) on the basis of the membership (non-membership) of two used (“linking”) words to the same semantic class. Also, it is not known, which pairs of sentences should be analyzed according to the event (we describe this problem later on).

### 3 The extraction method

#### 3.1 Preliminary definitions

At first, let us give some definitions of the terms used in the paper. They are as follows.

**Event** (denoted by  $E_n$ , where  $n$  stands for event's name) is an entity representing the event (conceptually it is an *occurred* that plays the central role in some situation, which represents a state of affairs) described in the text. The event is connected with a syntactic phrase (a verb phrase) that helps to identify it in a sentence, which is called an anchor. Also, there are some participants in the event – we identify them via thematic roles that are arguments of an anchoring phrase.

**Anchor** (marked as  $A_k$ , where  $k$  stands for an anchor name) is a verb or a verb phrase, which appearance in a derivation (i.e. a syntactically parsed sentence) triggers the process of recognition of an event (such as, for example, the kidnapping).

**Thematic role** (a semantic role label, marked as  $R_m$ , where  $m$  is a role name) is an entity representing an argument of a verb or a verb phrase (an anchor) denoting the event. For example, there may be such roles as Agent (in our considerations, a perpetrator), Patient (a victim), Instrument, Location, Time and others.

**Role filler** is a text phrase that instantiates a thematic role in the text (marked with the symbol  $R_pF_v$ , where  $p$  is a role name and  $v$  identifies a filler).

**Syntactic similarity.** Let us assume that the two argument function of syntactic similarity  $simsyn(W_1, W_2)$ , while given two words (or phrases) as arguments returns a binary value *true* or *false*. The function will return the *true* value if  $W_1$  and  $W_2$  have the same syntactic properties (i.e. number and gender), otherwise it returns *false*.

**Semantic class** (denoted by  $C_s$ , where  $s$  is a class name) is defined as an entity that is expressed by all of its verbalizations. For example, the verbalizations of the semantic class concerning kidnapping are  $C_{kidnapping} = \{\text{kidnap, seize, abduct, capture, intercept, take hostage}\}$ . It should be noted that we do not use all the meanings of the listed words, but only these fitting to a specific context.

**Atomic formula** is a triple of the form  $\langle \text{sub}, \text{pred}, \text{obj} \rangle$ , where  $\text{sub}$  means the subject of the sentence (and semantically it may play a thematic role  $R_m$ ),  $\text{pred}$  means the predicate (represents an event in terms of a certain semantic class  $C_s$ ) and  $\text{obj}$  means the object (semantically playing a role  $R_p$ ). An atomic formula could be considered as a rule representing a fact.

Let us illustrate the introduced notions with the exemplary message from DEV-MUC3-0018 (the text in this corpus is given in an upper case). We decorated the text with roles, role fillers, events and anchors. One of the considered sentences is:

OQUELI, LEADER OF THE NATIONAL REVOLUTIONARY MOVEMENT (MNR) AND HILDA FLORES, A GUATEMALAN SOCIAL DEMOCRATIC LEADER( $R_{\text{victim}}F_1$ ) WERE ABDUCTED( $E_{\text{kidnapping}}A_{\text{kidnapping}_1}$ ) AND KILLED IN JANUARY( $R_{\text{time}}F_1$ ) BY UNIDENTIFIED INDIVIDUALS( $R_{\text{perpetrator}}F_1$ ) IN GUATEMALA CITY( $R_{\text{location}}F_1$ ) AS THEY WERE HEADING TO THE LA AURORA AIRPORT.

Assume that there exists another sentence concerning the same event but with the new fillers for the victim and perpetrator roles:

IT TURNED OUT THAT POLITICIANS( $R_{\text{victim}}F_2$ ) WERE KIDNAPPED( $E_{\text{kidnapping}}A_{\text{kidnapping}_2}$ ) BY URBAN TERRORISTS OF FARABUNDO MARTI NATIONAL LIBERATION FRONT( $R_{\text{perpetrator}}F_2$ ).

After decorating the two sentences we are to check, whether two pairs:  $E_{\text{kidnapping}}A_{\text{kidnapping}_1}$  and  $E_{\text{kidnapping}}A_{\text{kidnapping}_2}$  concern the same event. We will show how to approach this issue in section 2.3.

We are motivated by VerbNet (VN) [1] thematic/semantic role methodology. VerbNet verb classes are organized according to the syntactic behavior of verbs. VerbNet uses 109 verb classes and 29 semantic role labels for arguments of the  $\langle \text{sub}, \text{pred}, \text{obj} \rangle$  triple pattern (which resembles our atomic formulae). We adhere to VerbNet semantics rather than to ontologies, because we are not aware of any publicly available ontology with adequate expressive power and rich verbalization of classes (ontological entities). We are in the process of using our CATIE ontology for the general extraction of facts from MUC-4 corpus [6].

We are interested in such event specifying verbs as: kidnap, abduct, seize (VN sense no 3), take (by force) (VN sense no 21, <http://verbs.colorado.edu/verb-index/vn/steal-10.5.php#steal-10.5>; sense number 3: take or capture by force or authority) belonging to class steal-10.5. However, instead of a role Agent [+animate | +organization] we need a role Agent/Patient [+person | +a group of persons | +organization]. In Unified Verb Index collection (VerbNet generalization) the word capture belonging to class steal-10.5.1 (<http://verbs.colorado.edu/verb-index/wn/wordnet.cgi?v3-0.capture.1.capture-2:36:00#1>) apparently has not been assigned a meaning kidnap.

### 3.2 Basic rules for identifying thematic roles

The next type of rules (besides the earlier described atomic formulae that represent facts) says that as the direct anchors we use all the interesting verbs ( $C_{\text{kidnapping}}$ ) in the past tense forms. Using a special function that retrieves a predicate of a given triple, namely  $\text{predicate\_of}(\langle s, p, o \rangle) = p$ , we denote such rules as triples of the form:  $\langle \text{predicate\_of}(\langle s, p, o \rangle), \text{tense\_of}, \text{"Past"} \rangle$ . We assume that  $\text{tense\_of}$  is a built-in predicate

representing verb tenses, i.e. “Past” and “Past Participle”. Another built-in predicate, named *voice\_of*, represents voice of a verb phrase, namely “active\_voice” and “passive\_voice”. The third built-in predicate, named *plays*, represents a fact concerning the deduced thematic role of a subject and an object of some triple (as it was assumed we only consider the agentive role (a perpetrator) and the patientive (beneficiary) role – a victim).

Now we are ready to give the rules to identify thematic roles of a predicate given in the past tense form. We are concerned with predicates expressed by verbs being members of a  $C_{\text{kidnapping}}$  semantic class.

The first rule states that for a given triple if its predicate is in the past tense and in the active voice then the subject plays the agentive thematic role of a perpetrator while the object plays the patientive thematic role of a victim (a kind of a person\_target). The rule (1) is as follows:

```
<predicate_of(<sub,pred,obj>), tense_of, “Past”> ∧
<predicate_of(<sub,pred,obj>), voice_of, “active_voice”> ⇒
<sub, plays, “agentive_role”> ∧ <obj, plays, “beneficiary_role”> (1)
```

The second (2) rule differs in the voice specification only that influences the order of the atomic formulae in the conclusion. The rule is as follows:

```
<predicate_of(<sub,pred,obj>), tense_of, “Past”> ∧
<predicate_of(<sub,pred,obj>), voice_of, “passive_voice”> ⇒
<sub, plays, “beneficiary_role”> ∧ <obj, plays, “agentive_role”>. (2)
```

### 3.3 Rules for event identification

In many cases information about certain roles and events is included in several sentences. Thus, matching different phrases to one thematic role constitutes one of a key tasks. We define a set of rules to identify such cases and eventually we either unify different events or differentiate them (the *are\_different* predicate). One of these rules bases on two sentences with a verb phrases denoted as two pairs containing an event and an anchor,  $E_{n1}A_{m1}$ ,  $E_{n2}A_{m2}$ . Each of these sentences contains a phrase that represents a filler of the same role, namely  $R_pF_{k1}$ ,  $R_pF_{k2}$ . To activate such a rule we need to find at least two sentences with these role fillers and event anchors. If we happen to find more than two sentences of such a kind, we need to analyze them in pairs. To describe such a rule, we need to define two predicates. The “*belongs\_to*” predicate is used if a given phrase belongs to a certain semantic class (this means that the main word in the phrase is a member of the considered class). The “*is\_equal\_to*” predicate decides whether either two semantic classes contain the same set of elements or role fillers are syntactically equivalent.

The process of analysis starts with searching of described pair of sentences. Let us denote the anchor and the role filler that were found in the first sentence as  $R_1F_1$  and  $E_1A_1$ , and the anchor and the role filler found in the second sentence as  $R_1F_2$  and  $E_2A_1$ . Once we have found these pairs we need to decide whether the described event anchors belong to the same semantic class (denoted as  $C_1$ ). This is formalized as:

$<E_1A_1, \text{belongs\_to}, C_1> \wedge <E_2A_1, \text{belongs\_to}, C_1>$ .

This basic condition should be considered as preemptive and its result decides if we are going to consider a pair of sentences as worth of executing this rule on.

The second part of the analysis starts with determining if role fillers belong to classes that are different, but there exists some relation between those classes. Furthermore we need to check if role fillers have the same syntactic properties. If those conditions are true, we can assume that phrases describe the same event. Additionally, there exists some relation among semantic classes, which may also be projected on role fillers (in particular it may be a subsumption). Let us formalize these considerations in the form of rule (3). In this rule, we mark “some relation” as a variable “?rel”.

$$\begin{aligned}
 & \langle R_1 F_1, \text{belongs\_to}, C_2 \rangle \wedge \langle R_1 F_2, \text{belongs\_to}, C_3 \rangle \wedge \langle C_2, ?\text{rel}, C_3 \rangle \wedge \\
 & \neg \langle C_2, \text{is\_equal\_to}, C_3 \rangle \wedge \text{simsyn}(R_1 F_1, R_1 F_2) \\
 \Rightarrow & \text{are\_the\_same}(E_1, E_2) \wedge (R_1 F_1, ?\text{rel}, R_2 F_2)
 \end{aligned} \tag{3}$$

However, if role fillers belong to the same class, but are different or role fillers have different syntactic properties, it is necessary to classify two events as different (4):

$$\begin{aligned}
 & (\langle R_1 F_1, \text{belongs\_to}, C_4 \rangle \wedge \langle R_1 F_2, \text{belongs\_to}, C_4 \rangle \wedge \neg \langle R_1 F_1, \text{is\_equal\_to}, R_1 F_2 \rangle) \\
 & \vee \neg \text{simsyn}(R_1 F_1, R_1 F_2) \Rightarrow \\
 & \text{are\_different}(E_1, E_2).
 \end{aligned} \tag{4}$$

We illustrate that rule with the following examples.

### Example 1

There are two consecutive sentences in the message:

- 1) John Smith ( $R_{\text{victim}} F_1$ ) has been kidnapped ( $E_{\text{kidnapping1}} A_1$ ).
- 2) President ( $R_{\text{victim}} F_2$ ) was taken hostage ( $E_{\text{kidnapping2}} A_2$ ) by unknown perpetrators.

The preemptive constraints are:

$$\langle "kidnap", \text{belongs\_to}, C_{\text{kidnapping}} \rangle \wedge \langle "take\_hostage", \text{belongs\_to}, C_{\text{kidnapping}} \rangle.$$

The following rule activation captures lexical associations between two neighboring sentences by pairing as similar each noun in the role of a victim (person\_target). This is similar to lexical bridge features used in [5]. The rule for those sentences goes as following:

$$\begin{aligned}
 & \langle "John Smith", \text{belongs\_to}, C_{\text{Person}} \rangle \wedge \langle "President", \text{belongs\_to}, C_{\text{Politician}} \rangle \wedge \\
 & \langle C_{\text{Person}}, \text{represents}, C_{\text{Politician}} \rangle \wedge \neg \langle "John Smith", \text{is\_equal\_to}, "President" \rangle \wedge \\
 & \text{simsyn}("President", "John Smith") \\
 \Rightarrow & \text{are\_the\_same}(E_{\text{kidnapping1}}, E_{\text{kidnapping2}}).
 \end{aligned}$$

As the result we obtain a fact (an atomic formula) of the form:

$$\langle "John Smith", \text{represents}, "President" \rangle.$$

The confidence of this rule could be measured in distance between the considered sentences (thus the distance is measured in the number of sentences). In particular this rule may be used only to analyze consecutive sentences.

## Example 2

We have three sentences, not necessarily in one document.

1. Ricardo Alfonso Castellar, mayor of Achi, ( $R_{\text{victim}}F_1$ ) who was kidnapped ( $E_{\text{kidnapping}}A_1$ ) on 5 January, apparently by Army Of National Liberation guerillas, was found dead.
2. Castellar ( $R_{\text{victim}}F_2$ ) was kidnapped ( $E_{\text{kidnapping}}A_1$ ) by a group of armed men.
3. A politician condemned kidnapping ( $E_{\text{kidnapping}}A_1$ ) of mayor of Achi ( $R_{\text{victim}}F_3$ ).

In this case we need to process sentences in pairs. First, we take sentences 1 and 2. We execute the rule and as a result we get the unification of  $E_{\text{kidnapping}}_1$  and  $E_{\text{kidnapping}}_2$ . This means that unification of  $E_{\text{kidnapping}}_3$  event, with both of the previous events would be redundant and we just need to clarify if  $E_{\text{kidnapping}}_3$  could be unified with any of those events. However, if  $E_{\text{kidnapping}}_1$  and  $E_{\text{kidnapping}}_2$  would not be unified, all events need to be compared separately. In this case we get three fillers of the victim role, furthermore the relation between those fillers is quite specific. That relation could be marked as “is\_substring\_of”. The left-hand side argument of this relation is always less expressive than its right-hand side and thus we could find the most expressive filler – “Ricardo Alfonso Castellar, mayor of Achi”.

Our method of unification is conceptually more powerful than the so far used for coreference resolution (for example in [11, 9]). But so far it is used only for establishing the agreement of semantic classes and also the noun-pronoun agreement features, that means features 2-3 and 8 out of 12 features proposed in [11].

### 3.4 Additional lexical rules

The examples shown in the previous subsection illustrate the need for rules that go beyond search of sentences with verb phrases corresponding to event related semantic class. To make the task of identifying event easier for the annotators, it is necessary to use the secondary semantic class containing words that are in a fuzzy relation to the core event term. We introduce a class:

$$C_{\text{fuzzy\_kidnapping}} = \{\text{disappear, release}\}$$

Following the Automatic Content Extraction (ACE) Programme guidelines:

An **event trigger** refers to the term within the event mention that most clearly expresses the occurrence of the event instance and is based on direct anchor – corresponds to  $C_{\text{kidnapping}}$ .

An **event mention** refers to the sentence within which an event instance is reported – corresponds to  $C_{\text{fuzzy\_kidnapping}}$ . An event can have multiple mentions associated with it. Apart from the sentence that initially reports the event, other coreferring sentences that contain anaphors of events (such as pronouns and definite descriptions of previously mentioned events) are taggable mentions of that event [9].

In general there always exists a direct connection between roles of events corresponding to  $C_1$  and  $C_{\text{fuzzy\_1}}$ . For example a victim of kidnapping directly corresponds to a subject of release or disappearance. To measure the confidence of fuzzy classes we look at the statistics of all words/stems in various part-of-speech forms, which directly or indirectly could indicate an event of kidnapping. They are words

corresponding to  $C_{\text{kidnapping}}$  and  $C_{\text{fuzzy\_kidnapping}}$  classes – verbs for kidnap (heads of verb phrases) in the past tense or attributive kidnapped, verbs in the past tense, verbs (infinitive, -ing form for a verb, gerund), nouns related to an act of kidnapping or a perpetrator, namely:

*kidnap, kidnapping, kidnapped, kidnapper*  
 stem *seiz, seized, seizing,*  
*abduct, abducted, abducting,*  
 stem *captur, capturing, captured,*  
*intercept, intercepting, intercepted,*  
 stem *releas, released, releasing,*  
*disappear, disappeared, disappearing,*  
*take/hold hostage.*

Finally, we apply coreference rules for both  $C_{\text{fuzzy\_kidnapping}}$  and  $C_{\text{kidnapping}}$  semantic classes.

### Example 3:

1. Ricardo Alfonso Castellar( $R_{\text{victim}}F_1$ ), mayor of Achi, was released( $E_1A_1$ ) on 15 January.
2. Kidnapping( $E_2A_1$ ) of Castellar( $R_{\text{object}}F_1$ ) was a brutal act.

Even though events  $E_1$  and  $E_2$  belong to different semantic classes we can unify specific role fillers within those events.

## 4 Word Statistics Tool

The process of designing pattern-based linguistic rules is a very tedious work, what constitutes the main disadvantage of such methods. To alleviate a burden we implemented a MUC Word Statistics Analyzer (Figure 1). The tool realizes several useful functions:

- 1) it presents graphically statistics of words across a document or a corpus
- 2) and it displays in two separate panels fragments of text pertaining to this statistics.

The considered in the paper extraction method relies on the quality of verb argument's typing (semantic classes). To obtain good results concerning the extensions of semantic classes  $C_{\text{kidnapping}}$  and  $C_{\text{fuzzy\_kidnapping}}$  we designed and implemented a statistic tool. It estimates the frequency of words (exactly, their stems) occurrences in the message or in the whole corpus. The tool also enables the analysis of sentences (or message) across which the stems appear. In the upper right corner of the screen given in Figure 1 the histogram is located that depicts the number of a word (stem) occurrences in the message and in the sentence. The exemplary message is shown in the lower left corner. In the bottom panel the list of sentences is located in which the stems with

different endings appear, for example: a stem kidnap, end words kidnapped, kidnap-  
per or kidnapping.

Summing up, by the quick inspection of the frequency of appearance of words and  
their correlation and varying the trigger term lists we can assess effectiveness of lin-  
guistic features.

## 5 Results

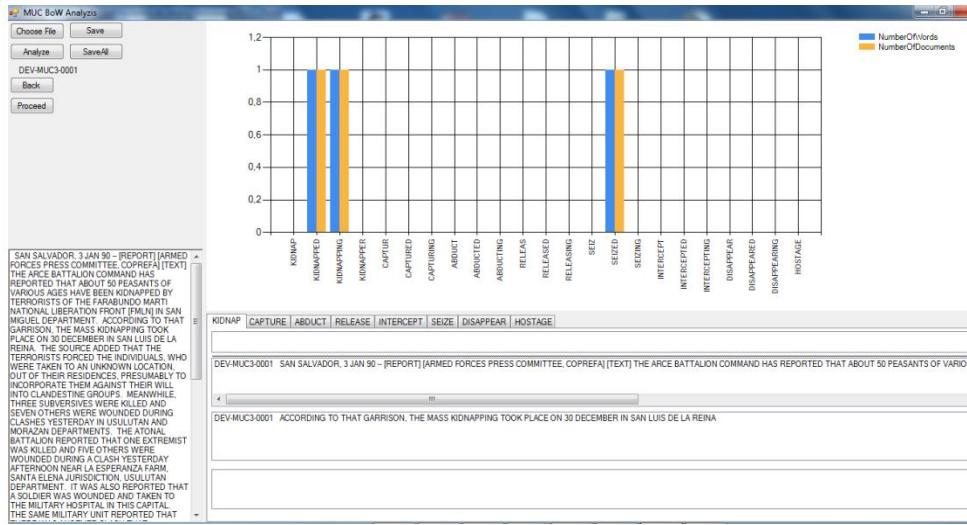
There are five overall IE related tasks that evolved from MUC.

- Named entity (NE) aims to extract all instances of persons, organisations, locations, dates, times, percentages and monetary entities.
- Coreference (CO) given a set of entities, this task aims to generate a set of entity coreference chains, such that mentions that coreference to the same entity appears in the same chain.
- Template element (TE) aims to extract all entity attributes. As an example, for the entity mention \ Castellar ", the aim is to extract its name ((Ricardo Alfonso Cas-tellar,"), type (PERSON") and descriptor ((the mayor of Achi").
- Template relation (TR) aims to extract all well-defined facts from each newswire text. In MUC-4 this was related to the knowledge frame (24 slots) of 8 terrorist type of events. In MUC-7 the facts were limited to relationships with organisations: employee of, product of and location of.
- Scenario template (ST) aims to extract pre-specified event information from anywhere in the given text, and relate it to the particular organisation and person enti-ties etc. involved in the event.

The figures presented in this table are based on the performance levels of systems participating in the MUC evaluations. More detailed figures can be found in Table 1.

**Table 1. MUC evaluation tasks**

| Year | Evaluation | MUC Tasks |        |        |       |           |
|------|------------|-----------|--------|--------|-------|-----------|
|      |            | NE        | CO     | TE     | TR    | ST        |
| 1991 | MUC-3      |           |        |        |       | F< 58%    |
| 1992 | MUC-4      |           |        |        |       | F<56% [9] |
| 1995 | MUC-7      | F< 94%    | F< 62% | F< 87% | F<76% | F< 51%    |



**Figure 1.** A snapshot of the results of the MUC Word Statistics Analyzer.

For many years these results have not been significantly improved. Only recently a significant progress [9,11] has been made.

There appear 159 events resolved as kidnappings out of 1700 documents as a result of assessment of the MUC-4 community [7].

We define the following numbers or word occurrences:

X1: at least a single occurrence of words from  $C_{\text{kidnapping}}$  or  $C_{\text{fuzzy\_kidnapping}}$

X2: only from  $C_{\text{kidnapping}}$  at least once

X3: only from  $C_{\text{fuzzy\_kidnapping}}$  at least once

X4: from  $C_{\text{kidnapping}}$  at least once and from  $C_{\text{fuzzy\_kidnapping}}$  at least once together

X5: only from  $C_{\text{kidnapping}}$  ending with -ed at least once

X6: only from  $C_{\text{fuzzy\_kidnapping}}$  ending with -ed at least once

X7: as in X1 from  $C_{\text{kidnapping}}$  at least once and from  $C_{\text{fuzzy\_kidnapping}}$  at least once, together ending with -ed

X8: only from {kidnap} set

X9: only kidnapped

Y1- Y9: occurrence as for X but for the set of documents that do not belong to a kidnapping event.

**Table 2. Statistics of MUC evaluation tasks**

|                                         | Number of occurrences |           |           |          |           |           |          |          |          |
|-----------------------------------------|-----------------------|-----------|-----------|----------|-----------|-----------|----------|----------|----------|
|                                         | X1                    | X2        | X3        | X4       | X5        | X6        | X7       | X8       | X9       |
| Documents concerning the kidnapping     | 144                   | 44        | 10        | 47       | 48        | 13        | 31       | 81       | 65       |
| Documents not concerning the kidnapping | Y1<br>394             | Y2<br>212 | Y3<br>128 | Y4<br>54 | Y5<br>156 | Y6<br>108 | Y7<br>30 | Y8<br>83 | Y9<br>32 |

The detailed analysis of these results will be presented at the Challenge event. We used the Sundance and AutoSlog systems for syntactic parsing and extraction patterns generation [12] together with Name Entity Extraction (with slightly modified dictionaries). Then we applied the semantic role filler and event resolution tool SRL Master.

**Table 3: The most effective patterns as determined from DEV 1-1300 texts**

|               |                   |                    |
|---------------|-------------------|--------------------|
| NP = EN       | VP(S)             | PP(BY) = Perp      |
| NP = EN       | VP(S)             | PP(IN) = Location  |
| NP = EN       | VP(S)             | NP(Date) = Date    |
| NP = EN       | PP(OF) = Victim   |                    |
| NP = EN       | NP(Date) = Date   |                    |
| NP = EN       | PP(IN) = Location |                    |
| NP = Victim   | PVP               | PP(ON) Date        |
| NP = Victim   | PVP               | PP(ON) Location    |
| NP = Victim   | PVP               | PP(BY) = Perp      |
| NP = Victim   | PVP               |                    |
| NP = Victim   | PVP               | NP(Date) = Date    |
| NP = Victim   | PVP               | PP(IN) Location    |
| NP = Victim   | PVP               | NP(Loc) = Location |
| Pron = Victim | PVP               | PP(BY) = Perp      |
| Pron = Victim | PVP               | PP(IN) = Location  |
| Pron = Victim | PVP               | NP(Date) = Date    |
| Pron = Victim | PVP               |                    |
| Pron = Victim | PVP               | PP(ON) = Date      |
| NP = Perp     | VP ActInf         | NP= Victim         |
| NP = Perp     | VP                | NP=Victim          |
| NP = Perp     | VP                | PP(IN) = Location  |
| NP = Perp     | VP                | PP(OF)= Victim     |
| NP= Victim    | AdjP              | NP(Date) = Date    |
| Pron          | PVP AuxVP         | NP = Victim        |
| NP= Victim    | AdjP              | PP(BY) = Perp      |

In Table 3 the meaning of symbols is the following: EN= event name (e.g. kidnapping, crime, etc.) – there are anchors, in all other patterns VP are anchors, NP = noun phrase, VP = verb phrase, PVP = passive verb phrase, AdjP=adjective phrase, PP= prepositional phrase starting with specific prepositions, Pron= noun phrase represented by a pronoun, Perp=perpetrator.

Effectiveness of our system is due to several factors:

- Our patterns are mostly triples, whether most previous works were based on syntax patterns consisting of 2 elements, see e.g Fig. 1 of [13].
- Non-triple patterns are more likely to generate extraction of nonrelevant patterns. For a pattern to be relevant we need to have at least either of two: location, date sentence part (first sought in a simple sentence, then in the complex sentence, and finally in adjacent sentences).
- One of the main contributions of this work is the introduction of VP(S) = supplementary verb phrase (particularly effective involving NP=EN are: *take place, claim responsibility, be responsible for, carry out*. To a lesser degree this helps to identify perpetrators and victims.

The correctness of extraction in this paper is providing all of the following kidnapping event roles (recall): *perpetrator individuals, perpetrator organizations, human\_target/victim, location and date*. These roles are narrower than 24 slots of the MUC-4 contest.

Table 4 presents the recall for the kidnapping events (here the same events in different documents are counted separately, similarly as for MUC-4 evaluation).

**Table 4: Recall for the kidnapping events for the MUC-4 development and test sets**

| Recall Measure [per cent] |          |
|---------------------------|----------|
| DEV set                   | TST sets |
| 78                        | 73       |

The recall numbers are significantly higher than in the MUC-4 contest (where the best contribution achieved around 60% for both precision and recall), but achieved for the easier task and for only one type of a terrorism event. They are also higher than in Table 3 of [5].

The system is presented at [http://draco.kari.put.poznan.pl/ruleml2013\\_Extraction](http://draco.kari.put.poznan.pl/ruleml2013_Extraction).

## 6 Conclusions

The recent wave of methods [11,9,8,3,4] is capable of significant improvement of extraction measures. The MUC Conferences provided benchmarks that decrease arbitrariness of a given method evaluation. For example open extraction system ReVerb gives a good precision but a poor recall [3]. We plan to apply against the full MUC-4 benchmark. The MUC Word Statistics Analyzer would be helpful for this task. There

are improvement possibilities in using the probable better syntax parser, Named Entity Recognition and using a wider set of coreference comparison.

Our choice of anchor words can be more optimal. In general, our patterns presented in Table 3 are more compatible with ontology-driven extraction than purely linguistic methods. Rather than use one general dictionary as used by most MUC related works, we can have lexicalization specific to ontology element. We are working in this direction.

**Acknowledgement.** This work was supported by the Polish National Centre for Research and Development (NCBR) No O ROB 0025 01 and DS 45-085/13 and DS-PB grants. We would like to thank Prof. Ellen Riloff for making Sundance and AutoSlog tools available to us, and Bartosz Zaremba for calculating some statistics.

## References

1. Bonial, C., Corvey, W., Palmer, M., Petukhova, V., and Bunt, H. A Hierarchical Unification of LIRICS and VerbNet Semantic Roles. Proceedings of the ICSC Workshop on Semantic Annotation for Computational Linguistic Resources (SACL-ICSC 2011), Sep, 2011.
2. Etzioni O., Banko M., Soderland S., and Weld D. S. 2008. Open information extraction from the web. *Commun. ACM* 51, 12 (December 2008), 68-74.
3. Etzioni O., Fader A., Christensen J., Soderland S., and Mausam: Open Information Extraction: The Second Generation. IJCAI 2011:3-10.
4. Huang, R. and Riloff, E.: Multi-faceted Event Recognition with Bootstrapped Dictionaries, Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT 2013).
5. Huang, R. and Riloff, E.: Modeling Textual Cohesion for Event Extraction, Proceedings of the 26th Conference on Artificial Intelligence (AAAI 2012).
6. Jedrzejek C., Cybulka J., CATIE ontology for the MUC-4 events extraction, in progress.
7. Lehnert, W.G., Cardie, C., Fisher, D., McCarthy, J., Riloff, E. and Soderland, S., Evaluating Information Extraction System, submitted to (*Journal of Integrated Computer-Aided Engineering*), 1(6), (1995), pp. 453-472.
8. Nakashole N., Weikum G., Suchanek F. M.: PATTY: A Taxonomy of Relational Patterns with Semantic Types. EMNLP-CoNLL 2012: 1135-1145.
9. Naughton M. Sentence-Level Event Detection and Coreference Resolution. School of Computer Science and Informatics, University College Dublin, PhD Thesis: October 2009.
10. Proceedings of the 4th Conference on Message Understanding, MUC 1992, McLean, Virginia, USA, June 16-18, 1992.
11. Soon, W. M., Ng H. T., and Lim D. C. Y. (2001). Learning approach to coreference resolution of noun phrases. *Computational Linguistics* 27 (4), 521–544.
12. Riloff E., Phillips M.. 2004. An Introduction to the Sundance and AutoSlog Systems Technical Report UUCS-04-015, School of Computing, University of Utah, <http://www.cs.utah.edu/~riloff/pdfs/official-sundance-tr.pdf>.
13. Patwardhan, S. and Riloff, E. (2006) "Learning Domain-Specific Information Extraction Patterns from the Web", ACL 2006 Workshop on Information Extraction Beyond the Document.

# **SBVR as a Semantic Hub for Integration of Heterogeneous Systems**

## **- A Case Study and Experience Report -**

Ling Shi<sup>1,2</sup>, Dumitru Roman<sup>3</sup>, and Arne J. Berre<sup>3</sup>

<sup>1</sup>Statsbygg, Pb. 8106 Dep, 0032 Oslo, Norway

<sup>2</sup>University of Oslo, Pb. 1072 Blindern, 0316 Oslo, Norway

ling.shi@statsbygg.no

<sup>3</sup>SINTEF, Pb. 124 Blindern, 0314 Oslo, Norway

{dumitru.roman, arne.j.berre}@sintef.no

**Abstract.** Extracting integration rules to handle semantic heterogeneity is one of the main challenges of achieving seamless connectivity between distributed systems. Semantics of Business Vocabulary and Rules (SBVR)'s machine and human readability and platform independence make it potentially suitable and interesting to study, as a central semantic hub of different systems. Semantic heterogeneity can be identified by comparing and analyzing vocabularies, fact models and business rules in the hub. Integration rules can then be extracted based on the semantic heterogeneity analysis. This article investigates and evaluates the usage of SBVR in heterogeneous systems integration. It provides a real-life case study and experience report on extracting integration rules based on an analysis of two Norwegian public sector's heterogeneous IT-systems modeled in SBVR.

**Keywords:** Integration rules, SBVR, Vocabulary, Fact models, Rules, Heterogeneity, Semantic heterogeneity, Ontology mismatch

## **1 Introduction**

The amount of data has been exploding in the last decades and therefore also the need for integrated information from distributed systems. There are numerous data sources available in organizational databases and on public information systems<sup>1</sup>. A typical integration scenario is that two heterogeneous systems A and B are built for different business purposes for different users at different times by different software developers using different information models. The two systems often have heterogeneous semantics, i.e. vocabularies, data structures and business rules are different. One of the main challenges of achieving seamless connectivity of related information from the different source systems is to extract the integration rules that can handle the heterogeneity of the source systems. Integration rules cover mainly what parts of re-

---

<sup>1</sup> <http://logic.stanford.edu/dataintegration/chapters/preface.html>

sources and properties from different source systems could be integrated, under what conditions, and how the transformations should be formed.

Extraction of integration rules is based on the analysis of the heterogeneity of the different systems. Therefore the quality of the heterogeneity description decides directly the quality of integration rules. In order to describe the heterogeneity, semantics of the source systems need to be described first. Machine readable semantic models are preferred compared to only human understandable semantic models because machine readability provides the possibility of utilizing automated reasoning and thereby the possibility of automation of the integration rules extraction. On the other hand, human readable semantic models make it easy to involve and interact with domain experts and decision makers. The domain experts can validate the semantics of the existing models and their documentation. The domain experts' involvement in capturing semantics is essential, especially when little existing semantic information could be found from the existing sources. A machine and human readable language is therefore preferred to model both the semantics of the systems and the heterogeneity between them.

Semantics of data models can be expressed in various forms ranging from schemas to system documentation, by annotation to ontologies, etc. Some well-known information modeling languages are evaluated below for the criteria of machine and human readability. XML provides structure but relies on e.g. schemas to provide semantics and the tree structure of XML is not always suitable for capturing various types of semantic relations. Ontology models support semantic integration by the machine readable meaning of terms [1]. However, ontology models are often not understandable by people without ontology training and lack the ability to address business rules and integration rules which are crucial to the integration challenges addressed in this paper. The Unified Modeling Language<sup>2</sup> (UML) is commonly used to model vocabularies and rules with the help of UML profiles, but its suitability for representing knowledge in a way that is easy to understand for non IT skilled users is questionable. Semantics of Business Vocabulary and Rules (SBVR) [5] is both machine readable and structured and human understandable. The date-time vocabulary<sup>3</sup> (DTV) is one of the SBVR examples of machine and human readable semantics. SBVR is potentially suitable as the modeling language for the central semantic hub of different systems.

Figure 1 below illustrates an SBVR-enhanced integration hub in the context of integration of system A and B which is a typical problem in practice as shown with a real-life example in Sections 2 and 3. The different source models as database schema (DB), UML, Web Ontology Language<sup>4</sup> (OWL) and Resource Description Framework<sup>5</sup> (RDF), together with domain experts' knowledge and system documentation, etc., from both source systems provide input to the central integration hub to be expressed as vocabularies, fact models and business rules. A transformation between SBVR and well-known modeling languages such as the OWL has been documented

---

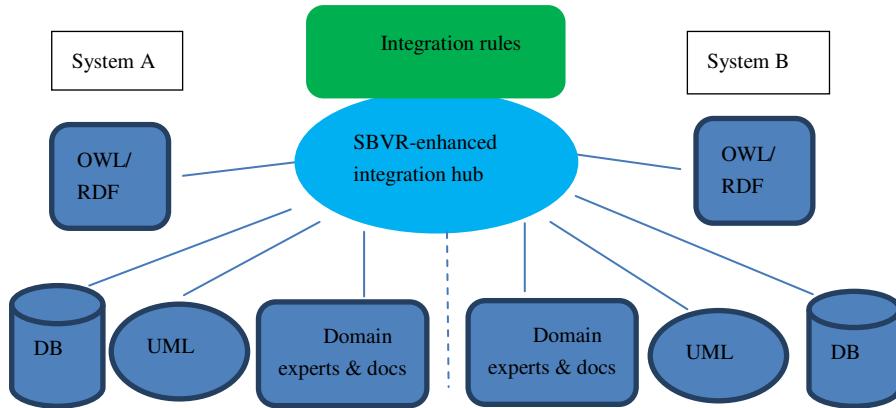
<sup>2</sup> [http://www.omg.org/gettingstarted/what\\_is.uml.htm](http://www.omg.org/gettingstarted/what_is.uml.htm)

<sup>3</sup> <http://www.omg.org/spec/DTV/>

<sup>4</sup> <http://www.w3.org/TR/owl-features/>

<sup>5</sup> <http://www.w3.org/RDF/>

in [2], and a transformation between SBVR and UML has been documented in [3, 7]. Heterogeneity can be identified by comparing and analyzing the SBVR models in the hub. The integration rules can be extracted based on the analysis result of heterogeneity.



**Figure 1.** SBVR-enhanced integration hub

The rest of the paper is organized as follows. Section 2 introduces a case study with focus on the integration challenges of two IT systems. Section 3 presents the realization of SBVR-enhanced integration hub in the case study. It first introduces the overall SBVR-based approach to address the identified challenges, and then reports on the realization of the hub in the case study presenting the transformation of source models to SBVR vocabularies, fact models and business rules, and extraction of the integration rules based on the heterogeneity analysis. Finally, Section 4 summarizes the work that has been done, draws lessons learned in using SBVR as a semantic integration hub, and suggests directions for future work.

## 2 Case Study

Statsbygg (SB) is the Norwegian government's key advisor in construction and property affairs, building commissioner, property manager and property developer<sup>6</sup>. This public sector administration company has more than 800 employees and manages approximately 2.7 million square meters of floor space<sup>7</sup>. Many of SB's business areas are dependent on up-to-date building information such as the buildings' name, address, status, tenant information, etc. The property management system is called Propman. Propman registers, among others, buildings and building related informa-

<sup>6</sup> <http://www.statsbygg.no/System/Topp-menyalg/English/>

<sup>7</sup> <http://www.statsbygg.no/FilSystem/files/omstatsbygg/aarsrapport/SBaarsmelding2011.pdf>

tion. Propman is client-server based and is built upon an Oracle relational database<sup>8</sup>. The available sources for building semantic models are database schemas, original system documentations and input from the domain experts.

The Norwegian national cadastral system called Matrikkel<sup>9</sup> stores the official data of buildings and building related information such as the buildings' name, usage area, status, address, land owner information, neighbor list and historical building changes. The Norwegian national map office is responsible for Matrikkel and it serves the data both as downloadable maps and map Web services. The available sources for building semantic models are class diagrams, a vocabulary and the Web services interfaces described with the Web Service Definition Language (WSDL).

Integration of the two systems is always a highly prioritized task among users and leaders in SB though there are integration difficulties, mostly at business and application layers. Building information such as address, status, municipal code, etc., is maintained manually in Propman though the information also exists in Matrikkel. Matrikkel is maintained frequently by municipal offices and other information providers. It often occurs that those types of data are out of date in Propman. One of the suggestions in SB to address the heterogeneity between the two aforementioned systems was to standardize the Propman with Matrikkel's building id and data structure for buildings. However, the heterogeneity of the two systems is inevitable since the two systems were originally built for different purposes and serve different users and customer groups. For example, SB administrates several properties abroad which are not registered in the Norwegian national Matrikkel. In such situations, standardized data structures do not guarantee interoperability at the scale expected by SB. Other examples of heterogeneity are shown in Section 3.

SB tried to update Propman data with building information from Matrikkel twice. In 2007 SB updated its building data based on the matching Matrikkel's building number in Propman though 25% buildings in Propman did not have the key registered. The quality of registered Matrikkel's building numbers was not good either. Some numbers were used several times on different buildings in Propman; some numbers did not belong to the right building. There were several kinds of errors after the updating process according to the feedback from the property managers. The second data updating was done in a half automated and half-manual way in 2012 and it tried to cover all the buildings including those without a Matrikkel's building number. Mismatches of data in the two systems were listed up and suggestions of modifications were delivered to the property management administrator who forwarded them to each property manager individually. Major reasons for this time and resource consuming process included the lack of a suitable vocabulary that could define terms like building, building's built area, building's address, etc., and also the lack of clear structures and rules that could be used to decide which building should be included in the register and what to do if mismatches between Matrikkel and Propman occurred. Each property manager then made an evaluation based on his/her domain knowledge and the corrections were therefore not 100% consistent.

---

<sup>8</sup> <http://www.oracle.com/us/products/database/overview/index.html>

<sup>9</sup> <https://www.matrikkel.no>

Neither of the integration approaches provided expected results and they were either time or resource consuming or error-prone. SB was looking for a more effective and systematic approach that could improve the data quality in Propman. The heterogeneity should be concretized and classified, and then the integration rules should be extracted to handle the heterogeneity problems with standardized processes, and to avoid the inconsistency caused by the missing rules. To achieve this goal, an approach has been developed as shown later in Section 3.1.

### **3 Realization of the SBVR Hub in the Case Study**

This section presents a real-life example of implementing the SBVR-enhanced integration hub for the two aforementioned heterogeneous systems. An overall approach is introduced first, and then the step by step realization of the approach is presented.

#### **3.1 Overall Approach**

The case was simplified to concentrate on the integration of buildings and related basic building information, though there are plenty of other kinds of building information that could be integrated such as addresses, owners, or land. The suggested approach focuses on the SBVR-enhanced integration hub and the extraction of integration rules based on a heterogeneity analysis. It is designed to include the following parts and steps:

- Part 1: Establishing the SBVR hub: Building the vocabularies, fact models, and business rules for the source systems.
- Part 2: Extracting Integration Rules: Identifying the mismatches of term definitions in the vocabulary, business rules and fact models; Building the integration rules after the mismatches.

The steps can be automated if proper tools are available, however manually approaches can also be used if necessary.

#### **3.2 Building the Vocabularies**

A template to define the data fields in the vocabulary was agreed upon first, and then two sets of SBVR vocabularies were established to separate the two system domains, the Matrikkel system and Propman respectively.

##### *Vocabulary Template*

SB has a cross domain working group called master data group whose main task is to define and maintain master data of the company. Domain experts from different business areas are invited to the group to work on their domain related terms and give input to the definitions. The group extended a template<sup>10</sup> from the Norwegian Agency

---

<sup>10</sup> <http://standard.difi.no/filearchive/2012-05-13-mal-begrepsbeskrivelser-1-0.pdf>

for Public Management and eGovernment (Difi)<sup>11</sup> to define the data fields included in the vocabulary as shown below.

- Difi fields:
  - Difi mandatory: Identifier, Recommended Term, Definition, Source, Discipline, Effective from, Responsible, Language, Expires, Reference to versions, Classification;
  - Difi recommended extension: Related concepts, Comment;
  - Difi administration documentation: Contact, Last Modified, Modified By, Status.
- SB extension fields: Synonym, System, Category, Approval Date.

#### *The Matrikkel Vocabulary*

The most challenging part of building a vocabulary was to reach a common sense on the definition in the organization or company. The Matrikkel system has already done part of the work and a definition page is provided on their website<sup>12</sup> with total of 92 terms dated September 1<sup>st</sup> 2012. Below is the original definition of the term “Building” in Matrikkel from the website.

**Table 1.** The original definition of the term Building in the Matrikkel system

|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Includes buildings and building changes.<br><br>The basis for the definition of buildings is Eurostat's definition: "Can be used separately listed for a permanent purpose and are suitable or intended to protect people, animals or things."<br><br>GAB system and DEK were two predecessors of the Matrikkel system. GAB system has from the beginning been associated with the Norwegian Standard NS 3940 Area and volume calculation of buildings as the basis for calculating area of buildings. This principle continued in the Matrikkel.<br><br>The area concept to be used in Matrikkel is called usage area. It is described in the standard and in the Matrikkel instructions. One precondition for a valid usage area is that the building meets the requirements of the standard of measurability. Based on these rules, carports (open garage), which only includes open space, fall outside of the definition. The same applies to tank constructions since such constructions are not normally accessible via doors or the like.<br><br>It indirectly implies that buildings in Matrikkel have usage area. It is however still open to register buildings that are otherwise not subject to registration. The area should then not be recognized as usage area but as an alternative type of land use.<br><br>All buildings with a usage area of 15 m <sup>2</sup> or more shall be recorded in Matrikkel, but smaller buildings can also be registered. |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

The above definition includes not only the definition of the term, but also the other valuable information as input to the vocabulary, fact model and business rules. The table below shows the original definition's input to the vocabulary template as described at the beginning of this section.

<sup>11</sup> <http://www.difi.no>

<sup>12</sup> <http://www.statkart.no/filestore/Matrikkelavdelingen/Foeringsinstruks/chapter02.html> (in Norwegian)

**Table 2.** Applying the vocabulary template to the term Building in Matrikkel

| Vocabulary Field | Value                                                                                                                                                                                                                                      |
|------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Identifier       | M.Building                                                                                                                                                                                                                                 |
| Recommended term | Building                                                                                                                                                                                                                                   |
| Definition       | Includes buildings and building changes.<br>The basis for the definition of buildings is Eurostat's definition: "Can be used separately listed for a permanent purpose and are suitable or intended to protect people, animals or things." |
| Source           | Eurostat, NS3940                                                                                                                                                                                                                           |
| Responsible      | The Norwegian National Map Office                                                                                                                                                                                                          |
| Related concepts | Usage area                                                                                                                                                                                                                                 |

The first sentence of the original definition indicates the term's relation to buildings and building changes. The original definition text also states a relation between Building and its attribute usage area. Both relations should be taken care of in building the fact models. Several business rules on what could be registered as a building in Matrikkel are included in the original definition text and they should be taken care of in building the business rules.

#### *The SB Vocabulary*

The master data group in SB arranged workshops with domain experts to collect the company's central terms and definitions. Some terms are cross-domain terms and domain experts had different understanding, focus and usage of the terms. The master data group worked as a negotiator in order to reach an agreement between the domain experts. When disagreement could not be resolved at this level, it had to be escalated to the higher administration level to reach a final decision. The result was an Excel sheet with more than 500 terms with names, definitions and related information.

For example, the definition of Building in SB is as follows:

**Table 3.** The original definition of the term Building in the Propman system

A building is a continuous building mass. In many cases, larger building masses can be defined as several buildings because the original building has been extended, and each gets its building number and building name. This is due to the preservation of historical data, due to different building status: e.g. building under construction, or has just been completed.

**Table 4.** Applying the vocabulary template to the term Building in the Propman system

| Vocabulary Field | Value                                     |
|------------------|-------------------------------------------|
| Identifier       | SB.Building                               |
| Recommended term | Building                                  |
| Definition       | A building is a continuous building mass. |
| Source           | Propman's system documentation            |
| Responsible      | SB                                        |
| Related concepts |                                           |

The above definition indicates that a normal criterion to identify a building with a unique building number is the continuous building mass. However, exceptions are allowed when a building is extended or under special conditions. These could be used as inputs to build the business rules.

### 3.3 Building the Fact Models

The Matrikkel system had UML models in its documentation and the database schemas of Propman were accessible. Those sources were transformed to SBVR fact models as described below.

#### *Matrikkel - Transformation from UML to Fact Models*

UML class diagrams could be transformed to fact models using some intuitive transformation rules based on ideas from [7, 8].

A single UML class may include name, attributes and operations [4]. The class name could be transformed to a term name in the fact model. The class attributes could be transformed to properties, one of the four special-purpose element of structure as described in [5]. The wording for property is “has”. There is no direct transformation from UML class operations to fact models though the operations could be analyzed and implemented as rules if applicable. The interpretation of operations to rules is not covered in this study.

The generalization between UML classes could be transformed to categorization, also one of the four special-purpose element of structure in fact models. The wording for categorization is “is a category of”. The generalization could also be transformed to classification if the subclass is an instance of the superclass, e.g., subclass Canada vs. superclass Country. The wording for classification is “is classified as”.

The composition could be transformed to composition in fact models, the wording for composition is “is composed of” or “is included in”. Other type of associations could be transformed to fact types.

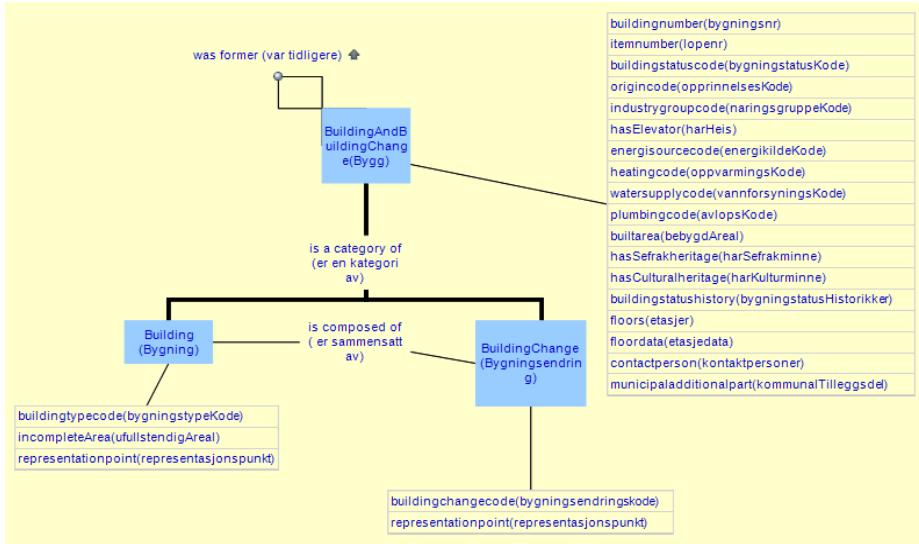
Due to the scope of the case study, the transformation rules used in the case study do not cover the transformation of, e.g., data types of attributes, multiplicity of attributes, multiplicity of associations, and aggregation in UML classes to fact models.

Figure 2 shows an example of a fact model modeled in a tool called FactXpress<sup>13</sup>. The model is generated from the UML model of building<sup>14</sup> in the Matrikkel system based on the transformation rules specified above. The text in parentheses is the original text in Norwegian.

---

<sup>13</sup> <http://www.rulearts.com/FactXpress>

<sup>14</sup> [https://www.test.matrikkel.no/matrikkel/docs/Domenemodell.html#analysemallene\\_bygg](https://www.test.matrikkel.no/matrikkel/docs/Domenemodell.html#analysemallene_bygg)

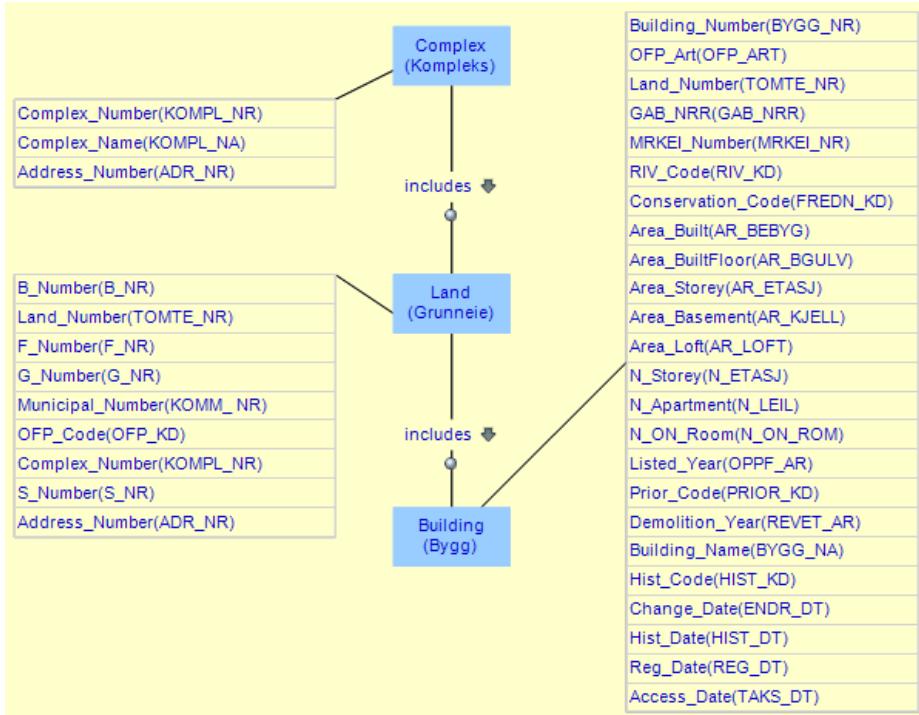


**Figure 2.** An SBVR fact model generated from a Matrikkel UML class diagram

#### *Propman - Transformation from Database Schemas to Fact Models*

The database schemas and system documentations were two of Propman's accessible source for semantics. Class diagrams were used as the bridge in transforming database schemas to fact models since transformation between database schemas and class diagrams can be automated by several database modeling tools. The database schemas were first imported to a class diagram using the existing reengineering tool in EA<sup>15</sup>. The legacy system Propman had not built relationships between the tables; thereby there were no associations between the classes. The information retrievable from the class diagram was therefore limited. There were other sources though, for example the system documentations. The statements that were helpful from the system documentations were, e.g., “One Complex includes one or more lands” and “Each land could build one or more buildings”. The figure below shows the fact model modeled based on the database schema generated class diagram and system documentations.

<sup>15</sup> <http://www.sparxsystems.com/products/ea/index.html>



**Figure 3.** The fact model of SB Building

#### Comparing Fact Models and Vocabularies

The fact models enrich the definitions in the vocabularies built as outlined in Section 3.2 by adding attributes of a concept and relations to other concepts. Moreover, fact models can also be used to validate the definitions from Section 3.2.

A common challenge in integration is using the same term for different meanings. A fact model will help with the identification of this kind of misleading information by showing attributes and relations of the concepts in a structured way. For example, the building definition in Table 2 states that a building “Includes buildings and building changes”. The fact model in Figure 2 shows three concepts, i.e. “BuildingAndBuildingChange”, “Building” and “BuildingChange”. The definition in the vocabulary should then be modified to reflect the fact models.

### 3.4 Building the Business Rules

#### *The Matrikkel Business Rules*

Some of the definitions in Matrikkel’s vocabulary also include rules. Those rules could be extracted from the textual definition, for example:

##### **R-M-bygg-4:**

Each building with usage area 15 m<sup>2</sup> or larger should be registered in Matrikkel system. Smaller buildings can also be registered, but is not compulsory.

### *The SB Business Rules*

The rules in SB were collected by interviewing the users and sending inquiries to responsible Propman system administrators. Here are some examples.

#### **R-P-bygg-1:**

SB registers both buildings larger than and smaller than 15 m<sup>2</sup>

#### **R-P-bygg-2:**

Each matriculated building in SB should be registered with a Matrikkel's building number.

## **3.5 Identifying the Mismatches of Vocabulary, Structures and Business Rules**

This section compares the vocabulary, structures and business rules of the two systems in the SBVR hub, and identifies the mismatches. Heterogeneity on resources usually could be confirmed already by comparing the term definitions in the two vocabularies. Further analysis could be done on comparing properties and fact types of each resource in the two fact models. Finally, the business rules defining the scopes or other attributes of resources could be compared to identify, e.g., the difference of the scopes.

### *The Vocabulary*

Comparing the definition of a term in the vocabularies provided direct indication whether the terms were identical or not. The sources of the term definitions were compared first and found out to be different for the term Building in the two systems. Then the definition texts were compared, Matrikkel's definition focused on a building's primary functions and SB's definition focused on the physical building mass. The term Building was therefore not identically defined in the two systems which means further analysis was necessary to identify the mismatches on the structure and business rules levels.

### *The Structures*

Fact models and their visualization provide an intuitive way of analyzing the structures. The properties and fact types of each resource in the two models were considered. One of the heterogeneity types is terminological heterogeneity [6]. It occurs when two properties with different names deal with the same information. For example, the builtArea is a data property of Building in Matrikkel, while Area\_Built is a property of Building in SB. This is a possible source of information duplication and mismatching. The integration rule should therefore address the issue by defining which system is the original source of the data property. For example, in this case, SB has the responsibility to report the Area\_Built value to the Matrikkel system. Therefore SB's Area\_Built is the original source of this data property.

### *The Business Rules*

Comparing the business rules of a specified term provided further information on semantic similarities and differences. The example below shows the different policies on what should be registered as Building by Matrikkel and SB. Those rules mismatch is caused by the heterogeneity in coverage of the resource Building. The heterogene-

ity in coverage is a subtype of conceptual heterogeneity as classified in [6]. The difference should be handled later in the integration rules.

### 3.6 Building the Integration Rules

This paper covers terminological and conceptual heterogeneity though other types of heterogeneity should be included in the identification and classification work in Section 3.5. The integration rules discussed below addresses the two types of heterogeneity.

An integration rule can be formed using a template like this: *What part of resource/property X in source system A should integrate with what part of resource/property Y in source system B with which integration keys under which conditions?*

To handle the terminological heterogeneity, an integration rule should define the original source of a data property in case of duplications. For example the following integration rule could solve this type of heterogeneity. This kind of integration rule can be interpreted as an ontology rule using, e.g., rdfs:subPropertyOf or owl:sameAs. The study of interpreting integration rules to ontology rules falls out of the scope of this paper.

**R-Int-bygg-1:**

*A Building's property Area\_Built in Propman is the same as a Building's property Built\_Area in Matrikkel with the integration key "Matrikkel's Building number".*

One alternative way to handle the “Difference in coverage” type of conceptual heterogeneity is to define the overlapping part as integration part. For example, the Building in Matrikkel and the Building in Propman are overlapping when both have the same Matrikkel's building number, then the integration rule below defines that only building with Matrikkel's building number can be integrated.

**R-Int-bygg-2:**

*Each building with Matrikkel's buildingnummer in SB can integrate with a building in Matrikkel on Matrikkel's building number.*

## 4 Conclusion and Further Work

The approach presented in this paper focused on extracting robust integration rules based on the heterogeneity analysis in an SBVR integration hub. Semantics of source systems could be modeled in or transformed to SBVR, and heterogeneity of different source systems could be identified by comparing those elements in the hub. Integration rules could then be extracted based on the identified heterogeneity. The information in the hub was understandable directly to both domain experts and IT personals. In this way, domain experts were able to follow the process all the way and their interaction with integration software developers was faster and more effective since they used the same mechanisms for communication. This also made software documentation easier since most of the explanations on why and how the integration rules

are extracted was already documented in the process in a human understandable language.

Currently, the transformation from source systems to SBVR hub is manually done and information could be lost in the transformation process. The identification of heterogeneity is also manually done, and so is the classification of heterogeneity types. The extracting of integration rules is based on human reasoning and still needs significant involvement of domain experts.

The case study presented in this paper is a preliminary attempt to design and implement an integration analysis framework based on machine and human readable SBVR. The approach in this paper can be further complemented and extended as follows. A transformation from source models to SBVR models needs to be automated to simplify the process of building SBVR-enhanced integration hub. A deeper review of the existing functionality of available tools should also be part of the further work. Comparing and analyzing heterogeneity needs to be standardized and automated if possible to reduce the inconsistency caused by human involvement. A state of art study of classification of ontology mismatches and their representation in SBVR would be natural to be done. Extraction of integration rules based on heterogeneity should be designed as an ontology reasoning process, where applicable, to reduce the inconsistency caused by human involvement. Other rule modeling languages such as Rule Markup Language (RuleML) could be evaluated for automated generation of integration rules.

**Acknowledgment.** The work of Dumitru Roman and Arne J. Berre is partly funded through the Semicolon II, PlanetData, and BigFut projects.

## References

1. Uschold, M., Gruninger, M.: Ontologies and Semantics for Seamless Connectivity (2004)
2. Karpovic, J., Nemuraite, L.: Transforming SBVR Semantics into Web Ontology Language OWL2: Main Concepts. In: Proc. 17th International Conference on Information and Software Technologies IT 2011 (2011)
3. Nemuraite, L., Skersys, T., Sukys, A., Sinkevicius, E., Ablonskis, L.: VETIS tool for editing and transforming SBVR business vocabularies and business rules into UML&OCL models. In: Information Technologies, IT 2010, Kaunas, Lithuania, April 21-23, 2010, 377-384 (2010)
4. Fowler, M., Scott, K. : UML Distilled Second Edition (2000)
5. Ross, R. G.: Business Rule Concepts, third edition (2009)
6. Euzenat, J., Shvaiko, P.: Ontology Matching (2007)
7. Cabot, J., Pau, R., Raventós, R.: From uml/ocl to sbvr specifications: a challenging transformation. In: Information Systems. Elsevier, Amsterdam (2009)
8. “Semantics of Business Vocabulary and Business Rules (SBVR), version 1.0, v1.0.0 Object Management Group (OMG), Jan. 2008. [online] [last accessed: April. 2012]. <http://www.omg.org/spec/SBVR/1.0/PDF/>

# Grailog KS Viz: A Grailog Visualizer for Datalog RuleML Using an XSLT Translator to SVG

Martin Koch<sup>1,2</sup>, Sven Schmidt<sup>1,2</sup>, Harold Boley<sup>1</sup>, and Rainer Herpers<sup>1,2</sup>

<sup>1</sup> University of New Brunswick, Faculty of Computer Science, Fredericton, NB, E3B 5A3,  
Canada

<sup>2</sup> Bonn-Rhein-Sieg University of Applied Sciences, Institute of Visual Computing & De-  
partment of Computer Science, Grantham-Allee 20, 53757 Sankt Augustin, NRW, Germany

{martin.koch, sven.schmidt, harold.boley} [AT] unb.ca,  
rainer.herpers [AT] h-brs.de

**Abstract.** Grailog embodies a systematics to visualize knowledge sources by graphical elements. Its main benefit is that the resulting visual presentations are easier to read for humans than the original symbolic source code. In this paper we introduce a methodology to handle the mapping from Datalog RuleML, serialized in XML, to an SVG representation of Grailog, also serialized in XML, via eXtensible Stylesheet Language Transformations (XSLT) 2.0/XML; the SVG is then rendered visually by modern Web browsers. This initial mapping is realized to target Grailog's “fully node copied” normal form. Elements can thus be translated one at a time, separating the fundamental Datalog-to-SVG translation concern from the concern of merging node copies for optimal (hyper)graph layout and avoiding its high computational complexity in this online tool. The resulting open source Grailog Knowledge-Source Visualizer (Grailog KS Viz) supports Datalog RuleML with positional relations of arity  $n > 1$ . The on-the-fly transformation was shown to run on all recent major Web browsers and should be easy to understand, use, and extend.

**Keywords:** visualization, graphs, directed hypergraphs, Grailog, computational logic, rules, Datalog, XML, RuleML, XSLT, SVG, JavaScript

## 1 Introduction

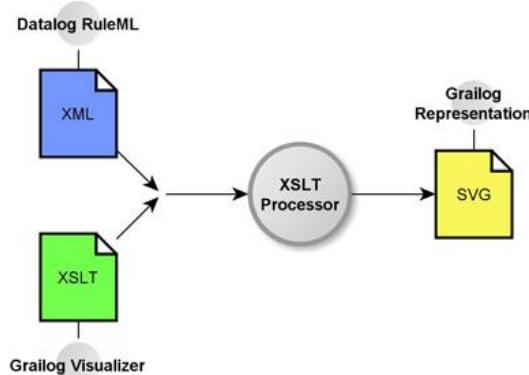
Datalog RuleML [4, 5] is an XML serialization of Datalog and the n-ary core sub-language of the RuleML family. Because of its interoperation usage in Artificial Intelligence (AI) and the Semantic Web, its normative syntax is in the Extensible Markup Language (XML), which is more suitable for machine processing than for human readability. To make the Datalog RuleML language more readable for humans, one natural approach is to translate its knowledge bases, consisting of facts and rules, in a human-oriented manner.

One method is to create a visualized representation from knowledge bases. Well-developed visualizations, as optional two-dimensional syntaxes, can help people to better understand logical constructs than through symbolic one-dimensional syntaxes alone. This has been explored in an approach of visualizing major (Semantic Web) formalisms in Graph inscribed logic (Grailog) [2,3]. It describes the mapping for several defined graph constructs to corresponding symbolic logic constructs of the considered sublanguage, which leads to better human readability. The current work constitutes a first step to try to automate this transformation as far as possible, so that ultimately each Datalog and other RuleML knowledge base can be easily visualized as a Grailog representation.

### 1.1 Objectives

The goal of this work has been to handle the task described before, i.e. the translation from Datalog RuleML to a visualized Grailog representation. This has resulted in an initial version of the Grailog Knowledge-Source Visualizer (Grailog KS Viz). The tool is open-source and deployed on the Web for access by users of RuleML, Grailog, Datalog and related systems (<http://www2.unb.ca/~mkoch/cs6795swt/index.html>).

The main objective of this work is to define a mapping from Datalog RuleML, serialized in XML [6], to the Scalable Vector Graphics (SVG) [8], both being W3C standards. This should be realized using eXtensible Stylesheet Language Transformations 2.0 (XSLT 2.0) [10], another W3C standard, which permits transformations from an XML source document to an (XML) target document. A visualization of the overall process can be seen in Figure 1.



**Fig. 1.** Visualization of the general transformation process: Datalog RuleML (written in XML) in combination with an XSLT file (Grailog KS Viz) leads to an SVG file (the corresponding Grailog representation)

By implementing different elements in the XSLT 2.0 document, each Datalog RuleML fact and rule in XML should automatically be transformed to its corresponding SVG element in XML. Elements can thus be translated one at a time, separating

the fundamental Datalog-to-SVG translation concern from the concern of merging node copies for optimal (hyper)graph layout and avoiding its high computational complexity in this online tool. A Grailog layout optimizer would deserve an entire R&D effort on its own. Our resulting visualization will instead stay in the “fully node copied” Grailog normal form, which means that the separation of clauses of the symbolic form in the Datalog RuleML source document is also kept in the resulting graphical form of the SVG Grailog representation.

Grailog is a comprehensive aid to visualize logics. The task of this work is to create an initial version of Grailog KS Viz, i.e. to use only a specific subset and functionality of Grailog corresponding to Datalog RuleML. This includes the important capability to translate Datalog’s n-ary relationships, for the current work with  $n > 1$ , to Grailog’s directed hyperarc arrows, which connect the argument-box nodes.

For usability, it should be possible to render the resulting SVG representation of the Datalog RuleML source document as a Grailog diagram “on the fly” by using suitable tools, e.g. one of the recent major Web browsers.

## 1.2 Languages

Several languages have constituted the backbone of this work. Datalog RuleML and Grailog have been important as initial source and final target of the envisioned translation. XML, SVG, XSLT and JavaScript are needed for the implementation. In the following, each language will be briefly described.

**Datalog RuleML.** RuleML is a family of rule based languages, which are used for sharing rule bases written in XML and publishing them on the Web. Datalog RuleML is one rule sublanguage of RuleML, providing the relational-view-like expressivity of Datalog in RuleML. Datalog [7] is at the foundation of RuleML and can be seen as an intersection of SQL and Prolog [4]. Datalog is used to define facts and rules and is therefore suitable to create function-free Horn logic knowledge bases and queries. XML files that contain stripe-skipped Datalog RuleML are the initial source files of this work. A tool called RuleML Official Compactifier (ROC) [14] can be used for the transformation from the fully expanded normal form to the stripe-skipped form.

**Grailog.** Grailog embodies a systematics to visualize knowledge representations by graphical elements. Each of the introduced graphical elements is mapped to its corresponding symbolic logic construct. The main benefit of Grailog is that its resulting visual representations of knowledge sources are much easier to read for humans than the original symbolic source code. Moreover, the transformation rules for the graphical-to-symbolic mapping are easy to learn and remember.

Grailog is based on several principles. One principle is that the used graphs should be natural extensions of Directed Labeled Graphs. Another principle mandates that the used graphs should allow stepwise refinements for logic constructs, like Description Logic constructors or general PSOA RuleML terms. Moreover, Grailog also implements the principle of orthogonality, which means that it consists of several inde-

pendent concepts which can be freely combined with each other. All these principles support the main goal of Grailog, namely easy understanding and application of the systematics.

A major part of Grailog's visualization elements are the directed hypergraphs, which are used for dealing with the n-ary relationships Datalog RuleML is capable of expressing.

**Scalable Vector Graphics (SVG).** SVG is an XML specification for two-dimensional graphics. It supports static and dynamic (i.e., interactive or animated) graphics, with different types of graphical objects and functions. Documents, written in SVG, are generally supported by all recent major Web browsers, but in fact all browsers have advantages and disadvantages. This is why this work has two final Grailog KS Viz versions. The following sections will describe this issue more precisely. To reproduce dynamic graphics, SVG needs the help of JavaScript, which was another important element of this work.

**JavaScript.** JavaScript is a scripting language and was defined in the ECMAScript language standard, developed by Ecma. Scripts do not need to be preprocessed before running and can make Web pages more dynamic. Due to these advantages, Web pages behave more like traditional software applications. Specific capabilities are interactive contents, the animation of page elements or the loading of new page contents without reloading of the whole page. [9]

**Extensible Stylesheet Language Transformations (XSLT).** The XSLT language permits transformations of XML documents into XML target documents or several other forms, like e.g. HTML, XHTML or SVG. The XSLT processor takes XML sources and a stylesheet, which describes the rules for the transformation. These template rules associate patterns, which match nodes in the source document, with a sequence constructor. In many cases evaluating the sequence constructor will cause new nodes to be constructed that can be used to produce part of a result tree. The structure of the result trees can be completely different from the structure of the source trees. In constructing a result tree, nodes from the source trees can be filtered and reordered, and any structure can be added.

## 2 Methodology

The result of this work should be an initial version of Grailog KS Viz, which automatically translates Datalog RuleML to the applicable Grailog representation, both written in XML. The translation should be done via XSLT. The final results should be described by SVG and be presentable through any suitable tool, like one of the recent major Web browsers.

The procedure of this work was divided into three parts. The first part was to determine all necessary elements of Grailog and Datalog RuleML. That means especially

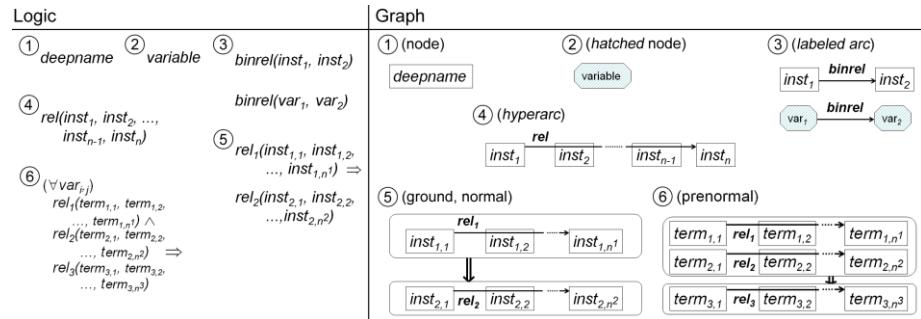
ly for Grailog the possible relationship elements, and for Datalog RuleML the supported XML declarations. The second part was to create all important Grailog representations in SVG, to have a pattern-like set of graphical elements for the last part of this work. The last part was the implementation of an XSLT file, which transforms a Datalog RuleML XML file into the Grailog representation. The graphical elements of the second step were necessary to lead the transformation to the right results. The following subsections will describe this procedure in more detail.

## 2.1 Determination of the Required Elements of Grailog and Datalog RuleML

The result of this work could only be an initial version of Grailog KS Viz because of the limited time of this work. Therefore, the first step was to determine the required elements of Grailog and Datalog RuleML.

The chosen subset of Grailog and Datalog RuleML can be seen in Figure 2. This subset contains all elements for an adequate Datalog RuleML knowledge base. As mentioned previously, the resulting visualization should be in a “fully node copied” normal form (in Grailog, all node occurrences with the same unique name remain one node).

To support the Grailog representations seen in Figure 2, Grailog KS Viz had to support the following XML elements of Datalog RuleML: “RuleML” (with and without namespace and schema declaration), “Assert”, “Implies”, “And”, “Atom”, “Rel”, “Var”, “Ind” and “Data”.



**Fig. 2.** Chosen Datalog RuleML subset visualized in Grailog: Both logical (left) and graphical (right) representations shown for (1) individual constants, (2) variables, (3) binary relations, (4) ( $n > 1$ )-ary relations, (5) single-premise rules and (6) multi-premise rules (modified from [2])

## 2.2 Grailog

The second step of this work was to create the Grailog representations in SVG. SVG allows three different types of graphical objects, the vector graphic shapes (e.g. paths or polygon lines), images and text. For the different graphical objects SVG provides further methods, like grouping objects, assigning different styles per object, and performing further transformations. Moreover, SVG provides a rich feature set which includes nested transformations, clipping paths, alpha masks, filter effects and tem-

plate objects. SVG is completely described through XML and has to be introduced by a “svg” element as the root. The dynamics of graphical elements is possible through JavaScript. JavaScript was especially essential to get the lengths of the different texts, to scale the elements and to position them. The importance of JavaScript was not obvious in the first considerations of this work and was therefore not mentioned in the objectives.

The creation of the Grailog representation started with the simplest element: a single individual constant. An individual constant is described in Grailog through a specific text surrounded by a rectangle. So, the first step was to create an SVG “text” element. This “text” element had to contain the desired text of the individual constant as well as a specific “id” and the coordinates “x” and “y”, as attributes. Every “text” element needs its own ID-number for distinction. The coordinates are relevant to place the text on a specific position in the viewBox which represents the image section on the screen. The very first text element always starts at the coordinates “x = 50” and “y = 50”. All further elements will be arranged to correspond to the first “text” element. Rectangles can be visualized in SVG through the “rect” element. For individual constants the “rect” element has to include the attributes “id”, “x” and “y” (as start coordinates like before), “height” and “width” (as determination for the size) and “style” (to assign stroke color and width). Apart from the “id” and the “style”, all attributes will be assigned through JavaScript code, to be independent of the varying text lengths. First, the JavaScript code computes the width of the “rect” element by calculating the length of the text of the “text” element. Then it determines suitable values for “height” and the coordinates “x” and “y” in consideration of the coordinates of the “text” element. Figure 3 shows the SVG source code for an exemplary individual constant and the resulting Grailog representation.

The second simplest element of the Grailog subset was the representation of a variable. A variable is a specific text surrounded by a hexagonal box, hatched by diagonally arranged blue lines. The hatched lines are a result of a specific “pattern” element. A “pattern” element of SVG can contain different other SVG elements. This self-implemented “pattern” element contains several “path” elements, which are in this case nothing more than diagonally arranged blue lines with specific start and end points. The pattern is allocated to a “polygon” element, which finally represents the hexagonal box of the variable. The “polygon” element of this variable representation contains attributes for “id”, “points” (determinations for the edges) and “style” (to assign the hatched pattern as well as stroke color and width). As seen for the individual constant, most of the work for positioning is done via JavaScript. In this case JavaScript especially computes the edges of the polygon to arrange the box around the text.

The last important elements of the different Grailog representations were the arrow for relations and the double-arrow and rectangles with rounded corners for rules. An arrow for relations consists only of a black “path” element and a “marker” element, which contains the arrow-head. The double-arrow is implemented as a single “path” element, which follows a particular track. To create a rectangle with rounded corners, one only has to assign the attributes “rx” and “ry” to a “rect” element.

The last step of the SVG part was to create different SVG documents out of the above described simple elements of the Grailog representation. The final documents included a representation for a single individual constant, a single variable, any combination of binary and n-ary relationships, as well as single- and multi-premise rules. After creating all these standard Grailog representations, the SVG elements could be used as patterns to transform any Datalog RuleML XML file through XSLT. The next subsection describes this transformation and the procedure of the implementation.

| SVG<br>source code                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | Grailog<br>representation |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------|
| <pre>&lt;svg version="1.1" xmlns="http://www.w3.org/2000/svg"&gt; &lt;text id="text1" x="50" y="50"&gt;inst1&lt;/text&gt; &lt;rect id="rect1" style="stroke:#000000; fill: none; stroke-width:1;"&gt; &lt;script&gt;     document.getElementById("rect1").setAttribute("width",         parseFloat(document.getElementById("text1").getComputedTextLength()) + 20);     document.getElementById("rect1").setAttribute("height",40);     document.getElementById("rect1").setAttribute("x",         parseFloat(document.getElementById("text1").getAttribute("x")) - 10);     document.getElementById("rect1").setAttribute("y",         parseFloat(document.getElementById("text1").getAttribute("y")) - 25); &lt;/script&gt; &lt;/svg&gt;</pre> | inst1                     |

**Fig. 3.** SVG source code (left) and the resulting Grailog representation (right) of an individual constant

### 2.3 Creating the Translation from Datalog RuleML to Grailog

In this part of this work, the implementation of the transformation will be explained, which is done by using XSLT 2.0. The transformation mainly depends on the part described before, i.e. emulating of the created patterns. In the following, the basic structure of Grailog KS Viz is illustrated. After that, an example of a small part of the actual transformation is given and explained in detail.

**Basic Structure of Grailog KS Viz.** To work with XSLT in general, the “stylesheet” or the completely synonymous “transform” element have to be used. For Grailog KS Viz, version 2.0 of XSLT is used, which brought along many useful functions, as it builds on XPath 2.0, instead of XPath 1.0 like its predecessor. With the help of the attributes of the “output” element, the output document is set to be of the type “XML” with the encoding “ISO-8859-1”, to allow a rich enough character set for the possible facts and rules that come with the source document. Moreover, the doctype of the resulting document is set to the official SVG doctype, to guarantee syntactic correctness of the produced SVG document.

The “apply-templates” element in conjunction with the “template” element is used to structure the transformation. The “template” element contains rules that are applied when a specific node is matched. With the help of “apply-templates” and its “select” attribute, the templates can be matched and the defined rules are executed. For Grailog KS Viz, these elements are used to divide the given knowledge base into

meaningful (sub)sections, like facts or the head of a single-premise rule. Another element which is used in this context is the “for-each” element. This element mostly loops through all terms of an atom. Because of the overall goal of a final SVG document as output, the first step of the transformation creates the SVG root element along with its attributes. Additionally, the definitions of the arrow head and the variable pattern are created.

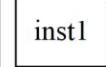
The next step of the transformation process by Grailog KS Viz is to differentiate between rules and relations. Moreover, it is important to differentiate single-premise rules from multi-premise rules, because they result in different Grailog representations. The differentiation is done by searching the Datalog RuleML document for the elements “Implies”, “And” and “Atom”. The idea is to use parent and child relationships and positions to determine the type of the considered atom. If an atom is a child of an “Implies” element, it means that it is part of a rule. If it is a direct child of “Implies”, it is either the head of a multi-premise rule or the head or premise of a single-premise rule. This can be clarified by checking how many atoms are direct children of a specific “Implies” element. If there are two atoms, it is a single-premise rule, with the first atom as premise and the second atom as head. If there is only one atom as direct child, it has to be the head of a multi-premise rule. If an atom is child of an “And” element, it is part of the premise of a multi-premise rule. The XSLT / XPath functions that are mostly used for the search are among other things “current()”, “count((item, item, ...))”, “position()” and “last()”. These can be used to gain information about positions of specific nodes, about the number of child elements and much more. The last differentiation is done between binary and n-ary relations. Therefore, simply the number of children of an atom can be used. The differentiation is used to determine if a simple arrow or a hyperarc is needed in the resulting SVG representation.

After knowing the kind of the currently processed atom, the suited SVG elements and corresponding JavaScript code will be created. The elements can be “text”, “rect”, “polygon” or “path”, depending on the current node. The “value-of” element offers the possibility to get the value of the “Rel”, “Var” and “Ind” nodes of an atom. Moreover, it is frequently used to create unique variable names for all the different elements of the resulting SVG document. This is a huge part of the transformation process, because the SVG patterns that were achieved in the first part of this work are now dynamically created as part of the transformation. To get the unique variable names, a function of XSLT for concatenation is used to create names based on the type of the relation or rule that is currently processed and on its position in the XML tree. Besides the creation of the SVG elements, also the corresponding JavaScript code is inserted. The “if” element of XSLT is used to alternate the code for different cases, e.g. if a term is the first term of a relationship or the last. The JavaScript code is also used for keeping track of the maximum height and width of the viewBox of the SVG document.

**Example for the Translation of an Individual Constant.** Figure 4 shows a small part of Grailog KS Viz source code, which for this example results in the Grailog

representation of an individual constant of an n-ary relation, as previously seen in Figure 3 for SVG.

The first line of this extracted source code checks if the considered atom is a direct child of the “Assert” element. In this case it is part of a fact. Then the variable “countRelations” is defined. It holds the number of facts and rules that exist before the actual considered fact plus one, which equals the exact position of the fact in the knowledge base. Then each child element of the atom is processed. If it is an “Ind” or “Data” element, which both lead to the same Grailog representation, an SVG “rect” element and an SVG “text” element are created. For this case, the rectangles name consists of the substrings “rect”, “Relation”, the value of “countRelations” and the position of the term in the atom. This ensures that it gets a unique name, so that no problems occur in the resulting SVG document, regardless of the number and kinds of individual constants. Beneath the SVG elements, the corresponding JavaScript code is inserted, with its adjusted variable names and some further adjustments.

| XSLT<br>source code                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | Grailog<br>representation                                                                     |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------|
| <pre> &lt;xsl:if test="parent::r:Assert"&gt;   &lt;xsl:variable name="countRelations" select="count(preceding-sibling::*) + 1"/&gt;   &lt;xsl:for-each select="./*"&gt;     &lt;xsl:if test="position()=2" &gt;!-- First term --&gt;       &lt;xsl:if test="(ancestor-or-self::r:Ind) or (ancestor-or-self::r:Data)"&gt;         &lt;xsl:element name="rect"&gt;           &lt;xsl:attribute name="id"&gt;             &lt;xsl:value-of select="concat('rect', 'Relation', \$countRelations, position())"/&gt;           &lt;/xsl:attribute&gt;           &lt;xsl:attribute name="style"&gt;             stroke:#000000; fill: none; stroke-width:1;           &lt;/xsl:attribute&gt;         &lt;/xsl:element&gt;         &lt;xsl:element name="text"&gt;           &lt;xsl:attribute name="id"&gt;             &lt;xsl:value-of select="concat('text', 'Relation', \$countRelations, position())"/&gt;           &lt;/xsl:attribute&gt;           &lt;xsl:value-of select=". /&gt;         &lt;/xsl:element&gt;         &lt;script type="text/javascript" language="JavaScript"&gt;           ... (relevant JavaScript code with adjusted variable names etc.) ...         &lt;/script&gt;       ... (second term, next terms, last term; each for „Ind“, „Data“, „Var“ and „Rel“) ...     &lt;/xsl:if&gt;   &lt;/xsl:for-each&gt; &lt;/xsl:if&gt;</pre> | <br>inst1 |

**Fig. 4.** XSLT source code (left) and its correspondent Grailog representation (right) for an individual constant

### 3 Results

The final result of the implemented translation process of Grailog KS Viz is an automatically generated SVG document, which can be used to render the Grailog representation of a given Datalog RuleML knowledge base. To demonstrate the reliability

and give a deeper understanding of the transformation and translation process, the following part shows and discusses two kinds of rules via examples.

### 3.1 Translation of Single-Premise Rules from Datalog RuleML to Grailog

The first kind of rule can be considered via an introductory example, which only consists of a single-premise rule and binary relations. In this case, the rule has no explicit meaning and only uses placeholder names. The rule in Datalog RuleML XML syntax and its Grailog representation, created by Grailog KS Viz, are shown in Figure 5.

| Datalog RuleML source code in XML                                                                                                                                                                                                                                                                                                                                                                                                                      | Grailog representation |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------|
| <pre> &lt;RuleML&gt;   &lt;Assert mapClosure="universal"&gt;     &lt;Implies&gt;       &lt;Atom&gt;         &lt;Rel&gt;Rel1&lt;/Rel&gt;         &lt;Var&gt;Var1&lt;/Var&gt;         &lt;Ind&gt;Inst1&lt;/Ind&gt;       &lt;/Atom&gt;       &lt;Atom&gt;         &lt;Rel&gt;Rel2&lt;/Rel&gt;         &lt;Ind&gt;Inst2&lt;/Ind&gt;         &lt;Var&gt;Var2&lt;/Var&gt;       &lt;/Atom&gt;     &lt;/Implies&gt;   &lt;/Assert&gt; &lt;/RuleML&gt; </pre> |                        |

**Fig. 5.** XML source code (left) and its correspondent Grailog representation (right) for a single-premise rule with binary relations

For the above shown example, Grailog KS Viz first recognizes that there is a rule, because there is an “Implies” element in the XML source document that can be matched. Then it checks if it is a single-premise or a multi-premise rule by looking at the number of atoms that are direct children of the “Implies” node. Because there are exactly two, it has to be a single-premise rule. Therefore, first the upper rounded rectangle is created, followed by its content. The premise of the rule consists of a binary relation “Rel1” and two terms, the variable “Var1” and the individual constant “Inst1”. The first term is created, followed by the text element with the value “Rel1”. This can be done because the text with the relator name is always at the same relative x-position, directly besides the representation of the first term. Then the second term is transformed. After that, finally the arrow can be created, starting from the first term, either polygon or rectangle and ending at the second. After the content of the rounded rectangle is complete, the width of the rectangle is adjusted and then the double-arrow is created. The creation of the head of the rule is done the same way, only that the vertical positions are adjusted.

### 3.2 Translation of Multi-Premise Rules from Datalog RuleML to Grailog

The second kind of rule can be considered via an advanced example, which consists of a multi-premise rule containing two binary relations and a 3-ary relation. In general, our subset of Grailog allows ( $n > 1$ )-ary relations, where the binary relation “be” is used to avoid unary relations ( $n=1$ ). The rule has the following meaning: If George knows a player and an arena is a hockey rink, then George plays with the player in the arena. Figure 6 shows the XML source code and the resulting Grailog representation.

| Datalog RuleML<br>source code in XML                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | Grailog<br>representation                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre> &lt;RuleML&gt;   &lt;Assert mapClosure="universal"&gt;     &lt;Implies&gt;       &lt;And&gt;         &lt;Atom&gt;           &lt;Rel&gt;know&lt;/Rel&gt;           &lt;Ind&gt;George&lt;/Ind&gt;           &lt;Var&gt;player&lt;/Var&gt;         &lt;/Atom&gt;         &lt;Atom&gt;           &lt;Rel&gt;be&lt;/Rel&gt;           &lt;Var&gt;arena&lt;/Var&gt;           &lt;Ind&gt;Hockey rink&lt;/Ind&gt;         &lt;/Atom&gt;       &lt;/And&gt;       &lt;Atom&gt;         &lt;Rel&gt;play&lt;/Rel&gt;         &lt;Ind&gt;George&lt;/Ind&gt;         &lt;Var&gt;player&lt;/Var&gt;         &lt;Var&gt;arena&lt;/Var&gt;       &lt;/Atom&gt;     &lt;/Implies&gt;   &lt;/Assert&gt; &lt;/RuleML&gt; </pre> | <p>The diagram illustrates the translation of the XML source code into a visual Grailog representation. It consists of two main parts: an upper rounded rectangle and a lower rounded rectangle, separated by a downward-pointing arrow.</p> <ul style="list-style-type: none"> <li><b>Upper Rounded Rectangle:</b> This represents the premise of the rule. It contains two nodes: "George" (rectangle) and "arena" (hexagon). Two arrows point from these nodes to a third node: "player" (hexagon) and "Hockey rink" (rectangle) respectively, connected by a double-headed arrow between them.</li> <li><b>Lower Rounded Rectangle:</b> This represents the head of the rule. It contains three nodes: "George" (rectangle), "play" (rectangle), and "player" (hexagon). A hyperarc connects "George", "play", and "player". An arrow points from "player" to "arena" (hexagon).</li> </ul> |

**Fig. 6.** XML source code (left) and its correspondent Grailog representation (right) for a multi-premise rule with binary and n-ary relations

In this example, the tool detects a multi-premise rule, because the “And” element is the first child element of the “Implies” element. Therefore, first the upper rounded rectangle is created, followed by its content. This affects the width and the height of the rounded rectangle, because a multi-premise rule can have an unbounded number of atoms in the premise. Because the premise of the rule only consists of binary relations, they are created as in the previous example. After the content of the upper rounded rectangle is complete, its width and height are adjusted and the double-arrow is created. The creation of the head of the rule cannot be done as in the previous example because it contains no binary relation, but the ( $n > 2$ )-ary relation “play”. Therefore, there can be an unbounded number of terms, and this leads to the usage of a hyperarc. The hyperarc is created after all terms are processed, and starts at the first term, cuts through intermediate ones and ends at the last. Finally, the lower rounded rectangle is created and then the SVG document is finalized. For this example the resulting SVG document consists of 289 lines of code.

There is no loss of generality with this example since a multi-premise rule with more than two premises can be reduced to multiple rules with two premises. However, Grailog KS Viz handles multiple premises directly.

The authors have started to complement the .ruleml examples (RuleML/XML) in the Datalog section of the public RuleML 1.0 exa library [13] with .grailog visualizations (RuleML/Grailog), including for the classical Datalog RuleML 'own' example (<http://ruleml.org/1.0/exa/Datalog/own.ruleml> paired with its visualization as <http://ruleml.org/1.0/exa/Datalog/own.grailog>).

### 3.3 Issues and Alternatives

Grailog KS Viz was realized by splitting the implementation part of this work into two mostly independent tasks, the creation of the Grailog representations in SVG and the actual transformation from Datalog RuleML to SVG. The independence of these two tasks was essential to be able to implement both tasks in parallel. This is also the reason for the usage of both, XSLT and JavaScript. Another reason that led to this approach was the fact that this was the first time to work extensively with SVG and XSLT for the first two authors, who implemented Grailog KS Viz. In retrospect, it might be possible to replace several parts of the JavaScript code with XSLT code. This would definitely have made the planning phase more complex and probably would have increased the time needed for completing this work. The benefit of such a solution would be much less code for the resulting SVG document.

Because one of the goals of this work was to support different recent major Web browsers, in the end two separate versions of the tool had to be implemented. Both use different methods to compute the length of the text elements, which is needed for determining the width and the position of the graphical elements. The “normal version” uses the JavaScript method “getComputedTextLength()”, which computes the length of the rendered text. This method unfortunately does not work for Firefox or Chrome at the on-the-fly transformation, i.e. by executing the transformation within the browser. After saving the resulting SVG document and opening it again, the method works for all tested browsers. The problem seems to be that at this moment when Firefox or Chrome use the “getComputedTextLength()”, no text is rendered, yet. To overcome this, a so called “monospaced version” of the tool was implemented. It uses the font “Monospace”, the “XMLSerializer()” and the “serializeToString()” method, which does work for Firefox, but not for Chrome or Safari, even after saving as SVG file and opening it again. The detailed support information can be seen in the appendix (Figure 7 and 8).

Currently, Grailog KS Viz consists of nearly 4000 lines of code, which is also based on the need for the support for two different source document versions, Datalog RuleML documents with and without the RuleML namespace and schema. Because this was realized in the last moments of this work, huge parts of the code exist in two forms, with the namespace in the XPath expressions and without it. By implementing a possible XSLT only-version, without the need for JavaScript and with a better implemented distinction between the two source document cases, it might be possible to

ultimately only have one, presumably monospaced and much shorter version of Grailog KS Viz.

Related work in the area of interest has been done for example by another RuleML/Grailog team [15]. This team also used the principles of Grailog for visualization purposes, but instead of building it from scratch, they used the Graphviz framework to visualize SWRL’s unary/binary Datalog RuleML in Grailog. The benefit of building it from scratch, however, is that the final results are identical to the given specifications of Grailog.

## 4 Conclusion

The result of this work is the successful implementation of an initial version of Grailog KS Viz, a Grailog knowledge-source visualizer based on RuleML-to-SVG translation.

The first section gave a short introduction to the topic and stated how the approach of this work relates to previous work. Moreover, a short preview of the actual work was given and each language used throughout this work was explained briefly. The methodology for the successful accomplishment of the objectives was developed in three parts. The first part dealt with the determination of the required elements of Grailog and Datalog RuleML. The result was a mapping from a Datalog RuleML subset to a Grailog subset. The second part was about the creation of the Grailog representations in SVG. It explained the used elements and functions of SVG and JavaScript and illustrated an example for an individual constant. The last part of this section described the creation of the translation from Datalog RuleML to the Grailog representation. Therefore, it explained the basic structure of Grailog KS Viz and also illustrated an example of the transformation of an individual constant. The results are shown in the third section of the report. First, the transformation of a single-premise rule containing binary relations was explained, and then a more complex multi-premise example. Instead of focusing on the actual code as in the previous examples, these two were explained on a higher level. After that, some issues of the current implementation were highlighted and a general picture of this work and its environment was given.

The implemented Grailog KS Viz meets the given requirements and offers a robust and usable functionality with response times suitable for online rendering. The tool is open source and all its versions, documents, source files and many more examples and explanations can be found on the official website of this work. Several links to different parts of the website of this work can be found in the appendix.

### 4.1 Future Work

As mentioned before, although this work is an overall success, there are different aspects that definitely can be improved in further work. Besides the already mentioned possible improvements in the previous section, the following changes are envisioned:

A simple but effective improvement would be to transform the source document in a way that the resulting SVG document appears in a pretty print layout. Currently, the structure of the resulting code is not pretty printed, which complicates further adjustments, users of the tool could want to do manually. Furthermore, the merging of the individual SVG elements of rules and facts to a “node copy-free” graph would be a big improvement of this initial version of the visualizer. This would make the tool more powerful, because the user could directly see connectivity, although the high computational complexity of merging node copies for optimal (hyper)graph layout might entail response-time issues in online Grailog rendering of large Datalog knowledge sources. Besides these Grailog-target improvements, also the support of more RuleML-source features such as unary relations, (positional-)slotted relations and typed variables is planned. Then, the development should proceed from Datalog RuleML to Hornlog RuleML and gradually cover the remaining Grailog-visualized branches of the RuleML family. Grailog generators for other rule and ontology languages could be similarly implemented as well.

Complementing the approach of this work, inverse translators parsing Grailog SVG/XML diagrams into RuleML/XML trees could be realized. This could result in an authoring tool that allows users to visually design rule bases in the graphically rendered SVG representation, which will then be parsed into the Datalog RuleML/XML representation. In combination with our current Grailog generator, this Grailog parser could ultimately lead to a complete Grailog IDE.

## 5 Acknowledgments

Financial support of the DAAD in the ISAP program line, project No 54890855, is gratefully acknowledged. Also, NSERC is thanked for its support through Discovery Grants. Finally, we would also like to thank the RuleML reviewers, Kenneth Kent, Diana Kraus and all others who have supported us during this work.

## References

1. Boley, H.: CS 6795 Semantic Web Techniques - Fall 2012 Projects. <http://www.cs.unb.ca/~boley/cs6795swt/fall2012projects.html>, visited on October 19th, 2012
2. Boley, H.: Grailog 1.0: Graph-Logic Visualization of Ontologies and Rules. Preprint: <http://www.cs.unb.ca/~boley/papers/GrailogVisOntoRules.pdf>, visited on May 9th, 2013. To appear: Proc. RuleML 2013, Springer LNCS 8035, July 2013
3. Boley, H.: Grailog. <http://wiki.ruleml.org/index.php/Grailog>, visited on May 24th, 2013
4. Boley, H., Athan, T.: RuleML Primer, August 2012. <http://ruleml.org/papers/Primer/RuleMLPrimer2012-08-09/RuleMLPrimer-p0-2012-08-09.html>, visited on October 19th, 2012
5. Boley, H., Athan, T., Paschke, A., Tabet, S., Grososf, B., Bassiliades, N., Governatori, G., Olken, F., Hirtle, D.: Schema Specification of Deliberation RuleML Version 1.0. <http://ruleml.org/1.0/>, visited on October 19th, 2012

6. Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., Yergeau, F.: Extensible Markup Language (XML) 1.0 (Fifth Edition) - W3C Recommendation, November 2008. <http://www.w3.org/TR/2008/REC-xml-20081126/>, visited on October 19th, 2012
7. Ceri, S., Gottlob, G., Tanca, L.: What You Always Wanted to Know About Datalog (And Never Dared to Ask). IEEE Transactions on Knowledge and Data Engineering, vol. 1 (1), pp. 146-166 (1989)
8. Dahlstroem, E., Dengler, P., Grasso, A., Lilley, C., McCormack, C., Schepers, D., Watt, J., Ferraiolo, J., Fujisawa, J., Jackson, D.: Scalable Vector Graphics (SVG) 1.1 (Second Edition) - W3C Recommendation, August 2011. <http://www.w3.org/TR/2011/REC-SVG11-20110816/>, visited on October 19th, 2012
9. Hazaël-Massieux, D.: JavaScript Web APIs. <http://www.w3.org/standards/webdesign/script.html>, visited on November 15th, 2012
10. Kay, M.: XSL Transformations (XSLT) Version 2.0 - W3C Recommendation, January 2007. <http://www.w3.org/TR/2007/REC-xslt20-20070123/>, visited on October 19th, 2012
11. Refsnes Data: W3Schools - SVG Tutorial. <http://www.w3schools.com/svg/default.asp>, visited on October 19th, 2012
12. Refsnes Data: W3Schools - XSLT Tutorial. <http://www.w3schools.com/xsl/>, visited on October 19th, 2012
13. RuleML: The Rule Markup Initiative - Library of Datalog Examples. <http://www.ruleml.org/1.0/exa/Datalog>, visited on June 12th, 2013
14. Singh, S., Aayush, B. R., Shah, P.: Testing, Inverting, and Round-Tripping the RON Normalizer for RuleML 1.0 in XSLT 2.0. <http://ruleml-roc.yolasite.com/>, visited on Januar 9th, 2013
15. Yan, B., Zhang, J., Akbari, I.: Visualizing SWRL's Unary/Binary Datalog RuleML in Grailog. <https://github.com/boliuy/SWRL-RULES-VISUALIZER>, visited on December 13th, 2012

## Appendix

### Useful Links to this Work's Website

**Index Page and Virtual Handout.** Index page with illustrations of the objectives, methodology and the final results of this work.

- <http://www2.unb.ca/~mkoch/cs6795swt/index.html>

**Results.** Exemplary overview of different SVG result outputs and downloadable versions of Grailog KS Viz (normal and monospaced font version).

- [http://www2.unb.ca/~mkoch/cs6795swt/media/html/project/result\\_s.html](http://www2.unb.ca/~mkoch/cs6795swt/media/html/project/result_s.html)

**Documentation.** Overview of this work's documentation. Proposal, final presentation and report are downloadable there.

- <http://www2.unb.ca/~mkoch/cs6795swt/media/html/project/documentation.html>

### Supported Web Browsers

| Representation in Browser       | Firefox (16.0.2) | Google Chrome (23.0.1271.64 m) | Internet Explorer (9.0.8112.16421) | Opera (12.10) | Safari (5.1.7) |
|---------------------------------|------------------|--------------------------------|------------------------------------|---------------|----------------|
| After on-the-fly transformation | not supported    | not supported                  | supported                          | supported     | supported      |
| Retrospectively saved as SVG    | supported        | supported                      | supported                          | supported     | supported      |

**Fig. 7.** Supported (green) and unsupported (red) Web browsers of the normal version of Grailog KS Viz

| Representation in Browser       | Firefox (16.0.2) | Google Chrome (23.0.1271.64 m) | Internet Explorer (9.0.8112.16421) | Opera (12.10) | Safari (5.1.7) |
|---------------------------------|------------------|--------------------------------|------------------------------------|---------------|----------------|
| After on-the-fly transformation | supported        | not supported                  | supported                          | supported     | not supported  |
| Retrospectively saved as SVG    | supported        | not supported                  | supported                          | supported     | not supported  |

**Fig. 8.** Supported (green) and unsupported (red) Web browsers of the monospaced font version of Grailog KS Viz

# Importation Closure that is Robust to Circular Dependencies

Tara Athan<sup>1</sup>

Athan Services, W Lafayette, IN, USA

taraathan@gmail.com

WWW home page: <http://athant.com>

**Abstract.** Any approach to the integration of distributed knowledge bases (KBs) through importation must make design decisions regarding circular importation dependencies: are they disallowed, ignored, or allowed? If they are allowed, how is the resulting infinite loop of importation handled in practice? We present preliminary results of Importations (<http://athant.com/projects/Importation/>), an exploration of the ramifications of embracing circular importation dependencies, with particular application to design decisions in the revision of ISO Common Logic (CL).

## 1 Introduction

The distributed, modular representation of knowledge (rules, ontologies, facts) in widely-available form such as on the Web is becoming increasingly common. Correspondingly, there is a growing need to merge, or import, such modules (in CL called “texts”) to create more comprehensive KBs. As authorship and maintenance becomes more distributed and less-closely coordinated, the likelihood of circular importation dependencies among texts increases.

While intentional circular dependencies are not a recommended practice in KB engineering, a robust reasoning engine should be able to recover gracefully from such occurrences. Languages may even be designed to provide unambiguous semantics for these situations. In particular, the CL community has recently addressed this issue[2] in regard to the proposed revision of the ISO CL[1].

This study is intended to provide further input regarding the ramifications of design decisions for the syntax and semantics of importation. Practical concerns include:

- Minimizing the burden involved in syntactic validation of texts with importations.
- Avoiding ambiguity in the semantics of texts with importations.
- Avoiding semantics that is non-intuitive.
- Ensuring that the set of expressions that must be handled for practical purposes, such as reasoning, remains finite.
- Minimizing the computational effort of resolving importations.

A typical practice in handling circular importation dependencies is to simply ignore an importation the second time it occurs, and in many settings this is a safe strategy that minimizes the effort of both syntactic validation and semantic evaluation and it

keeps things finite. However, in some applications the importation of the same text in different contexts creates different semantic effects. The result might depend on which importation was performed first, leading to ambiguity.

ISO CL is an example of a practical case where the importation of the same text in different contexts can create different semantic effects. One of the aims of CL is to allow the integration of segregated texts, e.g. where the vocabularies for individuals, functions and relations are disjoint, and nonsegregated texts with a so-called “higher-order syntax”, where the same name may be used to denote an individual and refer to a function and/or relation.

CL handles the integration of such disparate representations formally through a modification of the interpretation that is applied to an imported text. The modification, called restriction, has the effect of shrinking (i.e. restricting) the domain of discourse of the interpretation so that quantification occurs only over the intended set of entities. Such a restriction can have different semantic effects, depending on the composition of the restricted domain of discourse. Therefore, ignoring a second importation of a text may not produce the desired semantics.

It has been shown[3] that domain restriction of a text, in the particular language studied there, is logically equivalent to a well-defined rewriting of the text, including modifying the sentences inside quantifications as well as adding domain membership facts to avoid *free logic* behaviours. Such a rewriting can be considered a special case of a more general “text operator”. Other examples of text operators include a translation operator to convert a text from one KR language into another.

## 1.1 Propositions and Some Weird Things

The language that is created here to study circular importation dependencies contains the following components

- propositional statements, which encapsulate a text in an arbitrary knowledge representation (KR) language;
- text operators, as a model for CL domain restriction as well as translations;
- titling and importation statements modelled after the abstract semantics proposed for the CL revision[3];
- text construction statements, which form collections of texts.

Importation is handled at a metalogical level and it never mingles with the underlying logical language. For example, we do not allow conditional importation that depends on the interpretation of some logical sentence. Because of this, we may defer all logical considerations until after the resolution of importations, and this is why there is no need to include, e.g., logical connectives in the language.

## 2 Background

A semantics for importations of distributed KBs was recently proposed for CL in [2]. Two statements are used to implement importation, a titling statement, which assigns

a name from the vocabulary as the title of a text, and an importation statement, which invokes a text title.

The proposed CL semantics includes a relationship, called the title mapping, as a part of an interpretation. A titling statement with a name and text is true iff the image of that name under the titling mapping is indeed that text, verbatim.

The proposed CL semantics for importation statements is based on the concept of a corpus - a (possibly empty, finite or infinite) set of finite texts. The importation statement itself always interprets to true, but its semantic effect is to add a new text to the set of texts that must interpret to true if the corpus is to be considered satisfied by an interpretation. Intuititively, a set of new texts is generated from the old by replacing importation statements<sup>1</sup> that do not occur within a titling statement by their images under the title mapping, a process called *resolution*. These new texts are added to the corpus. Then if any of the new texts contain importation statements, the resolution process is repeated. At each resolution step a finite number of (finite) texts is added to the corpus, but the texts already in the corpus are not deleted or changed in any way. Thus importation resolution is a monotonic process.

In this approach, importation statements may occur within titled texts, and it is permitted to have circular importation dependencies, the simplest being a text that imports itself. However, if the importation dependencies are circular, then the importation closure process as defined above does not terminate in any finite number of steps.

The precise definition in [2] of the importation closure is the minimal fixed point of this corpus-extension procedure that contains the original corpus. It was shown there that such a minimal fixed point of such an importation closure exists for any interpretation. Although the proof was carried out for the language considered there,  $\mathcal{L}$ , a simplified language inspired by the CL abstract syntax and semantics, and the particular CL dialect called CLIF[1], the same proof applies to the even simpler language we consider below.

Note that the importation closure depends, in general, on the intepretation because the text susbstituted for an importation statement is determined by the title mapping of the interpretation, not the titling statements in the corpus. The connection between title mapping and titling statements is established semantically

In comparison to importation, titling and reification mechanisms of other KR languages:

- unlike the OWL importation mechanism, resolution of importation in our approach does not syntactically affect the text in which it appears. Also it does not introduce a “text” or “ontology” entity into the universe of reference.
- unlike the current CL semantics, our titling approach does not affect the denotation of the name that is used as a title. By analogy consider that ‘Wherever You Go, There You Are.’ is the title of a book, as well as the name of a proposition. Similarly, names in our vocabulary have dual, independent use as titles and as names of propositions.

---

<sup>1</sup> Alternative formulations of this transformation are (1) replace all importations in a text in the corpus simultaneously (2) replace each importation individually, leaving the others as they are, producing, in general, a set of new texts for each text resolved.

We first define the syntax, then the semantics of a family  $\mathcal{L}_0$  of languages. Some examples of circular importation dependencies are presented, and we then examine conditions under which a finite corpus will always be logically equivalent to some finite corpus having a certain closure property that will allow us to use that finite corpus for reasoning purposes.

### 3 Syntax of $\mathcal{L}_0$

#### 3.1 Lexical items

An  $\mathcal{L}_0$  language consists of the following lexical items:

- Logical lexical items: the text construction operator ‘*txt*’; a set of text operators ‘ $F_i$ ’,  $i \in \mathcal{I}$ ; the titling operator ‘*title*’; the importation operator ‘*import*’;
- Auxiliary lexical items: the parentheses: ‘(’, ‘)’; the Unicode SPACE character (U+0200).
- Names: a set  $V$  of strings of unicode text characters (i.e., no whitespace) other than those above.

#### 3.2 Grammar

An *expression* of  $\mathcal{L}_0$  is any (possibly empty) string of lexical items of  $\mathcal{L}_0$ . The basic syntactic categories of expressions, which are partially overlapping, are *names*, *propositions*, *statements*, and *texts*. Names were defined in the previous section 3.1.

We now define all other syntactic categories, including the subcategories of *import statement*, *text titling*, and *text constructions*, of  $\mathcal{L}_0$  simultaneously<sup>2</sup>:

We will use  $N$  for names,  $S$  for statements,  $\Gamma, \Delta$  for texts, and  $E$  for arbitrary expressions (with and without indices).

- A propositional statement is a name,  $N$  enclosed in parentheses ( $N$ ).
- A statement is either a propositional statement, an import statement, or a text titling statement.
- A text is either
  - a statement,
  - a text construction ( $txt \Gamma_1 \dots \Gamma_n$ ), where  $\Gamma_1, \dots, \Gamma_n$  ( $0 \leq n$ ) are texts , or
  - a text operator applied to a text ( $F_i \Gamma$ ) where  $\Gamma$  is a text.
- A text titling is ( $title N \Gamma$ ) where  $\Gamma$  is a text and  $N$  is a name.
- An import statement is ( $import N$ ) where  $N$  is a name.

A title mapping  $ttl$  is a total function from  $V$  of  $\mathcal{L}_0$  to the set of texts of  $\mathcal{L}_0$ .

A corpus of a  $\mathcal{L}_0$  language is a set of texts in that language. A corpus may be empty, finite or infinite. Although it is not necessary, and in some cases not possible, to serialize a corpus, for the examples in this paper we use the notation  $\{\Gamma_1 \Gamma_2 \dots \Gamma_n\}$  for a finite corpus, where  $\Gamma_i$  are texts, and the texts are separated by white space and enclosed in curly braces, which is similar to standard mathematical set notation. The curly braces are not part of  $\mathcal{L}_0$ .

---

<sup>2</sup> a LISP-like Cambridge Polish prefix notation is used for operator application

## 4 Semantics of $\mathcal{L}_0$

The semantics of languages in  $\mathcal{L}_0$  are described, but not specified, in this section. In particular, no specific connection is established between a text and the result of applying a text operator to that text, or the interpretation of that resulting text.

An *interpretation*  $I$  consists of

- a subset  $TP_I$  of  $V$ ;
- a subset  $TX_I$  of text expressions;
- a title mapping  $ttl_I$ .

Let  $I$  be an interpretation. Let  $E$  be an expression of  $\mathcal{L}$ . The interpretation of  $E$  under  $I$ ,  $I(E)$ , is defined in Table 1.

**Table 1.**  $\mathcal{L}_0$  Semantics

|    | If $E$ is                                             | Then $I(E) =$                                             |
|----|-------------------------------------------------------|-----------------------------------------------------------|
| R1 | a proposition: $(N)$                                  | true if $N \in TP_I$ ; otherwise false                    |
| R2 | a text construction:<br>$(txt\ E_1 \dots E_n)$        | true if $(txt\ E_1 \dots E_n) \in TX_I$ ; otherwise false |
| R3 | an application of a text operator:<br>$(F_i\ \Gamma)$ | true if $(F_i\ \Gamma) \in TX_I$ ; otherwise false        |
| R4 | a text titling $(title\ N\ \Gamma)$                   | true if $ttl_I(N) = \Gamma$ ; otherwise false             |
| R5 | an import statement $(import\ N)$                     | is always true                                            |

The importation closure  $\mathcal{C}'_I$  of a corpus  $\mathcal{C}$  under a particular interpretation  $I$  is defined as discussed in the previous section 2. In general, the importation closures of a text of  $\mathcal{L}_0$  with circular importation dependencies will not be finite.

A corpus  $\mathcal{C}$  is satisfied by an interpretation  $I$  if for every text  $\Gamma$  in the importation closure  $\mathcal{C}'_I$  of the text under the interpretation  $I$ ,  $I(\Gamma)$  is true. When  $I$  satisfies  $\mathcal{C}$ , we write  $I(\mathcal{C})$  is true; otherwise,  $I(\mathcal{C})$  is false. A model  $M$  of a corpus  $\mathcal{C}$  is a satisfying interpretation ( $M(\mathcal{C}) = \text{true}$ ).

One corpus  $\mathcal{C}_1$  entails another corpus  $\mathcal{C}_2$  if the latter is satisfied whenever the former is satisfied ( $I(\mathcal{C}_2)$  is true whenever  $I(\mathcal{C}_1)$  is true). Two corpora are logically equivalent if they entail each other ( $I(\mathcal{C}_1) = I(\mathcal{C}_2)$  for all  $I$ ).

Note that if two interpretations  $I$  and  $J$  have the same title mapping  $I = J$ , then the importation closures of a particular corpus  $\mathcal{C}$  are the same,  $\mathcal{C}'_I = \mathcal{C}'_J$ . However, the specification above is somewhat too weak to draw conclusions, and we will need to further restrict the semantics of titling.

It is expected that any actual KR language that adopts this importation mechanism will have a distinct, and typically compositional, specification of semantics, and such semantics will imbue the language with certain algebraic properties. In the following subsections, esp. 4.1, 4.5, we develop algebraic properties that we assume to hold for all  $\mathcal{L}_0$  languages.

## 4.1 Titling Semantics

We suppose that the title mapping affects only the semantics of texts containing titling statements. In particular, we suppose that individual titling statements may be extracted from text operators and text construction statements as follows:

A corpus  
 $\{(\dots(txt \dots (title N \Gamma) \dots) \dots)\}$   
is fully-equivalent to the corpus  
 $\{(\dots(txt \dots \dots))$   
 $(\dots(title N \Gamma) \dots)\}$   
Similarly, a corpus  
 $\{(\dots(foo \dots (title N \Gamma) \dots) \dots)\}$   
is equivalent to the corpus  
 $\{(\dots(foo \dots \dots))$   
 $(\dots(foo(title N \Gamma)) \dots)\}$

Therefore, it is sufficient to consider texts of only two kinds: (type I) those that contain no titling statements, and (type II) those that contain only individual titling statements, nested within some finite number of text operators and (unary) text constructions. We call the set of all type I(II) texts in a particular language  $\mathcal{T}_I$  ( $\mathcal{T}_{II}$ ).<sup>3</sup>

When an importation of a type II text into a type I text is resolved, the titling statement may be extracted as above, giving a new type II text while leaving the original type I text unchanged except for the removal of the importation statement. Note that the interpretation  $I(S)$  of a type I statement  $S$  is independent of the title mapping  $ttl_I$  of the interpretation.

A titling-model of a corpus  $C$  is an interpretation  $J$  that satisfies (at least) the type II statements of the importation closure  $C'_J$ . A corpus is *title-satisfiable* if it has at least one titling-model.

## 4.2 Self-Contained Corpora

If a corpus contains an import statement, say for name  $N$ , and does not have a corresponding titling statement to restrain the identity mapping of  $N$ , it may still be possible to draw conclusions based on the explicit statements and importations that are well-defined. However, the algorithms for addressing these cases become more complex, so for this study we wish to distinguish between these cases. To this end, we introduce a property of “self-containment”, indicating a corpus contains enough titling statements to uniquely define all the importations it “needs”.

Ideally, self-containment would be syntactically defined. However, the best we can manage at this point is a semantic definition of self-containment because of our non-compositional semantics.

Let a corpus  $\mathcal{C}$  be semantically self-contained iff

- $\mathcal{C}$  is title-satisfiable;

---

<sup>3</sup> The algebraic conditions on type II texts are restated more precisely in different notation in subsection 4.5.

- there exists a unique importation closure  $\hat{\mathcal{C}}'$  (called the *canonical importation closure*) for all titling-models.

The uniqueness of the  $\hat{\mathcal{C}}$  closure arises from the presence of “enough” titling statements - too few titling statements leads to multiple importation closures for titling-models, while its existence arises from the absence of “too many” titling statements, esp. contradictory ones that lead to lack of title-satisfiability.

### 4.3 Cover

When corpora have infinite importation closures, practical reasoning is only possible when entailments can be determined from some finite set of texts. To this end, we define the concept of a *cover*.

Let corpus  $\tilde{\mathcal{C}}$  be a cover of a corpus  $\mathcal{C}$  iff

- $\tilde{\mathcal{C}}$  is logically equivalent to  $\mathcal{C}$  and
- for every interpretation  $I$ ,  $\tilde{\mathcal{C}}$  is satisfied by  $I$  iff  $I(\Gamma)$  is true for every text  $\Gamma \in \tilde{\mathcal{C}}$ .

In other words, a cover captures the semantics of the corpus in a different, and potentially smaller, package than the importation closure. Reasoning may be carried out practically using covers, provided they exist and are finite for the corpora of interest.

The second item in the definition of a cover makes use of a compositional truth value of a corpus, without considering the importation closure. We now define a notation for this.

Given any interpretation  $I$  of  $\mathcal{L}_0$ , there is a mapping  $I_S$ , which we will call the superficial interpretation mapping, from corpora of  $\mathcal{L}_0$  into truth values such that  $I_S(\mathcal{C})$  is true iff  $I(\Gamma)$  is true for every text  $\Gamma$  in  $\mathcal{C}$ .

The importation closure of a corpus contains the original corpus. Therefore, a superficial interpretation  $I_S(\mathcal{C})$  of true is necessary, but not sufficient, for an interpretation  $I$  to satisfy a corpus  $\mathcal{C}$ . In this notation, the satisfaction condition for all corpora may be stated as follows.

Given any interpretation  $I$  of  $\mathcal{L}_0$  and any corpus  $\mathcal{C}$  of  $\mathcal{L}_0$ , then  $I(\mathcal{C}) = I_S(\mathcal{C}')$  where  $\mathcal{C}'$  is the importation closure of  $\mathcal{C}$  under  $I$ .

If  $\tilde{\mathcal{C}}$  is a cover of  $\mathcal{C}$ , the second condition of the definition implies  $I(\tilde{\mathcal{C}}) = I_S(\tilde{\mathcal{C}})$ , while the first condition implies  $I(\tilde{\mathcal{C}}) = I(\mathcal{C})$ , and so we have an equivalent statement of cover:

A corpus  $\tilde{\mathcal{C}}$  is a cover of a corpus  $\mathcal{C}$  iff for every interpretation  $I$ ,  
 $I(\mathcal{C}) = I(\tilde{\mathcal{C}}) = I_S(\tilde{\mathcal{C}})$

In order to develop algorithms for determining a cover of a corpus, we need to know what are the characteristics of covers. In particular, we note that the second condition in the definition of a cover is a property independent of the corpus the cover “covers”. Because any cover is a cover of itself, being a cover (of something) is equivalent to meeting the condition

$$I(\tilde{\mathcal{C}}) = I_S(\tilde{\mathcal{C}})$$

We now show that being a cover is equivalent to the condition: for every non-satisfying interpretation  $I$ , there is a text  $\gamma_I \in \mathcal{C}$  such that  $I(\gamma_I)$  is false.

First we consider that case that  $I(\mathcal{C})$  is true. Then by definition, any text  $\Gamma$  in  $\mathcal{C}'_I$ ,  $I(\Gamma)$  is true. But  $\mathcal{C}'_I$  contains  $\mathcal{C}$ , therefore  $I_S(\mathcal{C})$  is true, and hence  $I(\mathcal{C}) = I_S(\mathcal{C})$ . This holds for any corpus, not just covers.

Now suppose that  $I(\mathcal{C})$  is false, i.e.,  $I$  doesn't satisfy  $\mathcal{C}$ . If the condition above is satisfied, then there is some text  $\Gamma$  in  $\mathcal{C}$  such that  $I(\Gamma)$  is false, and thus  $I(\mathcal{C})$  is false and hence  $I(\mathcal{C}) = I_S(\mathcal{C})$ . Conversely, suppose  $\mathcal{C}$  is a cover. Then  $I_S(\mathcal{C})$  is false, which can only be the case if there exists some text  $\Gamma$  in  $\mathcal{C}$  such that  $I(\Gamma)$  is false.

Our basic goals for this paper are

- to determine conditions on a  $\mathcal{L}_0$  language such that for any finite corpus, even one with an infinite importation closure, there exists a finite cover;
- to describe an algorithm for finding a cover of a corpus for languages that satisfy these conditions.

We will accomplish this with the help of some semantics-preserving transformations, as defined in the next section.

#### 4.4 Semantics Preserving Transformations

We first define a variant form of logical equivalence of texts, which we call superficial-logical-equivalence (SL-equivalence or SLE).

With respect to a given  $\mathcal{L}_0$  language, let

- two texts  $\Gamma, \Delta$  be *SL-equivalent* ( $\Gamma \equiv_{SL} \Delta$ ) iff  $I(\Gamma) = I(\Delta)$  for every interpretation  $I$ ;
- two corpora  $\mathcal{C}_1, \mathcal{C}_2$  be *SL-equivalent* ( $\mathcal{C}_1 \equiv_{SL} \mathcal{C}_2$ ) iff  $I_S(\mathcal{C}_1) = I_S(\mathcal{C}_2)$  for every interpretation  $I$ ;

It is noteworthy that logical equivalence and SL-equivalence are not comparable characteristics; there exists pairs of texts that are logically equivalent but not SL-equivalent and v.v. (see Example 5.1). A stronger equivalence relation between texts can be defined by combining the two: a pair of texts (or corpora)  $\Gamma, \Delta$  are fully equivalent if they are both logically and SL-equivalent, and we write  $\Gamma \equiv_F \Delta$ .

A corpus  $\mathcal{C}$  containing two fully-equivalent texts  $\Gamma, \Delta$  is fully-equivalent to the corpus  $\bar{\mathcal{C}}$  obtained from  $\mathcal{C}$  by deleting  $\Delta$ .

A text transformation that always generates an output text that is fully equivalent to the input text is called a semantics-preserving transformation.

#### 4.5 Algebra-like Structures of the Syntax

Corpora are sets, and so are subject to the binary operations of union and intersection with their usual properties. The set of all corpora in a  $\mathcal{L}_0$  language is, naturally, closed under these two operations. There are four equivalence relations to consider: syntactic equality, and superficial-, logical- and full-equivalence, and there is a set of equivalence classes of corpora for each of the latter three relations. There is a set of operators on corpora corresponding to importation closure under a particular interpretation of a  $\mathcal{L}_0$  language. These operators are also well-defined on logical and full equivalence classes

of corpora. The subset of corpora that are self-contained also has a unique operator corresponding to importation closure under any titling-model. Because the semantics of corpora are defined compositionally in terms of the interpretation of texts (of the importation closure), certain properties hold for all  $\mathcal{L}_0$  languages; e.g., a corpus entails its subsets, the union of unsatisfiable corpora is unsatisfiable, and so on.

At a more fine-grained level, we may examine algebraic-like structures involving texts. We may consider individual texts, or ordered tuples of texts and the same four equivalence relations apply as were considered for corpora. Concatenation is a binary operator acting on tuples, and text construction is a polyadic operator (which we will denote by  $Q$ ) on texts ( $Q : \mathcal{T}^* \rightarrow \mathcal{T}$ ) and a unary operator on text tuples. The semantics of text construction is not defined compositionally, so for general  $\mathcal{L}_0$  we cannot assume algebraic properties of text construction, such as commutativity, associativity or distributivity of construction over concatenation. We will consider subsets of the  $\mathcal{L}_0$  languages that have certain algebraic properties.

The most restrictive assumption that could be made about the semantics of text construction is

$$\{Q(\Gamma_1, \dots, \Gamma_n)\} \equiv_F \{\Gamma_1 \dots \Gamma_n\}$$

where  $\Gamma_i \in \mathcal{T}^*, 1 \leq i \leq n$ , which would make text construction equivalent to corpus construction, giving text construction a semantics similar to polyadic conjunction.

Let  $\mathcal{L}_0^+$  be the subset of  $\mathcal{L}_0$  languages that satisfy this *conjunctive-text-construction* property. In  $\mathcal{L}_0^+$ , the set of full-equivalence classes of texts is an Abelian algebra loop (closed, associative, commutative, with identity element) under  $Q$  as a binary operation.<sup>4</sup>

The minimal assumption that is imposed on all  $\mathcal{L}_0$  languages, as described above in subsection 4.1, for all  $\mathcal{L}_0$  confers on text construction a semantics like polyadic conjunction only on type II texts, and also has the semantics similar to binary conjunction when a type II text embedded in a tuple of arbitrary texts. In particular, for all  $\mathcal{L}_0$  languages,

$$\{Q(\Gamma_1, \dots, \Gamma_n)\} \equiv_F \{\Gamma_1 \dots \Gamma_n\}$$

where  $\Gamma_i \in \mathcal{T}_{II}^*, 1 \leq i \leq n$  and

$$\{Q(\dots \Gamma_1, \Delta, \dots \Gamma_2)\} \equiv_F \{\Delta, Q(\dots \Gamma_1, \dots \Gamma_2)\}$$

where  $\Delta \in \mathcal{T}_{II}$  and  $\dots \Gamma_i \in \mathcal{T}^*, i = 1, 2$ .

Similarly to text construction, the semantics of text operators,  $F \in \mathcal{F}, F : \mathcal{T} \rightarrow \mathcal{T}$ , is not defined compositionally. To achieve the separability of type I and type II texts, it is imposed above in subsection 4.1, for all  $\mathcal{L}_0$  languages, that text operators are distributive over polyadic text construction on type II texts, as well as a type II text embedded in a text tuple. That is, for all  $\mathcal{L}_0$  languages,

$$F(Q(\dots \Gamma_1, \Delta, \dots \Gamma_2)) \equiv_F Q(F(\Delta), F(Q(\dots \Gamma_1, \dots \Gamma_2)))$$

where  $F \in \mathcal{F}, \Delta \in \mathcal{T}_{II}$  and  $\dots \Gamma_i \in \mathcal{T}^*, i = 1, 2$ .

We let  $\mathcal{L}_0^\Omega$  be the subset of  $\mathcal{L}_0$  languages where text operator distributivity holds over all texts. That is,

$$F(Q(\Gamma_1, \dots, \Gamma_n)) \equiv_F Q(F(\Gamma_1), \dots, F(\Gamma_n))$$

where  $F \in \mathcal{F}$ , and  $\Gamma_i \in \mathcal{T}, 1 \leq i \leq n$ .

---

<sup>4</sup> Polyadic text construction is a shortcut for repetition of the binary construction in this case.

Further let  $\mathcal{L}_0^{\&\Omega} = \mathcal{L}_0^+ \cap \mathcal{L}_0^\Omega$ ; in  $\mathcal{L}_0^{\&\Omega}$  languages, the set of full-equivalence classes of texts could be considered an Abelian algebra loop with distributive operators (Abelian  $\Omega$ -loop), a generalization of the well-studied algebraic structure "group with operators" or  $\Omega$ -group.

## 5 Examples

The following examples correspond to existing[2] as well as new Test Cases for the resolution of circular importation dependencies.

### 5.1 Noncomparability of Logical- and SL-Equivalence

In  $\mathcal{L}_0^+$ , the texts  $(\text{txt } (\text{import foo}))$  and  $(\text{txt } )$  are not logically equivalent, but they are SL-equivalent. On the other hand, the texts

$$\begin{aligned} &(\text{txt } (\text{title foo } (A)) (\text{import foo})) \\ &(\text{txt } (\text{title foo } (A)) (A)) \end{aligned}$$

are logically equivalent in  $\mathcal{L}_0^+$  but not SL-equivalent. This demonstrates that the concepts of logical and SL-equivalence are not comparable, in general.

### 5.2 No Cover if Not Self-Contained

Texts that are not semantically self-contained will, in general, have no cover (finite or otherwise). For example, the corpus

$$\{(\text{txt } \text{import foo})\}$$

has no cover. The importation closure is still defined for every interpretation, but its content is determined by the image of "foo" in the title mapping of the interpretation, and this can vary arbitrarily.

### 5.3 Simple Circular Importation

Consider a self-contained corpus consisting of two texts

$$\begin{aligned} &\{(\text{title foo } (F_0 (\text{import foo}))) \\ &(\text{import foo})\} \end{aligned}$$

The importation closure contains

$$\begin{aligned} &(F_0(\text{import foo})) \\ &(F_0(F_0(\text{import foo}))) \\ &(F_0(F_0(F_0(\text{import foo})))) \\ &\dots \end{aligned}$$

as well as the original texts. Similar examples can be constructed with three or more texts where multiple operators are composed as follows:.

$$\begin{aligned} &\{(\text{title foo } (F_0 (\text{import bar}))) \\ &\{(\text{title bar } (F_0 (\text{import foo}))) \\ &(\text{import foo})\} \end{aligned}$$

This example demonstrates that circular importation generates an infinite importation closure from a finite corpus.

Examining the results of this example from the perspective of algebra-like structures, we focus on the interaction of elements of  $\mathcal{F}'_F$ , full-equivalence classes on the closure of the set of text operators  $\mathcal{F}$  under the composition operation. Text operators  $F_0, F_1$  are fully-equivalent iff

$$(F_0 \Gamma) \equiv_F (F_1 \Gamma) \text{ for every } \Gamma \in \mathcal{T}.$$

We can achieve a finite cover for the corpora in this example if the closure under composition of a finite set of  $\mathcal{F}'_F$  is finite and is itself a subset of  $\mathcal{F}$ , a kind of compactness property. Let  $\bar{\mathcal{L}}_0$  denote the subset of  $\mathcal{L}_0$  languages that satisfy this *compositional-compactness* property.

#### 5.4 Concatenation of Multiple Operators applied to a Circular Import

A more complex circular importation dependency is demonstrated here:

$$\{(title \text{ foo} (txt(F_0 (import \text{ foo})) (F_1 (import \text{ foo})) (F_2 (import \text{ foo})))) \\ (import \text{ foo})\}$$

This example demonstrates the need for distributivity of text operators over text construction ( $\mathcal{L}_0^Q$ ) if we are to make progress on determining a finite cover for such a corpus. Assuming distributivity of the text operators over text construction, the importation resolution generates arbitrary finite compositions of the text operators  $F_0, F_1$  and  $F_2$ . If the language has the compositional-compactness property, each member of this infinite set of operator compositions is fully-equivalent to some member of a finite subset of  $\mathcal{F}'$ , and thus a finite cover of the original corpus may be obtained.

#### 5.5 Concatenation of Proposition and Operator applied to a Circular Import

Consider the corpus consisting of two texts

$$\{(title \text{ foo} (txt(A) (F_0 (import \text{ foo})))) \\ (import \text{ foo})\}$$

Importation resolution yields

$$\begin{aligned} & (txt(A) (F_0 (import \text{ foo}))) \\ & (txt(A) (F_0 (txt(A) (F_0 (import \text{ foo}))))) \\ & \dots \end{aligned}$$

We again make use of the distributive property of text operators over text construction:

$$(txt(A) (txt(F_0(A)) (F_0(F_0(import \text{ foo})))))$$

Further resolution of importation produces

$$(txt(A) (txt(F_0(A)) (txt(F_0(F_0(A))) (F_0(F_0(F_0(import \text{ foo})))))))$$

...

We may take advantage of the compositional-compactness property to rewrite this as

$$(txt(A) (txt(F_0(A)) (txt(F_1(A)) (F_2(import \text{ foo})))))$$

However, in the absence of at least some aspects of conjunctive semantics for text construction there is little more that can be done towards deriving a finite cover.

Let us suppose that text constructions can be flattened,

$$Q(\dots \Gamma_1, Q(\dots \Gamma_2), \dots \Gamma_3) \equiv_F Q(\dots \Gamma_1, \dots \Gamma_2, \dots \Gamma_3)$$

for all  $\Gamma_i \in \mathcal{F}, 1 \leq i \leq n$ . Then the text above may be transformed to the fully-equivalent

$(\text{txt } (A) (F_0 (A)) (F_1 (A)) (F_2 (\text{import foo}))).$

Further suppose that the semantics of text construction is independent of the order of texts or duplication of texts in the tuple. Then application of compositional-compactness implies that there are a finite number of full-equivalence classes for texts of this form, given that the  $F_i$  are generated by composition of  $F_0$  with itself.

## 6 Summary of Conditions for Applicability of Cover Determination Algorithm

Let  $\bar{\mathcal{L}}_0^{+\Omega} = \bar{\mathcal{L}}_0 \cap \mathcal{L}_0^+ \cap \mathcal{L}_0^\Omega$  be the subset of  $\mathcal{L}_0$  with the following properties.<sup>5</sup>

- $(\bar{\mathcal{L}}_0)$  The language is compositionally-compact: the closure under composition of a finite subset of  $\mathcal{F}'_F$  is finite and is itself a subset of  $\mathcal{F}_F$ , where  $\mathcal{F}'_F$  is the closure, under composition, of the set  $\mathcal{F}_F$  of fully-equivalent classes of text operators ( $\mathcal{F}$ ) in the language;
- $(\mathcal{L}_0^\Omega)$  Text operators distribute over text construction:  

$$F(Q(\Gamma_1, \dots, \Gamma_n)) \equiv_F Q(F(\Gamma_1), \dots F(\Gamma_n))$$
where  $F \in \mathcal{F}$ , and  $\Gamma_i \in \mathcal{T}$ ,  $1 \leq i \leq n$ ;
- $(\mathcal{L}_0^+)$ 
  - Text constructions can be flattened (associativity, identity):  

$$Q(\dots \Gamma_1, Q(\dots \Gamma_2), \dots \Gamma_3) \equiv_F Q(\dots \Gamma_1, \dots \Gamma_2, \dots \Gamma_3)$$
for all  $\Gamma_i \in \mathcal{F}$ ,  $1 \leq i \leq n$ ;
  - Text construction is independent of the order (commutativity):  

$$Q(\Gamma, \Delta) \equiv_F Q(\Delta, \Gamma)$$
 for all  $\Gamma, \Delta \in \mathcal{F}$ .
  - Text construction is independent of duplication (idempotency):  

$$Q(\Gamma, \Gamma) \equiv_F Q(\Gamma)$$
 for all  $\Gamma \in \mathcal{F}$ .

In any  $\bar{\mathcal{L}}_0^{+\Omega}$  language, we define an elementary statement to be a single proposition, titling or importation statement. Further, let an elementary text to be an elementary statement or the application of finite set of text operators and/or unary text construction to at most one elementary statement.

Each elementary text is a member of a full-equivalence class of elementary texts, and by the assumed closure of composition, each elementary text may be represented using at most one text operator.

By application of the assumed algebraic properties to a given text in a  $\bar{\mathcal{L}}_0^{+\Omega}$  language, the text may be converted to a fully-equivalent text having the form of a text construction over some finite set of elementary texts. This representation is similar to a normal form in logic, so we call it a normal form of the text (it is not necessarily unique). A normal form  $\bar{\mathcal{C}}$  of a corpus  $\mathcal{C}$  is a set of texts such that each text of the original corpus  $\mathcal{C}$  has a normal form in  $\bar{\mathcal{C}}$ .

Given a self-contained corpus  $\mathcal{C}$ , there is a unique importation closure  $\hat{\mathcal{C}'}$  for all title models. Let  $\bar{\mathcal{C}'}$  be a normal form of  $\hat{\mathcal{C}'}$ . Because of the compositional-compactness property, only a finite number of text operators are needed to represent all FE-classes of texts in  $\bar{\mathcal{C}'}$ .

---

<sup>5</sup>  $\bar{\mathcal{L}}_0^{+\Omega}$  languages are Abelian  $\Omega$ -loops where every finitely-generated sub- $\Omega$ -loop is compact relative to the discrete topology of the full-equivalence relation (i.e. has a finite number of FE-equivalence classes).

## 7 Data Model

We describe here a data model providing a compact representation of a normal-form text or FE-class from a particular  $\bar{\mathcal{L}}_0^{+\Omega}$  language. The representation is designed to facilitate explanation of the algorithm for determining the cover of a self-contained corpus from such languages.

### 7.1 Corpus Table

A corpus is represented in one or two tables or arrays (if split, one for the type I texts and another for the type II texts). Each FE-class of the corpus is represented in a set of rows, one row for each elementary text in a normal form. A row is associated with an FE-class by its "classID" field, which contains an identifier.

In each row of the corpus table, the remainder of the fields contain a compact representation of an elementary text. The "txt" field contains a binary value indicating the presence of a unary text construction operator. The "op" field contains an identifier for a text operator, if present, with null or a special value to indicate its absence. The final fields describe the elementary statement, providing either the name of a proposition ("prop"), the title invoked by an importation statement ("import"), or the primary key of a titling statement("titleID"), to be discussed in the next section.

### 7.2 Titling Table

The texts associated with all titling statements that occur within the corpus under consideration are represented in the titling table, whether or not they are *active* (are asserted in the type II corpus table). Each row in the table corresponds to one elementary text in a normal form of the titled text.

It is possible for the same title to be assigned to two different texts within a corpus. There are certain circumstances when such a corpus may even be satisfiable, such as the second titling statement occurring within another titling whose text is never imported, or a text operator is applied that translates to a non-conflicting title. Therefore we cannot rely on the text title (the "title" field of the table) as a primary key for titling statements; the "id" field contains an identifier that creates the primary key for titling statements (optionally jointly with the title).<sup>6</sup>

The remaining fields of the row contain a representation of the elementary text in the same form introduced in the previous section for the corpus table(s).

### 7.3 Representation of Example 5.5

The corpus from Example 5.5 is represented in the corpus Table 2 and titling Table 3.

---

<sup>6</sup> Alternatively the table schema could be normalized by factoring out the first column of the titling table into another table where the title name associated with each titling statement identifier is recorded. This table is particularly important in the more general case, not implemented here, where a text operator may modify the title of a titling statement.

**Table 2.** Example 5.5 Corpus Table (combined types I and II) - Initial State

| classID | txt | op | prop | import | titleID |
|---------|-----|----|------|--------|---------|
| 1       | 0   |    |      |        | foo1    |
| 2       | 0   |    |      | foo    |         |

**Table 3.** Example 5.5 Titling Tables

| title | id   | txt | op | prop | import | titleID |
|-------|------|-----|----|------|--------|---------|
| foo   | foo1 | 1   |    | A    |        |         |
| foo   | foo1 | 1   | F0 |      | foo    |         |

## 8 Determination of Cover

We first state the algorithm procedurally and then illustrate for a particular example.

### 8.1 Overview of Cover Determination Algorithm

The procedure for updating the corpus table is described for the case of resolution of importations individually and assuming titling statements are unaffected by text operators:

- Construct the corpus and title tables, providing separate titleID's for each syntactically distinct titling statement.
- Locate an importation statement in some class in the corpus type I table.
- Determine if the text to be imported is well-defined. If there are multiple titling statements for this title asserted in the corpus type II table, the corpus is unsatisfiable. If there is no asserted titling statement for this title, defer the resolution of this importation until later - a titling statement for this title may be imported into the corpus elsewhere.
- Perform an importation resolution and update the corpus table.
  - copy the class containing the selected importation into a new set of rows, providing a new, unused classID.
  - locate the row, in the new class, corresponding to the selected importation to be resolved.
  - Update the text construction field in this row by binary OR between the existing value and the value in the FE-class to be imported.
  - Update the text operator field with the FE-class identifier obtained (e.g. from table lookup) for the composition of the existing operator applied to the operator in the class to be imported.
  - Compare the new FE-class to existing classes, and discard if it is equal to any existing class. The order of rows is not significant.
- Repeat the resolution and update process, applying only once to each individual importation statement in the corpus. Note that the order of application is immaterial except in the case of ill-defined imported texts. If all texts to be imported are well-defined, a fixed point will be reached.

- If a fixed point cannot be achieved due to ill-defined import(s), throw an error that a cover does not exist due to missing or conflicting titling statement(s), as appropriate.

We present the cover determination algorithm by illustration as it would apply to Example 5.5. The second text in the original corpus has an importation statement (row 2 of the corpus table) which invokes the name “foo”. There is no text operation applied to the importation, so no name translation need be performed.

There is only one active titling statement (titleID “foo1”) in the corpus table and it applies to the name “foo”. Therefore the text to be imported is well-defined. Now if we carry out the importation resolution explicitly and normalize we obtain the text

$(\text{txt } (A) (F_0 (\text{import foo})))$

We then tentatively add the representation of this text to the corpus table. In terms of the tabular representation, this is a sort of nonlinear join.

**Table 4.** Example 5.5 Corpus Table - After Step 1(new rows only)

| classID | txt | op | prop | import | titleID |
|---------|-----|----|------|--------|---------|
| s3      | 1   |    | A    |        |         |
| 3       | 1   | F0 |      | foo    |         |

We compare the new text with existing texts in the corpus. In this case, the new text is different and thus is retained. Further, the new text has an importation so resolution is repeated. We assume, for illustration purposes, that  $F_0 \circ F_0 = F_0$ . In general there will be a finite set of FE-classes of operators containing  $F_0^n$ ,  $n \geq 1$ .

**Table 5.** Example 5.5 Corpus Table - After Step 2 (new rows only)

| classID | txt | op | prop | import | titleID |
|---------|-----|----|------|--------|---------|
| 4       | 1   | F0 | A    |        |         |
| 4       | 1   | F0 |      | foo    |         |

The new class is again different than existing classes and is retained. It also has an importation so resolution is repeated. However, this time the new class is the same as the class it was derived from. The procedure has reached a fixed point and terminates.

## 9 Discussion

The following feedback to the ISO CL language design was developed on the basis of the analysis above:

- This approach allows a design decision regarding whether all importations in a corpus should be resolved simultaneously, or individually. The algorithm reveals:

- Individual importation resolution is more convenient to implement.
- For simultaneous resolution, there are cases where the algorithm as defined above fails to determine a cover when one does exist. For example, with the two text corpus
 

```
{(title bar (title foo (A)))
(txt (import foo) (import bar))}
```

 simultaneous resolution of the two importations in the second text fails because of a missing titling statement for “foo”. Individual resolution introduces the missing titling statement when the importation for “bar” is resolved.
- There is much redundancy in copying an entire text whenever an importation resolution is performed, which could be eliminated if text construction has conjunction-like semantics, so that every corpus can be expanded to a fully-equivalent corpus containing only elementary texts. This would not only reduce the size of the cover; it would make the termination criterion easier to check, since only single rows would need to be compared. However, conjunction-like semantics implies monotonicity; this simplification would not be applicable to nonmonotonic logics, such as defeasible logics.

## 10 Conclusion

We have investigated a family of propositionally-based languages containing generic text operators and a titling/importation mechanism, identifying a set of characteristics that allow a practical implementation of importation closure, in preparation for reasoning, that is robust in the presence of circular importation dependencies. The study has demonstrated its utility by providing feedback to the revision of a standard KR language, ISO CL. The extension of the analysis to the case of text operators that modify titling statements warrants further consideration.

**Acknowledgements** Thanks to Fabian Neuhaus for valuable suggestions on this paper.

## References

- [1] ISOIEC. *Information technology Common Logic (CL): a framework for a family of logic-based languages*.
- [2] Fabian Neuhaus and Tara Athan. *CL Semantics Strawman @ONLINE*. Apr. 2013. URL: <http://philebus.tamu.edu/pipermail/cl/attachments/20130405/153ad554/attachment-0001.pdf>.
- [3] Fabian Neuhaus and Pat Hayes. “Common Logic and the Horatio problem”. In: *Appl. Ontol.* 7.2 (Apr. 2012), pp. 211–231. ISSN: 1570-5838. URL: <http://dl.acm.org/citation.cfm?id=2351667.2351672>.

# Transforming Association Rules to Business Rules: EasyMiner meets Drools

Stanislav Vojíř, Tomáš Kliegr, Andrej Hazucha, Radek Škrabal, Milan Šimůnek

Department of Information and Knowledge Engineering, University of Economics,  
Nám. Winstona Churchilla 4, Prague 3, 130 67, Czech Republic  
`{stanislav.vojir|tomas.kliegr|andrey.hazucha|xskrr06|simunek}@vse.cz`

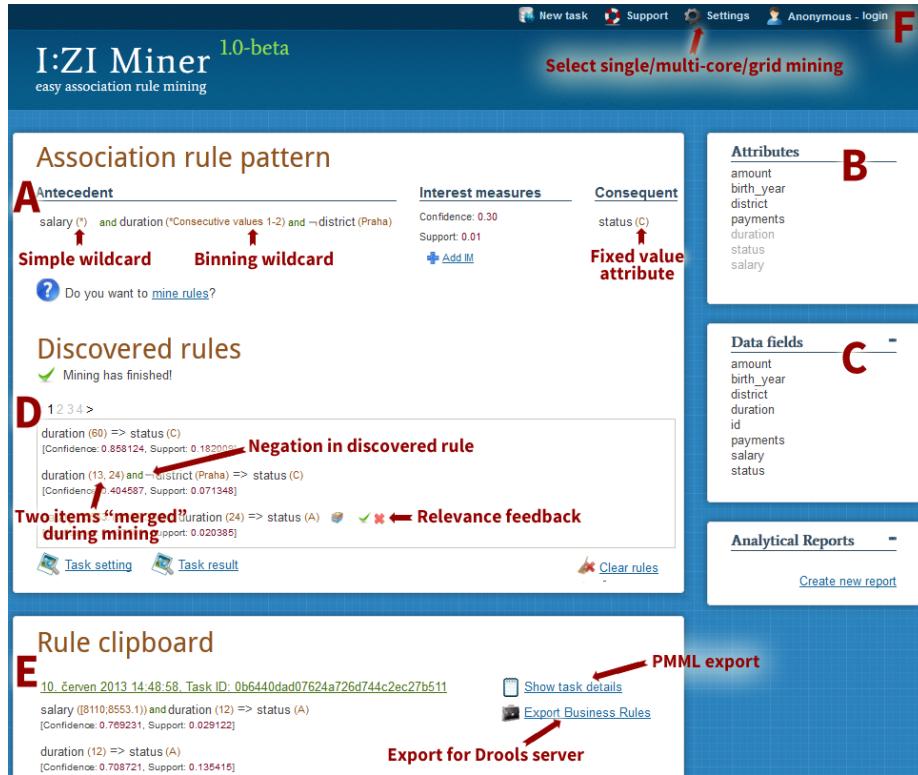
**Abstract.** EasyMiner ([easyminer.eu](http://easyminer.eu)) is a web-based association rule mining software based on the LISp-Miner system. This paper presents a proof-of-concept workflow for learning business rules with EasyMiner from transactional data. The approved rules are exported to the Drools business rules engine in the DRL format. The main focus is the transformation of GUHA association rules to DRL.

## 1 Introduction

The EasyMiner association rule mining system discovers rules from a table of objects. The system outputs all rules which hold in the given dataset in a certain predefined statistical sense. An example of a rule is  $\lceil \text{Amount} = \langle 100.000; 200.000 \rangle \wedge \text{District} = \text{Prague} \rightarrow_{0.7, 100} \text{Status} = \text{A} \rceil$ . Such a rule is learnt from a table (data matrix), where each row corresponds to one client of a bank, and it contains at least the following data: amount borrowed, district of the customer and loan status. Rule confidence 0.7 denotes that in this table, it is true that for 70% of clients from Prague who borrowed 100 to 200 thousand Czech crowns, the loan was A-grade. The *support* of the rule is 100, which means that there were at least 100 such clients.

The discovered rules are either exploited in a qualitative way by an expert, or used to perform classification (scoring) of incoming objects (e.g. [7]). With EasyMiner we attempt for a midway between these approaches: expert selects only some of the discovered rules, which are then interpreted as business rules. While the idea of interaction of a domain expert with discovered rules is not new [2], to the best of our knowledge, EasyMiner is the only web-based system which supports the complete cycle: data upload, preprocessing, mining, user interaction with the discovered rules, and export of selected rules to a business rules engine.

This paper is organized as follows. Section 2 describes EasyMiner and its workflow. The syntax of association rules output by EasyMiner is detailed in Section 3. Section 4 describes the transformation of rules to the DRL format. The description of the demo and the access details are listed in Section 5. The paper is concluded with some remarks on the applicability of the described transformation setup to other rule learners and with outlook for further work.



**Fig. 1.** EasyMiner screenshot

## 2 EasyMiner

EasyMiner is a sister project of the association rule learning system LISp-Miner ([lispminer.vse.cz](http://lispminer.vse.cz), [10]), which is a desktop/server-based system developed since the mid-1990s. The original paradigm of rule mining in LISp-Miner was that the discovered rules are pieces of knowledge intended for “human consumption”. EasyMiner, introduced at ECML’12 [12] as I:ZI Miner,<sup>1</sup> is both an interactive web application, which allows interactive pattern mining, and a web-service layer on top of LISp-Miner.

EasyMiner allows the user to perform the complete association rule mining task and review the discovered rules from an Internet browser.

<sup>1</sup> The first predecessor of EasyMiner called Association Rule Query Designer was introduced in [4]. This system was used for querying mining results stored in a knowledge base, not for performing live mining.

## 2.1 Data import

The imported data are in tabular form (a CSV file or a MySQL table). For columns with many distinct values it is strongly recommended to perform pre-processing by grouping similar values into a smaller number of *bins*. This can be done either automatically by a built-in heuristic algorithm on data import (numeric fields only), or manually after the mining task is setup. An example of a binning result is replacing all the say 60 distinct values of attribute “age” with just five values such as  $\langle 15; 23 \rangle, \langle 23; 37 \rangle, \langle 37; 49 \rangle, \langle 49; 53 \rangle, \langle 53; 75 \rangle$ .

## 2.2 Defining the Mining task

Once the data are imported, the user is presented with the main EasyMiner screen. The mining task is defined in the *Pattern Pane* (Fig. 1A) by selecting interest measures and placing attributes from the *Attribute Palette* (Fig. 1B) on the left and right side of the rule.

The set of interest measures includes the industry-standard confidence, support and lift measures and about 10 additional measures. All measures can be freely combined. The setting of an interest measure also involves a selection of a threshold value.

By dragging attributes on the left and right side of the rules respectively, the user decides which attributes *might* appear in the rule. For each attribute, it is also possible to define the set of its values considered during mining using the following options:

- fixed value: attribute must use a specific bin as its value if it appears in a rule
- simple wildcard: the system tries all single bins for the attribute value
- dynamic binning wildcard: during mining time, the system creates broader bins by merging bins created in the preprocessing stage into one bin. An example of a dynamically created bin is  $\langle 15; 23 \rangle \vee \langle 23; 37 \rangle$ .

It should be noted that while dynamic binning wildcard is convenient, it can significantly increase the computation time. To alleviate this problem, the user can select from several dynamic binning wildcards and thus restrict the size of the hypothesis space (e.g. only consecutive values are attempted to be merged). When the originally created preprocessing does not produce satisfactory results and dynamic binning does not help or is computationally infeasible, the recommended action is to create new attributes by dragging the names of columns from the input data from the *Data Field Palette* (Fig. 1C) to the *Attribute Palette* (Fig. 1B). After dropping the column to the *Attribute Palette*, the user defines custom preprocessing (binning). In this way, the mining task can contain multiple attributes derived (different binning) from the source data field.

Manual binning has also one significant advantage in the business rules context: bins can have user friendly names. Instead of bin  $\langle 53; 75 \rangle$  (result of automatic binning), the user can create more meaningful bins, e.g. by creating a bin  $\langle 60; 75 \rangle$  and naming it “senior”.

### 2.3 Mining

Once the user completes the setting of a mining task and clicks on the *mine rules* link, EasyMiner converts all the user settings to a variant of the PMML format [6] and submits the task via a web service to LISp-Miner. Depending on the configuration, defined via the Settings link (Fig. 1F), the task is executed in a single or multi-threaded LISp-Miner instance, or on the grid [11].

The discovered rules are returned to the EasyMiner front-end incrementally, as LISp-Miner progresses through the search space. Real-time results are shown in the *Result Pane* (Fig. 1D).

### 2.4 User Interaction with Results

The user oversees the discovered rules and tries to select the ones, which he or she thinks would bring value when deployed. The system offers two aids to the user: the strength of the rule and filtering based on a knowledge base.

The strength of the rule is indicated by the value of interest measures which the user selected in the *Pattern Pane*. The values for all discovered rules displayed there meet or exceed the preset thresholds. Generally, the higher the interest measure value above the threshold, the better the rule. Despite this simple “metarule of thumb”, the user should understand the semantics of the interest measures. As a future extension of the system, we plan to provide a representation of the rule in a human-friendly textual form, which should lower the requirements on user training (see Sec 6).

Discovered rules can be checked against a knowledge base of stored rules by issuing a *confirmation* or *exception* query [5]. Confirmation query returns rules from knowledge base, which contain in the antecedent only attributes contained in the discovered rule’s antecedent, and for each of these attributes, there is at least one overlapping value. The same must apply for the consequent. Exception query returns rules with the same antecedent and a consequent which share at least one attribute, and at least in one of the shared attributes there is no overlap in attribute values.

EasyMiner makes the check of the discovered rules against the knowledge base transparent for the user by embracing a relevance feedback paradigm: if the discovered rule is only a confirmation of a rule in the knowledge base, it is visually suppressed by gray font. In contrast, if the rule is an exception, it is highlighted in red. A green tick, moving the rule to the Rule clipboard, also stores the rule into the knowledge base. The relevance feedback module is a Java application running on top of the XML Berkeley database, which communicates with EasyMiner via a web service.

### 2.5 Rule Clipboard

The rules confirmed by the user are moved to the *Rule clipboard* (Fig. 1E). The rules in the clipboard are grouped according to the task, in which they were discovered. By clicking on the “Show task details” button, the user is presented

an HTML page with a complete definition of the mining task and the description of the data. Technically, this report is generated with an XSLT transformation from the GUHA AR PMML [6] XML export of the LISp-Miner system, which is available under the *Task result* link in the Result Pane (Fig. 1D).

The *Export Business Rules* link exports the rules in the clipboard for a specific task to the Drools server. For demo purposes, this link shows the DRL serialization of the rules.

### 3 BR-GUHA Association Rule

In theory, the LISp-Miner system used by EasyMiner mines generic GUHA association rules [9]. The high expressivity of GUHA rules is not suitable for this initial work on the transformation of association rules to business rules. While EasyMiner contains some simplifications in comparison with the full LISp-Miner implementation, the “EasyMiner” rules are still too expressive. In this section, we describe BR-GUHA 0.1, a constrained version of GUHA rules, which is suitable for transformation to Business Rules.

In the formal definition of GUHA rules, antecedent and consequent of the rule are defined in terms of boolean attributes, which are, in turn, defined as conjunction or disjunction of boolean attributes or literals. EasyMiner simplifies this generic recursive structure to a fixed three layer model, which eases the manipulation with the discovered rules:

- Layer 1: Antecedent is a conjunction of *derived boolean attributes*, Consequent is a non-empty conjunction of *derived boolean attributes*,
- Layer 2: A derived boolean attribute is a conjunction or disjunction of *literals*,
- Layer 3: A *literal* is an attribute-value pair or its negation.

Further, it should be noted that:

- *Attribute* refers to the result of preprocessing, not to a field in the original data table,
- *Value* is a bin created during preprocessing, or a dynamically created bin (a disjunction of multiple bins).

By default, EasyMiner (and GUHA) allows the consequent of the rule to have the same rich structure as the antecedent. The consequent of the rule can thus contain for example a disjunction of multiple attributes, or a disjunction of values of one attribute.

In contrast, with Business Rules, a rule needs to have a *definite* outcome. To quote from the Drools documentation: *It is bad practice to use imperative or conditional code in the RHS of a rule; as a rule should be atomic in nature - "when this, then do this", not "when this, maybe do this"*<sup>2</sup>. In BR-GUHA the

---

<sup>2</sup> [http://docs.jboss.org/drools/release/6.0.0.Beta3/drools-expert-docs/html\\_single/index.html#d0e7386](http://docs.jboss.org/drools/release/6.0.0.Beta3/drools-expert-docs/html_single/index.html#d0e7386)

consequent of the rule is constrained to contain a positive literal (negation not allowed). Furthermore, the attribute value must correspond to a single value in the underlying data table (no binning or dynamic binning allowed). No restrictions are made to the antecedent of the rule.

The second important component of an association rule are the interest measures, the 4ft-quantifier in *GUHA* terminology [3, 9]. A 4ft-quantifier is composed from one or more 4ft-partial quantifiers, each associated with one or more quantifier values. While EasyMiner embraces the more commonly used term “interest measure”, in other respects it does not impose additional constraints.

In BR-GUHA, we constrain the EasyMiner setup to two interest measures. Only the most commonly used interest measures are supported: *confidence*, *support* and *lift*, all with just one associated value. The first measure must be support, and the second measure is either lift or confidence.

Technically, the constraints described in this section are imposed by not allowing certain features in the mining setup. Most BR-GUHA constraints are readily supported by EasyMiner.<sup>3</sup>

## 4 Representing EasyMiner Association Rule in DRL

This section describes an initial specification of the conversion procedure of the simplified GUHA rules (“BR GUHA 0.1”) to the Drools Rule Language (DRL). In this preliminary work, this specification is done informally, through examples of transformation result for the relevant syntactic features.

### 4.1 Running Examples

Throughout this section, two example GUHA rules will be used. The first takes up the simple rule from the Introduction, while in the second all the syntactic features are used.

#### Rule 1

$\neg \text{Amount} = \langle 100.000; 200.000 \rangle \wedge \text{District} = \text{Prague} \rightarrow_{0.7, 100} \text{Status} = \text{A}^\top$ ,  
where 0.7 is the confidence value and 100 the support.

#### Rule 2

$\neg (\text{Amount} = \langle 100.000; 200.000 \rangle \vee \text{Duration} = 1\text{year}) \wedge \neg (\text{District} = \text{Bruntal})$   
 $\wedge (\text{Age} = [\text{Senior} \vee \text{Student}] \vee \text{Payments} = \langle 5.000; 10.000 \rangle)$   
 $\wedge \text{Education} = \text{university} \rightarrow_{0.95, 20} \text{Status} = \text{B}^\top$ ,  
where 0.95 is the confidence value, and 20 the support.

### 4.2 Attributes

To comply with the Drools object-oriented principles, each attribute in a rule is transformed to an instance of the Drools `Attribute` class. In the following, we will refer to this instance as `DrlObj`.

---

<sup>3</sup> With the exception of disabling binning in the preprocessing stage

The names of the attributes in the discovered rules may not necessarily match the names of fields in the underlying data table. Since it is expected that the requests to the business rules engine will use the names of the fields from the underlying data table, rather than the custom names introduced during data preprocessing, the name of the instance is set to the name of data field on which the attribute is based. The same applies to attribute values.

Rule 1 features attribute-value pair `District=Prague`. Assuming that the name of the underlying data field is “district”, and the underlying data value “Praha” was renamed during preprocessing to “Prague”, the resulting DRL fragment is as follows:

```
DrlObj (name == "district", value == "Praha")
```

### 4.3 Interest measures

The action of a user confirming the rule and exporting it to the business rule system, strips away the “fuzziness” from the rule, replacing the interest measure with a causal relationship. The original value of interest measures can, however, be used to define the conflict resolution strategy.

Consider object 1 depicted in Table 1.

| ID | amount  | district | age | duration | payments | education  |
|----|---------|----------|-----|----------|----------|------------|
| 1  | 120.000 | Praha    | 63  | 1year    | 11.000   | university |
| 2  | 110.000 | NA       | 61  | 1year    | 9.000    | university |

**Table 1.** Example objects

Both Rule 1 and Rule 2 match this object, however, the consequents of these rules are conflicting, since the status cannot be both A and B.

Drools offers multiple conflict resolution strategies. Interest measure values can be utilized in the *salience* strategy, by setting the salience property of a rule according to the value of lift or confidence interest measures, whatever is used in the rule. Since salience in Drools is an integer, while confidence is a float in the range of  $(0; 1]$  and lift is a float in the  $(0; inf]$  range, the original value of the interest measure needs to be multiplied by a scaling factor, e.g. 100, before it can be used as salience.

In association rule mining, it can be generally observed that with the increasing specialization of the antecedent, the confidence of a rule rises at the expense of decreasing rule support (as exemplified by Rule 1 and Rule 2). Specific rules are therefore preferred as their consequents are more likely to hold than for a consequent of a conflicting rule with a smaller number of conditions. To this end, the Drools *complexity* conflict resolution strategy, which favours rules with more conditions, should yield similar results as the salience strategy.

It should be noted that the statistical validity of a rule decreases with increasing specificity as each condition filters out some objects that would contribute to the *support* of the rule. However, we suggest not to take support into account during conflict resolution, since the fact that all rules considered have sufficiently high support is ensured by:

- support of the rule exceeding the minimum threshold set by the user during the mining setup,
- the user has explicitly approved the rule by placing it into the rule clipboard.

The use of the complexity strategy rather than the salience strategy also has the advantage that it naturally solves the situation when there are multiple conflicting rules with different interest measures. In this case, a comparison of salience would not make sense: the salience of 70, derived from confidence 0.7, and salience 110, derived from lift value 1.1, are incomparable.

Our preliminary conclusion is that the first approach to handle interest measures in the DRL export is to ignore them, and to use the complexity resolution strategy instead.

In our example, this strategy would favour Rule 2 over Rule 1.

#### 4.4 Binning

The values of attributes are a result of binning. The names of bins can be automatically generated, user-defined, or the same as the values of the underlying fields in the data table.

Since it is expected that the requests to the business rules engine will use values from the underlying data table, rather than the bin names, it is necessary to translate the bin names back to the values of the underlying datafield. In this step, one bin will be replaced by one or multiple values.

The resulting DRL depends on the data type of the attribute (numerical, nominal).

**Numerical attributes** The bins of the Amount attribute from Example 1 are created on a numeric range.

$\lceil \text{Amount} = \langle 100.000; 200.000 \rangle \rceil$

The result of transformation to DRL:

```
DrlObj(name == "amount", numVal >= 100000, numVal < 200000)
```

**Nominal attributes** The bins of the Education attribute were created by enumerating nominal values in the preprocessing stage for data mining. For example, bin “university” was created by merging values “undergraduate” and “graduate” of the underlying “education” data field.

$\lceil \text{Education} = \text{university} \rceil$

The result of transformation to DRL:

```
DrlObj(name == "education", value == "undergraduate"
|| value == "graduate")
```

#### 4.5 Dynamic Binning

A dynamic bin (multiple bins merged into one during mining) is present on attribute *Age* in Rule 2.

$$\lceil \text{Age} = [\text{student} \vee \text{senior}] \rceil$$

The result of transformation to DRL:

```
DrlObj(name == "age", (numVal >= 18 && numVal < 25) ||
(numVal >= 60 && numVal <= 75))
```

#### 4.6 Conjunction

Conjunction can be featured on the top level within the antecedent or consequent as in Rule 1 and Rule 2, or in a subexpression as in Rule 2.

##### Top level

$$\lceil \text{Amount} = (100.000; 200.000) \wedge \text{District} = \text{Prague} \rceil$$

This rule fragment is represented in DRL as

```
DrlObj(name == "amount", numVal >= 100000, numVal < 200000)
and DrlObj(name == "district", value == "Praha")
```

##### Subexpression

$$\lceil \text{Age} = \text{senior} \wedge \text{Payments} = (5.000; 10.000) \rceil$$

This rule fragment is represented in DRL as

```
DrlObj(name == "age", numVal >= 60, numVal <= 75)
and
DrlObj(name == "payments", numVal >= 5000, numVal < 10000)
```

## 4.7 Disjunction

Disjunction in a simplified GUHA rule can be present only as a subexpression within antecedent or consequent. Disjunction is present in Rule 2:

```
⊤(Amount=<(100.000; 200.000) ∨ Duration=1year)⊤.
```

The result of transformation to DRL:

```
DrlObj(name == "amount", numVal >= 100000, numVal < 200000)
or DrlObj(name == "duration", value == "1year")
```

## 4.8 Negation

Negation in an EasyMiner rule can be present only on a specific attribute-value pair. Negation is present in Rule 2:

```
⊤¬(district=Bruntal)⊤
```

The result of transformation to DRL:

```
DrlObj(name == "district", value != "Bruntal")
```

This assumes that the value of District is known. An alternative DRL rule, more truthful to the above EasyMiner rule, which would be fired even if the value of District is not available (as in object 2 in Table 1):

```
not DrlObj(name == "district", value == "Bruntal")
```

## 4.9 Consequent

The specification of the code in rule consequent is currently not yet finalized and the authors would welcome any input from the RuleML community. The provisional option currently implemented in the system is as follows:

```
then
 processResult(kcontext, "Status", "A");
end
```

The `processResult` is a static method which collects the results of fired rules. Its first argument is a rule context (provided by Drools) followed by the attribute name and its value from consequent of the association rule. As the next step, it is necessary to resolve the situation when multiple rules with different consequents have been activated. One of the options to accomplish this is to use the Drools `accumulate` function.

#### 4.10 Complete DRL

This section lists the complete DRL code for the two example rules.

```
import cz.vse.droolsserver.drools.DrlObj;
import function cz.vse.droolsserver.drools.DrlResult.processResult;

rule "ExampleRule1"
when (
 DrlObj(name == "amount", numVal >= 10000, numVal < 200000)
 and
 DrlObj (name == "district", value == "Prague")
)
then
 // a provisional construct
 processResult(kcontext, "Status", "A");
end

rule "ExampleRule2"
when (
(
 DrlObj(name == "amount", numVal >= 100000, numVal < 200000)
 or
 DrlObj(name == "duration", value == "1year")
)
and (not DrlObj(name == "district", value == "Bruntal"))
and
(
 DrlObj(name == "age", (numVal >= 18 && numVal < 25)
 || (numVal >= 60, numVal <= 75))
 or
 DrlObj(name == "payments", numVal >= 5000, numVal < 10000)
)
)
then
 // a provisional construct
 processResult(kcontext, "Status", "B");
end
```

### 5 Demo Scenario

The demo, accessible at <http://easyminer.eu/demo/ruleml2013>, shows the EasyMiner workflow supporting the business rules integration. All the data-mining steps described in Section 2 are shown as a screencast and as a live demo system. The demo finishes with the user clicking on the *Export as Business Rules* link, which shows the result of converting the rules in the rule clipboard to DRL.

## 6 Conclusion and Future Work

This paper presents a proof-of-concept system for learning business rules with EasyMiner from transactional data. The main focus of this paper is the transformation of GUHA association rules to the DRL format, used by the open source Drools business rules engine. In this preliminary work we have imposed some restrictions on the form of the GUHA rules being transformed. Nevertheless, the specification proposed here should support all features of conventional association-rule learning algorithms, i.e. those with output similar to the apriori [1] algorithm, plus some advanced features such as disjunctions or negations.

As a future work, we would like to investigate the possibilities for using and extending the human readable serializations of business rules, SBVR “Structured English” [8] in particular, as an alternative way of presenting the discovered rules to the user.

**Acknowledgements** The work described here was supported by grant IGA 20/2013 of the University of Economics, Prague and by the LinkedTV EU FP7 project.

## References

1. Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. Mining association rules between sets of items in large databases. In *SIGMOD*, pages 207–216. ACM Press, 1993.
2. Bart Goethals and Jan Van Den Bussche. On supporting interactive association rule mining. In *Proceedings of the 2 nd International Conference on Data Warehousing and Knowledge Discovery*, pages 307–316. Springer, 2000.
3. Petr Hájek and Tomáš Havránek. *Mechanizing Hypothesis Formation*. Springer-Verlag, 1978.
4. Tomáš Kliegr, David Chudán, Andrej Hazucha, and Jan Rauch. SEWEBAR-CMS: A system for postprocessing data mining models. In Monica Palmirani, M. Omair Shafiq, Enrico Francesconi, and Fabio Vitali, editors, *RuleML-2010 Challenge*, volume 639 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2010.
5. Tomáš Kliegr, Andrej Hazucha, and Tomáš Marek. Instant feedback on discovered association rules with PMML-based query-by-example. In *Web Reasoning and Rule Systems*. Springer, 2011.
6. Tomáš Kliegr and Jan Rauch. An XML format for association rule models based on guha method. In *RuleML-2010, 4th International Web Rule Symposium*, Berlin, Heidelberg, 2010. Springer-Verlag.
7. Bing Liu, Yiming Ma, Ching Kian Wong, and Philip S. Yu. Scoring the data using association rules. *Applied Intelligence*, 18(2):119–135, March 2003.
8. OMG (Object Management Group). Semantics of Business Vocabulary and Business Rules (SBVR), v1.0, 2008.
9. Jan Rauch. *Observational Calculi and Association Rules*. Studies in Computational Intelligence. Springer-Verlag, Berlin, 2013.
10. Jan Rauch and Milan Šimůnek. An alternative approach to mining association rules. *Foundation of Data Mining and Knowl. Discovery*, 6:211–231, 2005.

11. Milan Šimůnek and Teppo Tammisto. Distributed data-mining in the LISp-Miner system using Techila grid. In *Networked Digital Technologies'10*, pages 15–21, Berlin, 2010. Springer.
12. Radek Škrabal, Milan Šimůnek, Stanislav Vojíř, Andrej Hazucha, Tomáš Marek, David Chudán, and Tomáš Kliegr. Association rule mining following the web search paradigm. In Peter A. Flach, Tijl De Bie, and Nello Cristianini, editors, *ECML/PKDD (2)*, volume 7524 of *Lecture Notes in Computer Science*, pages 808–811. Springer, 2012.

# Constructing Controlled English for Both Human Usage and Machine Processing

Ping Xue<sup>1</sup>, Steve Poteet<sup>1</sup>, Anne Kao<sup>1</sup>, David Mott<sup>2</sup>, Dave Braines<sup>2</sup>

<sup>1</sup>Research & Technology, The Boeing Company, USA

{ping.xue, stephen.r.poteet, anne.kao}@boeing.com

<sup>2</sup>Hursley Emerging Technology Services, IBM, UK

{mottd, dave\_braines}@uk.ibm.com

**Abstract.** We present our on-going research on constructing and extending a version of Controlled English (CE) in support of knowledge sharing and decision-making for effective and efficient operations in the military coalition environment. This work would be useful for any multinational English speaking environment. This CE is intended for both human use and machine processing, providing:

- (i) A user-friendly language in a form of English enabling the user to use it in a fairly intuitive way.
- (ii) A precise language that enables clear, unambiguous representation of information that is amenable to rule-based interpretation and inferencing.

The paper focuses on the discussion of methods for CE construction while optimizing a balance between the naturalness for humans and machine readability of the CE language in light of theoretical considerations and empirical experimentations. We discuss certain aspects of CE syntax, semantics and the lexical model as examples. We also show sample CE-based knowledge-sharing capabilities.

**Keywords:** Ambiguity, coalition operations, Controlled English, decision-making, syntax and semantics, information extraction, linguistic variations, knowledge sharing, multi-nation collaborations, unstructured data

## 1 Introduction

As science and technology continue to advance, the volume of available data and information has been rapidly growing in both structured and un-structured forms, generated from a variety of sources including business processes, government and organizational policies, scientific activities, public web commentary as well as sensors and intelligence reports. These large data sets created by modern technologies, diverse in form and content, contain valuable and often critical information that, if

acquired and properly represented, can provide significant insights to improve knowledge-sharing and to support decision-making. However, the data are creations of human design with bias, and correct interpretation of data requires domain knowledge. As a result, information acquisition from the available data and knowledge-sharing among or across organizations are difficult. A major difficulty, as we observed in the military coalition context, comes from the fact that organizations (even related organizations) may have somewhat different underlying conceptual models of the world in addition to linguistic variations in terms of terminology, sentence structure, language usage and style. For structured data, metadata may also vary in semantics between domains; identical metadata elements may be used to refer to similar but distinct concepts. It is clear that we not only need automated data analysis tools but also a user-friendly common language that enables unambiguous information representation and facilitates a closer human-machine interaction as well.

In this paper, we present our on-going research on a controlled English being developed within a collaborative research alliance called the International Technology Alliance (ITA)<sup>1</sup> to support information-acquisition and knowledge-sharing for decision-making during coalition operations. Section 2 introduces ITA Controlled English (ITA CE, hereafter CE) and discusses our on-going effort to extend CE. Section 3 discusses tools and applications based on CE, including our initial implementation of CE for fact-extraction by various users, from knowledge engineers and linguists to non-technical domain-specialist users. Finally, section 4 summarizes CE and future work on CE extension in syntax and semantics for general expressivity in order to be able to capture and represent a diversity of concepts and to support a wider range of applications in the context of military coalitions.

## 2 ITA Controlled English

### 2.1 Controlled Natural Language

A controlled natural language (CNL) is a subset of a natural language using a restricted set of grammar rules and a restricted vocabulary. Well-known examples of controlled languages include ACE [1], CPL [2], PENG [3], Rabbit [4], Caterpillar Fundamental English [5], and STE (Simplified Technical English) [6] , with a common goal: to eliminate (or reduce) the ambiguity and complexity of a natural language, and thus to improve (or at least maintain) readability of the text for humans while allowing it to be processable by machines. ITA CE is an ITA variant of a controlled natural language originally developed by John Sowa [7].

It should be noted that there is often a tension between the human user-friendliness and machine predictability [8]. Predictable interpretation and reliable computation of

---

<sup>1</sup> In 2006, the US Army Research Laboratory (ARL) and the UK Ministry of Defence (MoD) established a collaborative research alliance with academia and industry partners called the International Technology Alliance (ITA) to address fundamental issues in Network and Information Sciences to enhance the abilities of the US and UK to conduct coalition operations.

a CNL requires deterministic property of the language. But a deterministic CNL is not necessarily easy for users who have not had any training in the CNL, because the restricted grammar and lexicon of the CNL may compete with his/her normal English intuition (i.e., the grammar and lexicon that the user has been exposed to since his/her birth). In general, the closer the CNL to the normal natural language, the more natural and the easier to use by humans, but the less predictable and the more computationally complex it will be for machines. How to achieve a good balance between these two is an important criterion as we design and develop the CE.

We need to make a distinction between ease of reading and ease of writing. In general, CNLs will be easier to read than to write, since the specific restrictions in syntax and vocabulary are more difficult for the writer to remember and follow. To this end, we have developed the CE Query Builder tool to help with the writing of CE, as discussed below in Section 3.1.

## 2.2 Goals of ITA CE

In addition to the general goal of balancing human usability and machine interpretability, CE should provide:

- A single, standardized language for various users, from different groups (e.g. the UK and the US) but also with different roles in the overall system (for example, end-users or soldiers in the battlefield, military analysts and planners, and system developers)
- The ability to express the basic facts (or propositions) required for a particular domain or application
- The ability to express the epistemic status of propositions (i.e. whether they are true or false, or just an assumption, possibly with a specified degree of certainty), or who believes the proposition; not all propositions will be assumed to be true with the same confidence
- Logical inference rules to allow the inferencing of additional facts from an initial set; in order to encode anything more than the most trivial kinds of knowledge, the ability to infer new facts from existing ones is necessary
- The ability to express the rationale behind a particular proposition; given the ability to infer new facts from old ones, decision makers need to understand the provenance and rationale of the facts on which they base their decisions
- Extensibility: the ability for users with various roles to extend the language in different ways; no system will ever be complete and the easier it is for various types of users to add new knowledge, the more adequate the system will be

### 2.3 Examples of ITA CE

The current version of CE is roughly consistent with First Order Predicate Logic and with existing ontology modeling languages such as OWL (Web Ontology Language).<sup>2</sup> It provides an unambiguous representation of information for machine processing, while aspiring to provide a human-friendly representation format that is directly targeted at non-technical domain-specialist users (such as military planners, intelligence analysts or business managers) to encourage a richer integration between human and machine reasoning capabilities [9]. In addition to more traditional areas such as knowledge or domain model representation and corresponding information, CE also encompasses the representation of logical inference rules [10], rationale (reasoning steps) [11], assumptions, and statements of truth (and certainty).

The CE currently permits a set of “plain” English sentences for stating propositions referring to entity existence, properties and relations:

- there is a person name Fred.
- the person Fred has French as language.
- the person Fred is married to the person Jane.

The CE also permits meta-statements that specify information about propositions such as their truth status or assumption:

- it is true that there is a person named Fred
- it is assumed that the person Fred is married to the person Jane.
- it is true to degree CV that Fred is a father.

Queries of a set of facts represented in CE can be made in forms like the following:

- for which X is it true that the person X is married to the person Jane

---

<sup>2</sup> It's the semantics rather than the syntax of CE that is compatible with FOL and existing modeling languages. The current relation between CE semantics and FOL is as follows: 1) the basic CE sentences are all given a FOL semantics in the definition of CE reference; 2) there are some parts of CE that have not yet been given a formal semantics in CE (such as the assumption-based logic); 3) not all of FOL can be represented in CE, for example certain combinations of existential quantifiers embedded in the scope of a universal quantifier. The relationship between CE and OWL (semantics) is as follows: 1) there are some parts of CE that cannot be represented in OWL, e.g. rules, assumptions, although potentially these could be represented in extensions to OWL such as RIF. 2) There are some parts of OWL that cannot (easily) be represented in CE, e.g. lists of explicit values for properties 3) There are one or two fundamental differences in philosophy, for example we prefer to make (nearly) all of the rules of inference to be explicit, whereas in OWL there are many implicit rules of inference. For more on the relationship between CE, FOL, and web modeling languages, see [10].

- for which X, Y and Z is it true that
  - the person X is the brother of the person Y and
  - the person Y is the father of the person Z

The creation or extension of a domain model (or a general model across domains) using CE is accomplished by the definition of (domain) concepts, relationships and properties. These are all achieved through the “conceptualise”<sup>3</sup> statement:

- conceptualise a ~ person ~ P.

A conceptualise statement creates the concept in question within the CE domain model. The concept is assumed to have a unique meaning, allowing unambiguous interpretation of a text string such as “person” in the above examples. The conceptualise statement introduces new concepts (including entity and event types, as well as their attributes and relations they can enter into) by putting them between tildes and using capitalized letters/strings for variables that would be replaced in a fact assertion or proposition statement.

Slightly more advanced examples are:

- conceptualise a ~ person ~ P that is an agent.
- conceptualise the person P
  - that has the value H as ~ height ~ and
  - has the value W as ~ weight ~
- conceptualise the person P
  - ~ is married to ~ the person P2

The first CE sentence creates “person” as a sub-concept of “agent” and the second indicates that it can have the properties “height” and “weight”. The last sentence asserts that it can have a “married” relationship with someone.

The meaning of a concept can be more fully defined by rules representing the logical relations between concepts and their properties. Logical rules can be represented as follows:

- if PREMISES then CONCLUSION
  - if ( the person X has the person Y as brother ) and
  - ( the person Z has the person X as parent )
  - then
  - ( the person Z has the person Y as uncle )

---

<sup>3</sup> The spelling of "conceptualise" is due to the origin of CE at IBM, UK.

If there are CE facts in the repository satisfying the first two premises (where the matching variables must have matching values), then the conclusion can be asserted, again with variables filled in from the premises, in accordance with traditional modus ponens inferencing.

Rationale for a particular proposition is based on the following form of statement:

- CONCLUSION because PREMISES

the task T1 has the agent A1 as executor  
because  
the plan P1 has the agent A1 as executor and  
the plan P1 contains the task T1.

A rationale would consist of a chain or network of such statements, tracing the provenance of a particular assertion and allowing the user to see the source and status (assumption or fact, or degree of certainty) of the different premises the fact is ultimately based on. The rationale also provides a basis for more complex processing, for example propagating degrees of certainty according to some particular theory uncertain inferencing, or simply highlighting assumptions or propositions with degrees of certain below a particular threshold, although we have not investigated this yet.

Note that the meaning of a concept is given by the inferences that can be made from CE statements that use that concept. These, in turn, are related to the place of the concept in a domain concept is-a hierarchy, the various attributes it can take and the types of relationships it can enter into, and ultimately the logical inference rules that it participates in.

Clearly, the examples given above are simplistic and basic, but with these simple mechanisms, CE has been used in practical applications with reasonable coverage, which we will discuss in Section 3.

## 2.4 Extending CE

As mentioned above, an important goal of CE is extensibility of the language. The conceptualise construct already allows for lexical extensibility, introducing new concepts for things and their associated properties and relationships<sup>4</sup>. However, it does not provide an ability to extend the syntax of CE. It is desirable to allow extension of the syntax because there are some areas where, although the concept can be expressed in CE, the expression is not very natural. For example, CE currently does not have adjectives per se. Adjectival expressions are captured via noun-like concepts. So in order to say that “my car is red”, we have to say:

- The car mycar is a red thing.

where “red thing” is an unanalyzed concept, despite the space in its name.

---

<sup>4</sup> Note that “thing” here is at the highest level in the ontology, including things like events and situations.

To allow more felicitous expression of facts like these, we have been working on a set of possible extensions to CE syntax and semantics. While it would be possible to introduce some of these extensions directly into the language, we have been trying to develop a more general means of extending the syntax and associating semantics with it to allow developers and advanced users to make additional extensions as needed in the future. One approach we have experimented with is to use simple transformation rules to capture the linguistic structures that share the same semantics with the existing CE sentences [12]. For example, the construction “there is a red car named mycar” could be transformed into “the car mycar is a red thing” by the following rule:

- there is a <name1> <noun2> named <name3>  
==>  
the <noun2> <name3> is a <name1> thing

where items contained in angle brackets are patterns that match certain components in the sentence (for example a <noun> is a word that is contained as a concept in the current conceptual model), and the double arrow expresses a mapping from extended to basic CE:

- extended CE ==> basic CE.

This allows straightforward extensions without the need to define new semantic interpretation rules. The only “hard” extension we would have to build into CE is the mapping syntax. The obvious drawback of this approach is that the possible extensions are rather limited. Later, we will briefly mention another more flexible approach to defining extensions to the language that we have begun to explore.

### 3 CE-based Capabilities and Applications

An extensible CE is most useful in situations that have the following characteristics:

1. A high degree of human-computer interaction, usually involving specialist users with complex needs in non-trivial environments.
2. A likelihood of rapidly evolving or uncertain tasks, queries or other knowledge-based activities.
3. The need for collaboration, either between different people or teams, and/or across different disciplines.

CE is of less value if there is no human-involvement, little complexity, or very firm and stable requirements, and in such circumstances traditional application develop-

ment processes are a much more straightforward and low risk solution. In cases where there is a high degree of customization, development, uncertain requirements or short lead times, especially in areas where human-led planning, thinking or decision-making are required, then CE (or similar human-friendly information processing environments) could be a very useful capability. Along these lines, we have developed (and are continuing to develop) CE-based capabilities such as “CE Store” to support military coalition applications.

### **3.1 CE Store: a Development Environment for CE-Based Applications**

CE Store is a research-grade runtime implementation of the CE language and ecosystem. It provides a basic CE processing environment that allows for the relatively quick and efficient testing of new concepts in CE and the development of prototype applications. CE Store includes the following high-level capabilities:

1. Basic CE sentence parsing
2. Definition and extension of any concept model (i.e. the possible types of things, including events and situations, and their possible attributes and relationships)
3. Assertion of any CE sentence conforming to a concept model
4. Loading and querying of any existing concept model and associated sets of facts
5. Definition and execution of any CE query including an example “visual query composition” element
6. Definition and execution of any logical inference rule, in the form of a “query with conclusion clauses” that can be used to assert new CE information
7. Definition and execution of “CE agents” which conform to a simple “CE Store” interface

CE Query Builder (CEQB) is a visual query drawing tool embedded in the CE Store environment, which makes use of drag-and-drop and contextual (popup) menus to allow the user to draw, execute and save a CE query or rule. It provides one means to help the user write more accurate rules and queries. The intention of the CEQB is that it is a useful “exploratory” environment in which CE queries can be constructed in a convenient manner. Once the query (or rule) is constructed it can then be saved, executed, etc. The CEQB is a “model aware” component of the environment and is directly integrated into the CE Store APIs. Therefore it will allow one to create queries relating specifically to the information that is currently loaded into the CE Store, indicating the existing concepts but also allowing the user to create new concepts via the conceptualize statement and new rules employing these and other concepts. Note also that, while the query is constructed graphically, the results are displayed in CE.

The purpose of the CE Store is to demonstrate an (almost) “pure” CE-based implementation of an information-processing environment within which human and machine agents can contribute and interact with complex information based on common conceptual models of a domain. In addition, the concept of CE Agent is core to the CE Store approach, allowing domain-specific modules to be constructed and integrated into a CE application. CE Agents may be constructed completely in CE or, if

necessary or more convenient, constructed in Java with a CE interface or API to the rest of the CE Store. Java-based CE Agents are used mainly in two cases: 1) When the functionality required is not expressible in if-then rules - e.g. low level text preprocessing or complex algorithmic or statistical processing like that needed for spatial information processing, or 2) When the writing of rules to express the required behavior is too complex or too tedious. It is more likely that, people with roles related to the infrastructure of the system and with IT backgrounds would develop the Java-based agents, if necessary and other users would simply interact with them in CE.

There is a publically available version of the CE Store, known formally as the “IBM Natural Language Processing Environment”, available for download from the IBM developerWorks site, here:

<http://ibm.co/RDIa53>

### 3.2 Example Applications

CE has been used in the development of a number of prototype systems, including fusion of hard data (from sensors) and soft data (human reports) for situation awareness on the battlefield [13], real-time integration of maps, photos, and messages about events [14], and collaborative planning [15]. The different applications have been based on different versions of CE and have focused on different areas of development of the language. The focus of this paper is not on applications, but the following gives a brief overview of some of the applications that have been developed with CE. See the particular papers for more details.

The data fusion application uses CE to manage the direction, collection, processing, and dissemination of data to support decision making. These require the expression of information needs, the description of asset capabilities, and the conversion of information products generated by each asset into a machine-processable form, consistent with the metadata specified for that asset. This supports the processing and ultimate delivery of data to meet the original information needs [13].

The second application provides for the assignment of objects like buildings and vehicles to locations on a map, the association of photos taken by agents in the field of those objects with icons on the map, and the identification and location of objects extracted from short human generated messages on the map [14].

The “Collaborative Planning Model” is a multi-layer set of conceptual models to enable collaborative planning as a specialized form of general problem solving with support from higher level models for spatial and temporal reasoning [15].

### 3.3 CE-based Fact Extraction

If a system is developed from scratch, then CE can be defined as the language to use in interacting with the system or with other humans through the system. However, in many applications, there is already a great deal of information available in unstructured form (i.e. free text) that has not been written in controlled English. In order to

make this information available, we are currently developing a more extensive and in-depth Fact Extraction system based on CE. Our motivation for using CE to develop this system is the many sub-domains that need to be developed and the rapid rate of change of the conceptual model and language, requiring users at various levels and with various roles to participate in its extension, from system infrastructure builders, to linguists and knowledge engineers, to end-users.

Our method employs natural language processing techniques to parse the language text, recognize the sentence structures, detect properties of the analyzed sentence units, and identify and extract the targeted information items, such as entities, events, relations and facts. CE is used in two different roles in our system: 1) to express the content of the extracted information, using domain specific terminology as specified in the underlying conceptual model, and 2) to describe the linguistic structure of the natural language text being analyzed and to express the processing rules used to get from the natural language text to that expression of the content in CE.

In order to map between the syntax of the sentence and the semantics of the domain, we are currently employing an open source parser, (specifically the Stanford Parser [16]) to provide a basic syntactic parse tree, allowing users to focus on the mapping of this parse tree into the meaning of the sentence, i.e. the specific entities, events, and situations represented in the analyst's conceptual model for the domain. The parse tree produced by the Stanford parser is converted into CE, so that it is amenable to processing by other CE rules.

A lexical model has been constructed in CE to support language processing and the construction of application demonstrations, such as the second application described above. This model is constantly being developed further to provide more complex linguistic concepts, while continuing to support the more basic applications. Recent extensions include:

8. representation of morphologically related sets of words
9. representation of lexical semantics

In the extended lexical-model, we introduce the notion of grammatical form, which includes grammatical information about the word such as the “part of speech” and inflectional features. In the lexical model, grammatical forms are represented in CE, For example, the following is a partial CE representation of a verb:

- conceptualise
  - the grammatical form GF ~ is an inflection of ~ the grammatical form GF1
- conceptualise
  - the grammatical form GF has the value V as ~ person ~ and
  - has the value V1 as ~ tense ~

The concept of grammatical form permits the association of different forms of the same word (for example the singular and plural forms of a noun, or the various forms

of a verb) with the lemma or base form of the word and therefore with each other, as well as linking them with their shared lexical meaning while distinguishing their inflectional meanings.

As information requirements vary from one domain to another, we are taking an ontology-based information extraction approach. The domain conceptual model that provides explicit specifications of concepts within the domain plays a crucial role in our information extraction process. Entities and events are primary types of information to be extracted, with entities typically corresponding to noun phrases and events or situation typically corresponding to verbs (together with their arguments and modifiers). Our system correlates the conceptual representations and lexical/grammatical representations by means of the “expresses” relation between words and phrases on the one hand and concepts in the domain model on the other, e.g.:

- the singular noun NN expresses the entity concept EC<sup>5</sup>

In order to specify the semantics of an entire phrase or sentence, we are developing the idea of a linguistic frame, which specifies syntactic structures with grammatical relations and other necessary (or optional) components. For each linguistic frame, there is a unique semantic interpretation rule. Thus, a linguistic frame is a complete description of a grammatical structure including the syntax and semantics. For example, a basic transitive verb phrase defines a relationship based on a transitive verb (with certain morphological, syntactic and semantic features) followed by an (object) noun phrase of certain type, and has the semantics of the relation concept defined. These are defined in such a way as to preserve the semantics of the phrases they cover and pass the resulting semantics up, thus allowing them to incorporate the principle of compositional semantics [17] while at the same time allowing specific construction-level semantics to be added, if needed.

For example, the following linguistic frame describes a predicate nominative construction with “is”, e.g. “Ms. Davis is a professor”:

- there is a linguistic frame named vp1 that

defines the verb phrase VP and

has the sequence

( the present third singular verb '|is\_VBZ|' , the determiner '|a\_DT|' , and  
the singular noun NN )

as syntax and

has the statement that

---

<sup>5</sup> Note: a more recent model has “word sense” linked to as an additional entity linked between the grammatical form and the domain conceptual model.

( the singular noun NN expresses the entity concept EC )<sup>6</sup>  
as preconditions and

has the statement that  
( the thing X realises the entity concept EC )  
as semantics and

has the thing X as phantom variable.

The “phantom variable” is used to specify an unrealized variable in the definition of the transitive verb that will be matched with the subject at the sentence level, somewhat in the manner of the lambda calculus [17].

This is similar to a feature-based phrase structure approach [18] [19], but there are two differences: (i) the linguistic frames for CE are typically more specifically defined (e.g. the above example including a specific verb); (ii) the linguistic frames in our approach are written in CE. Note that the current CE adopts the “one meaning per CE word”<sup>7</sup> principle and has transparent syntax-semantics mapping, allowing no ambiguity. This will ensure deterministic interpretation for each linguistic frame.

We are currently exploring the use of lexical resources like WordNet [20] and VerbNet [21] to aid both developers and end-users in lexical development, helping them to determine which sense of a word they wish to add and whether it already exists in the model (perhaps under a different name). VerbNet is organized into verb classes based on Levin classes [22], and incorporates both syntactic and semantic information. It encodes detailed information about possible syntactic realization of the argument structure for English constructions such as transitive intransitive-alternations. With its assignment of different roles and types to different arguments (both noun phrases and other phrases), VerbNet can aid the system in the correct parsing of sentences with those verbs. We have developed code to convert these resources into CE and are developing the machinery within CE to use them to perform these functions.

## 4 Conclusion and Future Work

In Summary, given large volumes of structured and unstructured data, information acquisition and knowledge sharing will need common information structures and representations that are unambiguous to support information sharing and interoperability among teams and team members across domain boundaries. This is the primary motivation for our work on CE and CE-based capabilities.

---

<sup>6</sup> Note: as above, a more recent model has “word sense” linked to as an additional entity linked between the grammatical form and the domain conceptual model.

<sup>7</sup> A “word” in CE is actually referred to as a “concept term”, since it is assumed that each such “word” maps uniquely and unambiguously to a single concept.

While the current CE is still basic, it has been used in a number of example applications as discussed above to model and interact with complex real-world environments, with a reasonable number of concepts, relationships, and rules. We are currently focused on three efforts: 1) extending the CE language to make it more natural, 2) improving the fact extraction capabilities of the system, and 3) continue leveraging external linguistic sources like WordNet and VerbNet, both for the basic fact extraction system and to assist end-users with only folk-linguistic knowledge to in extending conceptual models for their domain along with the associated lexical knowledge.

Our major effort currently focused on extending the CE language to make it more natural. For CE extension, we have no intention to include all the grammatical structures of English. Instead, we will continue to focus on basic English phrase and sentence structures, especially those that are structurally unambiguous. We have explored the use of transformational rules to extend CE basic syntax. However, there is probably a more powerful and flexible way of doing that. As discussed above, we are using linguistic frames to define natural language structures. We are planning to investigate the use of the same mechanism to extend CE itself. CE is an example of a natural language and the linguistic frame provides a way of specifying new syntactic structures and their associated semantics. This means we would be using CE to extend CE, which should not be a problem as long as it is ultimately grounded in some very fundamental CE.

Managing ambiguity is fundamentally important for the work of extending CE. As we extend CE, potential ambiguity may arise from either lexical resources or potentially ambiguous syntactic patterns. In English, the majority of words are inherently ambiguous. As mentioned throughout the paper, ITA CE is intended to be highly predictable with no ambiguity allowed. We define the relation between a word and a concept by using a ‘conceptualise’ statement but the definition exercise will need to be based on domain relevance including the data used in the relevant domain and the conceptual domain model so as to ensure the right definition of the lexical meaning for the lexical item in question. Possible ambiguities can also result from syntactic structures that allow multiple interpretations, including the well-known example of prepositional phrase attachment. Extended CE will define the most natural interpretation as the only possible structure, again based on the empirical data.

**ACKNOWLEDGMENT.** This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

## References

1. Fuchs, N. E., Schwertel, U., and Schwitter, R.: Attempto Controlled English. Proceedings of LOPSTR'98 (1998)
2. Clark, P., Harrison, P., Jenkins, T., Thompson, J., and Wojcik, R.: Acquiring and Using World Knowledge Using a Restricted Subset of English. In Proceedings of FLAIRS'05 (2005)
3. Schwitter, R.: Processable English. See <http://web.science.mq.edu.au/~rolfs/peng/> Retrieved August, 18 (2010).
4. Engelbrecht, P., Hart G., and Dolbear, C.: Talking Rabbit: a User Evaluation of Sentence Production. Ordnance Survey. Workshop on Controlled Natural Language (CNL 2009). 8-10 June 2009. Marettimo Island, Italy. Appears in Controlled Natural Language, Volume 5972, of Springer's LNCS/LNAI series (2009)
5. Verbeke, C. A.: Caterpillar Fundamental English, Training and Development Journal, 27, 2, 36-40, Feb 73 (1973)
6. MacDonald, M.L.: Simplified Technical English for All: A Customer-friendly Specification. AeroSpace and Defence Industries Association of Europe (ASD), [http://www.x-pubs.com/resources/2008conf/downloads/4X-Pubs2008\\_Maria\\_McDonald\\_Simplified\\_Technical\\_English\\_For\\_All.pdf](http://www.x-pubs.com/resources/2008conf/downloads/4X-Pubs2008_Maria_McDonald_Simplified_Technical_English_For_All.pdf) (2008)
7. Sowa, J.: Common Logic Controlled English, March 2007, <http://www.jfsowa.com/clce/clce07.htm> (2007)
8. Clark, P., Harrison, P., Murray W. R., Thompson J.: Naturalness vs. Predictability: A Key Debate in Controlled Languages. Workshop on Controlled Natural Language (CNL 2009). 8-10 June 2009. Marettimo Island, Italy. Appears in Controlled Natural Language, Volume 5972, of Springer's LNCS/LNAI series (2009)
9. Mott, D.: Summary of Controlled English, ITACS, <https://www.usukita.org/papers/5658/details.html> (2010)
10. Mott, D.: The representation of logic within semantic web languages, ITACS, url: <https://www.usukita.org/papers/5242/details.html> (2009)
11. Mott, D., Giannanco, C., Braines, D., Dorneich, M., and Patel, D.: Hybrid Rationale and Controlled Natural Language for Shared Understanding. In Proceedings of the Fourth Annual Conference of the International Technology Alliance, London, UK, September (2010)
12. Mott, D and Handler, J.: Layered Controlled Natural Languages, In Proceedings of the Third Annual Conference of the International Technology Alliance, Maryland, USA (2009)
13. Preece, A., Pizzocaro, D., Braines, D., Mott, D., de Melz, G., and Pham, T.: Integrating Hard and Soft Information Sources for D2D Using Controlled Natural Language, April 2013, SPIE Defense, Security, and Sensing (2013)
14. Braines, D., Mott, D., Laws, S.: Controlled English to Facilitate Human/machine Processing. April 2013, SPIE Defense, Security, and Sensing (2013)
15. Dorneich, M. C., Mott, D., Bahrami, A., Allen, J., Patel, J., Giannanco, C.: Lessons Learned from an Evaluation of a Shared Representation to Support Collaborative Planning. In, 7th Knowledge Systems for Coalition Operations, KSCO (2012)
16. Klein D. and Manning, C. D.: Accurate Unlexicalized Parsing. Proceedings of the 41st Meeting of the Association for Computational Linguistics, pp. 423-430 (2003)
17. Cann, R.: Formal Semantics: An Introduction, Cambridge University Press, Feb 6 (1993)
18. Gazdar, G., Klein, E., Pullum G., and Sag, I.: Generalized Phrase Structure Grammar. Harvard University Press, Cambridge, MA (1985)

19. Pollard C. and Sag, I.: Head-Driven Phrase Structure Grammar. University of Chicago Press, Chicago, IL (1994)
20. Fellbaum, C. (ed.). WordNet: An Electronic Lexical Database. Cambridge, MA: MIT Press, Cambridge, MA (1998)
21. Edward, L., Yi, S., and Palmer, M.: Combining Lexical Resources: Mapping Between PropBank and VerbNet. Proceedings of the 7th International Workshop on Computational Semantics. Tilburg, the Netherlands (2007)
22. Levin, B.: English Verb Classes and Alternations: A Preliminary Investigation, University of Chicago Press, Chicago, IL (1993)

# RECON – A Controlled English for Business Rules

Ed Barkmeyer<sup>1</sup> and Fabian Neuhaus<sup>1,2</sup>

<sup>1</sup> National Institute of Standards and Technology, Gaithersburg, MD

<sup>2</sup> Prometheus Computing, Cullowhee, NC

**Abstract.** Capturing business rules in a formal logic representation supports the enterprise in two important ways: it enables the evaluation of logs and audit records for conformance to, or violation of, the rules; and it enables the conforming automation of some enterprise activities. The problem is that formal logic representations of the rules are very difficult for an industry expert to read and even more difficult to write, and translating the natural language of the enterprise to formal logic is an unsolved problem. RECON – Restricted English for Constructing Ontologies – is a subset of English that can be easily read by an industry expert, while having a formal grammar and an unambiguous translation to formal logic. This paper describes the principal features of the RECON language, with examples, and shows the corresponding formal logic constructs that are produced by the RECON tool.

## 1 Introduction

The Restricted English for Constructing Ontologies (RECON) language is a close relative of English that has a well-defined interpretation in a formal logic language. The need for RECON arose in a project concerned with the consistency, completeness, and timeliness of information received from supply-chain business partners via electronic messages. To automate the validation of the information, the facts, rules and definitions of terms must be stated in a form suitable for machine reasoning – a formal logic language. On the other hand, capturing definitions, facts, and rules for an industrial domain requires the contributions of experts in the domain. Formal logic languages are very difficult for an industry expert to understand. So, to facilitate capturing the knowledge of the industry experts, the project developed an intermediate language – a “restricted English” called RECON.

RECON looks like English, but is carefully restricted in grammar, so that every statement and most definitions have an unambiguous equivalent in the formal logic language. The experts in the industry domain may require the assistance of a knowledge engineer to state their intent in the RECON language, but it is most important that they can read the restricted English formulation and verify that it captures their intent. The translation of the stated definitions, facts, and rules is used directly in validating incoming information. For example, the following is a business rule expressed in RECON:

**Example 1** Any shipment that consists of more than 1000 gallons of gasoline must be shipped via some registered tanker.

The formal logic version of this rule, as output from the RECON translator, is used directly by the validator in determining that messages describing shipments are (or are not) consistent with the rule.

Properly, RECON is only the grammar for the language. The vocabulary of the language – the nouns, verbs and adjectives – is defined by the industry experts. The RECON tooling is designed to capture a vocabulary based on English words and to parse sentences in the language that use the terms in that vocabulary. The sentences may be definitions of terms, facts about the domain, or rules. The parsed sentences are then translated into a formal logic text in the IKRIS Knowledge Language (IKL) [5], which is an extension of ISO Common Logic Interchange Format [6] that supports nominalized propositions.

The existing RECON tool is the engine that processes the vocabulary and translates the sentences. It was designed as a plug-in for an authoring tool that does not yet exist. It is currently run via a simple command-line interface program that invokes the RECON tool to process a set of files containing vocabularies, facts and rulesets.

This paper describes RECON’s capabilities, with a particular focus on representing business rules. We will do that by discussing various RECON examples and their translation into IKL. The grammar of the RECON language is formally specified in [1].

## 2 Related work

It is important to distinguish restricted English from natural language processing. The objective of natural language processing is to produce a formal interpretation of text as published. Natural language can be ambiguous, and formal interpretations are not necessarily reliable. The objective of a “controlled” English is to ensure that text written in the language can be converted in every case to a particular formal language by a particular algorithm. So we will here consider only restricted English languages.

The simplest restricted English languages are “template” languages, in which the knowledge engineer defines a set of sentence forms with parameter markers, and defines the interpretation as a pattern for text in the formal language, with slots for parameter substitutions. Sometimes called “domain-specific languages” (DSLs), these are supported by tooling such as XTEXT [15]. Several production rules technologies have languages of this kind.

The more interesting controlled natural languages have grammars and parsers that convert the intent of the statements to formal logic. The target logic language determines what a restricted English tool can export, and thus limits what the restricted English language can usefully express. First-order logic and its extensions are the most expressive, while Horn clauses, description logics and production rules are subsets of first-order logic that allow more efficient implementation.

Rabbit [2, 4], for example, is a controlled English language for the Web Ontology Language (OWL) [14], which is based on description logic. It permits only simple sentences and complex noun phrases that have clear renderings into OWL. Similarly, Common Logic Controlled English [13] is intended to express first-order logic propositions using explicit quantifiers and variables, and although it apparently allows more natural expressions, that part of the published grammar is incomplete. The Controlled English to Logic Translator (CELT) [10] exports formal statements in Knowledge Interchange format [9], a first-order logic language. Although its grammar is not published, CELT tooling accepts a more natural English, and interprets common terms using synonymies and the Suggested Upper Merged Ontology (SUMO) [7].

The Attempto project developed the Attempto Controlled English (ACE) [3] that can be rendered into formal languages for various computational purposes. ACE eliminates aspects of natural English that interfere with unambiguous interpretation, but like natural language parsers, it integrates multiple sentences in a text corpus, and supports back references, using “discourse representation” technology.

The Semantics of Business Vocabulary and Rules (SBVR) is a standard for capturing vocabularies, and introduces a Structured English for facts, definitions and rules that use those vocabularies [8]. The language, however, is not standardized; it is described informally in an annex. The parse relies on terms being marked up in the text. For example, “Mary goes to the store” is written “Mary goes to the store. ” SBVR tools must export an enhanced first-order logic language that includes proposition nominalization and modalities for possibility and obligation/permission.

Unlike natural language tooling, which depends on dictionaries for terms and their possible meanings, ACE, SBVR and RECON require terms to be declared and provide for formal definitions. For multi-word terms, however, ACE requires hyphenated terms and SBVR requires term markup, while RECON requires neither. RECON and SBVR provide for verb usage templates, resulting in n-ary logical relations with formal definitions, while ACE and CELT objectify most verbs as classes of states or events, whose logic model is an event object with a set of binary “role relations”. This is a major difference in the resulting formal logic structures. Neither ACE nor SBVR supports compound noun phrases, which RECON does. SBVR Structured English is the most comprehensive of the above, and anything written in SBVR Structured English can be written in RECON, without special markups. Finally, like ACE and unlike SBVR, the RECON grammar is formally defined in [1]. This document also contains a more detailed comparison of RECON with related work.

### 3 Dictionary and Vocabulary

The most basic linguistic notion in RECON is the word form. Roughly speaking, English word forms are the strings in an English text that are delineated from the rest of the text by whitespace or punctuation; e.g., ‘ACME’, ‘the’, ‘registered’ are

word forms in Example 1. A RECON dictionary entry (a 'word') is a collection of word forms associated with a grammatical category (noun, verb, other) that are treated as representations of the same word. (Verb participles have special significance, not discussed in this paper.) For example, Dictionary 1 contains one word.

### **Dictionary 1** *Dictionary Verb: run runs ran running run*

The same word form can belong to multiple dictionary entries. Note that RECON assigns no semantics to dictionary entries; they are purely syntactic.

A *vocabulary* consists of terminological entries. A terminological entry consists of one or more declarations. A terminological entry always begins with the declaration of a term for the vocabulary item, called the primary term. A term is a sequence of words, each of which will be recognized in any word form. The primary term declaration may be followed by alternative forms and definitions, and perhaps other declarations, that are part of the terminology entry and are associated with the primary term. A formal definition in the RECON language will cause RECON to produce a formal definition (of the corresponding IKL term) in IKL. (But we will not exemplify that.)

*Name: ACME Inc*  
*Name: Bride of Neptune*  
*Type Noun: supplier*  
*Type Noun: party*  
*Type Noun: shipment*  
*Type Noun: vessel*  
**Vocabulary 1**  
*Mass Noun: gasoline*  
*Adjective: (thing) is registered*  
*Property: (party) is the customer () of (shipment)*  
*Verb: (party) ships (shipment)*  
    *Alternative: (shipment) is shipped by (party)*  
*Verb: (shipment) is shipped via (vessel)*  
*Property: (quantity) is the volume () of (thing)*  
*Unit: gallon: volume*

Vocabulary 1 contains declarations of six different kinds of terms: names, type nouns, a mass noun, an adjective, two properties, and verbs. Adjectives and properties are treated as verbs that also have other syntactic usages. Note that adjectives, properties, and verbs take arguments called 'roles'. In the primary entry, the position and the type of a role is indicated by a noun term enclosed in parentheses. For example, the first argument of "ships" must be a party, the second must be a shipment. In the declaration of an alternative form (e.g., 'is shipped by') the same roles can appear in different positions. The two forms are different syntactic forms for the same verb concept with the same roles.

## 4 Starting simple

In first-order logic, an atomic sentence consists of a predicate and a number of arguments. The analog in RECON is a sentence that consists of a verb phrase that is a verb form from the vocabulary, where the arguments are replaced by names.

**Example 2** ACME Inc is registered.  
ACME Inc ships SH12345.

**Output 2** (thing . is\_registered ACME\_Inc)  
(party . ships . shipment ACME\_Inc SH12345)

These examples illustrate how simple RECON sentences are translated into simple IKL formulas. Note that RECON will recognize an alternative form and convert it to the primary form. For example, sentence Example 3 will be translated into the same formula as the second sentence in Example 2.

**Example 3** SH12345 is shipped by ACME Inc.

In the following sections we will consider several ways in which these simple sentences can become more complex. First, the roles in the verb phrase can be filled by complex noun phrases. Second, simple sentences can be combined to form more complex sentences.

## 5 Type nouns

The most commonly used noun phrases consist of a quantifier (e.g., ‘a’, ‘the’, ‘any’, ‘every’, ‘some’) and a type noun, as in Example 4. As Output 4 illustrates, the result of the translation into first-order logic involves one (or more) quantifiers.

**Example 4** ACME Inc ships a shipment.

**Output 4** (exists (?shipment1)  
(and  
(shipment ?shipment1)  
(party . ships . shipment ACME\_Inc ?shipment1)))

In English quantified noun-phrases can lead to ambiguity. For example, in English both sentences in Example 5 have two theoretical readings: (1) for each supplier there is a shipment that is shipped (but different suppliers might ship different shipments), and (2) there is one shipment that is shipped by every supplier.

**Example 5** Every supplier ships some shipment.  
Some shipment is shipped by every supplier.

A human reader uses contextual knowledge about the business practices of suppliers to disambiguate the sentences and to decide that (1) is likely the intended meaning. Since RECON has no such contextual knowledge, however, and because RECON has to provide an unambiguous parse, it must have a rule for deciding how each combination of quantifiers is interpreted. In this case, the rule is: The quantification for the subject of the main verb encloses any quantification for other verb roles. The rules for some constructs are more complex. As Output 5 shows, the two sentences in Example 5 are translated into different first-order logic formulas. The disadvantage of this is that the user must be careful, because the order of the quantified type nouns in RECON sentences influences the translation. The advantage is that the experienced author can express either intent, and know how each will be translated.

```
(forall (?supplier1)
 (if
 (supplier ?supplier1)
 (exists (?shipment2)
 (and
 (shipment ?shipment2)
 (party.ships.shipment ?supplier1 ?shipment2)
))))
```

#### **Output 5**

```
(exists (?shipment1)
 (and
 (shipment ?shipment1)
 (forall (?supplier2)
 (if
 (supplier ?supplier2)
 (party.ships.shipment ?supplier2 ?shipment1)
))))
```

## 6 Adjectives

As we have seen in Example 2, adjectives can be used within verb phrases. But, of course, their primary use in English is as noun modifiers. As illustrated by Example 6 the adjective (in this case ‘registered’) translates into a conjunct in Output 6.

**Example 6** Bride of Neptune is a registered tanker.

```
(exists (?tanker1)
 (and
 (and
 (tanker ?tanker1)
 (thing.is-registered ?tanker1))
 (= Bride_of_Neptune ?tanker1))))
```

## 7 Qualifiers

Another way to modify noun phrases is with quantifiers, that is subordinate sentences starting with ‘that’, ‘which’, ‘who’, or ‘whom’.

**Example 7** Any shipment that is shipped via Bride of Neptune is registered.

```
(forall (?shipment1)
 (if
 (and
 Output 7 (shipment ?shipment1)
 (shipment.is_shipped_via.vessel
 ?shipment1 Bride_of_Neptune))
 (thing.is_registered ?shipment1)))
```

RECON allows for arbitrarily complex and nested qualifiers. However, for the same reasons that style guides discourage the use of overly complex qualifiers in English, we recommend some restraint on their use in RECON.

## 8 Properties

In Vocabulary 1 we declared two properties. Properties can be used in verb phrases, but they also give rise to possessive noun phrase structures, as illustrated in Example 8. Note that in Output 8 the second sentence is translated into a universally quantified formula. This is too weak, it would be more appropriate to use an  $\iota$ -operator as defined in Russell’s theory of description [12].

**Example 8** ACME Inc is the customer of SH12345.  
The customer of SH12345 ships SH12345.

```
(party.is_the_customer_of.shipment ACME_Inc SH12345)
(forall (?thing1)
 (if
 (and
 Output 8 (thing ?thing1)
 (thing.is_the_customer_of.shipment
 ?thing1 SH12345))
 (party.ships.shipment ?thing1 SH12345)))
```

## 9 Mass-nouns

The semantics of mass expressions is a well-known challenge in philosophy and linguistics [11]. RECON does provide some limited support for expressions involving mass nouns. Occurrences of mass nouns without any units of measurement (e.g., ‘some gasoline’) are translated into quantifications over discrete portions of stuff (e.g., portions of gasoline). Hence, mass nouns are translated by RECON into predicates that apply to countable entities (the portions). To illustrate the point, Example 9 is interpreted by RECON into Output 9: there is [*a portion of*] gasoline that the shipment SH12345 consists of.

**Example 9** SH12345 consists of some gasoline.

**Output 9**

```
(exists (?gasoline1)
 (and
 (gasoline ?gasoline1)
 (shipment.consists_of.thing SH12345 ?gasoline1))))
```

## 10 Measurements, collections, and quantities

RECON is able to handle expressions that involve quantities of things expressed in measurement units. It contains a rather complex model of units of measure, which supports derived SI units and their definitions; e.g.,  $1N = 1kg\frac{m}{s^2}$ . The details are beyond the scope of this paper; however, we illustrate the approach with two examples. Example 10 is similar to Example 9, the difference is that the amount of gasoline is specified. The difference between Output 9 and Output 10 is an additional conjunct, which uses the quantity value ‘(Qvalue 1000 “gallon”)’ and the property *volume of*. It is used because the unit gallon was declared to measure volume. Quantity values are functional expressions consisting of the special function term ‘Qvalue’, a number, and a name that denotes a unit of measurement.

**Example 10** SH12345 consists of 1000 gallons of gasoline.

**Output 10**

```
(exists (?gasoline1)
 (and
 (and
 (gasoline ?gasoline1)
 (quantity.is_the_volume_of.thing
 (Qvalue 1000 "gallon") ?gasoline1))
 (shipment.consists_of.thing SH12345 ?gasoline1))))
```

**Example 11** SH12346 consists of 1000 widgets.

RECON uses a similar method to translate sentences that involve numerically quantified type nouns (as in Example 11) into quantifications over collections and their cardinality.

## 11 Connectives

RECON supports the use of compound sentences (Example 12) and compound noun phrases (Example 13).

**Example 12** ACME Inc is registered or ACME Inc is not registered.

**Output 12**

```
(or
 (thing.is_registered ACME_Inc)
 (not
 (thing.is_registered ACME_Inc))))
```

**Example 13** ACME Inc ships both SH12345 and SH12346.

(and  
**Output 13** (party . ships . shipment ACME\_Inc SH12345)  
                  (party . ships . shipment ACME\_Inc SH12346))

Note that Output 13 consists of two conjuncts; thus, there is no connection between the two shipments. There is an alternative interpretation of the English sentence, according to which ACME ships a collection that consists of the two shipments. This can be expressed in the RECON grammar by adding the key word ‘together’ (see Example 14).

**Example 14** ACME Inc ships both SH12345 and SH12346 together.

## 12 Rules

All of the examples we have considered so far are similar to Example 15 below in the following sense: they are statements about how the world is. In contrast, Example 16 is about how the world should be; it expresses a requirement. The existence of an unregistered shipment would make Example 15 false; but it would have no affect on the truth of Example 16. It would, however, make the state of the world unacceptable.

**Example 15** Every shipment is registered.

**Example 16** Every shipment must be registered.

Since business rules can be statements about actual characteristics of the world as well as normative characteristics, RECON has been designed to capture the difference. As a comparison of Output 15 and Output 16 illustrates, the difference is represented in IKL with the help of a modal predicate (‘obligation’) that operates on nominalized sentences. (The ‘that’-operator in Output 16 is a feature of IKL that allows nominalization of arbitrary sentences.)

(forall (?shipment1)  
      (if  
            (shipment ?shipment1)  
            (thing . is-registered ?shipment1)))  
  
(obligation (that  
              (forall (?shipment1)  
                (if  
                  (shipment ?shipment1)  
                  (thing . is-registered ?shipment1))))  
**Output 16**

## 13 A real business example

We have illustrated some of the features of RECON by discussing small RECON sentences and the formal logic translations that are produced by the RECON tool. Of course, any real business rule will usually combine several features; e.g., in the introduction Example 1 involves type nouns, an adjective, a mass noun, and a unit of measurement.

**Example 1** Any shipment that consists of more than 1000 gallons of gasoline must be shipped via some registered tanker.

As RECON's translation of Example 1 below shows, the resulting IKL formulas are often quite verbose. Some of that could be simplified by an expert knowledge engineer. Ultimately, however, the complexity of the formula below is a reflection of the complexity of the state of affairs expressed in Example 1. Hence, capturing its content in a logic language will be a challenge for any person who is not very familiar with such languages. And, of course, the result is unreadable for anybody without these skills. Therefore, the example demonstrates the need for a tool, like RECON, that enables business rules to be captured formally in a way that is more accessible to the business experts.

### Output for Example 1

```
(obligation (that
 (forall (?shipment1)
 (if
 (and
 (shipment ?shipment1)
 (exists (?gasoline2)
 (and
 (and
 (gasoline ?gasoline2)
 (forall (?quantity3)
 (if
 (and
 (quantity ?quantity3)
 (quantity . is_the_volume_of . thing
 ?quantity3 ?gasoline2)))
 (quantity . is_less_than . quantity
 (Qvalue 1000 "gallon") ?quantity3))))
 (shipment . consists_of . thing ?shipment1 ?gasoline2)
)))
 (exists (?tanker4)
 (and
 (and
 (tanker ?tanker4)
 (thing . is_registered ?tanker4)))
 (shipment . is_shipped_via . thing ?shipment1 ?tanker4)
)))))
))))
```

## 14 The RECON tool

At this time, the grammar of RECON has been finalized [1], and version 1.0 of the RECON engine is available on Sourceforge.<sup>3</sup> The engine supports most of the features of the grammar. Internally, it consists of a *dictionary manager* for words and word forms, a *vocabulary manager* for term declarations, a *parser*, and a *logic generator*. The parser converts definitions, facts and rules into a syntactic parse graph. Because RECON supports multiword terms, and the same word can begin or appear in multiple terms, the parser first produces a lattice of all possible interpretations of the input string as a sequence of terms and keywords. The parser then tries to parse the first such sequence, and if it fails then the next one, and so on, until it finds a successful parse or discards the last alternative. Heuristics are used to choose the first and next sequence in each case. The actual parsing algorithm is a recursive descent algorithm, based on the formal RECON grammar. Logic generation begins with a *rewrite step* that revises the parse graph for compound phrases and quantities, converts adjectives and properties to qualified nouns (using their verb forms), and resolves back references. The *interpret step* then produces a formal logic structure from the revised parse graph, converting type nouns to quantified variables and properly placing and interpreting quantifiers and modalities. The resulting logic structure is exported in IKL form. The architecture, data structures, and algorithms are described in detail in a forthcoming publication.

## 15 Conclusions and Future Work

We are currently experimenting with the use of RECON for capturing engineering requirements and static rules written into information exchange standards. One area of future work on the tool is the support of collections (mentioned in the discussion of Examples 11 and 14). While we have successfully demonstrated feasibility of using the RECON language to author business rules, we have not built the infrastructure to make it a user-friendly product. For that, one would need to develop a front-end for dictionary and vocabulary management, and an authoring tool that supports the user in writing RECON.

## References

- [1] Edward Barkmeyer and Andreas Mattas. *A Restricted English for Constructing Ontologies (RECON)*. NISTIR 7868. National Institute of Standards and Technology, 2012.
- [2] Ronald Denaux et al. “Rabbit to OWL: ontology authoring with a CNL-based tool”. In: *Controlled Natural Language*. Springer, 2010, pp. 246–264.
- [3] Norbert Fuchs, Uta Schwertel, and Rolf Schwitter. *Attempto Controlled English (ACE) Language Manual Version 3.0*. University of Zurich, 1999.

---

<sup>3</sup> <http://sourceforge.net/projects/nistreconst/>

- [4] Glen Hart, Catherine Dolbear, and John Goodwin. “Lege Feliciter: Using structured English to represent a topographic hydrology ontology”. In: OWLED. 2007.
- [5] Patrick Hayes and Chris Menzel. *IKL specification document*. 2006. URL: <http://www.ihmc.us/users/phayes/ikl/spec/spec.html>.
- [6] ISO/IEC 24707-2007. Information technology Common Logic (CL): a framework for a family of logic-based languages. 2007.
- [7] Ian Niles and Adam Pease. “Towards a standard upper ontology”. In: *Proceedings of the international conference on Formal Ontology in Information Systems-Volume 2001*. ACM. 2001, pp. 2–9.
- [8] Object Management Group (OMG). *Semantics of Business Vocabulary and Rules v1.0*. Tech. rep. 2010. URL: <http://www.omg.org/spec/SBVR/1.0/>.
- [9] Adam Pease. *Standard Upper Ontology Knowledge Interchange Format*. 2009. URL: <http://sigmakee.cvs.sourceforge.net/viewvc/sigmakee/sigma/suo-kif.pdf>.
- [10] Adam Pease and William Murray. “An English to Logic Translator for ontology-based knowledge representation languages”. In: *Natural Language Processing and Knowledge Engineering, 2003. Proceedings. 2003 International Conference on*. IEEE. 2003, pp. 777–783.
- [11] Francis Jeffry Pelletier. “Mass Terms: A philosophical Introduction”. In: *Kinds, Things, and Stuff: Mass Terms and Generics*. Oxford University Press, USA, 2009.
- [12] Bertrand Russell. “On denoting”. In: *Mind* 14.56 (1905), pp. 479–493.
- [13] John Sowa. *Common Logic Controlled English*. 2004. URL: <http://www.jfsowa.com/clce/specs.htm>.
- [14] World Wide Web Consortium (W3C). *OWL Web Ontology Language Reference*. 2004. URL: <http://www.w3.org/TR/2004/REC-owl-ref-20040210/>.
- [15] *Xtext 2.3 Documentation*. URL: <http://www.eclipse.org/Xtext/documentation/2.3.0/Documentation.pdf>.

# A Study on Translating Regulatory Rules from Natural Language to Defeasible Logic

Adam Wyner<sup>1</sup> and Guido Governatori<sup>2</sup>

<sup>1</sup> Department of Computing Science, University of Aberdeen, Aberdeen, United Kingdom

azwyner@abdn.ac.uk

<sup>2</sup> NICTA, Brisbane, Australia

guido.governatori@nicta.com.au

**Abstract.** Legally binding regulations are expressed in natural language. Yet, we cannot formally or automatically reason with regulations in that form. Defeasible Logic has been used to formally represent the semantic interpretation of regulations; such representations may provide the abstract specification for a machine-readable and processable representation as in LegalRuleML. However, manual translation is prohibitively costly in terms of time, labour, and knowledge. The paper discusses work in progress using the state-of-the-art in automatic translation of a sample of regulatory clauses to a machine readable formal representation and a comparison to correlated Defeasible Logic representations. It outlines some key problems and proposes tasks to address the problems.

## 1 Introduction

Legal regulations are expressed in natural language.<sup>3</sup> To make them automatically processable for reasoning or information extraction, they must be represented in a machine-readable form. There are several approaches to making regulations machine-readable, e.g. linked documents and annotated documents. We focus on the translation of statements in regulations into formal semantic representations that could then be provided to automated deduction engines, which can then be used to check for consistency and redundancy, draw inferences given ground facts, and provide users with meaningful explanations following a consultation, among other processing tasks. The use cases for such translations are very widespread: extracting and formalising relevant rules from regulations to form *rule books* for particular industries; checking for compliance to regulations; serving expert system web-front ends to users, and others.

The language of regulations seems particularly problematic to process. In [1], a range of issues were identified such as the sentence length, clausal embedding, and list structures, which contributed to long parse times or failures to parse. Beyond parsing issues, we want to translate the expressions in regulations into a formal semantic representation to support the sorts of reasoning tasks and use cases mentioned above. Efforts along these lines appear in early work in artificial intelligence and law [2], though without natural language processing (NLP). Some commercial products are available that support aspects of this process and serve the resultant expert systems to users on the web Oracle Policy Management. However, the source material is heavily preprocessed into a controlled language with limited expressivity (on controlled languages, see [3]). An

---

<sup>3</sup> Copyright ©2013 for the individual papers by the papers' authors. Copying permitted only for private and academic purposes. This volume is published and copyrighted by its editors.

open-source, implemented, controlled-language, *Attempto Controlled English* (ACE), has been applied to clinical practice guidelines [4] and to policy-making statements [5] with some, but limited, success. Pilot studies of parsing and semantic representation of regulations with broad coverage, open source tools, C&C/Boxer [6], have been carried out [7]. On the side of logical representations of regulations, there have been efforts to formalise portions of regulation using Defeasible Logic [8]. Machine-readable representations for legal rules, LegalRuleML, have been developed [9].

The studies with ACE and C&C/Boxer highlight two limitations: the output parse and semantic representation given by the tools must be manually checked to accurately correlate to the intended semantic interpretation of the input expression; relatedly, the outputs have not been associated with logical or machine-readable representations that could serve as requirements for the semantic representation. On the other hand, studies using Defeasible Logic and LegalRuleML do not systematically relate to natural language or the issues of acquiring the formal representations from the source material that is represented in natural language. There remains, then, a significant *gap* between natural language source material and formal, machine-processable representations.

In this paper, we discuss a pilot study in which we use C&C/Boxer to translate regulatory statements to semantic representations and then compare the output representations against logical representations in *Defeasible Logic* (DL) that have been *manually* created. By doing so, we gain a better idea of what each form of representation contains, what is gained or lost, how to scope and evaluate such work, the overall process in the analysis, and what next steps are required in order to improve automatic processing of regulatory text.

In Section 2, we provide information about our method, briefly covering the corpora, C&C/Boxer, and DL. A sample of the output from C&C/Boxer applied to the corpora are reported in Section 3. The DL representation of the sentences is presented in Section 4. In Section 5, we discuss the C&C/Boxer and DL representations in comparison as well as future work.

## 2 Materials and Method

In this section, we present the materials and the method we apply to the materials (e.g. C&C/Boxer and DL).

### 2.1 Materials

We examine a selection of Section 8.2 of Australia's *Telecommunications Consumer Protections Code* (2012) on complaint management. Broadly speaking, we take a *piece-meal* approach to the overall problem of processing the text, *filtering* and *preprocessing* the original material to some degree to make it *amenable to automatic processing*, yet leaving most of the relevant structure intact. Each of the preprocessing editorial moves is recorded, justified, and systematically applied; however, to economise on space, we suppress discussion of the edits here. The orginal material contained 173 words, and given the structure of the document, an unclear number of sentences. As the original data has formatting conventions that are not relevant at this point for the semantic content, we have reformatted the data, which we refer to as the *Source Data*. An additional layer of filtering is applied to the *Source Data*, which contains a range of complications which are not relevant to our current exercise such as lists, subordinate clauses, and references. We have manually preprocessed the data, resulting in *Modified Source Data* of 125 words in five sentences:

### *Modified Source Data*

8.2.1.a.xii. Suppliers must advise consumers in everyday language of the resolution of their complaint as soon as practicable after the supplier completes its investigation of the complaint.

8.2.1.a.xiii. A. Suppliers must complete all necessary actions to deliver the resolution offered within 10 working days of the consumer's acceptance of that resolution unless otherwise agreed with the consumer.

8.2.1.a.xiii. B. Suppliers must complete all necessary actions to deliver the resolution offered within 10 working days of the consumer's acceptance of that resolution unless the actions are contingent on actions by the consumer that have not been completed.

8.2.1.a.xi. Suppliers must provide a means for the consumer to monitor the complaint's progress.

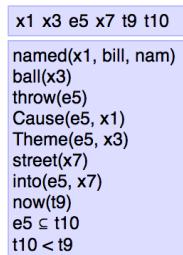
8.2.1.a.xiv. Suppliers must only close a complaint with the consent of the consumer or if clause c below has been complied with.

While this is a small corpus, it still allows for instructive semantic representations as well as challenges.

## 2.2 Method

**C&C/Boxer** C&C/Boxer automatically parses the sentences of the Modified Source Data and gives an associated semantic representation.<sup>4</sup> C&C/Boxer consists of a fast, robust *combinatory categorial grammar* (CCG) parser and *Boxer* [6], a tool that provides semantic representations in Discourse Representation Structures (DRSs) of Discourse Representation Theory (DRT) [10] for discourses, including pronominal anaphora and discourse relations. DRSs have equivalent First-order Logic statements in representations that are suitable for FOL theorem provers, e.g. *vampire*.

To economise our presentation, we omit parses and only consider DRSs. We provide a simple illustration of the DRS output for *Bill threw the ball into the street* in Figure 1. In Figure 1, there is a box notation, where boxes represent a knowledge base or



**Fig. 1.** DRS of *Bill threw the ball into the street*.

Discourse Unit of FOL expressions: a top sub-box represents discourse referents and a lower sub-box the FOL predictions. In the example, we have six discourse referents and ten predictions (including set and order relations). We discuss some of these. In

<sup>4</sup> <http://urd.let.rug.nl/basile/gsb/webdemo>

the example, a *named* entity relation is introduced between a variable *x1*, the string *bill*, and the type *nam*. There is an entity which is a ball, another which is a street. Given the neo-Davidsonian, event-theoretic representation [11], we have an event *e5*, which is a throwing event, and thematic roles, one for the *Cause* of the event and another for the *Theme*. Bill is associated with the cause of the throwing and the ball with the theme (the object) that is thrown. Finally, there is temporal information. While there may be some disputes about the semantic representation (e.g. about thematic roles or the interpretation of the preposition), by and large we find this an acceptable semantic representation.

Applying C&C/Boxer to longer, more complex sentences such as in our corpus results in correlatively more complex derivations and semantic representations. Such complex sentences and discourses must be carefully checked that the parse is correct and, more importantly, that the semantic output corresponds to semantic intuitions for an interpretation of the meanings of the sentences (assuming some way to determine these). We illustrate this further later.

**Defeasible Logic and Deontic Logic** In this section, we give a brief overview of *Defeasible Logic* (DL) [12], which we use to represent rules that are defeasible, and *Deontic Logic*, which represents concepts of *obligation*, *prohibition*, and *permission*.

In the legal domain, rules are well-known to be *non-monotonic*, that is, they admit of exceptions where the rule does not apply or where new information blocks the inference from the rule. DL takes an approach to non-monotonicity that is easy to implement and has been used in various application domains, e.g. regulations, business rules, and contracts [12]. In DL, there are five key features:

- facts - indisputable statements, e.g. *Bill is happy* is *happy(bill)*;
- strict rules - material implication in classical logic, e.g. *Emus are birds* is *r'*: *emu(X) → bird(X)*;
- defeasible rules - rules from which we draw inferences, unless the rule is defeated by superior, contrary evidence, e.g. *Birds typically fly* is *r''*: *bird(X) ⇒ fly(X)*;
- defeaters - rules that prevent conclusion of a defeasible rule from holding. They produce contrary evidence, e.g. *If an animal is heavy then it might not be able to fly* is *heavy(X) ↼ ¬fly(X)*, which only prevents the conclusion *fly(X)* where *heavy(X)*;
- a superiority relation among rules - the relation allows us to draw a “winning” conclusion from rules with opposition conclusions, e.g. where *r'''*: *brokenWing(X) ⇒ ¬fly(X)* and *r''' > r''*, the bird with a broken wing does not fly.

A defeasible theory is a program or knowledge base with these features.

In addition to defeasibility, legal reasoning engages the *deontic concepts*, that is, the concepts bearing on *obligation* (**O**), *prohibition* (**PR**), and *permission* (**PER**) along with related concepts of *violation*, where a violation obtains if what is obligated has not been fulfilled or if what is prohibited has come to pass. There area range of subsorts of obligations (see [13] for the subsorts and definitions), where **OM** is relevant to our example:

Maintenance obligation (**OM**) - obligations that, once introduced, require that a state be maintained for a given period of time, e.g. *After opening a bank account, customers must keep a positive balance until bank charges are taken out*.

### 3 Semantic Representation

In section 2, we presented the corpora and analysis method using C&C/Boxer and Defeasible Logic with deontic operators. C&C/Boxer was applied to the five sentences in our *Modified Source Data*, and every sentence was parsed and given a semantic representation. Essential for our purposes is to consider the semantic representation. In this paper, we only have space to discuss one of the examples.

In Figure 2, we have the representation for statement 8.2.1.a.xi, containing one main DRS, 10 entities, 16 predication, and one subordinated DRS. The main clause *Suppliers must provide a means for the consumer* is paraphrased: the modal *must*, given as  $\square$ , has wide scope over the whole representation; the agent of the event of providing is the supplier and the means is the theme; the means are in the *for* relation with the consumer; the time of the event is in the future with respect to *now*. The portion representing the subordinate clause *to monitor the complaint's progress* is paraphrased: a proposition *p2* represents a monitoring event with the supplier as agent, with a progress entity in the *of* relation to a complain, where the progress entity is the location of the monitoring event.

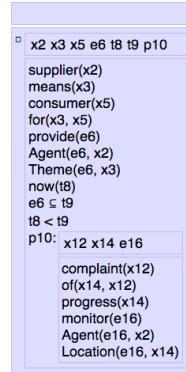
There are several issues to note about the semantic representation. The main clause has an acceptable representation. Semantic operators, e.g. *must* and predication relations, e.g. *for*, are semantically underspecified. Other predication have some intuitive sense, e.g. *Agent*. For the law, some *bearer* of the obligation is required, even if this is universal or generic; in the semantic representation above, there is no such indication of bearer. An important point is that the *generic*, law-like meaning of the sentence, signalled by the plural subject noun in combination with present tense, is not represented. Substantive problems arise with the subordinate clause: *progress* is taken as a location of the event of monitoring rather than a theme, which arises given the lexical specification of the verb *monitor*; the agent of the monitoring is the supplier, rather than the consumer. The first problem relates to the lexical specification of verbs, which are often polysemous. The second problem relates to what is known as *control* such as appears in the difference between the inferred subjects of *leaving* found in *Bill promised Jill to leave* and *Bill persuaded Jill to leave*; there are classes of verbs that behave one way or the other; in the example, *provide* is like *persuade*, not like *promise*. These issues may be resolved through better implementations of thematic role structure and control.

This is an example of the sort of output and analysis available for each of the sentences in our corpus. However, it is difficult to generalise about the outputs or the issues of the semantic representations, as each sentence has particularities that bear further discussion. In the next section, we discuss the related DL representation.

### 4 Representation in Defeasible Logic

In 2.1, we indicated the *Source Data*, which was used to manually translate into 10 DL rules and one rule ordering. However, as space is limited, we only present the DL representation associated with our C&C/Boxer output and mention aspects of the others.

- Sentence: *Suppliers must provide a means for the consumer to monitor the complaint's progress.*
- DL 8.2.1.a.xi:  $\text{complaint}(X), \text{complaint\_acknowledgment}(X) \Rightarrow [\text{OM}]\text{customer\_monitor\_progress}(X)$



**Fig. 2.** 8.2.1.xi: Suppliers must provide a means for the consumer to monitor the complaint’s progress.

The method of translation is entirely manual and intuitive. Between the sentence and DL, we see a range of differences: in the DL representation, the *supplier* is missing; complex predicates are introduced into the DL representation that are presuppositions, e.g. *the compliant is acknowledged*; what is linguistically complex is rendered as a DL predicate, e.g. *customer\_monitor\_progress*; while the deontic operator appears in both the sentence and DL, it appears as complex operator, *maintenance obligation*, which is not clearly associated with the linguistic source (which arises from the generic meaning of the plural subject with present tense); tense is not represented; the bearer of the obligation, e.g. *suppliers*, is not explicit.

## 5 Discussion

In this section, we discuss observations about the two approaches, how they relate, draw out some general points, and end with future work.

C&C/Boxer automatically provides a parse and a correlated semantic representation for each of our sentences; however, there may be some issues with the accuracy and completeness of the semantic representation. This is, in many respects, an issue to be addressed by refinements to C&C/Boxer itself. The semantic representations are highly articulated, identifying all the individuals, events, and relations, whether found in explicit linguistic forms (e.g. *noun objects*) or implicit (e.g. *thematic roles*). Temporal relations are represented. However, as noted, the generic interpretation is not represented, nor is the bearer of the obligation indicated.

In contrast, DL is a manual translation that represents the meaning of the source clause at a high level of syntactic and semantic abstraction in several respects and in contrast to the C&C/Boxer representations. In DL, complex combinations of words that form a phrase are represented as complex predicates; complex operators such as **OM** are not composed from their parts; temporal order is lost (or subsumed under the interpretation of the defeasible conditional); fine-grained elements of the source material are omitted, e.g. *means*; different participants and their roles in the actions are either omitted or incorporated into a predicate, e.g. suppliers are omitted and the customer

appears in *customer\_monitor\_progress*. The disadvantage of such complex predicates is that syntactic structure and semantic compositionality are largely obscured.

Asides from issues about granularity, the two most significant differences between the C&C/Boxer and DL representations are the representation of defeasibility and the scope of the modal. In the C&C/Boxer examples, no conditional representations arise without explicit, linguistic conditionals (or related operators) in the sentences. Moreover, C&C/Boxer provided only specific rather than generic interpretations, which could be taken to represent defeasibility. In combination, we can say that C&C/Boxer outputs do not represent defeasible rules as in DL. Yet, in natural language semantics, non-monotonicity is usually treated quantificationally, whether with adverbs of quantification [14], as generalised quantifiers [15], or in terms of genericity [16]. The first question is, then, what is the most useful or appropriate representation of defeasibility where we are concerned with the automatic translation from natural language into a formal representation? A second related question is to what extent can a tool such as C&C/Boxer accommodate the chosen representation or, turning it around, to what extent ought DL be revised to accommodate natural language semantics of non-monotonicity? Turning to the modal, the scope with respect to a conditional is a complex, largely unresolved matter in natural language semantics [17].

Some of the differences outline above may be taken informatively, in the sense that they indicate how each approach might incorporate or adapt to useful components of the other. There may be ways to bridge the differences; for example, complex predicates can be systematically related to component parts, and quantificational representations of non-monotonic operators could be translated into correlated statements of defeasible logic. But, bridging the differences requires first identifying what those are and whether to bridge them.

We have not discussed evaluation. In statistical or machine-learning analyses, results are usually provided in terms of precision and recall measures, where the performance of a proposed algorithm is measured against a *gold standard* corpus. However, in the absence of such gold standards, we cannot provide such measures; and the creation of such corpora rest on the *specification of what the corpora ought to encode*, which in our view, remains unclear in the research community. Rather, the results reported here bear on: (1) the extent to which existing technologies produce more or less intuitively accurate output; and (2) specific observations about the outputs in comparison; and (3), setting an agenda for future research. Nonetheless, for future work, some explicit measures for evaluation of each approach must be provided. This is tied to the issue of requirements; while initially it seemed that DL representations could be used as abstract specifications to which C&C/Boxer should fulfill, this is not clear. Indeed, this study only serves to highlight that the two approaches have rather different means and objectives, even if somewhat related. However, these topics must be for future work.

## References

1. Wyner, A., Peters, W.: On rule extraction from regulations. In Atkinson, K., ed.: Proceedings of the 24th International Conference on Legal Knowledge and Information Systems (JURIX 2011), Vienna, IOS Press (2011) 113–122

2. Sergot, M., Sadri, F., Kowalski, R., Kriwaczek, F., Hammond, P., Cory, T.: The British Nationality Act as a logic program. *Communications of the ACM* **29**(5) (1986) 370–386
3. Wyner, A., Angelov, K., Barzdins, G., Damljanovic, D., Davis, B., Fuchs, N., Hoefer, S., Jones, K., Kaljurand, K., Kuhn, T., Luts, M., Pool, J., Rosner, M., Schwitter, R., Sowa, J.: On controlled natural languages: properties and prospects. In: Proceedings of the 2009 conference on Controlled natural language. CNL'09, Berlin, Heidelberg, Springer-Verlag (2010) 281–289
4. Shiffman, R.N., Michel, G., Krauthammer, M., Fuchs, N.E., Kaljurand, K., Kuhn, T.: Writing clinical practice guidelines in controlled natural language. In: Proceedings of the 2009 conference on Controlled natural language. CNL'09, Berlin, Heidelberg, Springer-Verlag (2010) 265–280
5. Wyner, A., van Engers, T., Bahreini, K.: From policy-making statements to first-order logic. In: Proceedings of International Conference on Electronic Government and the Information Systems Perspective EGOVIS-2010. (2010) 47–61
6. Bos, J.: Wide-coverage semantic analysis with boxer. In Bos, J., Delmonte, R., eds.: Semantics in Text Processing. STEP 2008 Conference Proceedings. Research in Computational Semantics, College Publications (2008) 277–286
7. Wyner, A., Bos, J., Basile, V., Quaresma, P.: An empirical approach to the semantic representation of law. In: Proceedings of 25th International Conference on Legal Knowledge and Information Systems (JURIX 2012), Amsterdam, IOS Press (2012) 177–180
8. Antoniou, G., Billington, D., Governatori, G., Maher, M.J.: On the modeling and analysis of regulations. In: Proceedings of the Australian Conference Information Systems. (1999) 20–29
9. Athan, T., Boley, H., Governatori, G., Palmirani, M., Paschke, A., Wyner, A.: OASIS Legal-RuleML. In: Proceedings of the 14th International Conference on Artificial Intelligence and Law (ICAIL 2013), Rome, ACM (2013) xx–xx To appear.
10. Kamp, H., Reyle, U.: From Discourse to Logic: Introduction to Model-theoretic Semantics of Natural Language: Formal Logic and Discourse Representation Theory. Springer (1993)
11. Parsons, T.: Events in the Semantics of English: a Study in Subatomic Semantics. MIT Press (1990)
12. Antoniou, G., Billington, D., Governatori, G., Maher, M.J.: Representation results for defeasible logic. *ACM Trans. Comput. Log.* **2**(2) (2001) 255–287
13. Governatori, G., Rotolo, A.: Norm compliance in business process modeling. In: Proceedings of the 2010 international conference on Semantic web rules. RuleML'10, Berlin, Heidelberg, Springer-Verlag (2010) 194–209
14. Lewis, D.: Adverbs of quantification. In: Formal Semantics of Natural Language. Cambridge University Press (1975) 178–188
15. Barwise, J., Cooper, R.: Generalized quantifiers and natural language. *Linguistics and Philosophy* **4** (1981) 159–219
16. Pelletier, F., Carlson, G.: The Generic Book. The University of Chicago Press (1995)
17. Wyner, A.Z.: Violations and Fulfillments in the Formal Representation of Contracts. PhD thesis, Department of Computer Science, King's College London (2008)