

# **RuleML2012@ECAI Challenge and Doctoral Consortium**

## **6th International Rule Challenge**

**Proceedings of the RuleML2012@ECAI Challenge, at the 6th International Symposium on Rules**

**Montpellier, France, August 27th-29th, 2012.**

**Edited by**

**Hassan Aït-Kaci \***  
**Yuh-Jong Hu \*\***  
**Grzegorz J. Nalepa \*\*\***  
**Monica Palmirani \*\*\*\***  
**Dumitru Roman \*\*\*\*\***

\* IBM Canada

\*\* National Chengchi University, Taiwan

\*\*\* AGH University of Science and Technology, Krakow, Poland

\*\*\*\* CIRSFD-University of Bologna, Italy

\*\*\*\*\* SINTEF, Norway

---

## **Table of Contents**

- [Preface](#)

### **Part 1: RuleML2012@ECAI Doctoral Consortium**

#### **Doctoral Consortium Papers**

1. [Enabling Knowledge-Based Complex Event Processing](#)  
*Kia Teymourian*
2. [Cognitive System for Knowledge Representation of Elementary Pragmatics](#)  
*Shashishekar Ramakrishna*

3. [Combining Ontology and Rules to Model Judicial Interpretation](#)  
*Marcello Ceci*
4. [Checking Compliance in European Tender Documents through Ontologies and Rules](#)  
*Isabella Distinto*

## **Part 2: RuleML2012@ECAI Challenge**

### **Invited Demo Papers**

5. [Rule Based Business Process Compliance](#)  
*Guido Governatori and Sidney Shek*
6. [PSOATransRun: Translating and Running PSOA RuleML via the TPTP Interchange Language for Theorem Provers](#)  
*Gen Zou, Reuben Peter-Paul, Harold Boley, Alexandre Riazanov*

### **Challenge Demo Papers**

7. [Legal Rules, Text, and Ontologies Over Time](#)  
*Monica Palmirani, Tommaso Ognibene and Luca Cervone*
8. [Browsing case-law: an application of the Carneades Argumentation System](#)  
*Marcello Ceci and Thomas Gordon*
9. [A model driven approach for bridging ILOG Rule Language and RIF](#)  
*Valerio Cosentino, Marcos Didonet Del Fabro and Adil El Ghali*
10. [A loose Coupling Approach for Combining OWL Ontologies and Business Rules](#)  
*Amina Chniti, Patrick Albert and Jean Charlet*
11. [Diamond Debugger Demo: Rete-Based Processing of Linked Data](#)  
*Daniel Miranker, Rodolfo Depena, Hyunjoon Jung, Juan Sequeda and Carlos Reyna*
12. [Monitoring BPMN-Processes with Rules in a Distributed Environment](#)  
*Lothar Hotz, Stephanie Von Riegen, Alexander Pokahr, Lars Braubach and Torsten Schwinghammer*
13. [Loosely-Coupled and Event-Messaged Interactions with Reaction RuleML 1.0 in Rule Responder](#)  
*Zhili Zhao, Kia Teymourian, Adrian Paschke, Harold Boley and Tara Athan*
14. [Graph-based rule editor](#)  
*Maciej Nowak, Jaroslaw Bak and Czeslaw Jedrzejek*
15. [RuleTheWeb!: Rule-based Adaptive User Experience](#)  
*Adrian Giurca, Matthias Tylkowski and Martin Müller*
16. [PLIS+: A Rule-Based Personalized Location Information System](#)  
*Iosif Viktoratos, Athanasios Tsadiras and Nick Bassiliades*

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

## Preface

This volume collects the four selected contributions of the RuleML2012 Doctoral Consortium and the twelve demo papers accepted for presentation at the RuleML2012 Challenge.

The RuleML Doctoral Consortium is part of the RuleML International Symposium on Rules, and is intended to attract Ph.D. researchers in the area of Rules and Markup Languages, from different backgrounds (e.g. theoretical, application, vertical domain-specific), to encourage a constructive and fruitful interdisciplinary approach. The doctoral symposium provides two benefits to students. Firstly, the students can interact with academics and commercial experts in the field, who can evaluate their research projects from both theoretical and application points of view. Secondly, they have the opportunity to present and discuss their ideas in a dynamic and friendly setting.

The first RuleML Doctoral Consortium was included in the first part of the RuleML 5th International Symposium on Rules (RuleML 2011@IJCAI) held on July 19th, 2011 in Barcelona. We have organized this second Doctoral Consortium as part of the RuleML 6<sup>th</sup> International Symposium on Rules, held jointly with ECAI2012, the biennial European Conference on Artificial Intelligence. We include here the four papers of the doctoral consortium, selected from two different backgrounds: Computer Science, for the first two papers, and Law, for the other two. All contributions stressed their attention to temporal reasoning and complex event modelling; application is mostly in the legal domain.

Teymourian's work, supervised by Adrian Paschke, formalizes the combination of vocabularies/ontologies and declarative rules in the field of event processing, allowing to create more intelligent event processors capable of understanding the semantics of events.

Ramakrishna's work, supervised by Adrian Paschke, focuses on shedding light on the imminent need for an effective system for extraction, representation and specification of legal rules, especially in national Patent Law regulation.

Ceci's work, supervised by Monica Palmirani, defines an integrated methodology for modelling judgments, starting from legal texts and capturing both structural parts and arguments used by judges

to reach conclusions using ontologies and rules modelled in a tentative preliminary version of LegalRuleML.

Distinto's work, supervised by Monica Palmirani, describes a hybrid approach for combining a legal ontology on the EU Public Procurement Directives, developed in OWL 2.0, with the related rules modelled using the emerging LegalRuleML standard. The goal is to present a semantic and conceptual framework to support checking of compliance of European tenders with EU Directives.

The 6<sup>th</sup> RuleML International Symposium on Rules (RuleML2012@ECAI), took place on August 27th, 2012 in Montpellier, France. The RuleML Challenge was included in the symposium for the 6<sup>th</sup> time. The Rule Challenge is devoted to disseminating the most advanced practical experiences with rule-based applications, where state-of-the-art solutions and recent research proposals meet the concrete needs of the market.

The Challenge session features two invited demo papers. The first demo, from Governatori and Shek, reports on the development and evaluation of a business process compliance checker (BPCC), based on the compliance-by-design methodology proposed by Governatori and Sadiq in 2009. BPCC is implemented on top of the Eclipse Activity BPMN 2.0 plug-in for the representation of process models and has been extended with features to allow users to add semantic annotations to the tasks in the process model. The second invited demo, from Zou, Peter-Paul, Boley, and Riazanov, presents an online Positional-Slotted, Objective-Applicative (PSOA) RuleML reasoning service, PSOATransRun, consisting of a translator and an execution engine. The translator, PSOA2TPTP, maps knowledge bases and queries in the PSOA RuleML presentation syntax to the popular TPTP interchange language, which is supported by many first-order logic theorem provers.

This year, five main topics have emerged from the other contributions to the Challenge:

- 1) Legal rule modelling and tools enabling the integration between legal textual sources, metadata, ontologies and rules, including temporal reasoning and compliance checking.
- 2) Combination of rules, objects and ontologies, to support the development of integrated systems able to deal with knowledge-



intensive domains and hybrid reasoning, especially when applied in business processing environments.

- 3) Graphic tools for creating, visualizing, debugging, and modelling rules, and for presenting the outcomes of the reasoning.
- 4) Improvement of tools related to RuleML as a standardization effort with particular regard to complex event management using the Reaction Rules dialect.
- 5) Combining rules with adaptive user experience for improving the Semantic Web and providing personalized online services.

In particular, Palmirani, Ognibene, Cervone present an integrated prototype platform composed of several modules (web based rule editor and rule viewer, XML database, Drools reasoner) that are able to capture all the levels of legal document modelling simultaneously (text, metadata, and rules) and to manage legal changes over time. An application based on a simple fragment of the US copyright normative rules is presented. Ceci and Gordon present the application of the Carneades Argumentation System to case-law to demonstrate its abilities to: reconstruct the legal interpretations performed by the judge; present its reasoning path; suggest possible different or divergent interpretations in the light of relevant code- and case-law.

Cosentino, Didonet Del Fabro and El Ghali present an implementation based on a Model Driven approach for bridging the gap between JRules (part of IBM's WODM – WebSphere Operational Decision Management) to the W3C's RIF standard, to improve interoperability and reusability of the rules in a business process environment. Chniti, Albert and Charlet present two prototypes based on the Business Rule Management System (BRMS) in IBM's WODM: an OWL plug-in and a change-management plug-in able to detect inconsistencies that could be caused by ontology evolution and propose solutions (called *repairs*) to resolve them. Miranker, Depena, Jung, Sequeda, and Reyna present Diamond, a Rete-based rule system that evaluates SPARQL queries on Linked Data using a graphical rule debugging environment. Hotz, von Riegen, Braubach, Pokahr and Schwinghammer demonstrate an application of rules in a business process scenario using Business Process Model and Notation (BPMN) and using declarative rules to monitor the process execution in a distributed environment.

Zhao, Teymourian, Paschke, Boley and Athan present a recent instantiation of Rule Responder, a rule-based inference agent middleware, integrated with the event-messaging features of Reaction RuleML, which supports interaction based on a loosely-coupled interface using rule signatures and decoupled communication via event messages.

Nowak, Bak and Jedrzejek present a prototype implementation of a graphical tool for creating rules; it is also used to visualize data and results of reasoning.

Giurca, Tylkowski and Müller present RuleTheWeb!, an application using JSON-Rules to enrich the user navigation experience on the web. RuleTheWeb! uses adaptive user experience based on semantic data and reaction rules aiming to enable Social Web rules designed and shared by web users.

Viktoratos, Tsadiras and Bassiliades present Personalized Location Information System (PLIS+), a system able to provide personalized, location-based information services via rule-based policies. PLIS+ proves that combining contextual data coming from the end-user and policy rules of the online service can lead to powerful personalized information services.

We would like to warmly thank all students, supervisors, referees, co-chairs, members of the program committee and the organising team that made the RuleML2012 Doctoral Consortium and the RuleML2012 Challenge a great success.

August 2012

Hassan Aït-Kaci  
Yuh-Jong Hu  
Grzegorz J. Nalepa  
Monica Palmirani  
Dumitru Roman

# Enabling Knowledge-Based Complex Event Processing

Kia Teymourian

Supervisor: Prof. Adrian Paschke  
Freie Universitaet Berlin, Berlin, Germany  
{kia, paschke}@inf.fu-berlin.de

**Abstract.** Usage of background knowledge about events and their relations to other concepts in the application domain can improve the expressiveness and flexibility of complex event processing systems. Huge amounts of domain background knowledge stored in external knowledge bases can be used in combination with event processing in order to achieve more knowledgeable complex event processing. In this dissertation, I address the challenges of adding formalized vocabularies/ontologies and declarative rules to the area of event processing for enabling more intelligent event processors which can understand the semantics of events.

## 1 Motivation

In many business organizations some of the important complex events cannot be used in process management, because they are not detected from the workflows and decision makers can not be informed about them. Detection of events is one of the critical factors for the event-driven systems and business process management. Because of current successes in business process management (BPM) and enterprise application integration (EAI), many organizations know a lot about their own activities, but this huge amount of event information can not be used in the decision making process. The permanent stream of low level events in business organizations needs an intelligent real-time event processor. The detection of occurrence of complex events in the organization can be used to optimize the management of business processes.

Semantic models of events can improve event processing quality by using event meta-data in combination with ontologies and rules (knowledge bases). The combination of event processing and knowledge representation can lead to novel semantic-rich event processing engines. These intelligent event processing engines can understand what is happening in terms of events, can (process) state and know what reactions and processes it can invoke, and furthermore what new events it can signal. The identification of critical events and situations requires processing vast amounts of data and metadata within and outside the systems. Knowledge about event types and their hierarchies i.e. specialization, generalization, or other forms of relations between events can be useful. Semantic (meta) models of events can improve the quality of event processing

by using event metadata in combination with ontologies and rules (knowledge bases). Event knowledge bases can represent complex event data models which link to existing semantic domain knowledge such as domain vocabularies / ontologies and existing domain data. Semantic inference is used to infer relations between events such as e.g. transitivity or equality between event types and their properties. Temporal and spatial reasoning on events can be done based on their data properties, e.g. a time ontology describing temporal quantities.

The usage of background knowledge in event processing can have several use cases such as: e-health, business activity monitoring, fraud detection, etc.

**Use Case - High Level Stock Market Monitoring:** Companies have some business dependencies to each other, e.g., a company  $C_1$  produces raw material  $M_1$ , the business of another company  $C_2$  depends on this raw material for its production and might have big troubles if they can not supply the material. A third company  $C_3$  financed the company  $C_2$  and might have some financial problems if the company  $C_2$  have some material troubles. Let's consider that Mr. Smith is a stock broker and has access to a stock exchange event stream like:  $(Name, "VOW"), (Price, 20.24), (Volume, 8, 835)$  Mr. Smith might be interested in this dependency chain and can define a complex event detection pattern for this special complex event without even knowing what these companies are. He might be interested to know when the prices for these three companies have started falling.

Mr. Smith might also be interested in special kinds of stocks and would like to be informed if there are some interesting stocks available for sale. His special interest or his special stock handling strategy can be described in high level language which describes the interest using background knowledge about companies. Mr. Smith would like to start a query on the event stream similar to the following query: **Buy** Stocks of Companies, **Who** have *production facilities in Europe* **and** produce products from Iron **and** have more than 10,000 employees **and** are at the moment in *reconstruction phase* **and** their *price/volume increased stable in the past 5 minutes*.

As we can see, the above query cannot be processed without having background knowledge which can define the concepts in this query. Mr. Smith needs an intelligent system which can use background knowledge about companies. A background knowledge like the following should be integrated and processed together with the event data stream in a real-time manner so that interesting complex events can be timely detected.

```
{ (OPEL, belongsTo, GM), (OPEL, isA, automobilCompany),
  (automobilCompany, produce, Cars), (Cars, areFrom, Iron),
  (OPEL, hasProductionFacilitiesIn, Germany), (Germany, isIn, Europe),
  (OPEL, isA, MajorCorporation), (MajorCorporation, have, over10,000employees),
  (OPEL, isIn, reconstructionPhase), ... }
```

## 2 Research Problem

The existing event processing approaches are dealing primarily with the syntactical processing of low-level signals, constructive event database views, streams, and primitive actions. They provide only inadequate expressiveness to describe

the ontological semantics of events, actions, agents, states, processes, temporal/spatial concepts and other related concepts. They also do not provide adequate description methods for the complex decisions, behavioral logics including expressive situations, pre- and post-conditions, complex transactional (re-) actions, and work-flow like executions. All of these are needed to declaratively represent many real-world domain problems on a higher level of abstraction. My dissertation will address the following two main problems of the existing event processing approaches:

**1. Lacking Knowledge Representation Methods:** Event processing needs a knowledge (metadata) representation methodology. The current event processing systems do not provide any knowledge representation methods for events, and there is no precise logical semantics about events and other related concepts. There is a need for methods which can include ontological semantics of all related concepts to the event processing without affecting the scalability and real-time processing. A formal specification can build a stable foundation which is needed for any describing and reasoning about a system. It is also needed for comparing different systems without misunderstandings. Event processing needs as its basis a formalization and specification which can describe events, event patterns, situations, pre- and post-conditions, (re-) actions etc. Definition of events by logic is not addressed in current complex event processing solutions.

In this dissertation, I will address the challenge of knowledge representation for complex event processing (CEP) which integrates the domain and application specific ontologies for events, complex events, situations, actions and other concepts related to CEP.

**2. Limited Processing and Integration Method of Background Knowledge with Event Stream:** The processing approach of current event processing engines often rely on processing of simple event signals. They do not implement any usage of metadata about events or other related concepts from the application domain. The existing on-the-fly in-memory processing methods do not address the challenges of integration of background knowledge and semantic enrichment of events or event queries (complex event definitions/patterns). In this dissertation, I will address the nature of the trade-off real-time high-performance processing of events and expressiveness reasoning on background information. The advantages and disadvantages of alternative processing methods for the fusion of event stream and background knowledge should be investigated which can be used without effecting the real-time processing or scalability.

### 3 Fusion of Events and Background Knowledge

The fusion of background knowledge with data from an event stream can help the event processing engine to know more about incoming events and their relationships to other related concepts. I propose to use a *Knowledge Base (KB)* which can provide background knowledge (conceptual and assertional, T-Box and A-Box of an ontology) about the events and other non-event resources. This means that events can be detected based on reasoning on their type hierarchy, temporal/spatial relationships, or their relationship to other objects in the application domain. The connections to other relevant concepts and objects means for

example the relationship of a stock market event (price change) to the products or services of a company.

The benefits of using background knowledge in CEP are higher *expressiveness* and *flexibility*. Expressiveness means that an event processing system can precisely express CEP patterns and reactions. Flexibility means that a CEP system is able to integrate new business changes into the systems in a fraction of time rather than changing the whole event processing rules. Furthermore, complex event processing can benefit from the knowledge representation and semantic web technologies, because a central problem of event processing is information integration for which these technologies have already been proven to be a valid solution.

I propose to use external KBs for the storage and reasoning on background knowledge. The background knowledge about events and other non-event concepts/objects is described in description logic. The knowledge in the KB can be stored in the Resource Description Framework (RDF) data format<sup>1</sup> in an external triple store (special kind of databases for storage and management of RDF data). This knowledge can be queried from the event processing agents based on the demands of the event query rules. The external KB also includes a description logic to reason on the relations between events and other relevant non-event objects in the application domain. The KB can be queried by using SPARQL<sup>2</sup> queries and the results are then included in the event processing engine.

## 4 Knowledge Representation for CEP

Ontologies play an important key role in the knowledge-based CEP. They should be the conceptualization of the application domain to allow reasoning on events and other non-event concepts. I propose that event processing domain should be described by a modular and layered ontology model which can be reused in different application areas. Important general concepts such as event, action, situation, space/place, time, agent and process should be defined based on meta-models and pluggable ontologies which are in a modularized ontological top-level structure. These general concepts defined in the top-level ontologies can be further specialized with existing domain ontologies and ontologies for generic tasks and activities. The applications ontologies for specialize these domain and task concepts with respect to a specific application, often on a more technical platform specific level.

**Event Query Rules:** Event query rules (Complex event Patterns) can be considered as declarative rules which are used to detect complex events from streams of raw events. These event queries have a hybrid semantic, because they use event operation algebra to detect events and they use SPARQL queries to include background knowledge about these events and their relationships.

The event query rules allow simple event algebra operations, similar to Snoop [6] i.e. event operations like Sequence (Ordered), Disjunction (Or), Xor (Mutually Exclusive), Conjunction (And), Concurrent (Parallel), Any, Aperiodic,

---

<sup>1</sup> <http://www.w3.org/RDF/>

<sup>2</sup> SPARQL <http://www.w3.org/TR/rdf-sparql-query/>

Periodic, Operator (generic Operator). Further higher interval-based event operations like (BEFORE, MEETS, OVERLAP, ...) can also be used. My event query rules also include SPARQL query predicate to query external KBs, the SPARQL queries are used in a rule in combination with event operation algebra. This hybrid use of SPARQL query with event operation algebra can be categorized into several categories.

#### 4.1 Categorize of Event Query Rules

Event query rules can be categorized into several categories based on the usage of knowledge queries (SPARQL queries) inside the query rule. As previously described, the semantics of the whole event query is a hybrid semantic of description logic and event operation algebra which defines the semantics of event detection. In this section we describe the most important and interesting categories of event sQuery rules. This categorization is not a complete classification of all possible rule combinations, our aim is more to emphasize interesting rule combinations which can be processed using different event processing approaches.

**Category A - Single SPARQL Query:** In this category, the event query rule includes only one single knowledge query and uses its results in one or more variables within the event detection rule. A SPARQL query is used to import knowledge about event instances or types. One or more attributes of events are used to build the basic triple pattern inside the SPARQL query. Category A event sQuery rules can be categorized into three subcategories:

**Category A1 - Raw SPARQL:** This category of sQuery rule is the simplest form of these event query rule. The included SPARQL query is only about the resources in the background knowledge. The background knowledge query is independent from the event stream, however the complex event detection is defined on the results of this query in combination with the event stream. In some cases, on each event the SPARQL query should be resent to the KB to update the latest results from the KB.

**Category A2 - Generated SPARQL:** In this category of sQuery rules with each incoming event a different SPARQL query is generated and sent to the target knowledge base. The attribute/values of an event instance are used to generate basic triple patterns of a SPARQL query. Based on user definitions some of the tuples (attribute, value) of an event instance are selected and used to generate a single SPARQL Query.

**Category A3 - Generated SPARQL from Multiple Events:** The query is similar to A2, but the SPARQL query is generated from multiple events. Within a data window (e.g., a sliding time window) from two or more events a single SPARQL query is generated. Multiple events are used to generate the single SPARQL query, the event processing waits for receiving some new events and then generate a SPARQL query based on the emitted events, and query for the background knowledge about them.

**Category B - Several SPARQL Queries:** Queries of this category include several SPARQL queries and combine them with event detection rules. This means that several A category rules are combined together which can build a category B. The category B of rules are able to combine results from KBs with

events using event operation algebra.

**Category B1 - Several SPARQL Queries in AND, OR and SEQ Operations:** The category B1 is based on the category B, but the results from the SPARQL query predicates are combined with AND, OR, SEQ or similar event algebra operations. The whole query is evaluated on sliding windows of event streams. The SPARQL query predicates are not depending on each other, i.e., the results from one is not used in another SPARQL predicate, so that they are not depending on the results of the other SPARQL query.

**Category B2 - Chaining SPARQL Queries:** In category B2 several SPARQL queries are generated and executed in sequence. They can be generated based on the results of the previous SPARQL query. Each SPARQL query can be generated from a set of events (e.g., included in a slide of event stream by means of a sliding window, a counting or timing window). This means that different data windows can be defined to wait until some events happened and then a SPARQL query is executed. SPARQL queries might be defined in a sequence chain. The results are directly used for event processing or used in another following SPARQL query.

**Category B3 - Chained and Combined SPARQL Queries:** In this category SPARQL queries are used in combination with all possible event algebra operations like, AND  $\wedge$ , OR  $\vee$ , SEQ  $\oplus$ , Negation  $\neg$ , etc. The event operations are used for combining the results from several SPARQL queries or several SPARQL queries are used in combination with event algebra operations like:  $((sparql_1 \oplus sparql_2) \wedge sparql_3 \vee \neg sparql_4)$

This category of event query rules is the general form of queries and has the highest possible complexity, because the results from external KBs are used in combination with event operations or the attribute/values from incoming events are used for generation of complex SPARQL queries.

## 5 Integration and Processing of Event Stream with KB

Based on the above discussed categories of event query rules, different event processing approaches are possible to satisfy the requirements of event processing agent (EPA), e.g., high performance, scalability and elasticity. In the following different processing approaches are discussed:

**Polling the Knowledge Base:** The basic approach is to execute a query on the KB on each incoming event. After events are emitted and received in EPA, the EPA sends one or more queries to KB for every event. The problems of this approach, scalability and real-time processing, makes it impossible to use it for time-sensitive use cases like algorithmic trading or fraud-detection systems.

**Knowledge Query First (KQF):** For the processing of some rule categories, it is possible to execute the SPARQL query in advance and offline to the live event stream, i.e. execution of SPARQL query, before the events are emitted to the system. The results of knowledge queries can be cached in the main memory and be processed together with the events. Nevertheless, the results of the knowledge query can be old results from the knowledge base, hence they should be updated from time to time, e.g., by executing the whole query or pushing the result differences to the event processing agent.



**Plan-Based Processing (PBP)** This approach is about processing of event query based on an optimal plan for its sub-queries to avoid any unnecessary costs or losses of time. Some rule categories like category B1 rules, have several SPARQL queries which use multistep knowledge acquisition from external KBs. These SPARQL queries are combined in AND, OR, SEQ or similar event operations and the whole query should be evaluated in a time window. This makes it possible that the SPARQL queries can be executed in a sequence one after another or in a parallel setting. An execution plan can be generated to find out which execution plan is the low cost plan and which execution plan can be considered as high performance execution plan.

**Event Query Preprocessing (EQPP)** Event Query Preprocessing (EQPP) means that the complex query is preprocessed before the query is executed against the incoming stream event data. The original complex event query can be preprocessed by use of a KB and rewritten into several simple *new queries*. The original complex event query  $Q_a$  is preprocessed under the usage of a KB and divided into a set of simple event queries like  $\{q_1, \dots, q_n\}$ . A simple query is here a query which can be processed only with the information from the event stream and there is no need for using background knowledge. In the next step, these *new queries* can be syntactically processed on a network of event processing agents. The complex query  $Q_a$  can be considered as a propositional formula which can be converted to conjunctive normal form (CNF)  $Q_a \leftarrow q_1 \wedge \dots \wedge q_n$ , i.e. if all of the simple queries are given, then the complex event query is satisfied. The preprocessing is done by a processing agent which can access the KB and divide the complex query into several simple queries. The complex query  $Q_a$  can also be mapped in disjunctive normal form (DNF)  $Q_a \leftarrow q_1 \vee \dots \vee q_n$ , i.e. when one of the simple queries is triggered, then the complex event query will be satisfied and triggered.

## 6 Related Work

The state of the art approaches for event processing can be distinguished into two categories, rule-based approaches and non-rule-based approaches. Some of the event processing systems use non-deterministic finite state automata like Cayuga[4] or ESPER<sup>3</sup>. Many event processing languages have been proposed like, Snoop [6], Cayuga Event Language (CEL)[4], XChangeEQ [5]. Also several data stream processing systems have been proposed like Telegraph[8] which are targeted at handling continuous queries over high-throughput data streams. These systems are also related to the event processing systems[7].

Some stream reasoning languages and processing approaches are also proposed. Barbieri et.al. propose Continuous SPARQL (C-SPARQL) [3] as a language for continuous query processing and Stream Reasoning. Stream reasoning approaches for reasoning on RDF stream are not designed for fusion of background KBs and event stream. One of the recent rule-based systems is ETALIS [2]. ETALIS is a rule-based stream reasoning and complex event processing (CEP). ETALIS is implemented in Prolog and uses Prolog inference

<sup>3</sup> Esper: <http://esper.codehaus.org> , May 2012

engine for event processing. EP-SPARQL [1] is a language for complex events and stream reasoning. The formal semantics of EP-SPARQL is along the same lines as SPARQL. EP-SPARQL can be used in ETALIS for reasoning on RDF triple stream (event stream can be mapped to RDF stream). I have discussed CEP approaches which are most related for our knowledge-based CEP.

## 7 Future Work

My future steps are to work on more details of knowledge representation for events, situations, actions, and other related concepts. One of my tasks is to work on details of event query preprocessing algorithms for rewriting of complex event queries to several simple queries which can be distributed over an event processing network. Furthermore, I have to work on the described plan-based approach and specification of heuristics which can be used for selection of the optimized processing plan for a given query.

## References

1. Darko Anicic, Paul Fodor, Sebastian Rudolph, and Nenad Stojanovic. Ep-sparql: a unified language for event processing and stream reasoning. In *Proceedings of the 20th international conference on World wide web*, WWW '11, pages 635–644, New York, NY, USA, 2011. ACM.
2. Darko Anicic, Paul Fodor, Sebastian Rudolph, Roland Stühmer, Nenad Stojanovic, and Rudi Studer. Etalis: Rule-based reasoning in event processing. In Sven Helmer, Alexandra Poulouvasilis, and Fatos Xhafa, editors, *Reasoning in Event-Based Distributed Systems*, volume 347 of *Studies in Computational Intelligence*, pages 99–124. Springer Berlin / Heidelberg, 2011.
3. Davide Francesco Barbieri, Daniele Braga, Stefano Ceri, and Michael Grossniklaus. An execution environment for c-sparql queries. In *Proceedings of the 13th International Conference on Extending Database Technology*, EDBT '10, pages 441–452, New York, NY, USA, 2010. ACM.
4. Lars Brenna, Alan Demers, Johannes Gehrke, Mingsheng Hong, Joel Ossher, Biswanath Panda, Mirek Riedewald, Mohit Thatte, and Walker White. Cayuga: a high-performance event processing engine. In *SIGMOD '07: Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 1100–1102, New York, NY, USA, 2007. ACM.
5. François Bry and Michael Eckert. Rule-based composite event queries: The language xchangeeq and its semantics. In *Proceedings of First International Conference on Web Reasoning and Rule Systems, Innsbruck, Austria (7th–8th June 2007)*, volume 4524 of *LNCS*, pages 16–30, 2007.
6. S. Chakravarthy and D. Mishra. Snoop: an expressive event specification language for active databases. *Data Knowl. Eng.*, 14:1–26, November 1994.
7. Sharma Chakravarthy and Qingchun Jiang. *Stream Data Processing: A Quality of Service Perspective Modeling, Scheduling, Load Shedding, and Complex Event Processing*. Springer Publishing Company, Incorporated, 1st edition, 2009.
8. Sirish Chandrasekaran, Owen Cooper, Amol Deshpande, Michael J. Franklin, Joseph M. Hellerstein, Wei Hong, Sailesh Krishnamurthy, Samuel R. Madden, Fred Reiss, and Mehul A. Shah. Telegraphcq: continuous dataflow processing. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, SIGMOD '03, pages 668–668, New York, NY, USA, 2003. ACM.

# Cognitive System for Knowledge Representation of Elementary Pragmatics

Shashishekar Ramakrishna<sup>1,2</sup>

<sup>1</sup>AG Corporate Semantic, Department of Computer Science Freie Universitaet Berlin, Germany

<sup>2</sup>TelesPRI GmbH Berlin, Germany  
s.ramakrishna@teles.de

**Abstract.** The focus of this article is to throw light on the imminent need for an effective system for extraction, representation and construction of legal norms, especially the national patent law norms. This system, complementary to the FSTP-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 research.

**Keywords.** Facts Screening and Transformation Processor (FSTP), Innovation Test, Innovation Expert System (IES)

## 1 Motivation

The need of a sub-system for automating the application of elementary pragmatics<sup>1</sup>, ‘EP’ and National Patent Laws into the existing Facts Screening and Transformation Processor, ‘FSTP’[1]/Innovation Expert System, ‘IES’. This enables a person of pertinent skill, who is needed for recognizing non-elementary pragmatics, to recognize automatically and/or guided interactively by the FSTP ES to consider whether the properties an innovation at issue can be considered as Anticipate (A), Not-Anticipate (N) /Contradicts (C) to its prior arts/ considered reference set (RS).

## 2 Background - The Fact Screening and Transformation

As described in [1], “an innovation/creation over existing knowledge, provided as a reference set RS of prior art documents, is representable by a technique teaching TT.p which goes beyond the knowledge of the RS – just as in a patent/application.

---

<sup>1</sup> Elementary pragmatics are disclosures (explicit/implicit) of certain art which can be easily understood by a person of pertinent skill

This compound of knowledge, representing an innovation, is called “PTR”, standing for a “pair of TT.p 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 a PTR all technical facts for determining the “creativity geometrical” height of TT.p over RS, and
2. Instantly recognizing and replying to any rational query as to any relation between this TT.p and this prior art RS.

The PTR Expert System (ES) is built around the PTR’s “FSTP Test” (FSTP = “facts screening & transforming processor”), and hence is also called FSTP ES. The FSTP Test of a PTR supports initially screening its documents for all technical informal fundamental facts, then transforming them into technical formal fundamental 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. **pragmatic** (alias **innovative**) facts. These technical secondary facts use metrics induced by the Highest Courts precedents’ on creativity/innovation – by their numbers of RS-orthogonal and independent thoughts embodied by TT.p. From the basic facts the classical yes/no answer to the question, whether TT.p 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.p over its RS. Based on plcs-height/-creativity, TT.p’s pragmatic/innovative height over RS additionally takes into account the PTR’s “patent monopoly granting pragmatics”. A pmgp, 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.p. Hence a sub-system capable of (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 pmgp based on NLS/ (NNI = National Normative judicial Interpretation of facts).

Figure 1, shows different Knowledge Representation (KR) domains with sub-domains which cause an impact on a PTR during FSTP Test. The object of our concern in this thesis is to create KR domain dealing with NPS, and having EU PS, US PS, AU PS etc. as sub-domains. The formal modeling involves modeling of NLS/NNI by ontologies and rules using deductive (non-monotonic) reasoning for legal interpretations and inductive logics for learning.

### 3 Goals/Aim

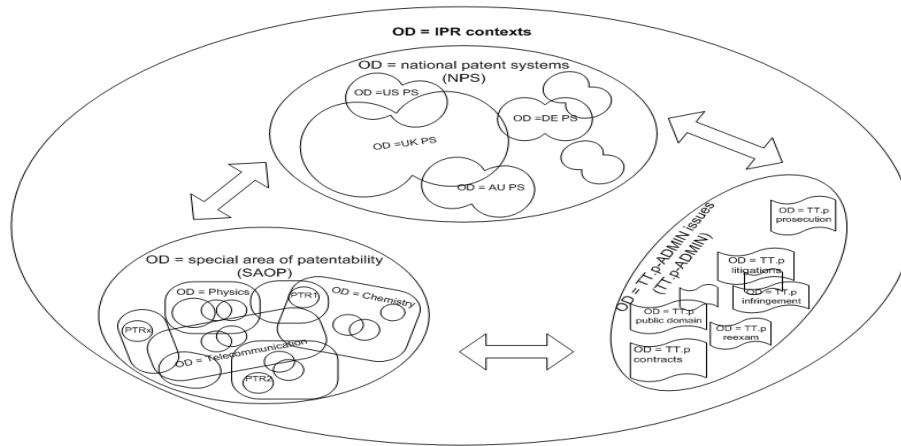
1. To analyze and extract the rules and ontological concepts described in the natural language descriptions of NPSs.

2. To identify the required semantics and inference rules needed for legal reasoning with NPSs and for the legal interpretation enabling the separation 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 ontologies domains representing the conceptualization of the NPSs 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 the 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 the knowledge. This addresses e.g., collaborative knowledge engineering and management (versioning, different roles such as author, maintenance), updates in the NPSs by new decisions which lead to corresponding isomorphic updates in the NPSs knowledge bases, integration of internal and external (semantic) background knowledge e.g. about skill, elementary pragmatics, usage data (annotations, proofs, etc.)

## 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?



1. Fig. 1. **Interdependence of domain ontologies** (Source: [1])

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).

- **Functional Requirements**
  - e.g., expressiveness, ...
- **Non functional requirements at design time**
  - e.g., composability and extensibility, interoperability, declarative implementability, modifiability and evolvability, reusability and interchangeability, ...
- **Non functional properties at runtime**
  - 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 [2], DeepDyve, etc... retrieves patents through large databases which forms the required RS (if previously not specified by the jury) for the TT.p. Thus formed PTR will be transformed from their natural language texts into some standard representation formats like XML, using text-mining and semantic recognition and annotation techniques supporting human knowledge engineers in the fact screening and transformation process.

Similar to the PTR, the existing patent rules from NPS have to be transformed from their natural language format to more standardized rule representation formats

like Reaction-RuleML [3], LKIF [4], or the upcoming Legal RuleML etc... 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 a flexible processing and legal reasoning [5]. The required (patent) rules/constraints are built by the rule creator module, which uses a rule-based agent networks like Prova [6] for realizing distributed rule inference services.

Such built rules may be;

1. Simple: Built based simple on Patentable Subject Matter (PSM) constraints, which are readily available out of any NPS. Like,

```
/* Invention dealing with plant, animals or seeds are not permitted to be patented */
or
/* Process of learning language, playing chess, teaching or operating machinery are
not patentable */
```

```
PSMCriteria1 ≡ (Invention ∧ (Product ∨ Process) ∧ (¬
    Plant ∨ ¬ Seed ∨ ¬ Animal))
PSMCriteria2 ≡ (Process ∧ (¬ LearningLanguage ∨ ¬
    PlayingChess ∨ ¬ OperatingMachinery
    ∨ ¬Teaching))
```

2. Complex : Built based on deductive logic [8] to match the elementary patent rules with background facts then using inductive logic in generalizing goal facts into rules that connect with background facts.

```
/* use of any radioactive substance or any process for atomic energy production,
control or disposal cannot be patented */
```

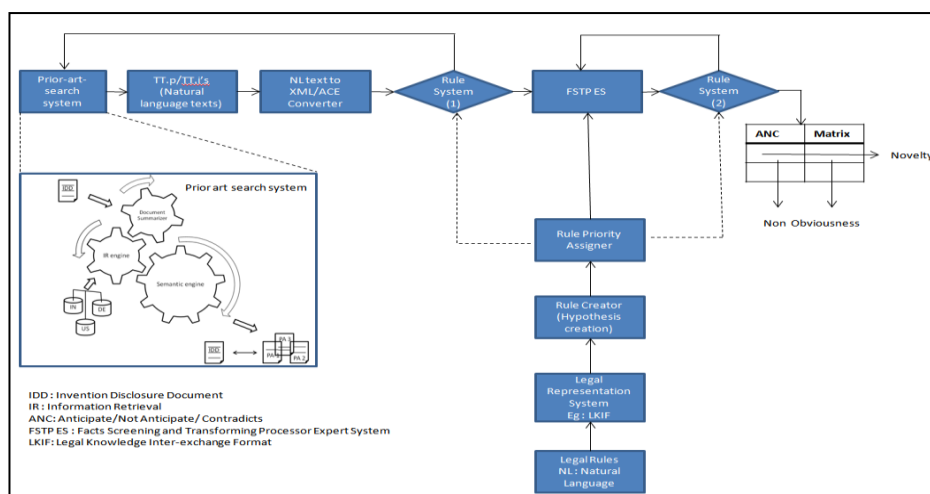
```
PSMCriteria3 ≡ ∀ Invention ∃ SubjectMatter (Process ∧ (¬
    AtomicEnergy ∧ (¬Production ∨ ¬ Disposal ∨ ¬
    Control)) ∧ (∃ Element (¬RadioactiveSubstance))).
```

3. Compound: Built based on combination of several rules (deductive rule chaining).

```
/*for prior-claiming, the invention claiming priority should have been patented in
US, the inventions priority-claim-date should be before the newly claimed invention
and publishing date should have been before the newly claimed invention*/
```

```
Criteria4 ≡ Invention ∧ (Product ∨ Process) ∧ (Country
    (US))
```

```
Criteria5 ≡ InventionPriorityDate (ClaimingInvention <
    ClaimedInvention)
```



**Fig. 2.** System model overview

$\text{Criteria6} \equiv \text{InventionPublishDate} (\text{ClaimingInvention} < \text{ClaimedInvention})$

$\text{PriorclaimCriteria7} \equiv \text{Criteria4} \wedge \text{Criteria5} \wedge \text{Criteria6}$

Such built rules are assigned priorities using, e.g. defeasible logic and scoped reasoning for distributed modularization of the knowledge bases (such as used in the Rule Based Service Level Agreement project and supported by Reaction RuleML (and the new upcoming Legal RuleML)).

Standardized rules with priorities enable arguments to be created, evaluated and compared. One such category of rules are Elementary Pragmatic (EP) rules, which including legal rules that can be applied at different fact gathering stages of the FSTP expert system on a PTR. Few examples for such discretization stages and their applicable elementary rules are as shown in Table 1.

Elaborating more on the concept identification stage, validation of identified concept/concepts is a process of filtering the concepts identified from a patent document, TT.0 based on existing EP's. Thereby, segregating them into non-patent-eligible, 'npe' and patent-eligible, 'pe' concepts. A concept under 'pe' may also be known as creative concepts, Cr-C. Certain complex concepts need a combination of EP's to be applied together to classify them as 'npe' which would otherwise have been considered as 'pe' concept.



## 5.1 Proposed Framework

We propose a legal information system framework as shown in Figure 3. The proposed framework is built based on a general information system research framework [7]. 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's, are fed by the Environment module and the syntax, semantics, pragmatics and instantiations encompassing a norm are fed by the Knowledge Base (KB) 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 norm's 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's 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 are sent back to KB module.

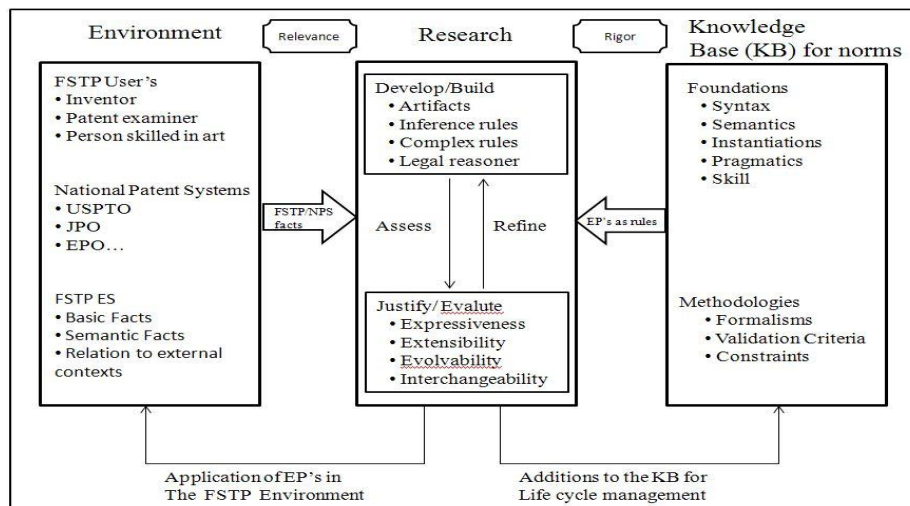


Fig. 3. Legal (esp. Patent) Information system framework.

## 5.2 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.

<b>FSTP discretization stages</b>	<b>Elementary rules applicable</b>	<b>Explanation</b>
Element identification stage	PSMCriteria1	Elements with Plant or Animals or Seeds not permitted
Attribute identification stage	PSMCriteria2	Attributes having below methods are not permitted a. Method of learning language b. Method of teaching c. Method of operating machine
Concept identification stage	PSMCriteria3	Concepts with below references are not permitted a. Musical notations b. Coloring substance for identification c. Atomic energy i. Production ii. Control iii. Disposal d. Radioactive substance

**Table 1.** FSTP discretization stages and their applicable elementary rule

EP can be divided into 4 types, table 2 shows few trivial example concepts considered as ‘npe’ for each EP mentioned above:

- EP from Natural Laws of Nature, EP-NL
- EP from Natural Phenomenon’s, EP-NP
- EP from National Legal(Patent) Systems, EP-NPS
- EP from Skill Documents/Standards, EP-STD

<b>EP</b>	<b>Concepts related to</b>
EP-NL	Speed of light Theory of relativity $E=mc^2$ Dijkstra Algorithm
EP-NP	Gravity Human metabolism
EP-NPS	Method of learning Language Method of teaching Production/Control/Disposal of atomic energy
EP-STD	Maximum delay for data transfer in ordinary telephone is 0.5 secs ISDN line has a stack of three protocols

**Table 2.** Shows few trivial example concepts considered as ‘npe’ for each EP mentioned above

### 5.3 Examples : Landmark cases

#### Mayo vs Prometheus.

*Invention summary:* Administration and use to thiopurine drugs to treat auto-immune disease

*Concepts:*

- C1: Physician administers the drug given to the patient using ‘administering step’
- C2: Physician measures the resulting metabolic levels in the patient’s blood
- C3: Physician compares the patients metabolic level against known safe and effective metabolic levels and then decided to increase or decrease the drug dosage.

Criteria8  $\equiv$  (Process  $\vee$  Manufacture  $\vee$  machine  
 $\vee$  composition of matter)

Criteria9  $\equiv$  Criteria8  $\wedge$  ( $\neg$  EP-NL  $\vee$   $\neg$  EP-  
NP  $\vee$   $\neg$  EP-STD  $\vee$   $\neg$  AbstractIdea)

{EP-NPS}	{C1}	‘npe’
{EP-STD}	{C2}	
{EP-NL}	{C3}	

Even though all concepts defined above seems to qualify all criteria’s at the first glance, On addition of pragmatic context (using micro-ontologies) to our analysis, Concepts, C2 and C3 identified in the above example fail to qualify the ‘criteria 9’, while concept C1 qualifies the ‘PSMCriteria1’, ‘Criteria 8’ and ‘criteria 9’, it fails to qualify ‘PSMCriteria2’. Thus, grouping all identified concepts as ‘npe’.

#### Newman vs United States Patent Office.

*Invention summary:* A device which will produce mechanical power exceeding the electrical power being supplied to it.

*Concepts:*

- C1: Electromagnetic energy can be rendered by a rotating permanent magnet spinning inside an electromagnetic pulsating conducting coil.
- C2: Rotating permanent magnet spinning inside an electromagnetic pulsating coil utilizes the coil mass energy and turns in into torque.

{EP-NL, EP-STD}	{C1}	‘npe’
{EP-STD}	{C2}	
{EP-NPS}	{C1, C2}	

Concepts C1 and C2 do not pass the ‘criteria 9’ while a concept formed by combining concept C1 and C2 would also fail to qualify the PSMCriteria1. Thus making the entire invention as ‘npe’.

## 6 Conclusion

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.

## 7 Acknowledgements

This work has been partially supported by the Fact Screening and Transformation Project (FSTP) funded by the TelesPRI GmbH: [www.fstp-expert-system.com](http://www.fstp-expert-system.com)”.

## 8 References

1. S. Schindler, “THE FSTP EXPERT SYSTEM (FSTP = Facts-Screening-and-Transforming-Processor),” 2010.
2. K. Dahlgren, “Technical Overview of Cognition’s Semantic NLP <sup>TM</sup> ( as Applied to Search ),” *ReCALL*, pp. 1-20, 2007.
3. Paschke, A., Boley, H., Zhao, Z., Teymourian, K., & Athan, T. (2012). Reaction RuleML 1.0: Standardized Semantic Reaction Rules. *RuleML 2012*.
4. Thomas F. Gordon. (2008). *The Legal Knowledge Interchange Format ( LKIF )*.
5. Hans Weigand, Adrian. Paschke. (2012). The Pragmatic Web: Putting Rules in Context. *RuleML 2012*.
6. Paschke, A., & Boley, H. (2011). RULE RESPONDER: Rule-Based Agents for the Semantic-Pragmatic Web. *International Journal on Artificial Intelligence Tools (IJAIT)*, 20(06), 1043-1081. doi:10.1142/S0218213011000528.
7. Hevner, Alan R.; March, Salvatore T.; Park, Jinsoo; and Ram, Sudha. 2004. "Design Science in Information Systems Research," *MIS Quarterly*, (28: 1)

# Combining Ontologies and Rules to Model Judicial Interpretation

Marcello Ceci<sup>1</sup>

<sup>1</sup> CIRSFD, University of Bologna.  
m.ceci@unibo.it

**Abstract.** This research aims to define an integrated methodology for modelling judgments, starting from legal texts and capturing not only structural parts but also arguments used by judges to reach conclusions. The goal is to build a complete ontology framework capable of detecting and modelling knowledge from the judgement's text. The formalized judgements provide the necessary metadata for the rule layer to enable argumentation towards the acceptance or rejection of a given interpretation. For pursuing this goal it is important to integrate the legal ontology construction with a rule formalization following legal reasoning-oriented theory and defeasible logics, just like the Carneades application (presented here) follows Walton's argumentation theory. The XML interchange uses OWL, RuleML, and the emerging LegalRuleML.

**Keywords:** OWL, Case-law, Legal Reasoning, LKIF-Rules, LegalRuleML

## 1 Introduction

Precedent is a main element of legal knowledge worldwide: by settling conflicts and sanctioning illegal behaviours, judicial activity enforces law provisions within the national borders, therefore supporting the validity of laws as well as the sovereignty of the government that issued them. Moreover, precedents (or case-law) are a fundamental source for law interpretation and it paradoxically happens that the exercise of jurisdiction can influence the scope of the same norms it has to apply, both in common law and civil law legal systems – even if to different extents.

The goal of the present research is to define a framework for case-law semantics, exploiting Semantic Web technologies to achieve isomorphism between the text fragment (the only binding legal expression) and the legal rule, thus "filling the gap" between document representation and rules modelling [15]. The formalization of the general structure of case-law, the metadata connected with the judicial legal concepts and the ontology set<sup>1</sup> constitute the basis for a semantic tool enriching the XML mark-up of precedents and supporting legal reasoning [17]. We believe that the features of OWL2 could greatly improve legal concepts modelling and reasoning [4], once properly combined with rule modelling. Our aim is hence to formalize the legal concepts and the argumentation patterns contained in the judgment in order to check,

---

<sup>1</sup> An ontology is a shared vocabulary, a taxonomy and axioms representing a domain of knowledge by defining objects and concepts with their properties, relations and semantics.

validate and reuse the elements of judgement as expressed by the text and the argumentation contained in it. To achieve this, four models are necessary:

- a *document metadata structure*, capturing the main parts of the judgment to create a bridge between text and semantic annotation of legal concepts;
- a *legal core ontology*, describing the legal domain's main elements in terms of general concepts through an LKIF-Core extension;
- a *legal domain ontology*, modelling the legal concepts of a specific legal domain concerned by the case-law, including a set of sample precedents;
- *argumentation modelling and reasoning*, representing the structure and dynamics of argumentation.

## 2 Research Methodology

This research is based on a middle-out methodology: top-down for modelling the core ontology, bottom-up for modelling the domain ontology and the argumentation rules. The research starts from the analysis of a sample set of 27 decisions of different grade Italian case-law (tribunal, court of appeal, Cassation Court) concerning consumer law<sup>2</sup>.

While DL is very powerful and semantically rich (making OWL an acknowledged standard for the modelling of document metadata), monotonic logics are not sufficient for modeling legal rules in a verifiable way, while at the same time maintaining the structure of the legislation and regulation they are ought to represent: legislation is typically organized as general rules subject to exceptions, and arguments made by applying legal rules are often defeasible. Moreover, the application of laws depends on time, and various legal rules may conflict with each other: these conflicts are resolved using legal principles about priority relationships between rules. Metarules such as these, however, need defeasible logics in order to be properly managed.

The research relies on the previous efforts of the community in the field of legal knowledge representation [2] and rule interchange for applications in the legal domain [9]. The issue of implementing logics to represent judicial interpretation has already been faced [1], albeit only for the purposes of an sample case. The aim of the present research is to apply these theories to a set of real legal documents, stressing the OWL axioms definitions as much as possible in order to enable them to provide a semantically powerful representation of the legal document and a solid ground for an argumentation system using a defeasible subset of predicate logics.

## 3 The Ontology Set

The Core Ontology (an extension of the LKIF-Core Ontology) introduces a model of the legal domain whose main elements are: legal rules, legal concepts, material circumstances, judicial claims, judicial interpretations, adjudications<sup>3</sup>.

---

<sup>2</sup> The matter is specifically disciplined in Italy through the "Codice del Consumo" (Consumer Law) and articles 1341-1342 of the Civil Code. This discipline is also present in all foreign legal systems, which will allow an extension of the research to foreign decisions and laws.

<sup>3</sup> Even though the core ontology should be domain-generic and not modeled upon a specific legal subject, the sample model was conceived only to successfully represent the interaction in the civil law subject, when contracts, laws and judicial decisions come into play.

Following this structure, the metadata taken from judicial documents are represented in the Domain Ontology. The modeling was carried out manually by an expert in the legal subject, which actually represents the only viable choice in the legal domain: automatic information retrieval and machine learning techniques, in fact, do not yet ensure a sufficient level of accuracy, even if some progress in the field has been made, for example in applying NLP techniques to recognize law modifications [14]. Building a legal domain ontology is similar to writing a piece of legal doctrine, thus it should be manually achieved in such a way as to maintain a reference to the author of the model, following an open approach.

Such a layered ontology creates an environment where the knowledge extracted from the decision's text can be processed and managed, in such a way as to enable a deeper reasoning on the interpretation instances grounding the decision itself. Example of this deeper reasoning include: finding relevant precedents which were not explicitly cited in the decision; validating the adjudications of the judge on the claims brought forward by the parties during the trial on the basis of applicable rules, accepted evidence and interpretation; suggesting legal rules/precedents /circumstances that could bring to a different adjudication of the claim. The layered structure of the ontology set also allows an efficacious scaling from legal concepts to factors, up to dimensions and legal principles: all these concepts can be represented in the domain ontology, and the hook of the core concepts to the LKIF taxonomy should allow semantic alignment between domain ontologies when different authors are concerned (yet, the ontology set alignment has not been tested by the present research).

## **4 Legal Argumentation Modeling**

The last part of the research relies on argumentation modeling to perform legal reasoning on the knowledge contained in the ontologies, in order to evaluate their capabilities. As a first test of the ontology set, the research developed a pilot case using the Carneades Argumentation System<sup>4</sup> [5] in order to verify the correctness of the OWL representation of precedents and to test how far defeasible rules can simulate judicial reasoning. Carneades implements Walton's argumentation schemes [10] to reconstruct and evaluate past arguments in natural language texts, but also as templates for manual generation of arguments graphs representing ongoing dialogues. It can therefore be used for studying argumentation from a computational perspective, but also to develop tools supporting practical argumentation processes. It is capable of importing knowledge from the ontology set [8] and of applying rules on them [7]. The new version of Carneades (2.x, under development) uses the Clojure language, while the latest complete version (1.0.2) relies on the LKIF-Rule language [2].

### **4.1. The implementation**

The present research developed applications [3] for Carneades in both LKIF-Rules and Clojure languages. Implementing Carneades involved the following interventions:

- enriching the semantic content of the ontology set by representing finer-grained knowledge contained in the decision's text, in an environment which does not overload the OWL reasoners, compromising computability;

---

<sup>4</sup> Carneades is a set of open source software tools for mapping and evaluating arguments, under development since 2006.

- modeling a rule system representing the dynamic relationships created by judicial interpretation and law application;
- importing knowledge from the ontology set in such a way as to allow successful interaction with the rule set and the Carneades model.

#### **4.2. Results and Issues**

The so-built system is capable of creating argumentation graphs pro or con a given legal statement: for example, it can state whether a contract clause can be judged as inefficacious, under which norms, and following which judicial precedents and judicial interpretations. These arguments are brought forward by the system not only when all the premises for the argument are accepted in the knowledge base: Carneades is in fact capable of “suggesting” incomplete arguments [3], thus highlighting critical aspects of the case which have not been taken into consideration by the judge (in the precedent case) or by the user (in the query).

As a result of their application to the ontology set, the LKIF-Rule and Clojure rule languages showed to be unsatisfactory in:

- identifying a border between semantic representation and syntactic modeling;
- importing ontologies: in the Carneades application no distinction is made between stated and inferred knowledge, and no feedback of new knowledge into the ontology reasoner is possible;
- providing basic legal deontics operators to represent obligation, violation, reparation and penalty;
- providing defeasibility logic operators defining the hierarchy between rules;
- modeling temporal dimensions to represent the three axes of enforceability, efficacy, applicability;
- assigning IDs for single parts of the rules, necessary to reach full isomorphism between the rule and the source legal document(s);
- qualifying rules with metadata such as author, data of creation of the rule, jurisdiction of the rule, etc.

#### **4.3. The Boundary Between Ontology and Rules.**

The critical point in importing ontologies, adding factors and writing rules was the design and management of information between the ontologies and the rules: some of the axioms already modeled in the ontologies, in fact, could better meet their potentialities if modeled as an LKIF or Clojure rules instead. The issue, anyway, should be solved with general criteria, since the two systems use different logics (description logic for OWL vs. first-order predicate logic for LKIF and Clojure). This suggests the distinction between static information (thesauri, taxonomies, administrative and procedural data) to be included in the ontology, and legal concepts (legal statuses, subsumptions, inclusion of a material circumstance into the scope of a norm) to be modeled as rules for argument evaluation.

To manage defeasibility of arguments (and thus rules) Carneades includes proof standards [6], which can in a way be interpreted as a kind of priority relation in defeasible logics [11]. The new Carneades 2.x also includes, in its Clojure-based rule system, the metadata block `<strict>` which allows to specify (through the values



true/false) if a `<scheme>` is either strict or defeasible. However, this block does not ensure full expressivity of the defeasible logics constituents, since it does not represent neither defeaters nor metarules.

## 5 A New Approach: LegalRuleML

The situation presented above is likely to present itself over and over again as long as modeling decisions (i.e. the introduction of elements of defeasible logics) are taken just for the purposes of a single application. In order to fully exploit its potentialities, AI&Law systems (such as legal argumentation systems) should instead rely on open and shared standards. The research community joined the efforts towards the definition of a standard for the syntax of legal rule extending LKIF-Rules with a modeling of temporal parameters, giving birth to the LKIF++ language [16]. Soon realizing that a standard in syntactic representation of norms would require a shared rule language to be built from scratch, the OASIS consortium started the development of a brand new syntax for legal rules, explicitly relying on the acquired standards in the underlying layers of the Semantic Web cake.

### 5.1. RuleML and LegalRuleML

LegalRuleML [18] is an extension of RuleML, an XML based language for the representation of legal rules using formal semantics [12]. LegalRuleML introduces features which are fundamental for modeling legal rules: isomorphism, defeasible logics, jurisdiction and authority, legal temporal parameters, legal deontics operators, qualifications, semantic of negation, behaviors. The language also allows to include elements and statements compliant with external ontologies. The syntax and structure of this language are a work in progress: therefore all tags, elements, attributes recalled here follow a syntax proposal presented mostly by Palmirani<sup>5</sup> in the TC, and some of them may not correspond to the definitive syntax of LegalRuleML<sup>6</sup>. The requirements expressed are nevertheless important for tackling some of the problems encountered using LKIF-Rules and Carneades.

### 5.2. Achieving isomorphism

Modelling the legal rules in the LegalRuleML language highlighted the potentialities of this tool in achieving isomorphism and representing defeasibility and temporal parameters. In the present proposal, the `<ruleInfo>` section introduces detailed information on the context of the rule. This rule-centric metadata approach favours the isomorphism with the legal text during the change management and the encapsulation of all information related to the rule in a unique XML node:

```
<lrml:ruleInfo id="ruleInfo2" appliesTo="#rule2">
  <lrml:sources id="sourceBlock2">
    <lrml:source element="#atom1" idRef="#art1341-com2"/>
    <lrml:source element="#atom2" idRef="#art1341-com1"/>
    <lrml:source element="#atom3" idRef="#art1341-com1"/>
  </lrml:sources>
</lrml:ruleInfo>
```

---

<sup>5</sup> General contents of the proposal can be found at <https://www.oasis-open.org/apps/org/workgroup/legalruleml/download.php/46379/1.2PRINCIPLES.004.doc>.

<sup>6</sup> Documents of the OASIS LegalRuleML TC are available at [https://www.oasis-open.org/committees/documents.php?wg\\_abbrev=legalruleml](https://www.oasis-open.org/committees/documents.php?wg_abbrev=legalruleml). The mailing list describing the work in progress can be browsed at <https://lists.oasis-open.org/archives/legalruleml/>.

```

        <lrml:source element="#atom4" idRef="#art1341com2"/>
    </lrml:sources>
    <lrml:strength iri="#dfsont;defeater"/>
    <lrml:jurisdiction iri="#jurisdictions;italy"/>
    <lrml:author idRef="#aut1"/>
    <lrml:times idRef="#t1"/>
    <lrml:creationDateTime idRef="#e1"/>
</lrml:ruleInfo>7

```

This section represents a context data container, since it introduces metadata which can be used in multiple circumstances to classify the rules. The bond between the text fragment, the legal rule and the author of the model is ensured by a list of `<sources>` for every element which constitutes the rule and by the element `<author/>`, linking the rule and its parts to specific individuals (a fragment of text and a person respectively), identified through an IRI. In this way it is possible to explicitly refer to the source documents of each part of a rule, also when a rule takes origin from multiple documental sources, allowing a clear distinction of which part of the rule comes from which document. At the same time, different rule authors are allowed to model the same text fragment in different ways, being always clear which author modelled which rules on a certain legal document.

### 5.3. Introducing Defeasible Logics and Temporal Parameters to the Rule

Inside the `<ruleInfo>` section, the element `<strength/>` defines the role of each rule in the defeasible logics dynamics (strict/defeasible/defeater). Priority relations are built through the `<Overrides>` element<sup>8</sup>, in the following form:

```

<Overrides id="ovr1">
  <Rule keyref="#rule_3"/>
  <Rule keyref="#rule_2"/>
</Overrides>9

```

The `<ruleInfo>` section contains also a `<times>` element, not indeed a normal attribute but rather a section introducing a whole different layer: it contains information on the time periods of the legal rule's coming into force, efficacy, application. The representation of temporal dimension of legal rules using three axes is a crucial addition towards the automatic management of legal rules in connection

<sup>7</sup> The proposal can be found at <http://www.oasis-open.org/apps/org/workgroup/legalruleml/download.php/45887/2.8isomorphism.001.doc>.

<sup>8</sup> This tag is developed jointly with RuleML (<http://ruleml.org/>) and in particular with the Defeasibility RuleML TG (<http://ruleml.org/1.0/defeasible.html>).

<sup>9</sup> This implementation of defeasibility should allow a better management of exceptions than in LKIF-Rule, where exceptions had to be made explicit in the rule syntax. If an exception presents itself in the form of two legal fragments not explicitly referring to each other but rather disposing opposing legal consequences (i.e. efficacy vs. inefficacy) it is possible to model these rules independently, and then create a priority relation reflecting the actual hierarchy between the two norms. Moreover, this solution allows a relative management of hierarchy, without the need to assign arbitrary "weight" values to each rule. The proposal can be found at <https://www.oasis-open.org/committees/download.php/46454/2.1.1defeasibility.006.doc>. Meaningful comments by TC member Tara Athan can be found at <http://www.oasis-open.org/apps/org/workgroup/legalruleml/download.php/45888/2.1defeasibility.002.002.doc>.

with their legally binding documents [16], and LegalRuleML allows to specify these time coordinates for each rule, starting from the identification of the relevant points in time through a list of <events> such as:

```
<lrml:event id="e2" value="1942-04-21T01:01:00.0Z"/>.
```

Events' IDS are recalled by the <timeBlock> element, which adds information on the event which occurs (start, end) and on the axis which is affected:

```
<lrml:timeBlock id="t1">
  <lrml:time start="#e2" refType="&lkif;#efficacy"/>
</lrml:timeBlock>10
```

## 6 Conclusions

The project presented here represents an effort towards the acquisition of an acknowledged standard for the rule layer of the semantic web layer cake, while at the same time trying to improve the state-of-the-art of legal knowledge representation by facing its main issues: the gap between document representation and rule modeling, and the need for a shared standard in the logic layer to represent legal reasoning. The Carneades Argumentation System has been used as a test field for the ontology set and the rules design, highlighting potentialities and issues of the approach.

Under these points of view, the approach proposed by the research represents a step towards the filling of that gap, relying on existing standards to achieve the isomorphism between legal document and rules; at the same time, the research defines the requirements for a reasoning engine to be capable of semantically managing knowledge coming from the ontology and applying legal rules to it. This engine will probably not be a closed one, embedded in some argumentation tool (as in Carneades): It should rather consist of a set of libraries to be implemented into existing engines in order to introduce a complete management of defeasibility and a standard language for interaction between these rules and OWL-encoded knowledge. The intention, in the upcoming research on this behalf, is to rely on a Drools<sup>11</sup> application under construction by CIRSFD and on NICTA's SPINDle<sup>12</sup> [13].

## Acknowledgements

A particular acknowledgement is due to Tom Gordon, provider of meaningful research inputs during my stage in Berlin, FOKUS, in the last autumn. Thanks also to Adam Wyner who gave me important feedbacks for improving the readability of the paper and also, as a legal domain expert, fruitful comments on the ontology modeling choices. Thanks go also to Harold Boley and Tara Athan of the RuleML Initiative for his helpful hints. Last but not least, I am grateful to my tutor and mentor Monica Palmirani, for her guidance in this challenging research.

---

<sup>10</sup> The proposal can be found at <http://markmail.org/message/oy34tkzr3r2ldhz?q=temporal+list.org%2Eoasis-open+list.org%2Exml+list.org%2Eebxml>.

<sup>11</sup> Drools ([www.drools.org](http://www.drools.org)) is a production rule system based on the Rete algorithm.

<sup>12</sup> SPINDle is an open source, Java-based defeasible logic reasoner which conducts efficient and scalable reasoning on defeasible logic theories.

## References

1. Boella G., Governatori G., Rotolo A., van der Torre L., A Logical Understanding of Legal Interpretation. In: KR 2010.
2. Boer, A., Radboud, W., Vitali, F., MetaLex XML and the Legal Knowledge Interchange Format. In: Casanovas, P., Sartor, G., Casellas, N., Rubino, R. (eds.), *Computable Models of the Law*, Springer, Heidelberg (2008), pp. 21-41.
3. Ceci, M., Gordon, T.: Browsing Case-Law: An Application of the Carneades Argumentation System, RuleML Challenge 2012 (under submission).
4. Gangemi, A.: Design Patterns for Legal Ontology Construction. In: Noriega, P., Bourcier, D., Galindo, F. (eds.), *Trends in Legal Knowledge. The Semantic Web and the Regulation of Electronic Social Systems*, pp. 171-191. European Press Academic Publishing (2007)
5. Gordon, T., Walton, D.: The Carneades Argumentation Framework: using presumptions and exceptions to model critical questions. In: Dunne, P.E.: *Computational Models of Argument. Proceedings of COMMA 2006: 1st International Conference on Computational Models of Argument*, The University of Liverpool, UK, 11th - 12th September 2006. IOS Press, Amsterdam (2006)
6. Gordon, T., Prakken, H., Walton, D.: The Carneades model of argument and burden of proof. In: *Artificial Intelligence*, Vol.171 (2007), No.10-15, pp.875-896.
7. Gordon, T.: Constructing Legal Arguments with Rules in the Legal Knowledge Interchange Format (LKIF). In: Casanovas, P.: *Computable models of the law: Languages, dialogues, games, ontologies (Lecture Notes in Artificial Intelligence 4884)*, pp. 162-184. Springer, Heidelberg (2008)
8. Gordon, T.: Combining Rules and Ontologies with Carneades. In: *Proceedings of the 5th International RuleML2011@BRF Challenge, CEUR Workshop Proceedings (2011)*, 103-110.
9. Gordon, T., Governatori, G., Rotolo, A., Rules and Norms: Requirements for Rule Interchange Languages in the Legal Domain. In: *Rule Interchange and Applications, International Symposium, RuleML 2009, BERLIN*, Springer pp. 282 - 296 (2009).
10. Gordon, T., Walton, D., Legal reasoning with argumentation schemes. In: *Proceedings of the Twelfth International Conference on Artificial Intelligence and Law*, ACM Press, New York, 2009, pp. 137-146.
11. Governatori, G.: On the relationship between Carneades and Defeasible logic. In: *ICAAIL '11 Proceedings of the 13th International Conference on Artificial Intelligence and Law*, ACM New York (2011)
12. Lee, J.K., Sohn, M.M.: The eXtensible Rule Markup Language, *Communications of the ACM*, Volume 46, Issue 5, pp. 59-64 (2003)
13. Lam, H.P., Governatori, G.: The making of SPINdle. In: Governatori, G., Hall, J., Paschke, A. (eds.): *RuleML 2009. LNCS 5858*, 315-322. Springer, Berlin (2009)
14. Palmirani, M., Brighi, R., Model Regularity of Legal Language in Active Modifications. In: *LNCS 6237/2010*, pp. 54-73.
15. Palmirani, M., Contissa, G., Rubino, R.: Fill the Gap in the Legal Knowledge Modelling. In: *Proceedings of RuleML 2009*, pp. 305-314.
16. Palmirani, M., Governatori, G., Contissa, G.: Temporal Dimensions in Rules Modelling. In: *JURIX 2010*, pp.159-162.
17. Palmirani, M., Ceci, M.: Ontology framework for judgement modelling. In: *AICOL 2011 Proceedings*. Springer, 2012 (under publication).
18. Palmirani, M., Governatori, G., Rotolo, A., Tabet, S., Boley, H., Paschke, A.: LegalRuleML: XML-Based Rules and Norms. In: *RuleML 2011*, pp. 298-312.

# Checking Compliance in European Tender Documents through Ontologies and Rules

Isabella Distinto

CIRSFID, University of Bologna  
isabella.distinto@unibo.it

**Abstract.** In this paper we present an overview of the PhD thesis, which aims to show the feasibility of a legal knowledge modeling based on the definitions included into legal texts using Semantic Web techniques in order to check compliance of tender documents to EU Directives. We show a hybrid approach, i.e. a theory and a system that combine the use of Akoma-Ntoso standard to describe legal texts, OWL 2.0 for modeling legal concepts and the emerging LegalRuleML standard for providing a rule-based extension of legal knowledge representation on the top of ontologies.

## 1 Introduction

The world of public procurements is a field of great significance for many reasons. It is one of the main economic activities of Governments, since through procurements are acquired services and goods for functioning of the *res publica* and for meeting needs of citizens' communities. This activity represents also a significant business opportunity for economic operators because public procurement in the EU accounts for 17% of EU GDP. In this scenario, the regular conduct of tendering processes plays a key role, in terms of transparency and fairness in bidding competitions, reduction of litigation-related costs, but also corruption fighting strategies.

A tender situation can be considered as a complex case, in which agents, documents and processes play an interactive role. Norms on public procurements contracts regulate either the content of documents and the conduct of processes. Through the representation of knowledge contained in normative texts it is possible to define the abstract, lawfully type of those documents or processes. Each single document or applied procedure is an instance of the abstract type defined by normative provisions. With respect to documents involved in processes related to European public procurements, both the abstract type and the actual token can be represented with languages of Semantic Web, in order to describe all the layers perspective of this kind of texts: structure (as organization of the texts), metadata (as any information that was not approved by the authority in the document), ontology (as any legal concept called from text that need a modeling), and finally legal knowledge representation (as modeling of rules). Through this articulate definition of legal knowledge, as showed in [16], it is possible to bridge the gap between legal text description of procurement documents and legal knowledge representation that on these same texts start.

This structured knowledge base can be exploited by a system that, making use of an inference engine, may enable automated reasoning useful to check compliance of tender documents and processes involved (ABox) with legal concepts and rules, as norm expressions (TBox).

We define *compliance* the conformity of a legal document and/or of a relevant behavior with the normative prescriptions regulating them. The difference between the prescription stated in legal texts and its actual application is a *compliance failure*. Through *compliance check* is possible to detect this failure.

The ultimate goal of the research is to produce a set of pilot cases in order to show the feasibility of such kind of system that would facilitate the monitoring by the administrations on the regular conduct of procedures for the award of public contracts.

## 2 Background and approaches

XML-based standards are widely adopted by Governments of several countries for structuring legal documents. Some Legal XML standard (such as Akoma-Ntoso standard [22]) envisages the connection of structured legal texts with ontologies. Upon this connection, ontologies can be intended as a common vocabulary for shared understanding of terms and a powerful tool to express the legal concepts in a formal and unambiguous way. In that way it is possible to have a link between legal language or terms and legal concept-based representation, as defined by the same text.

We distinguish between two types of legal knowledge, which can be represented through languages of Semantic Web [1]. The first is static knowledge, namely any concept that is used in legal domain as mean of classification of facts (or ontological instances), by applying certain rules. We think that the role of ontologies is to model all static and definitional aspects of these rules-contained knowledge components. Indeed, legal systems are built upon concepts, whose semantics is defined by the same text in which they are invoked and through which are enabled inferences, broadly coincident with teleological purposes of norms. These kinds of knowledge components are also called “intermediate legal concepts” [20], since through them are represented preconditions and legal consequences, in the same way of rules, or as “inferential links” [18].

The second type of legal knowledge is the dynamic one. The ontological level is used to classify instances (facts) into ontological classes (abstract types). Such classified facts are then subject to certain provisions. This is the point where ontologies and rules meet. Indeed, even if in many cases rules applied to classified instances may be represented through first order logic (necessary condition), in many others cases monotonic reasoning based on Open World Assumptions (OWA) is not suitable for reasoning with knowledge-base inconsistencies or conflicts among rules. In legal domain, accordingly to a set of information an argument can be considered true, but if the set under consideration is enriched, maybe the truth of the argument has to be revised. This type of situation is well known as defeasibility and typically is required non-monotonic logic with Closed World Assumptions (CWA) to represent it. So, our hypothesis is based on the assumption that it is possible to recognize rules that define

ontologically legal concepts (in terms of proof-of-concepts) and rules that are applied to legal concepts.

Over the last decades, many rules interchange languages has been produced. The most relevant are undoubtedly RuleML [3], SVBR [15], SWRL [13], RIF [14] and LKIF [6, 7]. As noted in [10], these languages are not suited to represent rules of legal domain. Both SWRL and RIF are not able to meet the requirements of legal knowledge representation, such as the isomorphism and the defeasibility, which can not be represented with a series of chained implications in first-order logic. It's also necessary to emphasize that the same type of rules can be implemented in ontologies through the additional features (Property Chain Inclusion) of OWL 2.0. Also SVBR suffer the same kind of limitation. Instead, LKIF, although designed specifically for modeling legal knowledge, is based on the ISO Common Logic standard that does not look as a candidate standard to be widely adopted within the World Wide Web community. RuleML has been designed to become the markup language for the Semantic Web Rules: it is based on XML, is suitable to integrate inferences from Web ontologies, is extensible since it is built with a modularity approach. However, it lacks of extensions required by the legal knowledge. Thus, the extension of RuleML with key features of rules extracted from legal knowledge is now the goal of LegalRuleML [17] emerging standard. This last language, indeed, allows to meet the requirements of isomorphism (i.e. the one-to-one correspondence between the atomic rule and the fragment of natural language text in which the rule is expressed), the specification of normative effects (such as deontic, qualificatory or potestative, just to name a few), the dynamic feature of norm and normative effects and the reification (namely the fact that rules are like objects with relevant properties, such as jurisdiction, authority and temporal constraints). For these reasons, we chose to represent rules in LegalRuleML.

### 3 Related works

Compliance checking is a field that is typically related to business domain. There is a vast literature on various techniques developed for supporting compliance checking of business processes to regulations. Just to name a few, Governatori et al. have been proposed approaches based on multi-modal logic [12] and on a specialized logic, namely Formal Contract Language (FCL) [11] to formalize contracts' rules in order to manage the compliance of contractual relationships in business processes. Many other works are based on the formal representation of business contracts and their performance [5, 8]. An approach based on the use of an ontology of compliance requirements for service processes is described in [21]: however this work represent an exception in field of business process compliance checking.

About the legal domain, under the NEURONA project, has been developed a legal ontology for the representation of data protection knowledge for reasoning about the correctness of the information regarding personal data files and the correctness of the measures of protection applied to these data files [4]. Some studies has been conducted on the complexity and limitations of reasoning on EU Directives with OWL [9].

## 4 Pilot Case Scenario

A pilot case scenario is used for explaining the described methodology. It is based on the modeling of a complex norm extracted from the European Directive 2004/18 on the coordination of procedures for the award of public works contracts, public supply contracts and public service contracts, namely the Art. 17. With this example, we intend to show what are both the limits and the role of ontologies versus rules and also that LegalRuleML allows to model articulated types of rules, which may be encountered in modeling of legal knowledge.

The article 17 of Directive 2004/18 states: “*Without prejudice to the application of Article 3, this Directive shall not apply to service concessions as defined in Article 1(4)*”. In this norm are contained: a reference to a the legal concept of *service concession* as defined by the same directive (in Art. 1 (4)); an exception to the application of the Directive (“*shall not applied*”) and an exception of exception about this rule in all the cases covered by the Art. 3 recalled by the Art. 17 (“*Without prejudice to the application of Article 3*”). In legal domain this mechanism (to recall another rule) is called *meta-rule*, since it is a rule about (the activation of) another rule, namely the Art. 3. This article states that if a contracting authority grants special or exclusive rights to carry out a public service activity to an entity other than such a contracting authority, the act by which that right is granted shall provide that, in respect of the supply contracts which it awards to third parties as part of its activities, the entity concerned must comply with the principle of non-discrimination on the basis of nationality<sup>1</sup>. Section 4.1 describes our proposal about modeling of legal concepts (static knowledge); Section 4.2 describes how it is possible to model complex metadata on the top of legal concepts with a syntax *look-like* LegalRuleML<sup>2</sup>.

### 4.1 Proof of legal concepts as basis of proof of legal compliance

As outlined in Section 1, legal concepts are basically identified by terms incorporating inferential links between preconditions and legal consequences, in the same way of rules. So, the role to express this kind of rules can be delegated to ontologies and in that way the proof of legal concepts can serve as basis of proof of legal compliance, since every individual belonging to a class of a specific legal concept have to comply with the restrictions on this class. For the purpose of the presented pilot case, we've developed an ontology<sup>3</sup> for describing particularly the class of *Service Concession* as defined in the Art. 1 (4) of the Directive

---

<sup>1</sup> Article 3. *Granting of special or exclusive rights: non-discrimination clause - Where a contracting authority grants special or exclusive rights to carry out a public service activity to an entity other than such a contracting authority, the act by which that right is granted shall provide that, in respect of the supply contracts which it awards to third parties as part of its activities, the entity concerned must comply with the principle of non-discrimination on the basis of nationality.*

<sup>2</sup> Documents of the OASIS LegalRuleML TC are available at [https://www.oasis-open.org/committees/documents.php?wg\\_abbrev=legalruleml](https://www.oasis-open.org/committees/documents.php?wg_abbrev=legalruleml). The mailing list describing the work in progress can be browsed at <https://lists.oasis-open.org/archives/legalruleml/>.

<sup>3</sup> Available at <http://codexml.cirsfid.unibo.it>



2004/18/ec and the class of Act of grant to entities others than authority as defined in the Art. 3 of the Directive 2004/18/ec. The main commitments of this ontology are to check if an individual of type Public Procurement Contract is also of type Service Concession, and which individual belongs to the class of Service Concession, but is also of type Act of grant to entities others than authority. In that way the rules are applied upon the materializations made by the ontology on legal concepts. According to the definition of Art. 1 (4) of the Directive 2004/18/ec the Class of Service Concession is modeled as SubClassOf a Public procurement contract, which has its object the right to exploitation the service or both the right to exploitation the service and the payment:

```
<EquivalentClasses>
  <Class IRI="#ServiceConcession"/>
  <ObjectIntersectionOf>
    <Class IRI="#PublicProcurementContract"/>
    <ObjectSomeValuesFrom>
      <ObjectProperty IRI="#hasObjectOfContract"/>
      <ObjectUnionOf>
        <Class IRI="#RightOfExploitationTheService"/>
        <ObjectIntersectionOf>
          <Class IRI="#Payment"/>
          <Class IRI="#RightOfExploitationTheService"/>
        </ObjectIntersectionOf>
      </ObjectUnionOf>
    </ObjectSomeValuesFrom>
  </ObjectIntersectionOf>
</EquivalentClasses>
```

According to the definition of Art. 3 of the Directive 2004/18/ec the Class of Act of grant to entities others than authority is modeled as SubClassOf a LegalDocument, by which is granted a right (special or exclusive) to a legal person that is not a Contracting authority:

```
<EquivalentClasses>
  <Class IRI="#ActOfGrantToEntityOtherThanAuthority"/>
  <ObjectIntersectionOf>
    <Class IRI="#LegalDocument"/>
    <ObjectSomeValuesFrom>
      <ObjectProperty IRI="#byWhichIsGrantedRightTo"/>
      <ObjectIntersectionOf>
        <Class IRI="#LegalPerson"/>
        <ObjectComplementOf>
          <Class IRI="#ContractingAuthority"/>
        </ObjectComplementOf>
      </ObjectIntersectionOf>
    </ObjectSomeValuesFrom>
  </ObjectIntersectionOf>
</EquivalentClasses>
```

## 4.2 Rules in LegalRuleML

We have modelled three rules<sup>4</sup> using a very preliminary syntax of LegalRuleML<sup>5</sup> for testing the emerging standard and providing useful feedback to the OASIS TC:

- the rule1 is a defeasible rule and involves this fragment of the Art. 17:

*“this Directive shall not apply to service concessions as defined in Article 1(4).”*

The formalization of this text is the following:

```
if
  X is a member of the OWL class public-procurement-contract
  X is a member of the OWL class called service-concession
  X enters in the definition of the Article 1(4)
then
  the Directive2004/18/EC shall not apply
• the rule2 has strength defeater and is also a meta-rule, since it activates the
  rule3:
  “Without prejudice to the application of Article 3”
  if
    X is a member of the OWL class public-procurement-contract
    X is a member of the OWL class called service-concession
  then
    the Directive2004/18/EC shall apply
    the rule3 is activated
• the rule3 is related to the content of Article 3;
  if
    X is a member of the OWL class called service-concession
    X is a member of the OWL class act-of-grant-to-entity-other-
    than-authority
  then
    X shall respect the principle of
      Non-discriminationClauseOnTheBasisOfNationality
    X shall respect of supply contracts which it awards to third
    parties as part of its activities
```

We concentrate our attention on the connection between rule2 and rule3 using `lrml:typeRule=&legalRuleML;metaRule` as attribute in the rule2 and the `<Ind iri="#rule3">rule3</Ind>`:

```
<Rule material="no" id="rule2" lrml:typeRule="&legalRuleML;metaRule">
  <!-- Art.17 Without prejudice to the application of Article 3 -->
  <if id="rule2-body">
    <And>
      <Atom id="rule2-atom1">
        <Rel iri="&lkif;#member_of">is a member of the public-
        procurement-contract</Rel>
        <Var type="&lkif;#public-procurement-contract">X</Var>
      </Atom>
      <Atom id="rule2-atom2">
        <Rel iri="&lkif;#member_of">is a member of class service-
        concession</Rel>
        <Var type="&lkif;#service-concession">X</Var>
```

<sup>4</sup> “Article 17 Service concessions - Without prejudice to the application of Article 3, this Directive shall not apply to service concessions as defined in Article 1(4).”

<sup>5</sup> We take in consideration the version available at the date of the paper submission: <http://www.oasis-open.org/apps/org/workgroup/legalruleml/download.php/45888/2.1defeasibility.002.002.doc>

```

        </Atom>
      </And>
    </if>
    <then id="rule2-head">
      <And>
        <Atom id="rule2-atom2">
          <Rel iri="&lkif;#shallApply">shall apply</Rel>
          <Ind iri="&DIRECTIVE2004_18_ec">Directive2004_18_</Ind>
        </Atom>
        <Atom id="rule2-atom2">
          <Rel iri="&lkif;#shallApply">shall apply</Rel>
          <Ind iri="#rule3">rule3</Ind>
        </Atom>
      </And>
    </then>
  </Rule>

```

## 5 Ongoing and future perspectives

In this paper we have shown how with an hybrid approach that integrates ontologies in OWL 2.0 (for contents related to legal concepts) and rules in a preliminary syntax of LegalRuleML (for the dynamic legal knowledge), it is possible to express in a computable formalism also complex norms, maintaining the isomorphic [2] relation with the text and with a clear distinction between roles of ontologies and rules.

Up to now, an ontology of European public procurement notices has been developed from scratch in OWL 2.0, in order to represent concepts related to these types of tender documents and processes involved. The ontology is based on both a top-down and a bottom-up approach. Indeed it represents concepts extracted from authoritative sources (Directive 2004/17/EC; Directive 2004/18/EC; etc.) compared with natural language patterns derived from standard forms in use for these tender documents<sup>6</sup>. The ontology is based on a modular approach and allows, for example, inferences for classifying a contract notice as contract notice with European relevance (i.e. upon EU threshold) or not. Another module of the ontology has the main commitment to classify contract notices covered by the Government Procurement Agreement: through this information it is possible to find out whether a call is open to the participation of economic operators from countries that are not EU members.

RDFa assertions in the XML enable the connection between the structural part of legal texts and the classes of the ontology so developed. Finally upon the ontological level, are represented rules applied to defined ontological classes as well as to materializations inferred through reasoners such as Hermit or Pellet.

We think that future perspectives on this work are strictly related to the investigation of the topic of legal ontology evolution, in order to meet the need to automatically detect changes in legal concepts and allow for a sustainable evolutionary system approach.

---

<sup>6</sup> These forms are available at [http://simap.europa.eu/buyer/forms-standard/index\\_en.htm](http://simap.europa.eu/buyer/forms-standard/index_en.htm)

## Acknowledgements

A particular acknowledgement is due to Enrico Motta, provider of meaningful research inputs during my stage at KMI - The Open University (UK), in the last spring. Thanks go also to the TC of the RuleML Initiative and to the LegalRuleML TC for their helpful hints. I am grateful to my tutor and mentor Monica Palmirani, for her guidance in this challenging research.

## References

1. Antoniou G., and Bikakis, A., DR-Prolog: A System for Defeasible Reasoning with Rules and Ontologies on the Semantic Web, *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 2, p. 233-245, 2007.
2. Bench-Capon, T. and Coenen, F., Isomorphism and legal knowledge based systems. *Artificial Intelligence and Law*, 1(1):65–86, 1992.
3. Boley, H., Tabet, S. and Wagner G., Design rationale for RuleML: A markup language for Semantic Web rules. In I. F. Cruz, S. Decker, J. Euzenat, and D. L. McGuinness, editors, *Proc. SWWS'01, The first Semantic Web Working Symposium*, pages 381–401, 2001.
4. Casellas, N., Nieto, J., Meroño, A., Roig, A., Torralba, S., Reyes, M., Casanovas, P., Ontological Semantics for Data Privacy Compliance: The NEURONA Project., In *AAAI Spring Symposium: Intelligent Information Privacy Management*, 2010.
5. Desai, N., Narendra, N. C. & Singh, M. P., Checking correctness of business contracts via commitments, in *Proc. AAMAS*, 2008.
6. ESTRELLA Project. The legal knowledge interchange format (LKIF). Deliverable 4.3, European Commission, 2008.
7. ESTRELLA Project. The reference LKIF inference engine. Deliverable 4.3, European Commission, 2008.
8. Farrell, A. D. H., Sergot, M. J., Sallè, M., Bartolini, C., Using the event calculus for tracking the normative state of contracts, *International Journal of Cooperative Information Systems*, 2005.
9. Gangemi A., Sagri M. T., Tiscornia D., A Constructive Framework for Legal Ontologies, in Benjamins, R., Breuker, J., Casanovas, P., Gangemi, A. (eds.): *Law and the Semantic Web*, LNCS, Springer, 2005
10. Gordon, T. F., Governatori, G., Rotolo, A., Rules and Norms: Requirements for Rule Interchange Languages in the Legal Domain, in: *Rule Interchange and Applications*, International Symposium, RuleML 2009, Springer, 2009.
11. Governatori, G. and Milosevic, Z. Dealing with contract violations: formalism and domain specific language. In *Proceedings of the Conference on Enterprise Computing EDOC 2005*, IEEE Press, 2005.
12. Governatori, G., Milosevic, Z., Sadiq, S.: Compliance checking between business processes and business contracts. In: *Proc. EDOC 2006*, pp. 221–232. IEEE, Los Alamitos (2006)
13. Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosz, B., & Dean, M. (May 2004). SWRL: A semantic web rule language combining OWL and RuleML. Available from <http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/>.

14. Kifer, M., Rule Interchange Format: The Framework, Proceedings of the 2nd International Conference on Web Reasoning and Rule Systems, October 31-November 01, 2008, Karlsruhe, Germany.
15. OMG: Semantics of business vocabulary and business rules (SBVR). <http://www.businessrulesgroup.org/sbvr.shtml>, 2008.
16. Palmirani, M., Contissa, G., Rubino, R., Fill the Gap in the Legal Knowledge Modelling, Proceedings of the 2009 International Symposium on Rule Interchange and Applications, November 05-07, 2009, Las Vegas, Nevada.
17. Palmirani, M., Governatori, G., Rotolo, A., Tabet, S., Boley, H., Paschke, A., LegalRuleML: XML-Based Rules and Norms.; in RuleML America (2011), 298-312.
18. Ross, A., Tû-Tû, Harvard Law Review, vol. 70, pp. 812-825.
19. Sartor, G., Legal Reasoning and Normative Conflicts, in Reasoning with Inconsistency, 1991.
20. Sartor, G., Understanding and Applying Legal Concepts: An Inquiry on Inferential Meaning. In: Concepts in Law. Ed. by Jaap C. Hage and Dietmar von der Pfordten. Springer, 2009.
21. Schmidt, R., Bartsch, C. and Oberhauser, R., Ontology-Based Representation of Compliance Requirements for Service Processes. In Proceedings of the Workshop on Semantic Business Process and Product Lifecycle Management, 2007.
22. Vitali, F.: Akoma Ntoso Release Notes, <http://www.akomantoso.org> (accessed August 20, 2009).

# Rule Based Business Process Compliance

Guido Governatori, Sidney Shek

NICTA, Australia

**Abstract.** In this paper we report on the development and evaluation of a business process compliance checker, based on the compliance-by-design methodology proposed by Governatori and Sadiq [9].

For a screencast see [http://www.youtube.com/watch?v=gFmDQJNai\\_4](http://www.youtube.com/watch?v=gFmDQJNai_4)

## 1 Introduction

Regulatory compliance is the set of activities an enterprise does to ensure that its core business does not violate relevant regulations, in the jurisdictions in which the business is situated, governing the (industry) sectors where the enterprise operates.

The activities an organisation does to achieve its business objectives can be understood as business processes, and consequently they can be represented by business process models. On the other hand a normative document (e.g., a code, a bill, an act) can be understood as a set of clauses, and these clauses can be represented in an appropriate formal language. Based on this [5] proposed that *business process compliance* is a relationship between the formal representation of a process model and the formal representation of a relevant regulation.

To gain compliance different strategies can be devised. [16] classifies approaches to compliance as *detective*, *corrective* and *preventative*.

*Detective measures* are intended to identify “after-the-fact” un-compliant situations. There are two main approaches: (a) *retrospective reporting* through manual audits by consultants or through IT forensics and Business Intelligence tools; (b) *automated detections* generating audit reports against hard-coded checks performed on the requisite system. Unlike the first approach, automated detection reduces the assessment time and consequently also the time of un-compliance remediation/mitigation.

*Corrective measures* are intended to limit the extent of any consequence caused by un-compliant situations. For example, situations that can arise from the introduction of a new norm impacting upon the business, to the organisation coming under surveillance and scrutiny by a control authority or to an enforceable undertaking.

The two approaches above suffer from lack of *sustainability*, caused by the extreme interest of companies in continuous improvements of the quality of services, and for changing legislations and compliance requirements. Indeed, even with automated detection means, the hard coded checking of repositories can quickly grow to a very large scale making it extremely difficult to evolve and maintain. To obviate these problem [17,13] propose a *preventative focus* based on the idea of *compliance-by-design*.

The key aspect of the compliance-by-design methodology is to supplement business process models with additional information to ensure that a business process is compliant with relevant normative frameworks before the deployment of the process itself.

## 2 BPCC Architecture

In this section we first introduce the architecture of BPCC, a business process compliance checker based on the business process compliance methodology proposed by Governatori and Sadiq [9].

As we have already discussed to check whether a business process is compliant with a relevant regulation, we need an annotated business process model and the formal representation of the regulation. The annotations are attached to the tasks of the process, and it can be used to record the data, resources and other information related to the single tasks in a process.

For the formal representation of the regulation we use FCL [4,8]. FCL is a simple, efficient, flexible rule based logic. FCL has been obtained from the combination of defeasible logic (for the efficient and natural treatment of exceptions, which are a common feature in normative reasoning) [1] and a deontic logic of violations [6]. In FCL a norm is represented by a rule

$$a_1, \dots, a_n \Rightarrow c$$

Where  $a_1, \dots, a_n$  are the conditions of applicability of the norm/rule and  $c$  is the *normative effect* of the norm/rule. FCL distinguishes two normative effects: the first is that of introducing a definition for a new term. For example the rule

$$customer(x), spending(x) > 1000 \Rightarrow premium\_customer(x)$$

specifies that, typically, a premium customer is a customer who has spent over 1000 dollars. The second normative effect is that of triggering obligations and other deontic notions. The deontic notions covered by FCL are obligations<sup>1</sup>, permissions, and reparation chains. For obligations FCL supports both maintenance obligations and achievement obligations, and for achievement obligations both pre-emptive and non-pre-emptive obligations (see [8] for full details). A reparation chain is an expression  $O_1 c_1 \otimes O_2 c \otimes \dots \otimes O_n c_n$ , where each  $O_i$  is an obligation, and each  $c_i$  is the content of the obligation (modelled by a literal). The meaning of a reparation chain is that we have that  $c_1$  is obligatory, but if the obligation of  $c_1$  is violated, i.e., we have  $\neg c_1$ , then the violation is compensated by  $c_2$  (which is then obligatory). But if even  $O_2 c_2$  is violated, then this violation is compensated by  $c_3$  which, after the violation of  $c_2$ , becomes obligatory, and so on.

It is worth noticing that FCL allows deontic expression (but not reparation chains) to appear in the body of rules, thus we can have rules like:

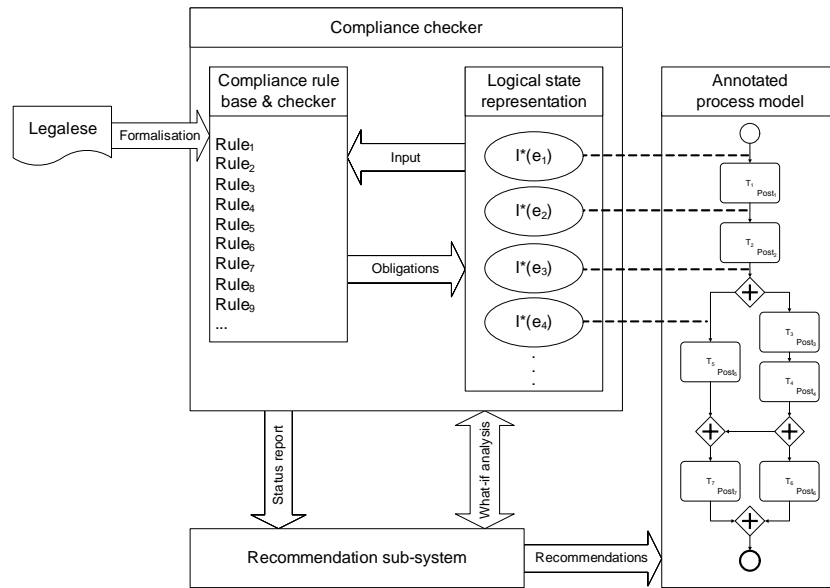
$$restaurant, [P]sell\_alcohol \Rightarrow [OM]show\_license \otimes [OAPNP]pay\_fine.$$

The rule above means that if a restaurant has a license to sell alcohol (i.e, it is permitted to sell it,  $[P]sell\_alcohol$ ), then it has a maintenance obligation to expose the license ( $[OM]show\_license$ ), if it does not then it has to pay the fine ( $[OAPNP]pay\_fine$ ). The obligation to pay the fine is non-pre-emptive (this means it cannot be paid before the violation). For full description of FCL and its feature see [4,8].

<sup>1</sup> Note the obligations allow us to capture prohibitions; a prohibition is an obligation plus negation, for example the prohibition to smoke can be understood as the obligation not to smoke.

Finally, FCL is agnostic about the nature of the literals it uses. They can represent tasks (activities executed in a process) or propositions representing state variables.

Compliance is not just about the tasks to be executed in a process but also on what the tasks do, the way they change the data and the state of artifacts related to the process, and the resources linked to the process. Accordingly, process models must be enriched with such information. [17] proposes to enrich process models with semantic annotations. Each task in a process model can have attached to it a set of semantic annotations. In our approach the semantic annotations are literals in the language of FCL, representing the effects of the tasks. The approach can be used to model business process data compliance [10]



**Fig. 1.** Architecture of BPCC

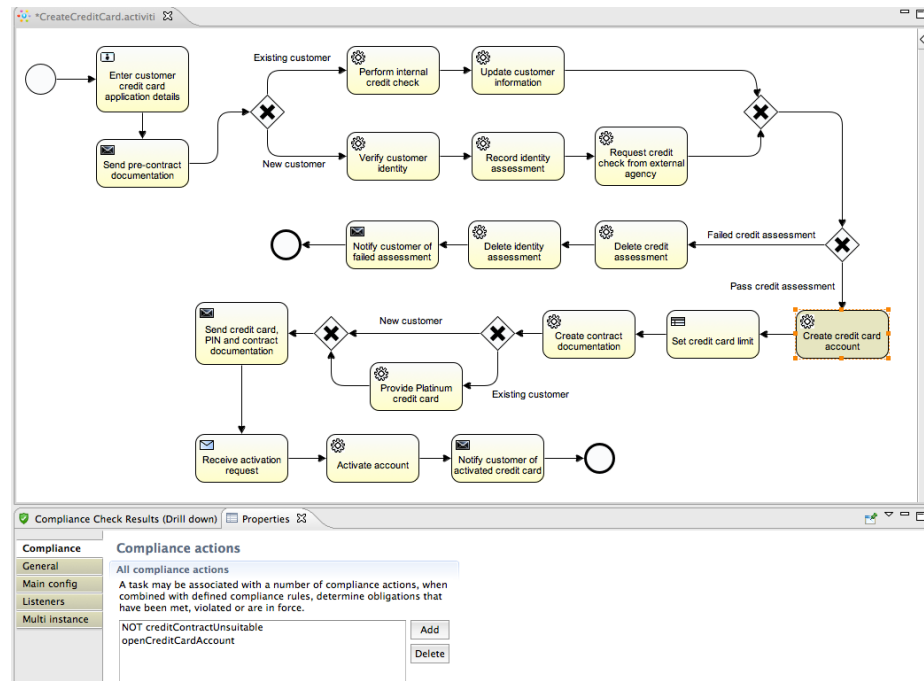
Figure 1 depicts the architecture of BPCC. Given an annotated process and the formalisation of the relevant regulation, we can use the algorithm propose in [7,8] to determine whether the annotated process model is compliant. The process runs as follows:

- Generate an execution trace of the process.
- Traverse the trace:
  - for each task in the trace, cumulate the effects of the task using an update semantics (i.e., if an effect in the current task conflicts with previous annotation, update using the effects of the current tasks).
  - use the set of cumulated effects to determine which obligations enter into force at the current tasks. This is done by a call to an FCL reasoner.
  - add the obligations obtained from the previous step to the set of obligations carried over from the previous task.



- determine which obligations have been fulfilled, violated, or are pending; and if there are violated obligation check whether they have been compensated.
- repeat for all traces.

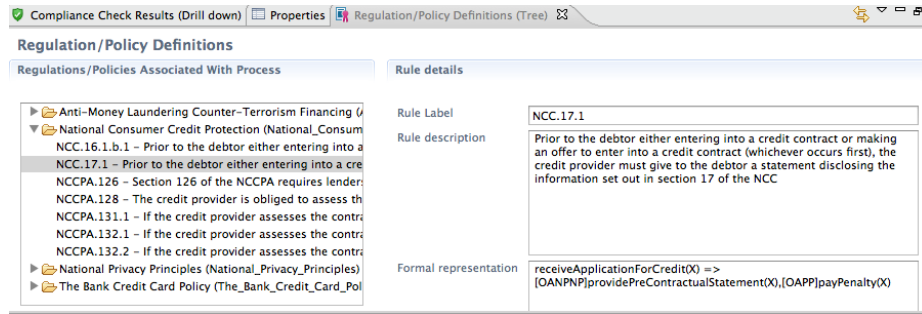
A process is compliant if and only if all traces are compliant (all obligations have been fulfilled or if violated they have been compensated). A process is weakly compliant if there is at least one trace that is compliant.



**Fig. 2.** An Opening Credit Card Account Process with Annotations in BPCC

### 3 Implementation and Evaluation

BPCC is implemented on top of Eclipse. For the representation of process models, it uses the Eclipse Activiti BPMN 2.0 plugin, extended with features to allow users to add semantic annotations to the tasks in the process model. BPCC is process model agnostic, this means that while the current implementation is based on BPMN all BPCC needs is to have a description of the process and the annotations for each task. A module of BPCC take the description of the process and generates the execution traces corresponding to the process. After the traces are generated, it implements the algorithm outlined in the previous section, where it uses the SPINdle rule engine [12] for the evaluation of the FCL rules. In case a process is not compliant (or if it is only weakly compliant) BPCC

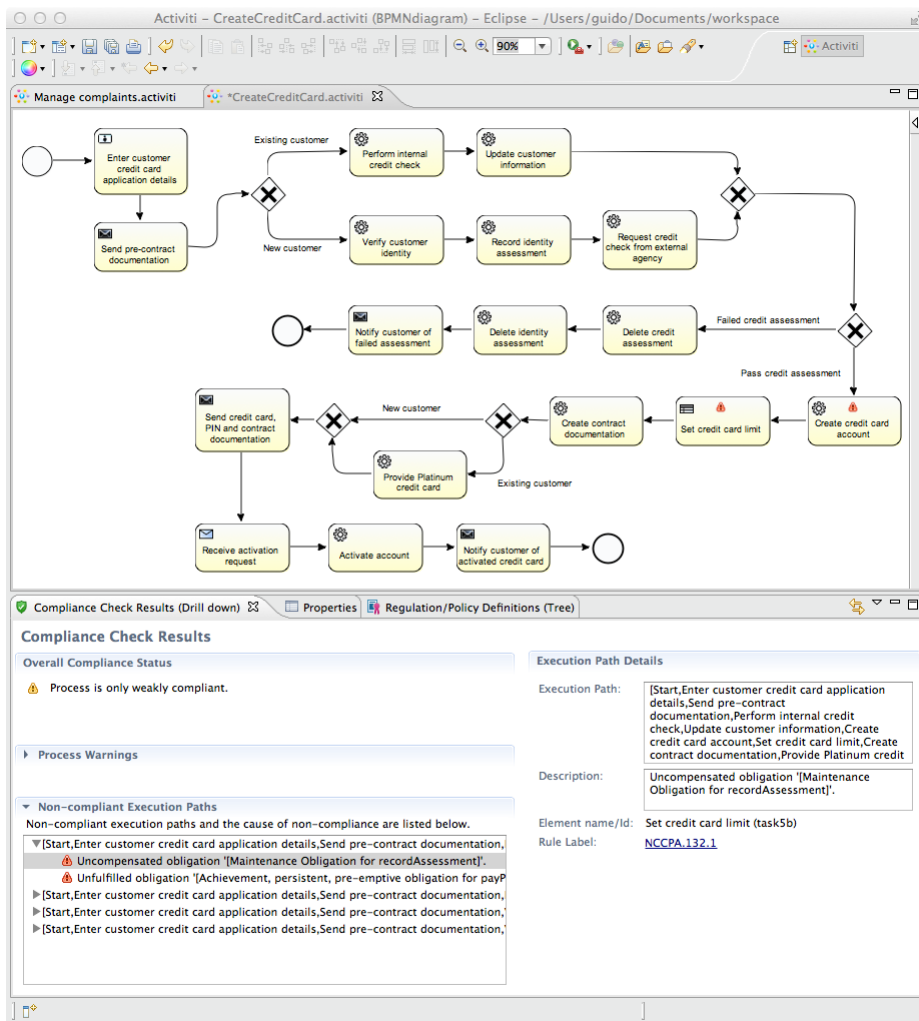


**Fig. 3.** Regulations Relevant to the Opening Credit Card Process

reports the traces, tasks, rules and obligations involved in the non compliance issues (see Figure 4).

BPCC was tested against an 2012 Australian Telecommunications Customers Protection Code (C628-2012). The code is effective from September 1st 2012. The code requires telecommunication operators to provide annual attestation of compliance with the code starting from April 1st 2013. The evaluation was carried out in May-June 2012. Specifically, the section of the code on complaint handling has been manually mapped to FCL. The section of the code contains approximately 100 commas, in addition to approximately 120 terms given in the Definitions and Interpretation section of the code. The mapping resulted in 176 FCL rules, containing 223 FCL (atomic) propositions, and 7 instances of the superiority relation. Of the 176 rules 33 were used to capture definitions of terms used in the remaining rules. Mapping the section of the code required all features of FCL: all types of obligations apart punctual obligations were used, reparation chains, permissions, defeasibility to easily capture exceptions, and obligations and permissions in the body of rules.

The evaluation was carried over in cooperation with an industry partner subject to the code. The industry partner did not have formalised business processes. Thus, we worked with domain experts from the industry partner (who had not been previously exposed to BPM technology, but who were familiar with the industry code) to draw process models for the activities covered by the code. The evaluation was carried out in two steps. In the first part we modelled the processes they were. BPCC was able to identify several areas where the existing processes were not compliant with the new code. In some cases the industry partner was already aware of some of the areas requiring modifications of the existing processes. However, some of the compliance issues discovered by the tools were novel to the business analysts and were identified as genuine non-compliance issues that need to be resolved. In the second part of the experiment, the existing processes were modified to comply with the code based on the issues identified in the first phase. In addition a few new business process models required by the new code were designed. As result we generated and annotated 6 process models. 5 of the 6 models are limited in size and they can be checked for compliance in seconds. The largest process contains 41 tasks, 12 decision points, xor splits, (11 binary, 1 ternary). The shortest path in the model has 6 tasks, while the longest path consists of 33 tasks (with 2 loops), and the



**Fig. 4.** BPCC report of traces, rules, and tasks responsible for non-compliance

longest path without loop is 22 task long. The time taken to verify compliance for this process amounts approximately to 40 seconds on a MacBook Pro 2.2Ghz Intel Core i7 processor with 8GB of RAM (limited to 4GB in Eclipse).

## 4 Conclusions

We reported on the development of a tool, BPCC, for checking the compliance of business processes with relevant regulations. The BPCC was successfully tested for real industry scale compliance problems. In the recent years, a few other compliance prototypes have been proposed: MoBuCom [15], Compass [2] and SeaFlows [14]. MoBuCom and Compass are based on Linear Temporal Logic (LTL) and mostly they just address “structural compliance” (i.e., that the tasks are executed in the relative order defined by a constraint model). The use of LTL implies that the model on which these tools are based on is not conceptual relative to the legal domain, and it fails to capture nuances of reasoning with normative constraints such as violations, different types of obligations, violations and their compensation. For example, obligations are represented by temporal operators. This raises the problem of how to represent the distinction between achievement and maintenance obligations. A possible solution is to use always for maintenance and sometimes for achievement, but this leaves no room for the concept of permission (the permission is dual of obligation, and always and sometimes are the dual of each other). In addition using temporal operators to model obligations makes hard to capture data compliance [10], i.e., obligations that refer to literals in the same task. SeaFlow is based on first-order logic, and it is well known that first order logic is not suitable to capture normative reasoning [11]. On the other hand FCL complies with the guidelines set up in [3] for a rule languages for the representation of legal knowledge and legal reasoning.

## Acknowledgment

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

## References

1. G. Antoniou, D. Billington, G. Governatori, and M.J. Maher. Representation results for defeasible logic. *ACM Transactions on Computational Logic*, 2(2):255–287, 04 2001.
2. A. Elgammal, O. Türetken, and W.-J. van den Heuvel. Using patterns for the analysis and resolution of compliance violations. *Int. J. Cooperative Inf. Syst.*, 21(1):31–54, 2012.
3. T.F. Gordon, G. Governatori, and A. Rotolo. Rules and norms: Requirements for rule interchange languages in the legal domain. In G. Governatori, J. Hall, and A. Paschke, editors, *RuleML 2009*, INCS 5858, pp. 282–296. Springer, 2009.
4. G. Governatori. Representing business contracts in RuleML. *International Journal of Cooperative Information Systems*, 14(2-3):181–216, 2005.

5. G. Governatori, Z. Milosevic, and S. Sadiq. Compliance checking between business processes and business contracts. In P.C.K. Hung, ed., *EDOC 2006*, pp. 221–232. IEEE Computing Society, 2006.
6. G. Governatori and A. Rotolo. Logic of violations: A Gentzen system for reasoning with contrary-to-duty obligations. *Australasian Journal of Logic* 4:193–215, 2006.
7. G. Governatori and Antonino Rotolo. An algorithm for business process compliance. In E. Francesconi, G. Sartor, and D. Tiscornia, eds, *Jurix 2008*, pp. 186–191. IOS Press, 2008.
8. G. Governatori and A. Rotolo. A conceptually rich model of business process compliance. In S. Link and A. Ghose, eds, *APCCM 2010*, CRPIT 110, pp. 3–12. ACS, 2010.
9. G. Governatori and S. Sadiq. The journey to business process compliance. In J. Cardoso and W. van der Aalst, eds, *Handbook of Research on BPM*, pp. 426–454. IGI Global, 2009.
10. M. Hashmi, G. Governatori, and M. Thandar Wynn. Business process data compliance. In *RuleML 2012*, LNCS 7438. Springer, 2012.
11. H Herrestad. Norms and formalization. In *ICAIL 1991*, pp 175–184. ACM, 1991.
12. H.-P. Lam and G. Governatori. The making of SPINdle. In G. Governatori, J. Hall, and A. Paschke, eds, *RuleML 2009*, LNCS 5858, pp. 315–322. Springer, 2009.
13. R. Lu, S. Sadiq, and G. Governatori. Compliance aware business process design. In A.H.M. ter Hofstede, B. Benatallah, and H.-Y. Paik, eds, *BPD’07*, LNCS 4928, pp 120–131. Springer, 2007.
14. L.T. Ly, S. Rinderle-Ma, K. Göser, and P. Dadam. On enabling integrated process compliance with semantic constraints in process management systems - requirements, challenges, solutions. *Information Systems Frontiers*, 14(2):195–219, 2012.
15. F.M. Maggi, M. Montali, M. Westergaard, and W.M.P. van der Aalst. Monitoring Business Constraints with Linear Temporal Logic: An Approach Based on Colored Automata. In *BPM 2011*, LNCS 6896, pp. 132–147. Springer, 2011.
16. S. Sadiq and G. Governatori. Managing regulatory compliance in business processes. In J. van Brocke and M. Rosemann, eds, *Handbook of Business Process Management*, volume 2, pp. 157–173. Springer, 2010.
17. S. Sadiq, G. Governatori, and K. Naimiri. Modelling of control objectives for business process compliance. In G. Alonso, P. Dadam, and M. Rosemann, eds, *BPM 2007*, LNCS 4714, pp. 149–164. Springer, 2007.

# PSOATransRun: Translating and Running PSOA RuleML via the TPTP Interchange Language for Theorem Provers

Gen Zou<sup>1</sup>, Reuben Peter-Paul<sup>1</sup>, Harold Boley<sup>1,2</sup>, and Alexandre Riazanov<sup>3</sup>

<sup>1</sup> Faculty of Computer Science, University of New Brunswick, Fredericton, Canada  
`gen.zou AT unb.ca`, `reuben.peterpaul AT gmail.com`,

<sup>2</sup> Information and Communications Technologies, National Research Council Canada  
`harold.boleyn AT nrc.gc.ca`,

<sup>3</sup> Department of Computer Science & Applied Statistics, UNB, Saint John, Canada  
`alexandre.riazanov AT gmail.com`

**Abstract.** PSOA RuleML is an object-relational rule language generalizing POSL, OO RuleML, F-logic, and RIF-BLD. In PSOA RuleML, the notion of positional-slotted, object-applicative (psoa) terms is used as a generalization of: (1) positional-slotted terms in POSL and OO RuleML and (2) frame and class-membership terms in F-logic and RIF-BLD. We demonstrate an online PSOA RuleML reasoning service, PSOATransRun, consisting of a translator and an execution engine. The translator, PSOA2TPTP, maps knowledge bases and queries in the PSOA RuleML presentation syntax to the popular TPTP interchange language, which is supported by many first-order logic theorem provers. The translated documents are then executed by the open-source VampirePrime reasoner to perform query answering. In our implementation, we use the ANTLR v3 parser generator tool to build the translator based on the grammars we developed. We wrap the translator and execution engine as resources into a RESTful Web API for convenient access. The presentation demonstrates PSOATransRun with a suite of examples that also constitute an online-interactive introduction to PSOA RuleML.

## 1 Introduction

Knowledge representation is at the foundation of Semantic Web applications, using rule and ontology languages as the main kinds of formal languages. PSOA RuleML is a recently developed rule language which combines the ideas of relational (predicate-based) and object-oriented (frame-based) modeling. In order to demonstrate the PSOA RuleML semantics, we have implemented an online PSOA RuleML reasoning service PSOATransRun. It enables PSOA RuleML deduction using the first order open-source VampirePrime reasoner via the interchange language TPTP (Thousands of Problems for Theorem Provers), which is supported by many reasoners, especially theorem provers. PSOATransRun is composed of a translator, PSOA2TPTP, and a run-time environment in the form of a TPTP-aware execution engine. The translator maps knowledge bases

and queries of PSOA RuleML in RIF-like Presentation Syntax (PSOA/PS) into a document in TPTP's First Order Form (FOF), which is then fed into the VampirePrime reasoner to deduce the query results.

Our implementation of PSOA2TPTP is built upon the ANTLR v3 parser generator framework.<sup>4</sup> The main components include a lexer, a parser and a tree walker generated from the input ANTLR grammars. The input document is first broken up, by the lexer, into a token stream; then converted, by the parser, into a structured Abstract Syntax Tree (AST); and finally traversed, by the tree walker, to generate a TPTP document via TPTP Abstract Syntax Objects.

We wrapped the PSOA2TPTP translator and the VampirePrime-based execution engine as resources into a RESTful Web API, and published a Web site demonstrating its use.<sup>5</sup>

## 2 Preliminaries

### 2.1 PSOA RuleML

PSOA RuleML [1] is an object-relational rule language generalizing POSL, OO RuleML, F-logic, and RIF-BLD. In PSOA RuleML, the notion of positional-slotted, object-applicative (psoa) terms is introduced:

$$o \# f ([t_{1,1} \dots t_{1,n_1}] \dots [t_{m,1} \dots t_{m,n_m}] \ p_1 \rightarrow v_1 \dots p_k \rightarrow v_k)$$

This notion generalizes (1) positional-slotted terms in POSL and OO RuleML and (2) frame and class-membership terms in F-logic and RIF-BLD. In a psoa term,  $o$  is the object identifier (OID) which uniquely identifies the object represented by the term. A psoa term integrates three types of information: (1) The class membership  $o \# f$  makes  $f$  the type of instance  $o$ ; (2) every slotted argument  $p_i \rightarrow v_i$  associates  $o$  with an attribute  $p_i$  and its value  $v_i$ ; (3) every tupled argument  $[t_{i,1} \dots t_{i,n_i}]$  associates  $o$  with a sequence of terms.

### 2.2 TPTP-FOF and VampirePrime

TPTP is a collection of test problems for automated theorem proving systems using the problem format of the same name. TPTP-FOF is the dialect allowing the use of arbitrary first-order formulas. A TPTP-FOF problem is a list of annotated formulas of the form:

$$fof(name, role, formula, source, useful\ info).$$

Here, *name* is a name given to the formula; *role* specifies the type of intended use of the formula, e.g. *axiom*, *theorem*, *conjecture*, etc. *formula* is the formula body (*source* and *useful info* are optional and irrelevant for our translation). Table 1 shows the most widely used TPTP constructors.

<sup>4</sup> <http://www.antlr.org/>

<sup>5</sup> <http://198.164.40.211:8082/psoa2tptp-trans/index.html>

**Table 1.** TPTP Constructors

Symbol	Logical Meaning	Symbol	Logical Meaning
$\sim$	<b>not</b>	$\neq$	<b>unequal</b>
$\&$	<b>and</b>	$\Rightarrow$	<b>implication</b>
$ $	<b>or</b>	$?[v1, v2, \dots]$	<b>existential quantifier</b>
$=$	<b>equal</b>	$![v1, v2, \dots]$	<b>universal quantifier</b>

VampirePrime is an open source reasoner derived from Vampire [2], a mature high-performance reasoner for first-order logic. VampirePrime supports not only standard theorem proving tasks like consistency checking and entailment, but also query answering using the Incremental Query Rewriting Technique [3].

### 3 System Architecture

In Figure 1, we present an architectural view of the PSOATransRun framework. We use Linux for our host environment, and VampirePrime can be re-compiled for any platform that supports gcc 4.x. We use a Java servlet container to host the PSOATransRun RESTful-Web-API web application, which we depict in Figure 1 as a *Web Archive* (WAR). The RESTful Web API WAR, basically consists of two JAX-RS<sup>6</sup> resources, and a static HTML page *application.html*. The Web API depends on the PSOA2TPTP-Translator Java application (see Figure 2) and it is also packaged into the WAR. The PSOATransRun application component, *application.html* is a static HTML Web page that accesses (via *XMLHttpRequests*<sup>7</sup>) the PSOATransRun RESTful resources (*Translate* and *Execute*) and composes them to provide an experimental PSOA Presentation Syntax (PSOA/PS) prototype for basic reasoning. The design and implementation of the RESTful Web API is described in more detail in Section 4.2.

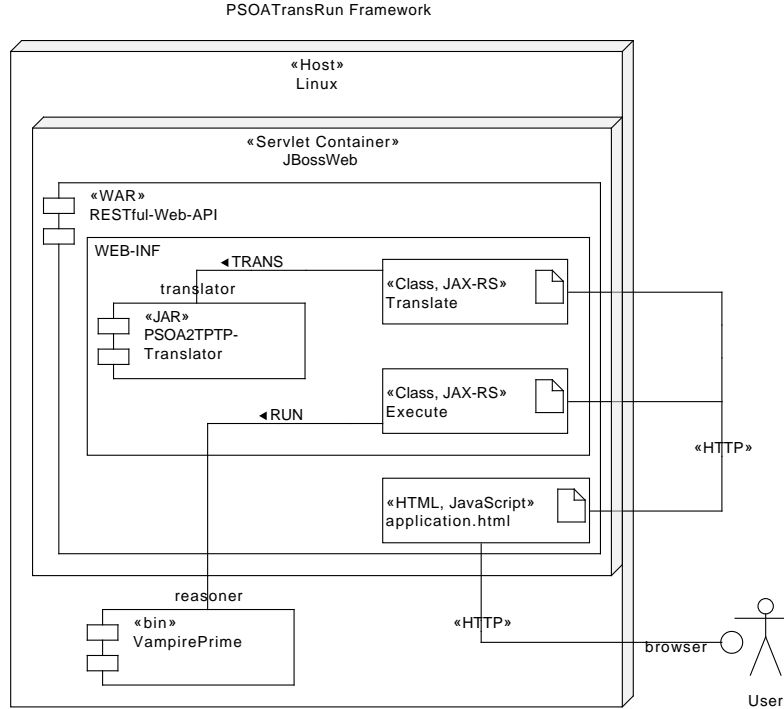
The architecture of the PSOA2TPTP translator is depicted in more detail in Figure 2. The translation consists of four phases:

1. The *PSOA/PS lexer* feeds off the input document as a character stream and does lexical analysis, grouping the characters into a stream of tokens.
2. The *PSOA/PS parser* operates on the token stream emanating from the lexer, and parsing the grammatical structure while constructing an intermediate data structure called Abstract Syntax Tree (AST), which is a highly structured and condensed version of the input.
3. The *tree walker* traverses the AST and builds an internal data structure, TPTP Abstract Syntax Objects (TPTP ASOs), representing semantically equivalent TPTP formulas, based on the translation rules.

<sup>6</sup> JAX-RS is a Java API for RESTful Web Services that facilitates the creation of Web services according to the Representational State Transfer (REST) architectural style.

<sup>7</sup> Used to send HTTP requests directly to a Web server.





**Fig. 1.** Architecture of PSOATransRun. The PSOATransRun application, *application.html*, composes the *Translate* and *Execute* (Run) resources for PSOA/PS queries.

4. The *TPTP renderer* reuses an existing parser/renderer library<sup>8</sup> for generating TPTP documents in concrete syntax from TPTP ASOs.

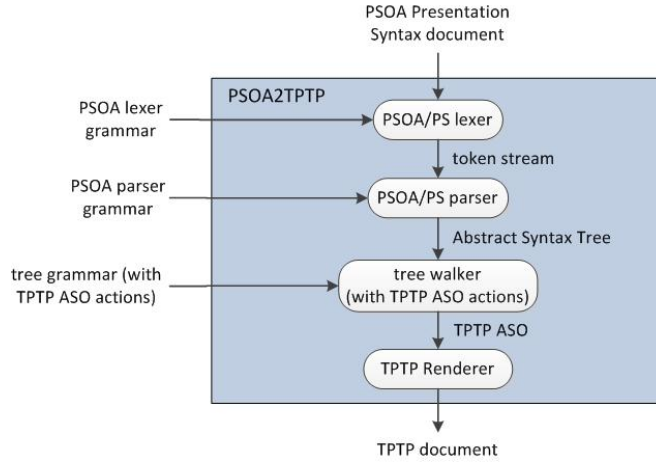
The lexer, parser and tree walker are generated by the ANTLR framework<sup>9</sup> from the provided lexer grammar, parser grammar and tree grammar, respectively.

Our intention was to create an application programming interface (API) and expose our growing set of translation tools and reasoner services over the *World Wide Web* via Web services. We chose to apply the REpresentational State Transfer (REST)<sup>10</sup> architectural style when designing our API for practical

<sup>8</sup> <http://riazanov.webs.com/tptp-parser.tgz>

<sup>9</sup> ANOther Tool for Language Recognition (ANTLR) is a parser generator widely used for building translators and interpreters for domain-specific languages. <http://www.antlr.org/>

<sup>10</sup> REST is an architectural style for distributed systems such as the World Wide Web. A RESTful Web API is an API that conforms to the RESTful architectural constraints specified in [4]



**Fig. 2.** Detailed architectural view of the PSOA2TPTP translator

reasons. While there are other architectural styles for distributed computing besides REST, RESTful Web APIs tend to be much easier to understand and use (see [5]).

## 4 Implementation

### 4.1 Translation

The semantics-preserving translation from PSOA RuleML to TPTP has two phases: (1) Normalization of composite formulas into a conjunction of elementary constructs and (2) translating them into corresponding TPTP forms.

In the first phase, every psoa formula of the form

$$o \# f([t_{1,1} \dots t_{1,n_1}] \dots [t_{m,1} \dots t_{m,n_m}] \ p_1 \rightarrow v_1 \dots p_k \rightarrow v_k)$$

is split into a conjunction of 1 class membership formula  $o \# f()$ ,  $m$  single-tuple formulas  $o \# \text{Top}(t_{i,1} \dots t_{i,n_i})$  and  $k$  (RDF-triple-like) single-slot formulas  $o \# \text{Top}(p_i \rightarrow v_i)$ . The rewriting preserves the semantics since the truth value of a psoa formula is *defined* by the conjunction.

In the second phase, we define the translation function  $\tau_{psoa}(\cdot)$  mapping each PSOA/PS elementary construct to a TPTP construct as shown in Table 2.

In the translation, we use ‘1’ and ‘Q’ as the prefixes for translated local constants and variables in TPTP, respectively.<sup>11</sup> The KB is translated sentence by sentence using  $\tau_{psoa}(\cdot)$ , while for the query we use a preserved answer predicate **ans** to show the bindings of variables. More explanations can be found in [6].

<sup>11</sup> In TPTP, constants and variables start with lower case and upper case letters, respectively.

**Table 2.** Mapping function  $\tau_{psoa}(\cdot)$  from PSOA/PS constructs to TPTP constructs

PSOA/PS Constructs	TPTP Constructs
$\_C$	$1C$
$?v$	$Qv$
$o \# \text{Top}(t_1 \dots t_k)$	$\text{tupterm}(\tau_{psoa}(o), \tau_{psoa}(t_1) \dots \tau_{psoa}(t_k))$
$o \# \text{Top}(p \rightarrow v)$	$\text{sloterm}(\tau_{psoa}(o), \tau_{psoa}(p), \tau_{psoa}(v))$
$o \# f()$	$\text{member}(\tau_{psoa}(o), \tau_{psoa}(f))$
$a \# b$	$\text{subclass}(\tau_{psoa}(a), \tau_{psoa}(b))$
$a = b$	$\tau_{psoa}(a) = \tau_{psoa}(b)$
$\text{And}(f_1 \dots f_n)$	$(\tau_{psoa}(f_1) \& \dots \& \tau_{psoa}(f_n))$
$\text{Or}(f_1 \dots f_n)$	$(\tau_{psoa}(f_1) \mid \dots \mid \tau_{psoa}(f_n))$
$\text{Exists } ?v_1 \dots ?v_n f$	$? [\tau_{psoa}(?v_1) \dots \tau_{psoa}(?v_n)] : \tau_{psoa}(f)$
$\text{Forall } ?v_1 \dots ?v_n f$	$! [\tau_{psoa}(?v_1) \dots \tau_{psoa}(?v_n)] : \tau_{psoa}(f)$
$\varphi :- \psi$	$\tau_{psoa}(\psi) \Rightarrow \tau_{psoa}(\varphi)$

## 4.2 RESTful Web API

Both the translation and execution operations are exposed as RESTful Web services as shown in Figure 1. This was accomplished by creating two REST resources: *Translate*, a REST resource for representing the PSOA2TPTP translator; *Execute*, a REST resource for representing a reasoner (VampirePrime).<sup>12</sup> Currently POST is the only HTTP operation supported by these resources along with application/json (JSON encoding) and text/plain (plain text) Internet media types.

To translate a PSOA/PS document into a TPTP document, the PSOA/PS document must be JSON-encoded and sent, in an HTTP POST request, to the *Translate* URI; the response is the result of the PSOA2TPTP translator encoded as a JSON array of TPTP-FOF sentences. See [7] for details.

The *Execute* Web service allows an application programmer to execute a reasoner; the reasoner we use is the VampirePrime reasoner, which accepts TPTP-FOF sentences as input. Therefore, to query an input knowledge base using PSOA/PS the application programmer must first request translation and then send the resulting TPTP-FOF sentences in an HTTP POST request to the *Execute* URI. The result will be the plain text output from the reasoner (see Listings 2-5 in [7]) and the example in the next section.

## 5 Examples

In this section we demonstrate some examples showing how input knowledge bases (KBs) and queries are translated into TPTP-FOF and executed by VampirePrime to get the query results. We start with a simple example with only ground facts in the KB, followed by an advanced one with rules.

<sup>12</sup> Note that the designation of *resource* is not in and of itself a Web service, which requires the combination of the resource URI, an HTTP operation and an Internet media type.

### 5.1 Example 1

– **Input KB:**

```
Document(
  Group(
    _f1#_family(_Mike _Amy _child->_Fred _child->_Jane)
    _Amy#_person([_married] [_bcs _mcs _phd] _job->_engineer)
  )
)
```

– **Translated KB:**

```
fof(ax01, axiom,
  member(lf1, lfamily) & tupterm(lf1, lMike, lAmy)
  & sloterm(lf1, lchild, lFred) & sloterm(lf1, lchild, lJane)).
fof(ax02, axiom,
  member(lAmy, lperson) & tupterm(lAmy, lbcs, lmcs, lphd)
  & tupterm(lAmy, lmarried) & sloterm(lAmy, ljob, lengineer)).
```

The KB has two psOA formulas as facts. The first fact has one tuple for the family's adults, where `_Mike _Amy` is equivalent to `[_Mike _Amy]`, a short-cut allowed only in single-tuple psOA terms; it has two slots for the family's children. The second fact has two tuples, of lengths 1 and 3, and also a slot. The two formulas are first broken into two conjunctions of elementary constructs, and then mapped to two TPTP conjunctions according to the function  $\tau_{psOA}(\cdot)$  defined in the last section.

– **Query 1.1:** `_Amy#_person(_job->_engineer)`

– **Translated Query:**

```
fof(query, theorem,
  ((member(lAmy, lperson) & sloterm(lAmy, ljob, lengineer))
   => ans)).
```

– **VampirePrime Output:**

```
Proof found.
...
... | «ans» ...
```

The translated query is combined with the translated KB into a document and executed by VampirePrime. In the output, `«ans»` indicates that the queried fact is true. Note that this query is a ground fact, so that the task here is to prove the fact rather than asking for variable bindings, which we will show next.

– **Query 1.2:** `_Amy#_person(_job->?Job)`

– **Translated Query:**

```
fof(query, theorem,
  ((member(lAmy, lperson) & sloterm(lAmy, ljob, QJob))
   => ans("?Job", QJob))).
```

– **VampirePrime Output:**

Proof found.

```
...  
... | «ans»("?Job = ",lengineer) ...
```

This query asks for the job of `_Amy`, and the answer `«ans»("?Job = ",lengineer)` means `?Job` can unify with `_engineer`.

## 5.2 Example 2

### – Input KB:

```
Document(  
  Group (  
    Forall ?X ?Y ?Z (  
      ?X#_person(_descendent->?Z) :-  
      And(?X#_person(_child->?Y) ?Y#_person(_descendent->?Z))  
    )  
    Forall ?X ?Y (  
      ?X#_person(_descendent->?Y) :- ?X#_person(_child->?Y)  
    )  
    _Tom#_person(_child->_Amy _job->_professor)  
    _Eva#_person(_child->_Amy)  
    _Amy#_person([_married] [_bcs _mcs _phd] _child->_Fred)  
    _Fred#_person(_school->_UNB)  
  )  
)
```

### – Translated KB:

```
fof(ax01,axiom,(  
  ! [QZ,QY,QX] :  
    ( ( member(QX,lperson) & sloterm(QX,lchild,QY)  
      & member(QY,lperson) & sloterm(QY,ldecendent,QZ))  
    => ( member(QX,lperson) & sloterm(QX,ldecendent,QZ) ) ) ).  
fof(ax02,axiom,(  
  ! [QY,QX] :  
    ( ( member(QX,lperson) & sloterm(QX,lchild,QY) )  
    => ( member(QX,lperson) & sloterm(QX,ldecendent,QY) ) ) ).  
fof(ax03,axiom,  
  ( member(lTom,lperson) & sloterm(lTom,lchild,lAmy)  
    & sloterm(lTom,ljob,lprofessor) ) ).  
fof(ax04,axiom,  
  ( member(lEva,lperson) & sloterm(lEva,lchild,lAmy) ) ).  
fof(ax05, axiom,  
  ( member(lAmy, lperson) & tupterm(lAmy, lbcs, lmcs, lphd)  
    & tupterm(lAmy, lmarried) & sloterm(lAmy,lchild,lFred) ) ).  
fof(ax06,axiom,  
  ( member(lFred,lperson) & sloterm(lFred,lschool,lUNB) ) ).
```

The KB has two rules and four facts. The facts shows the information of \_Tom, \_Eva, \_Amy, \_Fred. The rules define the descendent relationship.

– **Query 2.1:** ?Ancestor#\_person(\_descendent->?Who)

– **Translated Query:**

```
fof(query,theorem,(
    ! [QWho,QAncestor] :
        ( sloterm(QAncestor,ldecendent,QY)
          => ans("?Ancestor = ",QAncestor,"?Y = ",QWho) ) )).
```

– **VampirePrime Output:**

Proof found.

```
...
... | «ans»("?Ancestor = ",lAmy,"?Who = ",lFred) ...
...
... | «ans»("?Ancestor = ",lEva,"?Who = ",lAmy) ...
...
...
... | «ans»("?Ancestor = ",lEva,"?Who = ",lFred) ...
...
```

The query asks for all the descendent pairs <?Ancestor, ?Who> in the KB, and the output «ans»("?Who = ",lMike) and «ans»("?Who = ",lTom) from VampirePrime means gives all the unifications.

– **Query 2.2:**

```
And (?Ancestor1#_person(_descendent->_Fred)
    ?Ancestor2#_person(_descendent->_Fred))
```

– **Translated Query:**

```
fof(query,theorem,(
    ! [QAncestor2,QAncestor1] :
        ( ( member(QAncestor1,lperson)
          & sloterm(QAncestor1,ldecendent,lFred)
          & member(QAncestor2,lperson)
          & sloterm(QAncestor2,ldecendent,lFred) )
          => ans("?Ancestor1 = ",QAncestor1,
                "?Ancestor2 = ",QAncestor2) ) ) ).
```

– **VampirePrime Output:** Proof found.

```
...
... | «ans»("?Ancestor1 = ",lAmy,"?Ancestor2 = ",lAmy) ...
...
... | «ans»("?Ancestor1 = ",lAmy,"?Ancestor2 = ",lEva) ...
...
...
... | «ans»("?Ancestor1 = ",lTom,"?Ancestor2 = ",lEva) ...
...
```

– **Query 2.3:**

```
And (?Ancestor1#_person(_descendent->?Who)
    ?Ancestor2#_person(_descendent->?Who))
```

– **Translated Query:**

```
fof(query,theorem,(
    ! [QAncestor2,QWho,QAncestor1] :
    ( ( member(QAncestor1,lperson)
      & sloterm(QAncestor1,ldecendent,QWho)
      & member(QAncestor2,lperson)
      & sloterm(QAncestor2,ldecendent,QWho) )
    => ans("?Ancestor1 = ",QAncestor1,
           "?Who = ",QWho,"?Ancestor2 = ",QAncestor2) ) ).
```

– **VampirePrime Output:** Proof found.

```
...
... | «ans»("?Ancestor1 = ",lAmy,"?Who = ",lFred,"?Ancestor2 = ",lAmy)
...
.....
... | «ans»("?Ancestor1 = ",lTom,"?Who = ",lAmy,"?Ancestor2 = ",lEva)
...
.....
... | «ans»("?Ancestor1 = ",lTom,"?Who = ",lFred,"?Ancestor2 = ",lEva)
...
...
```

## 6 Conclusions and Future Work

PSOATransRun is the first implementation of PSOA RuleML. It translates a PSOA/PS knowledge base and queries into semantically equivalent TPTP documents, and then executes them through the VampirePrime reasoner to obtain the query results. Future work on the project includes: (1) Extend the capability of PSOATransRun to support all PSOA RuleML constructs; (2) build a complete benchmark suite for testing PSOA RuleML reasoners; (3) deploy PSOATransRun in real applications, e.g. the Clinical Intelligence use case [8], where PSOA rules are used to define semantic mappings for a hospital data warehouse.

The wiki page on PSOA RuleML<sup>13</sup> documents the ongoing development of PSOATransRun, gives further examples, and links to the online system. Users of PSOATransRun are encouraged to send their email feedback to the authors.

## References

1. Boley, H.: A RIF-Style Semantics for RuleML-Integrated Positional-Slotted, Object-Applicative Rules. In Bassiliades, N., Governatori, G., Paschke, A., eds.: RuleML Europe. Volume 6826 of LNCS., Springer (2011) 194–211

<sup>13</sup> [http://wiki.ruleml.org/index.php/PSOA\\_RuleML](http://wiki.ruleml.org/index.php/PSOA_RuleML)

2. Riazanov, A., Voronkov, A.: The Design and Implementation of Vampire. *AI Communications* **15**(2-3) (2002) 91–110
3. Riazanov, A., Aragao, M.A.: Incremental Query Rewriting with Resolution. *Canadian Semantic Web II* (2010)
4. Fielding, R.T.: Architectural Styles and the Design of Network-based Software Architectures. PhD thesis, University of California, Irvine (2000)
5. DuVander, A.: New Job Requirement: Experience Building RESTful APIs. <http://blog.programmableweb.com/2010/06/09/new-job-requirement-experience-building-restful-apis/> (July 2010)
6. Zou, G., Peter-Paul, R., Boley, H., Riazanov, A.: PSOA2TPTP: A Reference Translator for Interoperating PSOA RuleML with TPTP Reasoners. In Bikakis, A., Giurca, A., eds.: *RuleML 2012*. LNCS, Springer, Heidelberg (2012) 264–279
7. Zou, G., Peter-Paul, R.: PSOA2TPTP: Designing and Prototyping a Translator from PSOA RuleML to TPTP Format. Technical report [http://psoa2tptp.googlecode.com/files/PSOA2TPTP\\_Report\\_v1.0.pdf](http://psoa2tptp.googlecode.com/files/PSOA2TPTP_Report_v1.0.pdf).
8. Riazanov, A., Rose, G.W., Klein, A., Forster, A.J., Baker, C.J.O., Shaban-Nejad, A., Buckeridge, D.L.: Towards Clinical Intelligence with SADI Semantic Web Services: a Case Study with Hospital-Acquired Infections Data. In: *Proceedings of the 4th International Workshop on Semantic Web Applications and Tools for the Life Sciences*. SWAT4LS '11, New York, NY, USA, ACM (2012) 106–113



# Legal Rules, Text and Ontologies Over Time

Monica Palmirani<sup>1</sup>, Tommaso Ognibene, Luca Cervone<sup>1</sup>

<sup>1</sup> CIRSIFID, University of Bologna.  
{monica.palmirani, tommaso.ognibene, luca.cervone}@unibo.it

**Abstract.** The current paper presents the “Fill the gap” project that aims to design a set of XML standards for modelling legal documents in the Semantic Web over time. The goal of the project is to design an information system using XML standards able to store in an XML-native database legal resources and legal rules in an integrated way for supporting legal knowledge engineers and end-users (e.g., public administrative officers, judges, citizens).

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

## 1. Introduction

“Fill the gap” [28] is a project funded by the CIRSIFID-University of Bologna in 2009 as an extension of the outcomes of the ESTRELLA<sup>1</sup> IST project (IST-2004-027655) with the aim for performing a platform where legal documents are modelled using XML standards and the ontology layer is used as the interconnection technique between the pure text of the document and the embedded legal knowledge, including rules representing the norms expressed by the textual document. The ontology is used for modelling the legal concepts and to represent the properties and the T-Box axioms of the main legal values (e.g., copyright, work, etc.), including geo-spatial (e.g., jurisdiction) and legal temporal dimensions (e.g., enforceability, efficacy, applicability of the norms). The text, annotated in XML using the Akoma Ntoso standard [35] and the metadata, extracted using parsers and NLP techniques [22], are connected manually to the ontology framework [14][10] and finally, the rules, formalized in *defeasible logic*, are connected to the textual provisions and to general and abstract legal concepts modelled in the legal ontology.

In this way several applications are possible: i) to improve information retrieval of legal documents; ii) to facilitate navigation over time when norms and texts change; iii) to foster semantic indexing, classification and query; iv) to enhance the expressivity of formal models of legal reasoning; v) to provide appropriate legal explanations using textual provisions in order to justify the proof of reasoning and vi) finally to annotate legal resources with the legal knowledge output of the proof for permitting open data sharing (e.g., Linked data). The aim of this paper is to present a first demo of this approach applied on a fragment of the US code, Title 17, Section 504 in order to provide a *proof-of-concept* of the applicative architecture.

---

<sup>1</sup> <http://www.leibnizcenter.org/current-projects/estrella>

## 2. Relevance of the Problem

The last twenty years have seen a growing interest in the development of XML standards, methodologies and models for the management of legal knowledge [21][33][8]. This interest covers not only “proper” law documents, such as legislation, but it embraces all source documents with a relevant legal content, such as the vast area of internal regulations of public bodies/private companies, codes of conduct, codes of practice, often cited as “soft law”. This is particularly true in some financial/legal domains [6], such as those involving banks and insurance companies and as well as new emerging sectors like patent law , cloud computing and privacy [17], credit card company regulations and telecommunication policies.

The Computer Science and Law community itself [30] dedicated the last two decades to modelling legal norms using different logics and formalisms. 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. In the last ten years several Legal XML standards have arisen for describing legal text and rules (RuleML, RIF, SWRL, etc.). In the meantime the Semantic Web, in particular Legal Ontology research, combined with NLP extraction of semantics, has given a great impulse to modelling the legal concepts [23][10][14][31]. Certainly, one of the main challenges is to acquire the ability to capture, with the help of NLP techniques [30][36], all relevant legal knowledge embedded in a legal document and to represent it in an appropriate formal model. This enables the descriptiveness, meaning and semantics of the source document to be retained, and at the same time, the knowledge is machine-readable and computable.

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 in order to realize an integrated and self-contained representation. There are four 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 logic representation of the embedded rules strongly affects the real usage of the legal document knowledge in favour for citizens, public administrations and business (e.g., contracts, assurance regulation, bank soft law, etc.);
- management of changes to the legal document over time, especially act, 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 empowered bodies (e.g., contractors) should be preserved by any manipulation. 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;
- a theory of legal document modelling able to separate clearly the many layers of representation of the resource: content (text), structure of the text, metadata on the

document, ontology on the legal concepts expressed in the text, legal content modelling (regulatory part of the text) are fundamental to preserving over time the digital legal text enriched by many semantic annotations, including logic representation of the rules.

### 3. Filling the Gap: from Text to Rules

The first distinction that we should provide to understand goals in the legal domain is to distinguish between three conceptual layers:

- **norms** (abstract mandatory commands concerning rights or duties)
- **textual provisions** (sequences of texts) and
- **rules** (elaboration of the text in logical rules).

The **norm**, following Kelsen's definition [19], is an abstract mandatory command concerning rights or duties. The norm usually is expressed in written using legal text or in an oral way (e.g., social norm, oral contract) or in other representations (e.g., symbolic road signs).

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

The **legal rules** are interpretation of the provision(s) formalized using logical rules in the form of antecedent and consequent. Sometime several provisions determine a single rule or a single provision includes multiple rules.

Usually in the state of the art AI&Law scholars focused their attention only on the rules modelling and on the foundational logical theory, and apart the isomorphism principles [4] the connection with the text over time and the ontology aspects has been neglected. There is a theoretical and important debate in the AI&Law community on the interpretation of the legal textual provisions expressed in natural language and on canonization of the rules using logical formalisms [5]. The prevalent theory is now oriented towards *hybrid interpretation* [32] (rather than pure *textualism*, or pure *interpretation*). We want to make visible in the text a "scintilla of 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 forced the legal knowledge expert to split the original provision in two or more rules, or to duplicate the rules or to compact several sentences in one unique rule. In this scenario we have to manage an N:M relationship between norms, textual provisions and ontology that we want to capture and to represent maintaining the strong separation between these three levels.

The law changes over time and consequently change the rules and the ontological classes (e.g., the definition of EU citizenship changed in 2004 with the annexation of 10 new member states in the European Community). It is also fundamental to assign dates to the ontology and to the rules, based on an analytical approach, to the text, and analyze the relationships among sets of dates. The semantic web cake recommends that content, metadata should be modelled and represented in separate and clean layers. This recommendation is not widely followed from too many XML schemas, including those in the legal domain. The layers of content and rules are often confused to pursue a short annotation syntax, or procedural performance

parameters or simply because a neat analysis of the semantic and abstract components is missing.

Therefore in our vision all legal resources present a complex multilayered information architecture [2] that includes several perspectives of analysis:

- **TEXT.** The perspective of the document officially approved by a legally competent authority. This text is the only legal binding.
- **TEXT'S STRUCTURE.** The perspective of the document that describes the way the text is organized.
- **METADATA.** Any additional information that was not deliberated by the legally competent authority. Metadata can describe the document itself (e.g., by way of keywords), its workflow (e.g., procedural steps in the bill), its lifecycle (the document's history), or its identification (e.g., by way of an URI).
- **ONTOLOGY.** Any information specifying the legal or institutional setting in which the document plays a role—e.g., information identifying the document as a judgment or opinion about the legal system's concepts—or any legal concept which is invoked in the text and which needs modelling (e.g., jurisdiction, enter into force, applicability, etc.).
- **LEGAL RULES.** The legal interpretation and modelling of the text's meaning. The transformation of the norms in logical rules for permitting legal reasoning. Several XML standards are present in the state of the art for managing rules (RIF, RuleML, SWRL), nevertheless RuleML seems to provide a flexible language able to describe different possible theories or logical models (propositional, predicative, argumentative, non-monotonic, deontic, defeasible, etc.) but it is not fitted for the legal domain. For this reason we are starting a new technical group in OASIS called LegalRuleML with the aim to perform a specific RuleML module oriented to the legal peculiarities including defeasibility, deontic, temporal reasoning, qualification of the norms, institutions (e.g., authority, agents, authors), jurisdiction [29].

The aim of this project is to connect all the layers and to manage the temporal dimensions of the provisions (text layer) and rule levels to make them connected accordingly with respective change over time. In other words our aim is to track the modifications of the text especially concerning the temporal dimensions (e.g., application of the norm) and to model the changes in the rules accordingly with the change management of the text. Vice versa it often happens that rules are changed by external events or by the result of legal reasoning, so the validity of the text is affected by these inferred knowledge (e.g., international treaty rules suspended by war).

Secondarily, we would like to build a reasoner that is able to deduct the correct rules to apply depending on the time parameters of the facts. If we have a precedent in criminal law of the High Court in 1994, a new law in 2010 and while the facts of the case occurred in 2005, but the judge must decide the case-law in 2012, the reasoner has to take in consideration and compare two scenarios: the rules coming before the 2005 (date of the facts) and the rules coming after the 2005 till 2012 for permitting to the judge to evaluate the best opportunities for the criminal (principle of *favor rei*). The reasoner should be able to manage temporal reasoning on the rules at a meta-level in order to pre-filter the pertinent rules according to the temporal parameters and to build dynamically new rules for qualifying the rules (e.g., defeasible and defeater).

We use the Akoma Ntoso XML standard [35] for implementing the first three levels of the Semantic Web cake (mark-up the text, the structure and the legal metadata) and to provide hooks and mechanisms for referring to external ontologies and to legal knowledge modelling using URI and idRef to the proper nodes. LKIF-Core [10] provides general mechanisms for coping with the ontology level and it is able to manage the events, the roles, the authors, and other more fine grained legal knowledge models. Finally LegalRuleML (we used a draft preliminary version not official approved by the OASIS TC<sup>2</sup>) is able to model normative rules [29]. The fundamental part of this multilayer architecture is the URI reference based naming convention [16][3] that functions as the interface between levels. A specific resolver is able to point out the different sources and rules, over time, in correct way. This mechanism is included in the *RuleViewer* module presented in section 7.

#### 4. Applicative Scenario

The applicative scenario that this project aims to manage is showed in the following picture (see Fig.1) and organized in the following steps:

- i. legal text is marked-up in XML legal standard, in our case in Akoma Ntoso;
- ii. legal concepts derived by legal text are modelled in OWL and updated on the base of the changes over time;
- iii. rules interpreted by legal text, and integrated by the legal ontologies, are modelled in LegalRuleML;
- iv. a native XML database stores all the files and it is able to manage Xpath and Xquery in easy way on all the files XML, RDF, OWL;
- v. Drools engine provide the level of reasoning;
- vi. general assertions derived by knowledge base reasoning process (e.g., an invalid section) could be exported in RDF in order to enrich the original XML text and in such way to improve the sharable knowledge on the web.

This approach is based on several requirements near to the concept “Fill the Gap”:

- the **granularity** of the legal document marked-up in XML must be related, following the isomorphic principle [4], to the logic rules representation and to the ontological statements (word, paragraphs, etc.). For this purpose we recall the URI references of Akoma Notoso XML nodes in a special metadata block of LegalRuleML called <lrml: sources>, where we specify the provisions connected with the atoms and rules. This **relationship between rules and text** is an **N:M cardinality**, so the current state of LegalRuleML implements a mechanism able to represent the multiple associations;
- the **contextual information** related to the rules has to manage in atomic way in order to favour the correspondence between rule and the textual provision. This ensures that where there is a modification or cessation of a fragment of text, it is possible to manipulate only a node of the LegalRuleML XML tree without affecting the consistency of the other rules. This principle enables rule and its

<sup>2</sup> We take into consideration the version available at the date of the paper submission: <http://www.oasis-open.org/apps/org/workgroup/legalruleml/download.php/45888/2.1defeasibility.002.002.doc>

metadata to be encapsulated as an atomic *object*. For this reason we proposed a separate and atomic block <lrml:ruleInfo> in LegalRuleML (under discussion in the TC).

- the relationship between the document, rules and the ontology layers needs to be implemented considering the bidirectionality of the information and, among the other issues, the evolution of the legal concept over time. The system is not able to implement this feature in the current version;
- the legal document **changes over time**, therefore in the rule modelling layer, as well as in the legal ontology and in the text layers, it is necessary a mechanism for managing the dynamicity over time and a temporal logic able to manage retroactivity effect of the norms (e.g., annulment, forking of temporal lines, etc.) and applicability of the law (e.g., an Act about the earthquakes in Italy is applicable only to the events of 20 and 29 May 2012). The **legal temporal model** should be able to manage three main legal axes (enforceability, efficacy and applicability, so a rule could be effective but not applicable when concrete conditionals are not satisfied). For this purpose Akoma Ntoso and LegalRuleML include a temporal model event-based;
- the **deontic operators** need to be managed jointly with the temporal parameters for permitting the correct application of the obligations, rights, permissions and violations accordingly with the fact (e.g, crime) to evaluate;
- the non-monotonic dimension of the law, (e.g., the exceptions that are present often in legal documents including contract) would strongly suggests that a **defeasible reasoning** approach should be adopted.

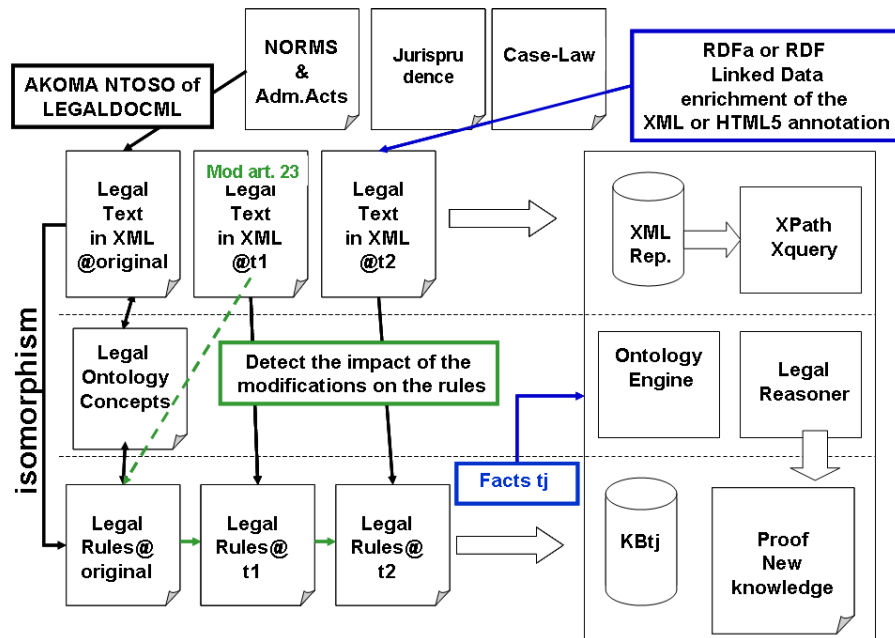


Fig. 1 – Applicative Scenario

## 5. Architecture Design

For implementing these goals we will adopt the following methodology and apply it to a fragment of the US copyright domain (sec. 504) in order to model, describe and represent the different levels of legal knowledge information:

- **text, structure, metadata:** legal documents (e.g., contracts) will be marked-up in Akoma Ntoso using a new web editor (*Rawe*) based on a previous .specialized editors features (Norma-Editor [24]) that was able to extract structure, references and metadata using parsers [2];
- **legal domain ontology:** an ontology for modelling and defining macro-concepts specific for the legal domain (e.g., IPR issues, jurisdiction, penalty, etc). The ontological classes built manually define the legal theoretical concepts describing their properties, relationships with other contents and their spatial and chronological dimensions. The ontology are stored in the native XML database that is able to manage versioning and evolution over time of OWL2 ontologies [7][1];
- **NLP tools:** the team uses NLP techniques for extracting the legal knowledge embedded in legal texts and to represent them as XML structured elements or as previously defined ontological structures. NLP techniques facilitate the development of parsers able to fill the gap between the text and its semantic level and accelerating the mark up process that usually is a time consuming task [25];
- **rules modelling and reasoning:** the legal document (e.g., law, judgments, contracts, etc.) will be modelled by the legal knowledge engineer in LegalRuleML. Rules using the web editor. The rules represented in LegalRuleML are imported inside of an inference engine properly customized by CIRSFD (based on Drools ver. 5.4) with a specific dialog interface for entering the facts. The rule engine aims to use a hybrid technology [9][34] (see among the others OntoRule project) using semantic rule reasoning taking benefits from the OWL2 ontology;
- **native XML DB repository and rule viewer.** All legal resources (text, metadata, ontologies and rules) are stored and delivered on the Web using a native XML database (based on eXist). In this way all the legal resources will be interconnected and presented for gathering the legal knowledge through an information retrieval engine, a reasoning engine and an application layer [26][27];
- **presentation of rules an text.** Finally the text and rules are presented using a web interface (*RuleViewer*) in order to connect text and rules for the end-user.

Some modules of this architecture (see Annex) are presented in this paper demo.

- A **specialized web editor** for marking up legal text and normative rules (*Rawe*<sup>3</sup>) in synchronized way, realizing so the isomorphism principle.

---

<sup>3</sup> <http://sinatra.cirsfid.unibo.it/rawe/>

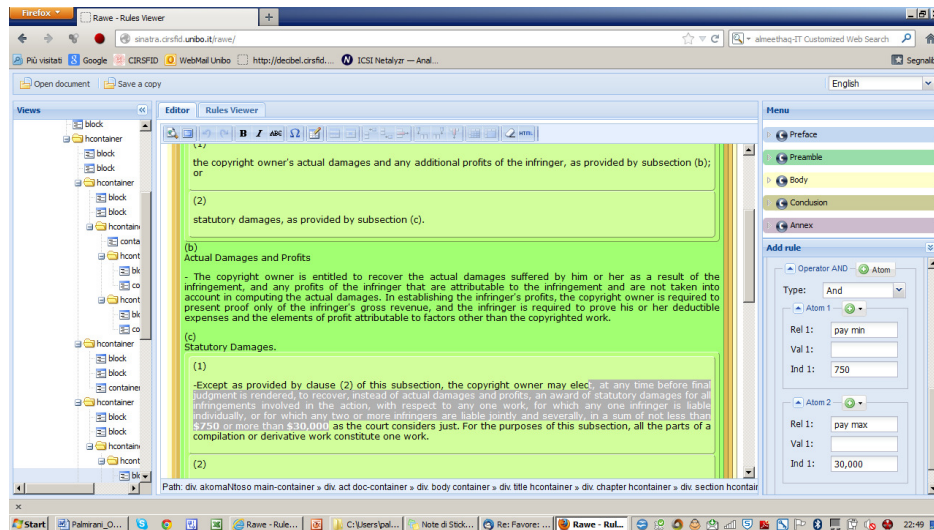


Fig. 2 – Rawe web editor for marking up legal text and normative rules.

The rules are converted to the LegalRuleML emerging standard and the text to Akoma Ntoso.

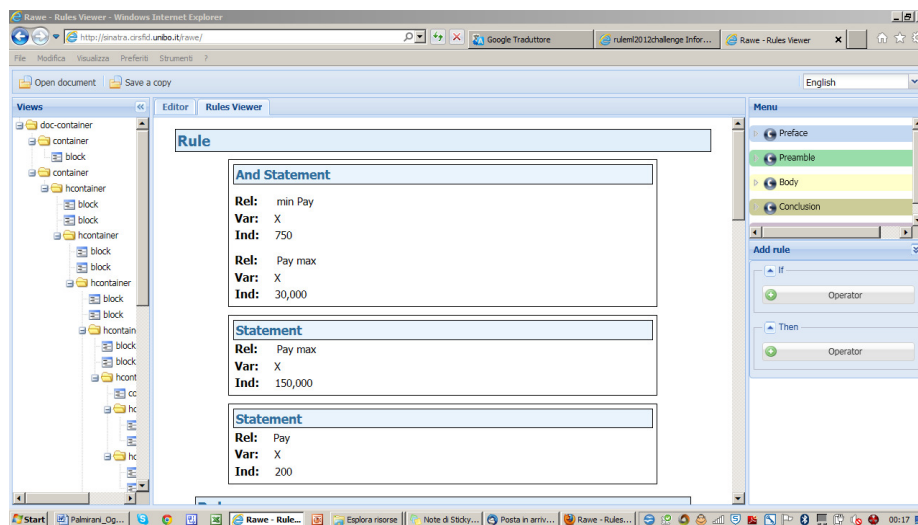


Fig. 3 – Rule Viewer inside of Rawe editor.

- In a second step the editor submits the text and the rules in XML format (respectively in Akoma Ntoso and in LegalRuleML) to the *eXist* native XML database.
- An interface API queries *eXist* and extracts the LegalRuleML files. In particular those files are used for populating the temporal information in a dynamic way stored in the Drools knowledge base.



- The Drools reasoner simulates defeasibility rules and reasoning.
- Finally a rule viewer (*RuleViewer module*<sup>4</sup>) presents the rules connected with the legal text, including version management over time.

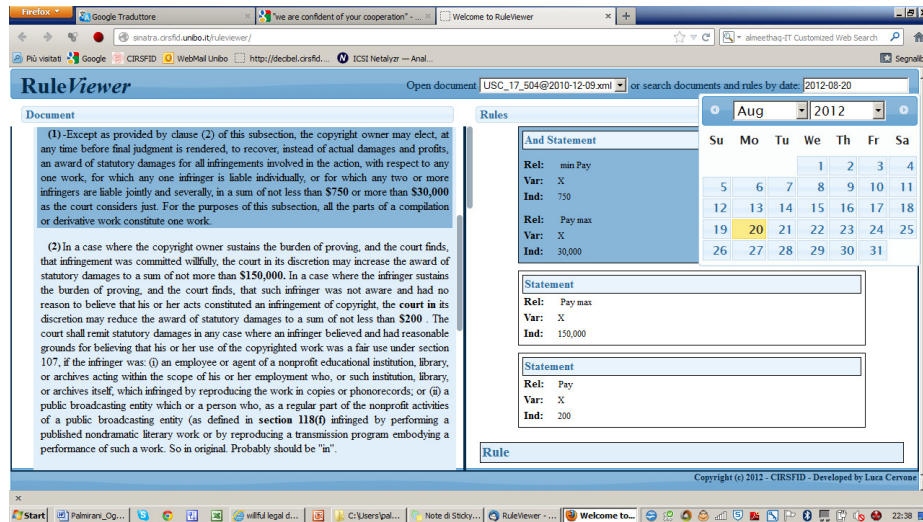


Fig. 4 – Text and RuleViewer.

## 6. Pilot Case

We definitely aim to provide a demonstration environment for testing the applicative scenario, using a pilot case coming from the US code, Title 17, sec. 504 on the copyright infringement.

17 USC Sec. 504

(c) Statutory Damages. –

(1) 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 \$250 or more than \$10,000 as the court considers just. For the purposes of this subsection, all the parts of a compilation or derivative work constitute one work.

(2) In a case where the copyright owner sustains the burden of proving, and the court finds, that infringement was committed willfully, the court in its discretion may increase the award of statutory damages to a sum of not more than \$50,000. In a case where the infringer sustains the burden of proving, and the court finds, that such infringer was not aware and had no reason to believe that his or her acts constituted an infringement of

<sup>4</sup> <http://sinatra.cirsfid.unibo.it/ruleviewer/>

copyright, the court at its discretion may reduce the award of statutory damages to a sum of not less than \$100.

This provision was modified three times and it is just a chance that the modifications are not retroactive (i.e., in legal domain it is necessary to use a non-monotonic temporal model). The table below lists the modifications with the temporal intervals of efficacy:

Interval name	Interval of efficacy of the norm	Statutory Damages	Willfully	Bona Fides
timeBlock1	[1976-10-19, 1995-03-01[	\$250 <= statutoryDamages <= \$10,000	Max \$50,000	Min \$100
timeBlock2	[1995-03-01, 2001-02-01[	\$500 <= statutoryDamages <= \$20,000	Max \$100,000	Min \$200
timeBlock3	[2001-02-01, •[	\$750 <= statutoryDamages <= \$30,00	Max \$150,000	Min \$200

The goal is to insert in the system a fact in a due date  $t_1$  and to check which statutory damages the infringer must pay. We are interested also to show the text of the proper version according to the rules applied and the time.

## 6.1. Modelling Text and Rules

We have translated the textual provisions (all the three versions) in XML using Akoma Ntoso and the corresponding rules in LegalRuleML<sup>5</sup>. The XML files were posted to the system, stored in *eXist* repository and also converted in Drools Rule Language for permitting to manage the rules in the Drools suite. In Drools a rule has the following format:

```
rule
    // attributes
    when
        // conditions
    then
        // actions
end
```

It is not a classical conditional structure IF-THEN, because Drools uses WHEN-THEN model. Drools implements a version of *Rete* algorithm and executes the conclusions whenever the patterns in the conditions are matched by a fact. Drools supports several temporal reasoning constructs (e.g., Allen's time model) as showed in the table below:

---

<sup>5</sup> Using the preliminary draft version of the OASIS TC.















	Point-Point	Point-Interval	Interval-Interval
A before B			
A meets B			
A overlaps B			
A finishes B			
A includes B			
A starts B			
A coincides B			

Fig. 5 – Allen’s temporal model managed by Drools.

Anyway, the implementation of the temporal reasoning by Allen’s model in Drools is effective only when dealing with business rules and business events such as commercial markdowns that live in the present and die in the future. On the contrary, when dealing with legal events and rules, there is not only present and future, but a complex mix of past, present and future to deal with (e.g., modifications in the past with retroactive effects). For this reason the temporal attributes (metadata) implemented by Drools to manage events’ and rules’ lifecycles are not enough in our case. This limitation has been partially supplied by adding temporal constraints as patterns to be matched.

```
@ID("statutory_damages_01")
date-effective "31-OCT-1988"           // start efficacy
//date-expires "dd-MMM-dddd"          // limitation of action (?)
```

Fig. 6 – date-effective and date-expires.

The rule attributes “date-effective” and “date-expires” are almost useless when dealing with legal rules (see Fig. 6). Drools is open source and written entirely in Java, so that we implemented a data structured using POJO. We inserted some instances in the reasoner in the following way:

```
infringements = new ArrayList<Infringement>();

infringements.add(new Infringement(1, 1, "1", new DateTime(1989,1,1,0,0,0,0),
    null, false, false, null));
infringements.add(new Infringement(1, 1, "1", new DateTime(1999,1,1,0,0,0,0),
    null, false, false, null));
infringements.add(new Infringement(1, 1, "1", new DateTime(2009,1,1,0,0,0,0),
    null, false, false, null));
```

Fig. 7 – Instances.

Furthermore, in the DRL (Drools Rule Language) file, we added besides the mere instances some semantics of Drools:

```
declare Infringement
    @role( event )
    @timestamp( getTimestamp() )
    @duration( getDuration() )
end
```

Fig. 8 – Data model of the instance query.

In this way, the reasoner considers the instances as events, with a precise *timestamp* and duration. Though the attributes “@timestamp” and “@duration” suffer the same limitations as the rule’s temporal attributes as “@stated” before. Every rule refers to a precise legal provision with precise temporal parameters for its efficacy. We can add these temporal conditions in every rule as in the following listing (see Fig. 9). Or we can consider the temporal parameters as meta rules that control the rule flow. For this we use the Drools & jBPM RuleFlow (see Fig. 10). In the following example we force the rule engine to fire (or rather, to give a chance to fire) only the rules that match with given certain temporal parameters. In this manner we can deal with rule versioning in a fast way (especially in case of modifications of the relative norms).

```
rule "rule of the statutory damages with burden of proving and willful infringement [from 03-01-1995 to 01-01-2001]"
when
    Date( $startEfficacy: time ) from new Date(getDate("3-1-1995")) // 03-01-1995
    Date( $endEfficacy: time ) from new Date(getDate("1-1-2001")) // 01-01-2001

    $infringement : Infringement
    (
        $infringement.getTimestamp() after $startEfficacy, // temporal reasoning
        $infringement.getTimestamp() before $endEfficacy,

        $infringement.getBurdenOfProving(), // normative reasoning
        $infringement.getMalicious()
    )
then
    System.out.println("The event happened in " + $infringement.getTimeStart() +
        " therefore I apply the rule in force [from 03-01-1995 to 01-01-2001].");
    System.out.println("Since the burden of proving is " + $infringement.getBurdenOfProving()
        + " and the willfulness is " + $infringement.getMalicious() + " then the rule applied is 'burden of proving + willfulness'");
    System.out.println();
end
```

Fig. 9 – Temporal conditions in Drools Rules.

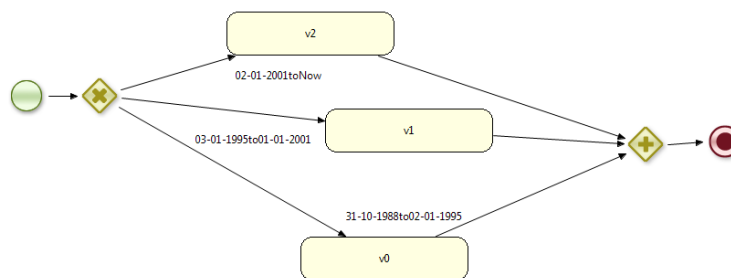


Fig. 10 – Workflow module for modelling the priority of the rules in Drools Rules.

Nevertheless, this kind of rule flow management cannot deal in an effective way with the phenomenon of multiple versions of legal rules applicable at the same time. Or rather, it can, by using an OR type instead of an XOR type, but it needs also a more accurate combination between the two rule set versions.

Drools has various way to deal with complex rule sets. In order to deal with conflicting rules we tried to apply the construct named “salience”. It allows to give to every rule an order of priority. This way the rule engine shall fire firstly the rules with the highest priority. The disadvantages is to define since the beginning the priority of all rules manually.

## 6.2. Goals

The system manages the following request for each fact on the basis of the below table of variables shown below:

Starting Time of the infringement	Ending Time of the infringement	Status of the willfulness	Status of the proving
T1	T2	Willful Not-willful	Burden of Proving Not burden of proving

Fig. 11 – Data model of the dialogue.

We need also the *status of the work* (public domain or not public domain) that for now it is stored in the local database in static way, but for the future we will insert the appropriate rules coming from the Title 17, US code, and we will deduct in real-time and dynamically the status of the work from the reasoning.

We have entered several cases testing different intervals of time, different status of wilful and proving. In the pilot case this produces the following output:

```
The event happened in 1998-01-01T00:00:00.000+01:00 therefore I must apply the rule in force [from 03-01-1995 to 01-01-2001].
Since the burden of proving is false and the willfulness is false then the rule applied is 'not burden of proving'

The event happened in 1994-01-01T00:00:00.000+01:00 therefore I must apply the rule in force [from 31-10-1988 to 02-01-1995].
Since the burden of proving is false and the willfulness is false then the rule applied is 'not burden of proving'

The event happened in 1989-01-01T00:00:00.000+01:00 therefore I must apply the rule in force [from 31-10-1988 to 02-01-1995].
Since the burden of proving is true and the willfulness is true then the rule applied is 'not burden of proving'
```

Fig. 12 – Output of the test.

For example, the event with *timestamp* “1989-01-01T00:00:00.000+01:00” has the field “burden of proving” set true and the field “willfulness” set true, therefore the legal rule to be applied should be the rule “burden of proving + willfulness”. Nevertheless, as the rule “not burden of proving” has the highest priority and it is a general rule with no constraints, the rule engine fires it and executes the consequences. In a second moment, when the rule engine fires the rules with a lower

priority, it fires our right rule, the rule “burden of proving + willfulness” and this leads to *override* the consequences of the general rule fired before.

Such kind of priority as a conflict resolution strategy between rules may be consequent to the following legal argument: first we apply the general rule, then we apply the more specific rule and override the general rule consequences with the more specific rule consequences.

### 6.3. Reasoner

Drools rule engine is made for dealing with thousands of facts and thousands of rules fired real time by different entry points. Its aim is to build expert systems able to take into consideration all the events happened during the time and matching the appropriate conclusions by means of rulebases. For our legal domain, the best option is to build a legal expert system very focalized on concrete, business application in order to exploit all the best features of Drools (If the aim is to exploit all the best feature of a Business Rule Management System). For example if we want a system for dealing with US Copyright law, with Drools we can build a rule engine always active, 24/7, that process several precise classes of events (such as possible violations, death of authors, payment of royalties) and compute different kinds of rules (we can have more abstract rules that just represents the logic within the norms, or we can have more concrete rules that realize a business policy conforming to the norms). In technical terms, Drools offers:

- Several clock types, for example, a real-time clock or a pseudo clock. The real-time clock is the default and should be used in normal circumstances. But in the legal domain the pseudo clock is especially useful as we have complete control over time and we can better deal with complex temporal reasoning schemas.
- The declaration and usage of events with both semantics: point-in-time events and interval-based events. This is useful in the legal domain since an infringement or, more generally, a crime, can be committed in a precise single moment, or in an interval of time (e.g., stalking crime that is a set of harassment during an interval of time).

## 7. Front-End Component: RuleViewer

At the end the system front-end *RuleViewer*<sup>6</sup> (see Fig. 5) shows the list of the versions of the US Code, Title 17, section 504 over times (seven versions). The end-user can navigate to one version (using a calendar or selecting the list of versions) and to detect immediately, by *mouse-hover* mechanism, the rules involved in a due fragment of text. Vice versa it is possible to navigate the rules in the right window and the end-user can see the correspondent fragment of legal text for integrating the legal interpretation. An important feature of *RuleViewer* is to provide a common environment where the models for representing legal knowledge are close to the

---

<sup>6</sup> <http://sinatra.cirsfid.unibo.it/ruleviewer/>

document text, thus bridging the gap between legal reasoning and textual representation. This means that each document (or each document package) can be connected with a set of rules representing the norms found in the legal document. It is for this purpose that we use the LegalRuleML language in our project in a very preliminary version<sup>7</sup>. The relationship between LegalRuleML files and the legal textual Akoma Ntoso documents is thus managed using a many-to-many (N:M) cardinality expressed in the `<lrml:source>` element that is able to connect, using URI references, rules and original textual sources. This makes it possible to navigate from a document to its corresponding norms and vice-versa, also considering the versioning. This can be done using the latest Web technologies (AJAX) and the mouse-hover technique. But this also means that searching for the relations dependent on a rule can be a quite complex task, typically involving a very large body of XML documents. The best way to query LegalRuleML resources related to a document is to use Xpath or Xquery languages and to foster the `< lrml:source> < lrml:ruleInfo>` LegalRuleML blocks and the temporal parameters related to the rules. So, if we store the LegalRuleML files in a particular instance of the *eXist* database, we can use the above-mentioned technology to do complex queries in a very customizable way and with good results. This feature of connecting documents with their rules and vice-versa is helpful especially where change management is concerned, for in this way we can navigate a set of documents stilled at a time *t* and extract a subset of rules already coordinated with respect to the timeline. An inference engine can process for purposes of reasoning.

## 8. Conclusion

We have presented the foundational architecture design of a project called “Fill the Gap” which aims to integrate and to interconnect legal text marked up in Akoma Ntoso with rules marked up with the emerging LegalRuleML and ontology expressed in LKIF-Core. Three modules are presented: i) a specialized web editor for marking up text and rules in XML; ii) an eXist repository with an API for Xpath and Xquery; iii) a prototype in Drools for implementing defeasible reasoning using first priority mechanism and after workflow steps definitions for simulating the hierarchy relationship among the rules (*override* mechanism); iv) a front-end visualization of the legal text and rules in *side-by-side* windows. We have also refined the temporal aspects, but the legal time parameters don’t follow the canonical sequence of Allen’s events because we need to manage the retroactive effects. Finally we have tested the Drools engine for producing some results. Unfortunately Drools needs new modules for managing defeasibility (not only simulating it) and temporal reasoning closer to the legal domain time parameters imported dynamically by LegalRuleML files. We are satisfied by these discovery because for the future we intend to proceed with the following experiments: 1) to refine the editor especially on the rule modelling side and for including boxes dedicated to metadata; 2) to extend the eXist API versus Drools;

---

<sup>7</sup> Documents of the OASIS LegalRuleML TC are available at [https://www.oasis-open.org/committees/documents.php?wg\\_abbrev=legalruleml](https://www.oasis-open.org/committees/documents.php?wg_abbrev=legalruleml). The mailing list describing the work in progress can be browsed at <https://lists.oasis-open.org/archives/legalruleml/>.

3) to extend the Drools environment with the following modules: i) to import in a more effective way in Drools the LegalRuleML rules and to export from the editor them in the LegalRuleML syntax (when fixed by the OASIS TC); ii) to manage defeasibility in a more abstract way, not embedded in the Drools code; iii) to implement an abstract legal temporal model of reasoning. The goal is also to understand if we can reach the same performance and expressiveness of SPINdle [20] using some Drools extensions; 4) to refine the front-end interface for the rule viewer (e.g., graphs representation) and to improve the Rest/API from and to Drools.

**Acknowledgement.** A particular thanks go to Guido Boella, University of Turin, for the precious inputs on the research topics and on the organization of the paper for improving readability. I would like to thank Llio Humphreys for the proof-reading, not limited to the English matter but fruitfully integrated with her competences in the legal informatics and logic.

## References

- [1] Ashley K. D.: Ontological requirements for analogical, teleological, and hypothetical legal reasoning. In: ICAIL 2009, pp. 1-10, 2009.
- [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] Bekiari C., Doerr M. and Le Boeuf P.: International Working Group on FRBR and CIDOC CRM Harmonization. 2008. FRBR object-oriented definition and mapping to FRBRER (v. 0.9 draft). Accessed 20 August 2009.  
[http://cidoc.ics.forth.gr/docs/frbr\\_oo/frbr\\_docs/FRBR\\_oo\\_V0.9.pdf](http://cidoc.ics.forth.gr/docs/frbr_oo/frbr_docs/FRBR_oo_V0.9.pdf).
- [4] Bench-Capon T. and Coenen F.: Isomorphism and legal knowledge based systems. *Artificial Intelligence and Law*, 1(1):65–86, 1992.
- [5] 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.
- [6] Boella G., Humphreys L., Martin M., Rossi P., and van der Torre L.: Eunomos, a legal document and knowledge management system to build legal services. In: Proceedings of AI Approaches to the Complexity of Legal Systems Workshop (AICOL), Berlin, Springer, 2012.
- [7] 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>.
- [8] Boer A., Radboud W., Vitali, F.: MetaLex XML and the Legal Knowledge Interchange Format. In: Casanovas P., Sartor G., Casellas N., Rubino R. (eds.), *Computable Models of the Law*, Springer, Heidelberg (2008), pp. 21-41, 2008..
- [9] Bragaglia S., Chesani F., Ciampolini A., Mello A., Montali M., Sottara D.: An Hybrid Architecture Integrating Forward Rules with Fuzzy Ontological Reasoning. *HAIS* (1) 2010: pp. 438-445, 2010.
- [10] 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.
- [11] 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.
- [12] 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.
- [13] 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.
- [14] 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.
- [15] <http://www.akomantoso.org/> naming convention of the URI
- [16] <http://www.ifla.org/publications/functional-requirements-for-bibliographic-records>



- [17] Hu Y., Wu W., Cheng D.: Towards law-aware semantic cloud policies with exceptions for data integration and protection. WIMS 2012: 26, 2012.
- [18] Karam N., Paschke A.: Patent Valuation Using Difference in ALEN. Description Logics 2012, 2012.
- [19] Kelsen H.: *Reine Rechtslehre*, 2d. ed., Wien, 1960.
- [20] Lam H., Governatori G.: The Making of SPINDle. RuleML 2009 proceeding, pp. 315-322, 2009.
- [21] 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.
- [22] Mazzei A., Radicioni D., Brighi R.: NLP-based extraction of modificatory provisions semantics. ICAIL 2009: pp. 50-57, ACM, 2009.
- [23] 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.
- [24] 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.
- [25] Palmirani M., Brighi R.: Model Regularity of Legal Language in Active Modifications. AICOL Workshops 2009: pp. 54-73, Springer, 2009.
- [26] Palmirani M., Cervone L., Vitali F.: Legal metadata interchange framework to match CEN metalex. ICAIL 2009, pp. 232-233, 2009.
- [27] 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.
- [28] Palmirani M., Contissa G., Rubino R: Fill the Gap in the Legal Knowledge Modelling. In *Proceedings of RuleML 2009*, pp. 305-314, Springer, 2009.
- [29] 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.
- [30] Proceeding of the 13th International Conference on Artificial Intelligence and Law, Pittsburgh 6-10 June, 2011, ACM, NY, 2011.
- [31] Sartor G.: Legal Concepts as Inferential Nodes and Ontological Categories. In *Artif. Intell. Law* 17(3) 2009, pp. 217-251, 2009.
- [32] Sartor G.: *Legal Reasoning: A Cognitive Approach to the Law*. Vol. 5. *Treatise on Legal Philosophy and General Jurisprudence*. Berlin: Springer, 2005.
- [33] Sartor G.; Palmirani M.; Francesconi E.; Biasiotti M. (eds.): *Legislative XML for the Semantic Web. Principles, Models, Standards for Document Management*, Dordrecht/Heidelberg/London/New York, Springer, 2011, *Law, Governance and Technology Series*, Vol. 4., 2011.
- [34] Sottara D., Mello P., and Proctor M.: A configurable rete-oo engine for reasoning with different types of imperfect information. *IEEE Trans. Knowl. Data Eng.*, 22(11): pp. 1535-1548, 2010
- [35] Vitali F., Palmirani M.: Akoma Ntoso Release Notes. [<http://www.akomantoso.org>]. Accessed 20 June 2012.
- [36] Wilcock G.: *Introduction to Linguistic Annotation and Text Analytics* Morgan & Claypool Publishers 2009.

## Annex

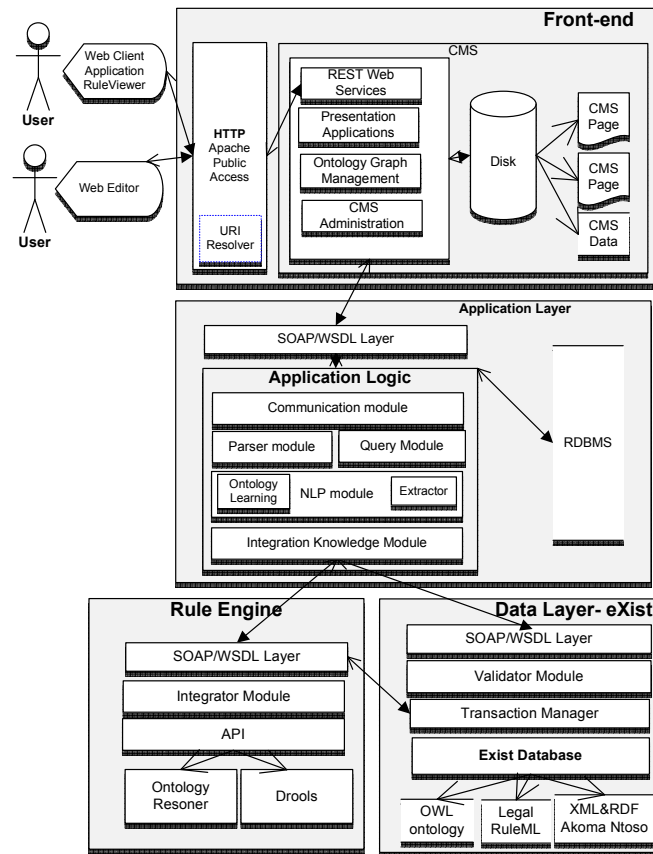


Fig. 13 – System Architecture

# Browsing case-law: an Application of the Carneades Argumentation System

Marcello Ceci<sup>1</sup>, Thomas F. Gordon<sup>2</sup>

<sup>1</sup> CIRSFD, University of Bologna, Italy

<sup>2</sup> Fraunhofer-FOKUS Institut, Berlin, Germany

[m.cecchi@unibo.it](mailto:m.cecchi@unibo.it)

[thomas.gordon@fokus.fraunhofer.de](mailto:thomas.gordon@fokus.fraunhofer.de)

**Abstract.** This paper presents an application of the Carneades Argumentation System to case-law. The application relies on a set of ontologies – representing the core and domain concepts of a restricted legal field, the law of contracts – and a collection of precedents taken from Italian courts of different grades. The knowledge base represents the starting point for the construction of rules representing laws and precedents which, in turn, are responsible for the argumentative reasoning. The system reconstructs the legal interpretations performed by the judge, presenting its reasoning path and suggesting possible different or divergent interpretation in the light of relevant code- and case-law.

**Keywords:** AI&Law, Legal Argumentation, Semantic Web, Legal Ontology.

## 1 Introduction

Precedent is a main element of legal knowledge worldwide: by settling conflicts and sanctioning illegal behaviours, judicial activity enforces law provisions within the national borders, therefore supporting the validity of laws as well as the sovereignty of the government that issued them. Representing the content of case-law, in terms of legal concepts taken into consideration and interpretations performed by the judge, is a very interesting task for the IT research, and the AI & LAW community has presented very significant outcomes in this topic since the '80, with different approaches: legal case-base reasoning (HYPO, CATO, IBP, CABARET), legal concepts representation through logics [1], rule interchange for applications in the legal domain [8] and more recently also argumentation.

### 1.1. Carneades

Carneades<sup>1</sup> [10] is a set of open source software tools for mapping and evaluating arguments, under development since 2006. Carneades contains a logical model of argumentation based on Doug Walton's theory of argumentation, and developed in

---

<sup>1</sup> <http://carneades.berlios.de>

collaboration with him. In particular, it implements Walton's argumentation schemes [11] not only to reconstruct and evaluate past arguments in natural language texts, but also as templates guiding the user as he/she generates his/her own arguments graphs to represent ongoing dialogues. It can therefore be used for studying argumentation from a computational perspective, but also to develop tools supporting practical argumentation processes. The main application scenario of Carneades is that of dialogues where claims are made and competing arguments are put forward to support or attack these claims [14], but it also takes into account the relational conception of argument<sup>2</sup> [4].

## 1.2. The application to case-law: objectives and methodology

In the present application, Carneades' potentialities will be exploited to conduct reasoning on case-law, whose knowledge has been previously modeled in an OWL/RDF ontology and in a set of rules (in LKIF-Rule language [2]).

The goal of the present approach is to define a framework for case-law semantics, exploiting Semantic Web technologies to "fill the gap" between document representation and rules modelling [13], ensuring isomorphism between the text fragment (the only binding legal expression) and the rule. Cornerstone of the framework is the ontology, intended in its computer science meaning: a shared vocabulary, a taxonomy and axioms which represent a domain of knowledge by defining objects and concepts together with their properties, relations and semantics. We believe that the features of OWL2 could greatly improve legal concepts modelling and reasoning, once properly combined with rule modelling. Our aim is hence to formalize the legal concepts and the argumentation patterns contained in the judgment in order to check, validate and reuse the elements of judgement as expressed by the text and the argumentation contained in it.

To achieve this, four models are necessary:

- a *document metadata structure*, capturing the main parts of the judgment to create a bridge between text and semantic annotation of legal concepts;
- a *legal core ontology*, describing the legal domain's main elements in terms of general concepts through an LKIF-Core extension;
- a *legal domain ontology*, modelling the legal concepts of a specific legal domain concerned by the case-law, including a set of sample precedents;
- *argumentation modelling and reasoning*, representing the structure and dynamics of argumentation.

Three kinds of knowledge are modelled in the ontology and the rules: legal rules (laws and other authoritative acts), case-law (precedents and their relevant interpretations), and material circumstances (the object of the judgment: i.e., a contract clause, a human behaviour, an event).

---

<sup>2</sup> The main difference between the two conceptions is that a proposition which has not been attacked is acceptable in the relational model of argument, while in most dialogues it would be not acceptable, since in most schemes making a claim involves having the burden of proof on it.

Aim of this application is to create a reasoning environment allowing a high level of human-machine interaction: the user can start from some basic concept (a legal concept, a fact, an exception, a law prescription) and query the database (which is contained in the OWL ontology) to get some “pilot cases” in return. The user can then ask about the outcome of the pilot case, and about the main interpretations made by the judge in his decision: these are presented in a graph which shows not only the logical process followed by the judge and the laws which he applied but also those who could be, and the precedents which – if accepted - could lead to a different judgement.

This is possible thanks to the mix of OWL/DL reasoning, semantically managing static information on the elements of the case, and rule-based defeasible reasoning, which is ought to represent the dynamics of norms and judicial interpretations.

The present approach focuses on the "argument from ontology" feature of Carneades [6]: the program is in fact capable of accepting (or rejecting) the premises of arguments on the basis of the knowledge contained in some imported OWL/RDF ontology (see below 3.1). This allows to build complex argumentation graphs, where the argument nodes represent legal rules and the statements are accepted or rejected on the basis of knowledge coming from the ontology and/or data inserted by the user.

In this perspective, the Carneades argument graph may either represent:

- a reconstruction of a judicial decision's contents in terms of laws applied, factors taken into considerations, interpretations performed by the judge. The conclusion of the argumentation represents the final adjudication of the claim, and the Carneades reasoner is expected to accept or reject the claim by semantically applying the judicial interpretations contained in the decision's groundings (this is the kind of representation which will be shown in the present application);
- a collection of argumentations paths leading to a given legal statement (such as "contract x is inefficacious"). On the basis of manually-inserted statements concerning the object of the case (statuses or factors concerning the material circumstance, i.e. contract x) the Carneades reasoner suggests possible argumentation paths leading to (the acceptance or rejection of) the desired legal statement.

In both cases, however, the system presents to the user not only argumentation paths which have been proved as valid (i.e. rules whose conditions have all been met), but also possible, incomplete argumentation paths where one or more of the premises is still undecided: under this perspective, the tool presents to the user a semantic environment where different laws, legal statuses and precedents are semantically related to each other.

From that point, the user can go further by querying the knowledge base to retrieve precedents where similar (or different) interpretations are made: in this way, he can realize which differences – if any – exist between two or more precedents. It is like browsing case-law in a law journal in order to compare different decisions, but in the Carneades environment this can be done directly with legal concepts, not only to verify a combination of circumstances and laws under a logical point of view, but also

to receive suggestions from the system on which law, precedent or circumstance could lead to a different outcome.

## **2 The case-law argumentation ontologies**

The “Core and Domain Legal Ontologies” are two OWL/RDF ontologies conceived to model the semantics of judicial interpretations, developed in the context of a research on judicial knowledge modelling [12].

### **2.1. The Core Ontology**

The core ontology (an extension of the LKIF-Core Ontology) introduces the main concepts of the legal domain, defining the classes which will be later filled with the metadata of judicial decisions. Even though the core ontology should be domain-generic and not modeled upon a specific legal subject, the sample model was conceived only to successfully represent the interaction in the civil law subject, when contracts, laws and judicial decisions come into play. An important modeling choice is the reification of legal statuses and the creation of the object property “applies” (subproperty of LKIF’s “qualifies”) to link a material circumstance to its status.

### **2.2. The Domain Ontology**

Following the structure outlined in the Core Ontology, the metadata taken from judicial documents are represented in the Domain Ontology. The modeling was carried out manually by an expert in the legal subject (a graduated jurist), which actually represents the only viable choice in the legal domain: automatic information retrieval and machine learning techniques, in fact, do not yet ensure a sufficient level of accuracy. Building a domain ontology is similar to writing a piece of legal doctrine, thus it should be manually achieved in such a way as to maintain a reference to the author of the model, while at the same time keeping an open approach. Here, the property “applies” is used to link the fact of the cases to its legal statuses, in such a way as to give evidence to interpretations made by the judge (represented by the object property “judged\_as”, subproperty of “applies”).

### **2.3. Features of the Ontology Set**

The so-built layered ontology creates an environment where the knowledge extracted from the decision’s text can be processed and managed, in such a way as to enable a deeper reasoning on the interpretation instances grounding the decision itself. Example of this deeper reasoning include: finding relevant precedents which were not explicitly cited in the decision; finding anomalies in the evaluation of material circumstances, in the light of cited precedents and similar cases; validating the adjudications of the judge on the claims brought forward by the parties during the trial on the basis of applicable rules, accepted evidence and interpretation; suggesting possible weak spots in the decision’s groundings; suggest possible appeal grounds and legal rules/precedents/circumstances that could bring to a different application of the rules and/or to a different adjudication on the claim.

The layered structure of the ontology set allows an efficacious scaling from legal concepts to factors, up to dimensions and legal principles: all these concepts can be represented in the domain ontology, and the hook of the core concepts to LKIF-Core

should ensure a good semantic alignment between different domain ontologies, as far as different authors are concerned (even if, at the present time, this research has not yet evaluated the alignment capabilities of the present ontology set).

#### **2.4. Changes made to the ontology set for the present application**

This application of Carneades involved the following modifications for the Core and Domain ontologies [3]:

- enriching the semantic content of the ontology set by representing finer-grained knowledge contained in the decision's text, in an environment where this expansion of the knowledge base does not entail an overloading of the OWL reasoners, which would compromise computability;
- modeling a rule system representing the dynamic relationships created by judicial interpretation and law application;
- importing knowledge from the ontology set in such a way as to allow successful interaction with the rule set and the Carneades model.

### **3 Constructing Arguments with Carneades**

Carneades is a tool which relies on a solid background theory involving Walton's argumentation schemes with an articulated conception of the burden of proof and of its allocation [9]. However, even though the present application takes advantage from the strong conceptual foundations of this tool, these two features will be set aside. The main reason for this is that (as of Carneades version 1.0.2) these functions are not implemented in the automatic argument construction process, which itself represents the pivot of this experiment. Particularly for the burden of proof this is unfortunate, since the present experiment would gain a lot of depth from an automatic management of burden of proofs and proof standards.

While the new version of Carneades (2.x, currently under development) will use the Clojure language, the latest complete version (1.0.2, the one used for the present application) relies on the LKIF-Rule language [7].

The features of Carneades which are thoroughly used in the present application are the ontology import module, the argument construction module, and the argument visualizer.

#### **3.1. Ontology Import Module**

The ontology import module allows the system to automatically import (stated and inferred) knowledge coming from any OWL/RDF ontology [6]. This knowledge is then used by Carneades during its argumentation process, in order to accept those assertions inside the argument graph which are recorded as true into the (stated or inferred) ontology. What the Carneades ontology import feature cannot do, as of version 1.0.2, is to show the reasoning process, followed by the Hermit reasoner<sup>3</sup> to

---

<sup>3</sup> HermiT is reasoner for ontologies written using the Web Ontology Language. Given an OWL file, a reasoner can perform tasks such as determining whether or not the ontology is consistent, identifying subsumption relationships between classes, inferring new knowledge.

infer those assertions. The ontology import is instead a black box: in the argument visualizer, assertions accepted on the basis of knowledge coming from the ontology actually appear as accepted under the condition that the whole ontology is valid, without any argumentation being provided nor a distinction between stated and inferred knowledge being made. This means that, at the present time, only the reasoning coming from the application of rules to the knowledge base is showed in the visualizer: nevertheless, having solved some modeling issues on this behalf, this is already enough to show the core part of the interpretation process followed by the judge.

### **3.2. Argument Construction Module**

The argument construction module relies on rules, ontologies and manually inserted statements to construct an argumentation tree trying to answer any query concerning a precedent contained in the knowledge base [10]. The target statement can be of two types: a "query-like" statement (i.e.  $?x$  Oppressive\_Clause, which means "give me all  $x$ , where  $x$  is an oppressive clause") or a simple assertion (ME/LaSorgente\_Clause8 Oppressive\_Clause, which means "Clause 8 of the contract between M.E. and La Sorgente is an oppressive clause"): in the first case, the system will return a list of results and arguments, while in the second case the system will construct a single argumentation tree towards the desired goal (pro or con, which means towards the acceptance or the rejection of the target statement).

### **3.3. Argument Visualizer**

The argument visualizer is a powerful tool that allows the user to manage the argumentation process (manually adding assumptions, accepting/rejecting statements, cutting out parts of reasoning, and so on) in order to investigate the concepts of the case in details and relevant case law, not by comparing their textual appearance, but rather by manipulating the concepts expressed in them [5].

## **4 The Scenario: Consumer Law**

In this demonstration of Carneades application, we will examine a piece of Italian Consumer Law<sup>4</sup> by:

- retrieving the case-law concerning a legal concept;
- analyzing the interpretations made by the judge on that case;
- searching possible alternative solutions to the case.

Please notice that, in order to simplify the understanding of the graph, in this presentation the search for arguments will be "progressive": this means that we will ask the reasoner to conduct argumentation on statements only to a limited extent, in order to unveil the argumentation graph step by step. This is done by limiting the number of nodes that the reasoner can follow before returning results. By raising this value, it is possible to obtain more detailed or complex graphs in a single step, either

---

<sup>4</sup> Actually regulated through legislative decree n. 206 of September 6th, 2005 - even though some cases still fall under artt. 1341 and 1342 of Italian Civil Code, which are still in force.



by deepening the search, to resolve exceptions and indirect argumentation paths in a single step, or by broadening it, to search for more "daring" argumentation paths.

#### 4.1. Modelling of the law

Preliminarily, the two norms involved in the reasoning will be presented:

**Article 1341 comma 1 of Italian Civil Code** – General contract clauses which have been unilaterally predisposed by one of the contract parties are efficacious only if they were known by the other contract party, or knowable by using ordinary diligence.

**Article 1341 comma 2 of Italian Civil Code** – Clauses concerning arbitration, competence derogation, unilateral contract withdrawal, and limitations to: exceptions, liability, responsibility, and towards third parties, are inefficacious unless they are specifically signed by writing.

The modelling of this information is based on both the ontology and the rules. In particular, the ontology contains "static" information on the law (such as the enacting authority, the subject, the legal concepts contained in the text, the URI of the legal expression), while the rules classify the material circumstances (in this case, the contract clauses) which share certain legal statuses as being relevant under that law. This is an example of a rule stating the relevancy for comma 2 of Article 1341:

```
<!ENTITY oss "http://www.semanticweb.org/ontologies/2011/8/proval.owl#">
<rule id="LAW_Art1341co2">
  <head>
    <s pred="Relevant_ExArt1341co2"><v>C1</v> falls under the
      discipline of Article 1341 comma 2 of Civil Code </s>
    <s pred="&oss;considered_by"><v>C1</v> falls under the
      discipline of <i value="&oss;Art1341co2cc">Article 1341 comma 2
        of Civil Code </i></s>
  </head>
  <body>
    <s pred="&oss;applies"><v>C1</v> applies <v>S1</v> </s>
    <s pred="&oss;Oppressive_Status"><v>S1</v> is an oppressive
      status</s>
    <s pred="&oss;applies"><v>C1</v> applies <i
      value="&oss;General">general status</i> </s>
    <s pred="&oss;applies"><v>C1</v> applies <i
      value="&oss;Unilateral">unilateral status</i> </s>
    <not>
      <s pred="&oss;applies"><v>C1</v> not applies <i
        value="&oss;SpecificallySigned">specifically signed</i>
      </s>
    </not>
  </body>
</rule>
```

Please notice that the rule does not contain the list of statuses, but rather refers to a class of “Oppressive statuses” (a naming acknowledged by the legal doctrine), whose modelling is left to the ontology (Fig. 1). This distribution in the representation of the law allows an open organization of legal knowledge, while

Description: Oppressive_Status	
SubClass Of (Anonymous Ancestor)	
Members +	
◆	ArbitrationAgreement
◆	CompetenceDerogation
◆	ContractWithdrawalUnilateral
◆	Exception_Limitation
◆	LiabilityLimitation
◆	LimitationTowards3rdParties
◆	ResponsibilityLimitation

Fig. 1. Members of the Oppressive\_Status class

at the same time keeping the full expressivity of the rule syntax.

The rule presented above only states which circumstances are subsumed under that legal rule; successively, another rule comes into play, verifying if any exceptions to the general rule apply. If not, the consequence of the legal rule (in this case, inefficacy) is related to the circumstance (the contract clause):

```
<rule id="LAWCONS_Inefficacy rule">
  <head>
    <s pred="&oss;Inefficacious"> <v>C1</v> Is inefficacious:
      has no effects </s>
  </head>
  <body>
    <and>
      <or>
        <s pred="Relevant_ExArt1341co1"><v>C1</v> falls under the
          discipline of Article 1341 comma 1 of Civil Code </s>
        <s pred="Relevant_ExArt1341co2"><v>C1</v> falls under the
          discipline of Article 1341 comma 2 of Civil Code </s>
        <s pred="Relevant_ExArt1342co2"><v>C1</v> falls under the
          discipline of Article 1342 comma 2 of Civil Code </s>
      </or>
      <not exception="true">
        <s pred="&oss;applies"><v>C1</v> applies <i
          value="&oss;ReproducingLawDisposition"> a law
            disposition</i> </s>
      </not>
      <not exception="true">
        <s pred="&oss;applies"><v>C1</v> applies <i
          value="&oss;International"> an international
            agreement</i> </s>
      </not>
    </and>
  </body>
</rule>
```

## 4.2. Modelling of the contract

The material circumstances which are taken into account by a precedent are modelled only in the domain ontology (not in the rules), and semantically classified depending on its characteristics (in the case of a contract clause: containing contract, contract parties, object of the contract, object of the clause). Two different properties represent the relation between a circumstance (the clause) and a legal status: "applies", which means that the status has been recognised by both parties as applicable to the circumstance, and "judged\_as", which means that status has been interpreted as being applicable to the circumstance by a judge.

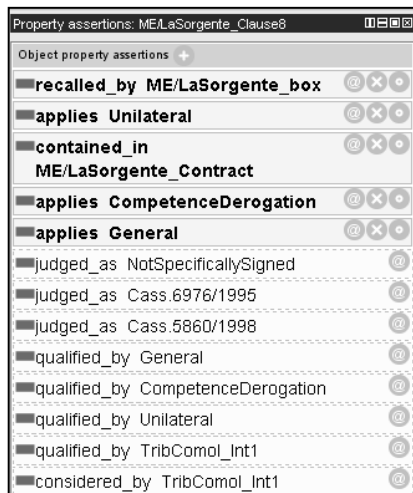


Fig. 2. Properties of a contract clause instance

So, for example, the contract clause ME/LaSorgente\_Clause8 (Clause 8 of the contract between M.E. and "La Sorgente") has the characteristics indicated in Fig. 2.

We can see that "Unilateral", "CompetenceDerogation", "General" are three characteristics which are acknowledged by both parties; "recalled\_by ME/LaSorgente\_box" represents a relation of this clause with another part of the contract; the "considered\_by" property links to the precedent (and therefore the authority) which produced the subsumptions: a legal status ("NotSpecificallySigned") and two precedents which have been cited (Cass. 6976/1995 and Cass. 5860/1998). In the next chapter we will see how this knowledge is managed in Carneades. Finally, please notice that the clause also applies the status of "CompetenceDerogation", an Oppressive\_Status.

### 4.3. Modelling of the case-law

The judicial interpretation instances are modelled both into the domain ontology and into the rules. The domain ontology contains static knowledge (enacting authority, object of the case, classification, a URI) as well as "surface" information such as those presented above: the circumstance taken into consideration, the legal status under which the circumstance is subsumed, the precedents cited.

The mechanics underlying the judicial interpretation are contained in the rules. This is an example of a rule representing a judicial subsumption:

```
<rule id="JINT_RecallNonOppressiveClauses">
  <head>
    <not>
      <s pred="&oss;applies"><v>C1</v> doesn't apply <i
        value="&oss;SpecificallySigned"> specifically signed
        status </i></s>
    </not>
    <s pred="RecallException"> <v>C1</v> is subject to the
      exception</s>
  </head>
  <body>
    <s pred="&oss;recalled_by"><v>C1</v> recalled by <v>B1</v> </s>
    <s pred="&oss;hasfactor"><v>B1</v> has factor <i
      value="&oss;RecallsNonOppressiveClauses"> recalls also non
      oppressive clauses</i></s>
    <s assumable="true" pred="&oss;judged_as"><v>C1</v> applies<i
      value="&oss;Cass.5860/1998">precedent Cass.5860/1998</i></s>
  </body>
</rule>
```

Please notice the particular role given to the precedent "Cass.5860/1998": it is an assumption, so it does not prevent the system from suggesting this particular interpretation (and the precedent) as a result of the reasoning process on cases which share the other conditions, even if that precedent is not explicitly recalled. At the same time, if the precedent is directly cited in the case, the system is capable of putting a stronger accent on that interpretation, not only by assuming its applicability but by directly stating it.

## 5 Carneades application

### 5.1. Reconstructing Precedents and their reasoning

Let's suppose we have a contract clause which - we are afraid - is oppressive. We start by querying the system (the ontologies and the rules) to retrieve a list of contract clauses (object of precedents) which have been considered oppressive - either as an undisputed fact or following a judicial interpretation (fig. 3).

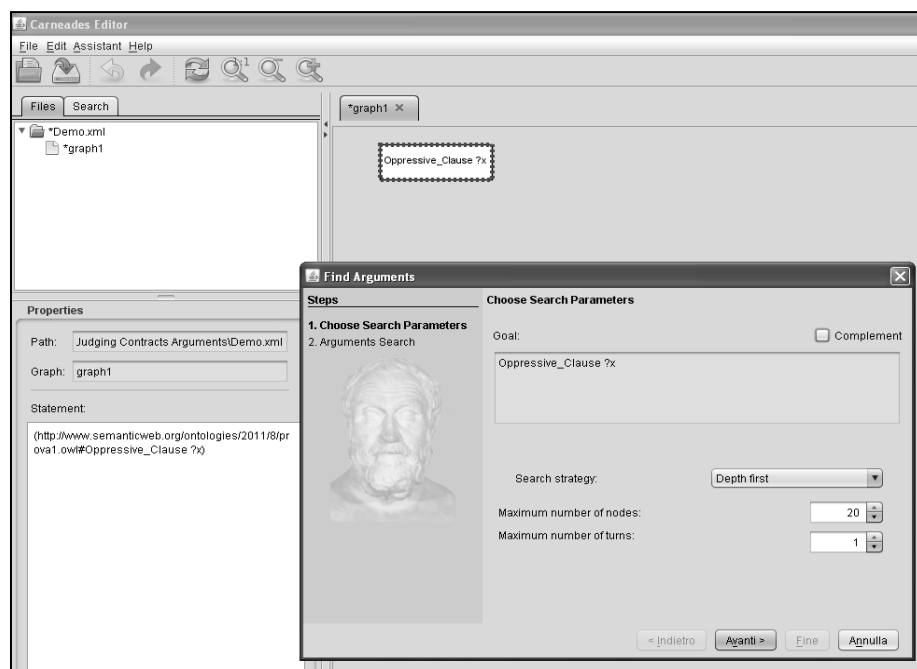
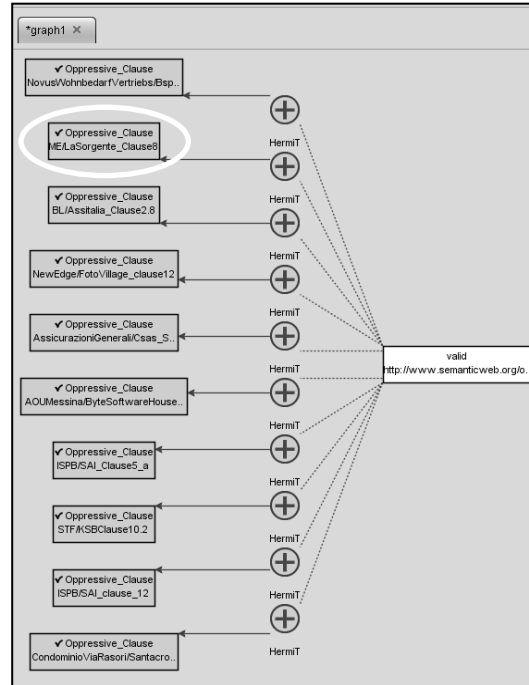


Fig. 3. The “Find Argument” window of Carneades

We take the second example: ME/LaSorgente\_Clause8. Please notice that this instance does not represent a legal case, but rather a contract clause. Information contained in the ontology allows us to retrieve in any moment the details concerning the case (court, date, parties, the decision's text, and so on) but no such information will be shown on the argumentation tree: in it, we will only find the contract clause and other material circumstances related to it (the contract it is contained in, for example), legal statuses, and interpretation instances: only the elements which are relevant to the dynamics of the judicial argumentation, while those who pertain to the identification of the

[illegible]

The system found two applicable laws to argument the inefficacy of the clause (left part of fig. 5): if the conditions of one of these two laws are met, and no exception exists (in this case, possible exceptions - broken lines - are the contract being an

international contract, and the contract reproducing law dispositions), the clause is inefficacious. The requirements for a clause to be relevant under one of these two laws are presented in the central part of fig.5.

In order for the clause to be relevant under Article 1341co1, it must be general, unilateral and not knowable by the other party by using ordinary diligence (please notice that, due to a mistake, knowable appears as "knowledgeable" in the figures).

In order for the clause to be relevant under Article 1341co2, it must be general, unilateral, oppressive and not specifically signed.

The clause was found to be oppressive (since a clause concerning competence derogation is part of the list of article 1341co2), general and unilateral (dark boxes). These statements come directly from the ontology set's knowledge base, which means

that the relative information has been manually inserted in (or inferred by) the database and it is not possible to further explain those positions ( the "HerMiT" argument followed by dotted lines pointing to the premise "valid [ontologyURI]"). The search was not deep enough to determine whether this clause is specifically signed or not, nor whether it was knowable or not (white boxes).

The next step is to ask Carneades to produce argumentation towards the acceptance/rejection of the yet undecided statements. We start asking to find arguments PRO the clause being specifically signed (Fig. 6):

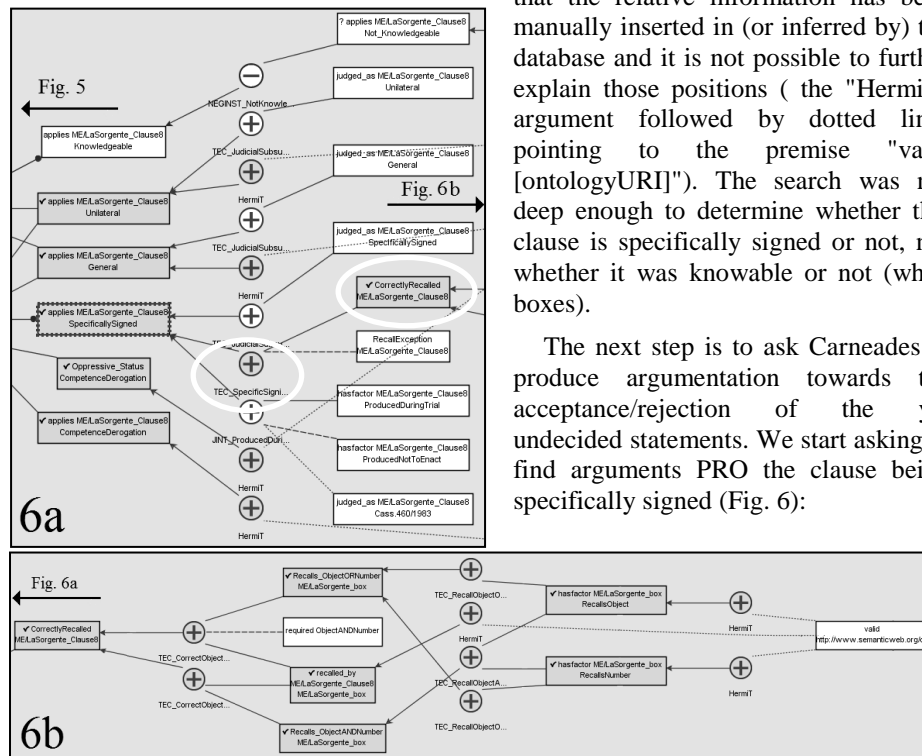


Fig. 6 (a-b). Argumentation PRO the clause being specifically signed.

Fig. 6a shows that the requirement for a judicial interpretation towards the specific signing of the clause is met: the clause is "correctly recalled" and therefore can be considered as specifically signed following some precedent (unless the "recall exception" applies).

Fig. 6b explains why the "correctly recalled" premise has been marked as "accepted" by Carneades: the ME/LaSorgente contract contained a distinct box







## 5.2. Suggesting New Argumentation Paths

Finally, we provide a brief example of how Carneades can suggest new interpretation for known facts, in the light of existing norms and relevant precedents. We consider the same clause (ME/LaSorgente\_Clause8) and ask the system to go through the analysis of its relevance under article 1342co2 (see fig. 1). Under that perspective, the clause lacked an acceptable argument PRO or CON its knowability. We try to find some arguments PRO the knowability and find out that Carneades has noticed the signed box which recalls clause 8. Using a different legal reasoning, he takes into account the rule JINT\_KnownDocumentRecall, introduced by TribPiacenza2.1 (a decision which is inside the knowledge base of the Domain Ontology). Following this interpretation, if a clause is recalled by a document which is known to the parties, the clause has to be considered knowable. Carneades also found out that the distinct box is a contractual agreement, and therefore presumes (dotted green line) that the interpretation TribPiacenza2.1 can be applied to the case. This specific interpretation of the case ME/LaSorgente was not contained in the decision text, which does not talk about the profile of knowability (since there is no judged\_as property linking the clause to one of its I/O statuses) and neither it cites the precedent of TribPiacenza2.1, but nevertheless Carneades suggested this way of arguing PRO the target statement.

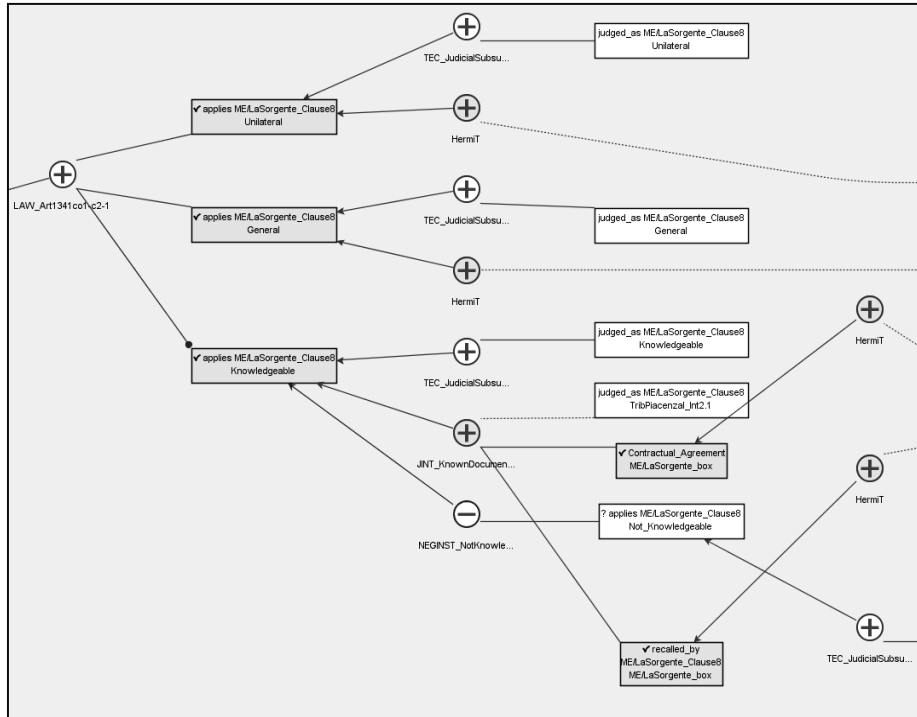


Fig. 10. Suggested argumentation PRO the clause being knowledgeable.

This could give hints on the practicability of such a strategy in a similar case, or arise comments on the difference between the legal concepts of "knowability" and "specific signing" in the terms of the relevance of the number and kind of clauses which are recalled in a separate document.

## **6 Other Applications**

The example only showed how to model one single case, albeit giving the idea of which different directions can be taken from there. In this environment, it is in fact possible to conduct many and more complex activities even with the small number of cases already contained in the OWL knowledge base: it is possible to query precedents (such as TribPiacenza2.1) in order to understand the characteristics of the case, and to compare them to other precedents or to a new case; it is possible to query about the relevancy of a clause having certain characteristics under a specific norm or judicial interpretation, and those characteristics can be either manually inserted as statements in the argumentation graph, or automatically extracted from precedents or knowledge bases (even outside the Domain Ontology). Finally, the defeasible logics behind the reasoner and the solid proof standards system - if properly implemented in the rule engine, and accompanied by an improvement in the reasoner's computability - can allow a complete analysis of possible exceptions to rules and interpretations, through a real "positions searching" activity: the system takes - in turns - the part of the attacker and the defendant and produces arguments PRO and CON the given statement.

## **7 Conclusions**

The Carneades application presented here was intended to show how an argumentation system can be used to process semantic data in a complex way. The arguments construction and the rules representing code- and case-law could never meet their full potentialities if not supported by a semantically rich knowledge base, such as the "legal ontology set" presented in section 2.

The present application therefore represents a demonstration of how a shared logics and syntax for legal rule representation, combined with a standard core ontology for legal concepts, would constitute the ideal starting point for a totally new conception of case-law classification, browsing and management.

The application represents an advancement of the work presented at the RuleML Challenge 2011 [6] because it creates a complete juridical environment in order to achieve a real benchmark of Carneades' capabilities: the sample (constituted by 27 precedents) has been completely represented in the ontology set and in the rules, thus heavily stressing the Carneades and OWL reasoners and showing their limits in terms of computability. Moreover, the ontology set used in the present application was not specifically modelled upon Carneades, rather representing an effort towards a standard representation of legal text's contents which ensures isomorphism with the source document and interoperability with different applications in the rules and logics layers.

## References

1. Boella G., Governatori G., Rotolo A., van der Torre L., A Logical Understanding of Legal Interpretation. In: KR 2010.
2. Boer, A., Radboud, W., Vitali, F., MetaLex XML and the Legal Knowledge Interchange Format. In: Casanovas, P., Sartor, G., Casellas, N., Rubino, R. (eds.), *Computable Models of the Law*, Springer, Heidelberg, pp. 21-41 (2008).
3. Ceci M., Combining Ontology and Rules to Model Judicial Interpretation, RuleML 2012 Doctoral Consortium (under submission).
4. Dung, P.M.: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. In: *Artificial Intelligence*, 77(2), pp. 321-357 (1995).
5. Gordon, T.: Visualizing Carneades argument graphs. In: *Law, Probability and Risk*, 6(1-4), pp. 109-117 (2007).
6. Gordon, T.: Combining Rules and Ontologies with Carneades. In: *Proceedings of the 5th International RuleML2011@BRF Challenge*, CEUR Workshop Proceedings, pp. 103-110 (2011).
7. Gordon, T.: Constructing Legal Arguments with Rules in the Legal Knowledge Interchange Format (LKIF). In: Casanovas, P.: *Computable models of the law: Languages, dialogues, games, ontologies*. Berlin: Springer. (Lecture Notes in Artificial Intelligence 4884), pp. 162-184 (2008).
8. Gordon, T., Governatori, G., Rotolo, A., Rules and Norms: Requirements for Rule Interchange Languages in the Legal Domain. In: *Rule Interchange and Applications, International Symposium, RuleML 2009, BERLIN*, Springer pp. 282 - 296 (2009).
9. Gordon, T., Prakken, H., Walton, D.: The Carneades model of argument and burden of proof. In: *Artificial Intelligence*, Vol.171, No.10-15, pp. 875-896 (2007).
10. Gordon, T., Walton, D.: The Carneades Argumentation Framework: using presumptions and exceptions to model critical questions. In: Dunne, P.E.: *Computational Models of Argument. Proceedings of COMMA 2006: 1st International Conference on Computational Models of Argument*, The University of Liverpool, UK, 11th - 12th September 2006. IOS Press, Amsterdam (2006).
11. Gordon, T., Walton, D.: Legal reasoning with argumentation schemes. In: *Proceedings of the Twelfth International Conference on Artificial Intelligence and Law*, ACM Press, New York, pp. 137-146 (2009).
12. Palmirani, M., Ceci, M.: Ontology framework for judgement modelling, AICOL 2011.
13. Palmirani, M., Contissa, G., Rubino, R.: Fill the Gap in the Legal Knowledge Modelling. In: *Proceedings of RuleML 2009*, pp. 305-314 (2009).
14. Walton, D.: *The New Dialectic: Conversational Contexts of Argument*. University of Toronto Press, Toronto/Buffalo (1998).

# A model driven approach for bridging ILOG Rule Language and RIF

Valerio Cosentino<sup>1,2</sup>, Marcos Didonet del Fabro<sup>3</sup>, Adil El Ghali<sup>1,4</sup>

<sup>1</sup> IBM France

<sup>2</sup> AtlanMod, INRIA & EMN, Nantes, France

<sup>3</sup> Universidade Federal do Paran, Brazil

<sup>4</sup> Lutin UserLab, France

valerio.cosentino@fr.ibm.com, marcos.ddf@inf.ufpr.br,  
elghali@lutin-userlab.fr

**Abstract.** Nowadays many companies run their business using Business Rule Management Systems (BRMS), that offer: a clear separation between decision logic and procedural structure; the ability to modify a rulebase set rather than processes and the reusability of rules across applications. All these factors allow a company to quickly react and align its policies to the ever-changing market needs. Despite these advantages, different BRMSs can be found on the market, each of them implementing a proprietary business rule language (ex.: JBoss uses Drools, IBM uses Ilog Rule Language, etc.). Rule Interchange Format(RIF) is a W3C open standard aiming at reducing the heterogeneity among business rule languages, which makes rules less reusable and interchangeable. Our work is focused on providing an implementation based on a Model Driven approach for bridging Ilog Rule Language to RIF.

## 1 Introduction

The field of BRMS is characterized by a number of normative, open source, or proprietary systems and languages (Ilog JRules, JBoss Drools, etc.), allowing the expression of various solutions to business problems at a high abstraction level, but with heterogeneous sets of capabilities and languages. Rule Interchange Format (RIF [1]) and Production Rules Representation (PRR [2]) are two standard proposals respectively developed at the W3C and at the OMG aiming at providing a level of standardization to the domain. An important difference between RIF and PRR is that RIF is a standard for distribution of shared rules at runtime, whereas PRR is a standard for interchanging design-time rules.

This work presents a case study of bridging a business rule language: the Ilog Rule Language (IRL) to a standard format: the Rule Interchange Format (RIF) and vice versa. We show how to make it possible with the help of Model Driven Engineering (MDE [3]).

RIF is the W3C Rule Interchange Format and it is part of the infrastructure composing the semantic web. It is an XML language for expressing rules that

computers can execute. One of the main goals of RIF is to promote a standard format to interchange rules between existing rule languages.

Because of the serious trade offs in the design of rule language, RIF provides multiple versions, called dialects:

- Core: it is the fundamental RIF language. It is designed to be the common subset of most rule engines
- BLD: it adds to Core dialect logic functions, equality in the then-part, and named arguments
- PRD: adds a notion of forward-chaining rules, where a rule fires and then performs some action (adding, updating or retracting some information)

In this work we have implemented transformations for the Core and PRD dialects. An example of a RIF example is shown in (Fig. 1).

```
Prefix(ex <http://example.com/2008/prd1#>)
(* ex:rule_1 *)
Forall ?customer ?purchasesYTD (
  If And( ?customer#ex:Customer
          ?customer[ex:purchasesYTD->?purchasesYTD]
          External(pred:numeric-greater-than(?purchasesYTD 5000)) )
  Then Do( Modify(?customer[ex:status->"Gold"]) ) )
```

**Fig. 1.** Example of a rule in RIF

IRL is a formal rule language used inside WebSphere ILOG JRules BRMS. It supports the two different levels of a full-fledged BRMS: the technical level targeted at software developers and the business action language targeted at business users. The technical rules of JRules are written in IRL. The complete specification of IRL can be found in the JRules documentation [4] an example of an IRL rule is provided in (Fig. 2).

```
rule rule_1 {
  when{
    customer : customer.Customer(purchaseYTD : purchase_YTD);
    evaluate(purchaseYTD > supportPackIBM.RifUtil.toDecimal("5000"));
  }
  then{
    modify refresh customer {status = "Gold";}
  }
}
```

**Fig. 2.** Example of a rule in IRL

## 2 Model Driven Engineering

The primary software artifacts of the MDE approach are models, which are considered as first class entities. Every model defines a concrete syntax and conforms to a meta-model (or grammar, or schema), which defines its abstract syntax. One of the most common operations applied on models is transformation, which consists in the creation of a set of target models from a set of source models according to specific rules.

In the work presented in this paper three main tools have been used:

- Ecore allows to handle and create meta-models in Eclipse Modeling Framework (EMF) [5]
- TCS (Textual Concrete Syntax) ([6],[7]) is a bidirectional mapping tool between meta-models and grammars. It is able to perform both text-to-model (injections) and model-to-text (extractions) translations from a single specification. TCS is used to map context-free concrete syntaxes to meta-models.
- ATL (Atlas Transformation Language) [9] is a model transformation language specified as both a meta-model and a textual concrete syntax. It allows developers to produce a number of target models from a set of source models by writing rules that define how to create target models from source model elements.

## 3 Transformations

The RIF2IRL transformation takes as input a RIF file, a BOM file (Business Object Model [4]) and a Dictionary model. The output generates an IRL model and the related IRL file, generated by a TCS parser.

From the first input, using a TCS parser, a model conforming to an implementation of the RIF meta-model is extracted. A BOM file is passed to the transformation to resolve the domain information declared inside the RIF file/model. The related BOM model is extracted out of the BOM file by a TCS parser and it conforms to the BOM meta-model implemented in [10]. A Dictionary is added as input parameter in the transformation in order to provide a translation from the element names declared in the RIF file to the corresponding ones appearing in the IRL file. The transformation has been implemented in ATL [8] and it defines how to convert RIF to IRL.

The IRL2RIF transformation takes as input a IRL file and two BOMs and returns a RIF file. From the first input, a model conforming to the IRL meta-model is extracted by means of a TCS parser. The first BOM contains the information of the domain model, while the second one is used to map IRL functions to the built-in RIF functions for handling dates, numbers, strings, etc. The output is a RIF model, that is translated into a RIF file by a TCS parser.

## 4 The demonstration

The IRL2RIF and RIF2IRL transformations will be first demonstrated using a set of simple examples (Appendix A) that illustrate the main functionalities of

the transformation. We will then show a real world application of the transformation using data from the Arcelor use-case studied in the ONTORULE project [11]. In the context of the Rule Xchanger application depicted in the (Fig. 3), we will focus on the transformation of IRL rule edited using JRules into RIF rules that will be executed using the Tight engine.

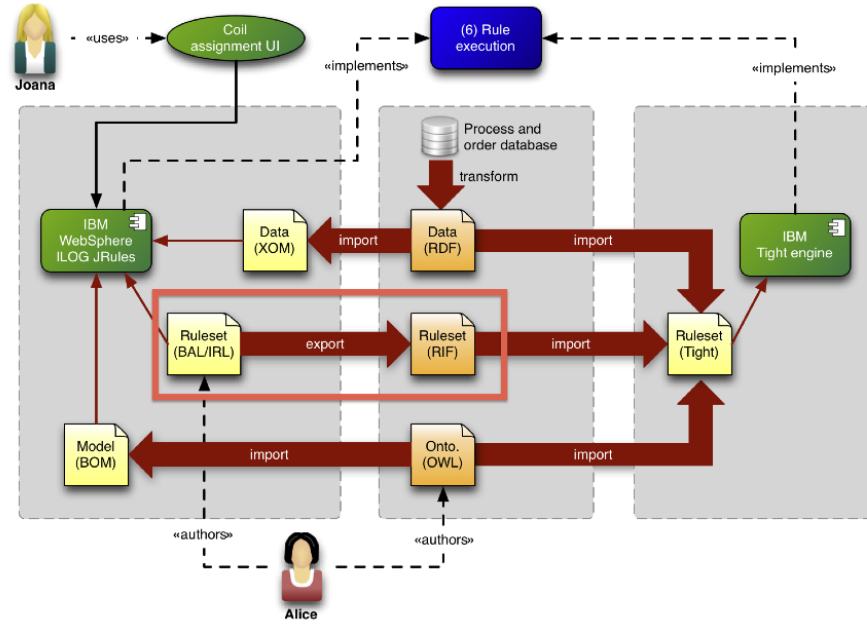


Fig. 3. Rule Xchanger scenario

## 5 Conclusion

The transformation we intend to demonstrate is a key component in the interoperability between two rule languages that has been used in some real world applications such as the Arcelor use-case. It shows that rule application developed using commercial BRMS such as JRules can be exported to other platforms that support the RIF standard.

## References

1. Rule Interchange Format <http://www.w3.org/TR/rif-overview/>
2. Production Rules Representation <http://www.omg.org/spec/PRR/1.0/>
3. Schmidt, D.C. Model-Driven Engineering. IEEE Computer 39 (2), Feb. 2006.

4. JRules documentation <http://pic.dhe.ibm.com/infocenter/dmanager/v7r5/index.jsp>
5. Eclipse Modeling Framework <http://www.eclipse.org/modeling/emf/>
6. Textual Concrete Syntax <http://wiki.eclipse.org/TCS>
7. Jouault F., Bézivin J., Kurtev I., TCS: a DSL for the Specification of Textual Concrete Syntaxes in Model Engineering. In proc. of GPCE'06, Portland, Oregon, USA, pp 249-254
8. Atlas Transformation Language <http://www.eclipse.org/at1>
9. Jouault F., Allilaire A., Bézivin J., Kurtev I. ATL: a Model Transformation Tool. Science of Computer Programming 72(3, Special Issue on Second issue of experimental software and toolkits EST):31-39, 2008
10. Didonet Del Fabro M., Albert P., Bézivin J., Jouault F., Industrial-strength Rule Interoperability using Model Driven Engineering, Technical report 6747, INRIA, Nov. 2008. <http://hal.inria.fr/docs/00/34/40/13/PDF/RR6747.pdf>
11. Gonzalez-Moriyon G., Final steel industry public demonstrators, ONTORULE Deliverable D5.5, Jan. 2012

## A Annex

```

Forall ?x (
  If And (?x#ex:Customer
    ?x[ex:status->"normal"]
    ?x[ex:discount->10] )
  Then Do ( Retract( ?x[ex:discount->10] )
    (Assert ?x[ex:discount->0] ) ))

rule p0_r0 {
  when{

    X : customer.Customer ();
    evaluate((X.status == "normal" && X.discount == 10));
  }
  then{
    X.discount = 0;
  }
}

*****

Forall ?x (
  If And (?x#ex:Customer
    ?x[ex:status->"normal"]
    Then Do ( Modify ( ?x[ex:discount->0] ) ))

rule p0_r0 {
  when{

    X : customer.Customer ();
    evaluate((X.status == "normal"));
  }
  then{
    modify X {discount = 0;}
  }
}

```

```

BOM

package customer;

public class Customer
{
    public int discount;
    public java.lang.String status;
    public Customer();
}

```

**Fig. 4.** Example 1



```

(* calculateFib1 *)
Group (
  Forall ?f such that (?f #ex:Fib)
    (If Exists ?n ?v (And (?f[ex:number->?n]
      ?f[ex:value->?v]
      pred:numeric-equal(?n 1)
      pred:numeric-equal(?v 0)))
    Then Do ( Modify (?f[ex:value]->1))))

package calculateFib1 {
  rule calculateFib1_r0 {
    when(
      f : fib.Fib (n : number; v : value);
      evaluate(n == 1 && v == 0);
    )
    then(
      modify f {value = 1;}
    )
  }
}

*****
(* calculateFib2 *)
Group (
  Forall ?f such that (?f #ex:Fib)
    (If And (Exists ?n ?v (And (?f[ex:number->?n]
      ?f[ex:value->?v]
      pred:numeric-equal(?n 2)
      pred:numeric-equal(?v 0))))
    Then Do ( Modify (?f[ex:value]->1))))

package calculateFib2 {
  rule calculateFib2_r0 {
    when(
      f : fib.Fib (n : number; v : value);
      evaluate(n == 2 && v == 0);
    )
    then(
      modify f {value = 1;}
    )
  }
}

*****
(* MakeRecursiveGoal *)
Group (
  Forall ?f ?f1 such that (?f #ex:Fib)
    (If And (Exists ?n ?v ?n1 (And (?f[ex:number->?n]
      ?f[ex:value->?v]
      pred:numeric-greater-than(?n 1)
      numeric-equal(?v 0)
      (INeg (Exists ?f1 (And (?f1 #ex:Fib
        ?f1[ex:number->?n1]
        pred:numeric-equal(?n1 pred:numeric-subtract(?n 1))
        )))) ))
    Then Do ( Assert(ex:Fib(pred:numeric-subtract(?n 1))))))

package MakeRecursiveGoal {
  rule MakeRecursiveGoal_r0 {
    when(
      f1 : fib.Fib (n1 : number);
      f : fib.Fib (v : value; n : number);
      evaluate(n > 1 && v == 0 && !((n1 == n - 1)));
    )
    then(
      insert(new fib.Fib (n - 1));
    )
  }
}

*****
(* computeValue *)
Group (
  Forall ?f ?f1 ?f2 such that (?f #ex:Fib)
    (If And (Exists ?n ?v (And (?f[ex:number->?n]
      ?f[ex:value->?v]
      pred:numeric-greater-than(?n 2)
      pred:numeric-equal(?v 0)
      ?f1 #ex:Fib
      ?f1[ex:number->?n1]
      ?f1[ex:value->?v1]
      pred:numeric-equal(?n1 pred:numeric-subtract(?n 1))
      pred:numeric-not-equal(?v1 0)
      ?f2 #ex:Fib
      ?f2[ex:number->?n2]
      ?f2[ex:value->?v2]
      pred:numeric-equal(?n2 pred:numeric-subtract(?n 2))
      pred:numeric-not-equal(?v2 0))))
    Then Do ( Modify (?f[ex:value]->pred:numeric-add(?v1 ?v2))))

package computeValue {
  rule computeValue_r0 {
    when(
      f2 : fib.Fib(n2 : number; v2 : value);
      f1 : fib.Fib(n1 : number; v1 : value);
      f : fib.Fib(v : value; n : number);
      evaluate(n > 2 && v == 0 && n1 == n - 1 && v1 != 0 && n2 == n - 2 && v2 != 0);
    )
    then(
      modify refresh f {value = v1 + v2;}
    )
  }
}

*****
(* setResult *)
Group (
  Forall ?f such that (?f #ex:Fib)
    (If And (Exists ?n ?v (And (?f[ex:number->?n]
      ?f[ex:value->?v]
      pred:numeric-not-equal(?v 0))))
    Then Do (pred:print(pred:concat("Fib" ?n "=" ?v))))

package setResult {
  rule setResult_r0 {
    when(
      f : fib.Fib(v : value; n : number);
      evaluate(v != 0);
    )
    then(
      out.println(supportPackIBM.RiFUtil.concat("Fib", n, "=", v));
    )
  }
}

```

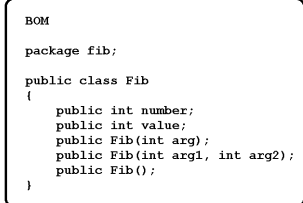


Fig. 5. Example 2

```

(* startSort *)
Group (
  Forall ?c
    (If Exists ?Id (And (?c #ex:Control
      ?c[ex:id->?Id]
      pred:numeric-equal(?Id 0)))
      Then Do ( Modify (?c[ex:id->1])))

package startSort {
  rule startSort_r0 {
    when{
      c : sort.Control(Id : id);
      evaluate(Id == 0);
    }
    then{
      modify refresh c {id = 1;}}
  }
}

*****
(* startDisplay *)
Group (
  Forall ?c
    (If Exists ?Id (And (?c #ex:Control
      ?c[ex:id->?Id]
      pred:numeric-equal(?Id 1)))
      Then Do ( Modify (?c[ex:id->2])))

package startDisplay {
  rule startDisplay_r0 {
    when{
      c : sort.Control(Id : id);
      evaluate(Id == 1);
    }
    then{
      modify refresh c {id = 2;}}
  }
}

*****
(* switchPosition *)
Group (
  Forall ?first ?second ?c
    (If And (Exists ?Id ?p1 ?p2 ?v1 ?v2 (And (?c #ex:Control
      ?c[ex:id->?Id]
      pred:numeric-equal(?Id 2)
      ?first #ex:Element
      ?first[ex:position->?p1]
      ?first[ex:value->?v1]
      ?second #ex:Element
      ?second[ex:position->?p2]
      ?second[ex:value->?v2]
      pred:numeric-greater-than(?v2 ?v1)
      pred:numeric-less-than(?p2 ?p1))))
      Then Do (Modify (?first[ex:position->?p2])
        Modify (?second[ex:position->?p1])))

package switchPosition {
  rule switchPosition_r0 {
    when{
      second : sort.Element(v2 : value; p2 : position);
      first : sort.Element(p1 : position; v1 : value);
      c : sort.Control(Id : id);
      evaluate(Id == 2 && v2 > v1 && p2 < p1);
    }
    then{
      modify refresh first {position = p2;};
      modify refresh second {position = p1;}}
  }
}

*****
(* incrementPosition *)
Group (
  Forall ?e1 ?e2 ?c
    (If And (Exists ?Id ?p1 ?p2 ?v1 ?v2 (And (?c #ex:Control
      ?c[ex:id->?Id]
      pred:numeric-equal(?Id 1)
      ?e1 #ex:Element
      ?e1[ex:position->?p1]
      ?e1[ex:value->?v1]
      ?e2 #ex:Element
      ?e2[ex:position->?p2]
      ?e2[ex:value->?v2]
      pred:numeric-greater-than(?v2 ?v1)
      pred:numeric-equal(?p2 ?p1))))
      Then Do (Modify (?e2[ex:position->pred:numeric-add(?p1 1)])))

package incrementPosition {
  rule incrementPosition_r0 {
    when{
      e2 : sort.Element(v2 : value; p2 : position);
      e1 : sort.Element(v1 : value; p1 : position);
      c : sort.Control(Id : id);
      evaluate(Id == 1 && v2 > v1 && p2 == p1);
    }
    then{
      modify refresh e2 {position = p1 + 1;}}
  }
}

*****
(* displayElement *)
Group (
  Forall ?c ?e
    (If And (Exists ?Id (And (?c #ex:Control
      ?c[ex:id->?Id]
      pred:numeric-equal(?Id 2)
      ?e #ex:Element
      ?e[ex:value]->?v
      ?p ?e[ex:position]->?p
      pred:print(pred:concat("value" ?v "is at position" ?p)))))

package displayElement {
  rule displayElement_r0 {
    when{
      e : sort.Element();
      c : sort.Control(Id : id);
      evaluate(Id == 2);
    }
  }
}

```

BOM

```
package sort;
```

```
public class Control
```

```
{
  public int id;
  public Control(int arg);
  public Control();
}
```

```
public class Element
```

```
{
  public int position;
  public int value;
  public Element(int arg1, int arg2);
  public Element();
}
```

# A Loose Coupling Approach for Combining OWL Ontologies and Business Rules

Amina Chniti,<sup>1,2</sup> Patrick Albert,<sup>1</sup> Jean Charlet<sup>2,3</sup>

<sup>1</sup> CAS France, IBM

{amina.chniti,albertpa}@fr.ibm.com

<sup>2</sup> INSERM UMRS 872, Eq 20, 15, Rue de l'école de médecine, 75006, Paris, France

Jean.Charlet@upmc.fr

<sup>3</sup> AP-HP, Paris, France

**Abstract.** In this demonstration we will show two prototypes based on the BRMS (Business Rule Management System) WODM, (1) The OWL plug-in and (2) the change-management plug-in. The OWL plug-in enables authoring and executing business rules over OWL ontologies. It consists of importing OWL ontologies into WODM and using all the functionalities offered by this BRMS to author and execute rules. The change-management plug-in enables the evolution of business rules with respect to the ontology changes. This component, implemented basically using an OWL ontology and rules, detects inconsistencies that could be caused by an ontology evolution and proposes solution(s), called repair, to resolve them.

**Keywords:** Ontology, Business Rule, Consistency maintenance, inconsistency.

## 1 Challenges faced

In the majority of BRMS, Business Models are represented using Object Models while OWL Ontologies offer a better power of expressiveness. The purpose of the proposed challenge is to bring the expressiveness of OWL ontologies to business users by means of business rules authored in a natural controlled language. For this, we exploited the infrastructure offered by the BRMS WebSphere Operational Decision Management (WODM) and developed two prototypes : The OWL Plug-in and the change-management plug-in for WODM. the OWL plug-in enables authoring and executing business rules over OWL ontologies. It consists of importing OWL ontologies into WODM and using all the functionalities offered by this BRMS to author and execute rules. The change-management plug-in enables the evolution of business rules authored over OWL ontology with respect to the ontology changes. This component, implemented basically using OWL ontology and rules, detects rules inconsistencies that could be caused by an ontology change and proposes solution(s), called repair, to resolve them.

During the development of this work we have been faced to the following challenges :

- how to enable business users to deal with their business knowledge formalized using OWL Ontologies?
- How to import OWL ontology into WODM?
- WODM is object model based, how to import the expressiveness and the reasoning capacity of OWL into such a BRMS?
- How to minimize the loose of information?

Ontologies evolve during their life cycle :

- What is the impact of such evolution on the rules?
- How to make the rule set evolving with respect to the ontology?
- How to maintain its consistency while it evolves?
- How to detect the impact of the ontology evolution on rules?

## 2 Method

In this section, we will describe the methods we developed to resolve the challenges described above. These methods are based on WODM.

WODM offers an infrastructure that enables business users to author - in a controlled natural language - execute and manage business rules in a collaborative way. As the majority of BRMS, it uses an object oriented models to formalize the domain knowledge. In WODM, this object oriented model is called BOM (Business Object Model). The BOM represents the entities of a given business (e.g. *Client*, *age*). It is generated over from the XOM (eXecutable Object Model) then verbalized. The XOM is the model enabling the execution of rules. It references the application objects and data, and is the base implementation of the BOM. The XOM can be built from compiled Java classes (Java execution object model) or XML Schema (dynamic execution object model). The verbalization of the BOM consists of generating a controlled natural language vocabulary (VOC) which enables to edit the business rules. The VOC, add a layer of terminology on top of the BOM (e.g. “*the client*”, “*the age of the client*”). This vocabulary is used to compose the text of the rules.

### 2.1 OWL plug-in

To enable business users to author business rules over OWL ontologies, we developed the WODM OWL plug-in. This plug-in exploits infrastructure offered by WODM to import OWL ontologies within it. The main component for authoring rules in WODM is the BOM. For this, we performed a mapping of OWL concepts (TBox) into the BOM. Thus, when we import an OWL ontology within WODM, the BOM is automatically generated and the functionalities offered by the BRMS can be used [1].

Among these functionalities, we will focus on authoring and executing rules. Once we have the BOM, its verbalization is also available and the business users are able to edit the business rules in a natural controlled language or using the decision tables or the decision tree.

To execute business rules authored over ontologies, we performed a second mapping of OWL/BOM entities to a XOM using Jena . Jena is a Java framework, including an ontology API for handling OWL ontologies, which allows to generate Java objects from the entities of the ontology. These Java objects then constitute the XOM. The use of Jena provides an execution layer for the OWL ontologies. This execution layer provides inference mechanisms on this model and the mapping of OWL concepts, properties, and individuals to a Java object model. When the business user launches the rule execution process, the ontology individuals (ABox) are loaded into the working memory of the rule engine and mapped into java objects. The rule engine evaluates the rule conditions against these objects and fires the rules for the objects that meet the condition. During this process, the ABox mapped into Java Objects, is updated with respect to the action parts of the fired rules. At the end of the execution process, the “new” ABox is loaded into the ontology.

Another important point in the process of executing rules is the interaction between the classification engine and the rule engine. In the following we will present some examples of this kind of interaction as achieved with the system presented in this work. The classification engine assigns the type of the individuals, then the rule engine uses this inferred knowledge to trigger a computation that could not be easily represented in an ontology. In other words, the rule engine asks the classification engine for the type of the individual, then it executes the rule(s) matching with the returned type.

*Example 1:* In the ontology, we define the concept **CarDriver** as a **Driver** who has a **CarDrivingLicense** :

**CarDriver**  $\equiv$  *hasDrivingLicense some CarDrivingLicense*

and we declare the following individuals :

**Driver**(Joe); *hasAge*(Joe, 20) ; *hasName*(Joe, "Joe")

**Person**(Toto); *hasAge*(Toto, 8) ; *hasName*(Toto, "Toto")

**Driver**(John); *hasAge*(John, 25) ; *hasName*(John, "John")

then we author the following rule (i.e. car driving license)

*car driving license :*

*IF the age of the driver is more than 18*

*THEN add the car driving license to the driving licenses of the driver ;*

After the execution of this rule, we see in the ontology that for each driver who is more 18 (i.e. John and Joe), a car driving license is attributed. Then, we execute the *car driver* rule, that lists the names of all the car driver in the ontology. The result is : Joe is a car driver and John is a car driver because of the reclassification of Joe and John after executing the *car driving license* rule.

*car driver :*

*THEN print the name of the car driver + " is a car driver" ;*

*Example 2 :* In the ontology, we also define a concept **ChildCarDriver**, subclass of a concept **Child**. A **ChildCarDriver** is a **Child** who has as father a

**CarDriver**. Then we define *Toto* who has father *John*.

```
ChildCarDriver subclassOf Child subclassOf Person  
ChildCarDiver  $\equiv$  hasFather only CarDriver  
hasFather(Toto, John)
```

after executing the *car driving license* rule, we execute the *child car driver* rule that lists the names of all the child car driver. The result is that *Toto* is a *child car driver* because that in the ontology we define a child car driver as a person who has a car driver as father.

```
child car driver :  
THEN print the name of the child car driver + " is a child car driver" ;
```

In the ontology, we also define the concept **Contravention** that has an amount, a driver could have 0 or more contraventions and a contravention amount to pay.

```
Contravention(c1), Contravention(c2)  
hasContravention (John, c1), hasContravention (John, c2)
```

Using this definition, we author the *remove car license* rule that removes the car driving license for each car driver who has a contravention and calculates the contravention amount to pay. After executing this rule, John will loose his car license. Thus, when we re-execute the *car driver* rule and the *child car driver* rule, we will only have *Joe* is a car driver.

```
remove car license :  
definitions  
set 'a contravention' to a contravention ;  
set 'a car driver' to a car driver where the contraventions of this car driver  
contain a contravention;  
IF 'a car driver' is not null  
THEN remove the car driving license from the driving licenses of 'a car driver';  
for each contravention in the contraventions of 'a car driver' :  
- set the contravention amount to pay of 'a car driver' to the contravention  
amount to pay of 'a car driver' + the contravention amount of this contraven-  
tion ;
```

In this example, the rule engine uses this inferred knowledge to trigger a computation that could not be easily represented in an ontology

*Example 3* In the ontology, we define the concept **Contravention** and the concept **RiskyDriver** as a **Driver** who has more than 3 **Contravention** (a cardinality restriction), *Frank* as a **RiskyDriver** and we give to John 4 **Contraventions**.

```
RiskyDiver  $\equiv$  hasContravention min 3  
RiskyDriver(Frank) ; hasName(Frank, "Frank")
```

*hasContravention (John, c1), hasContravention (John, c2), hasContravention (John, c2), hasContravention (John, c4)* such as **Contravention(c1)**, **Contravention(c2)**, **Contravention(c3)**, **Contravention(c4)**.

If we execute the *risky driver* rule that lists the name of the risky driver we will only see that *Frank is a risky driver* which means that the classification engine is not able to classify John, who has four contraventions, as a risky driver.

*risky driver* :  
*THEN print the name of the risky driver + " is a risky driver" ;*

## 2.2 Change-Management plug-in

The change-Management plug-in is for WODM enables to analyse the impact of ontology evolutions on business business rules. Ontology evolutions consist of changes that impact an ontology. These changes may be structural changes, conceptual changes, entity definition changes,... Business rules depends on the entities of the ontology and its evolution has an impact on the rule set that may causes inconsistencies. Thus, we developed the *MDR* approach (*Model, Detect, Repair*), which ensures the consistency maintenance of business rules while ontology evolution [3].

The *MDR* approach is based on design patterns and especially *Change Management Patterns* (CMP). This approach has been inspired from *ONTO-EVO<sup>A</sup>L* [2], which deals with the consistency maintenance of OWL ontologies while they evolve. In our approach the CMPs are proposed to guide the evolution process of a rule set while maintaining its consistency. They consist of three categories of patterns :

1. *Change Pattern* : used to model the ontology change knowledge that are important to detect its impact;
2. *Inconsistency Pattern* : used to detect the inconsistencies caused by a change;
3. *Repair Pattern* : used to propose solutions, called repair, to resolve the inconsistencies.

The consistency maintenance process that we propose in our approach consists of three steps :

1. Model the change to apply to the ontology using the change pattern;
2. Detect the eventual inconsistency that could be caused using the inconsistency pattern;
3. Propose repair to solve the inconsistency using the repair pattern.

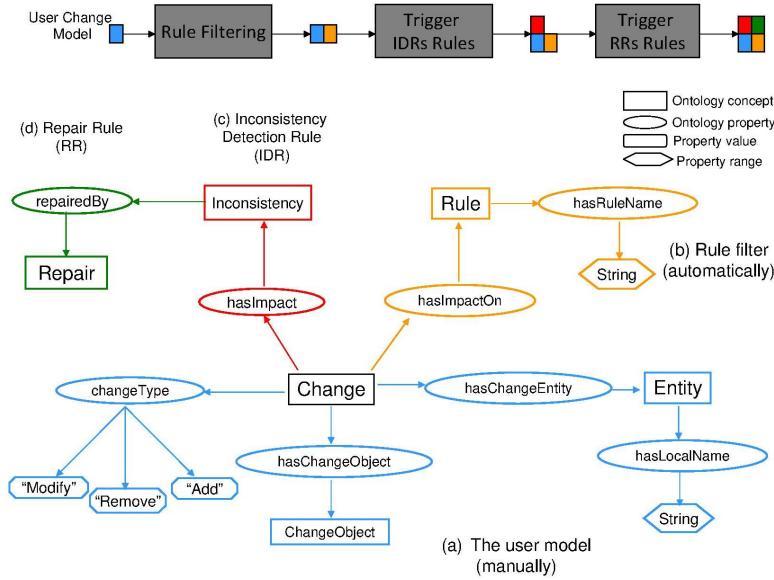
Change pattern is designed using an OWL ontology, called *MDR<sub>Ontology</sub>*, which model the ontology changes and their description, the inconsistencies that impact a rule set and their repairs. Each change has constraints to verify to avoid inconsistencies. Depending on the violated constraint the inconsistencies are detected using the inconsistency pattern. Thus, the inconsistency patterns are designed using a set of rules, called *Inconsistency Detection Rules* (*IDR*),

which in their condition part define the constraint that each change should verify and in their action parts define the inconsistency that will be caused by the change. The repair patterns are also designed using rules, called **Repair Rules** (RR), which in their condition parts test on the detected inconsistency and on the modelled change then, in their action parts assign the repair(s) to apply. The rules designing the inconsistency and repair patterns have been authored over the *MDROntology* using the OWL plug-in (see section 2.1).

Figure 1 illustrates the *MDR* process. As input of our system, the user models the change description as *MDROntology* individuals. The change description consists of :

- Change type : add, remove, modify;
- Change object : conceptual change (i.e. subclass change, add concept, remove property,...) or entity definition change (enumeration change, restriction change, rename entity...);
- Change entities : concept or property that will be impacted by the change;
- Impacted rules and the scope of the impact (i.e. the impacted rule part).

The change type, the change object and the change entities must be provided manually by the user. The impacted rules and the impacted rule parts are detected automatically. Nevertheless, depending on the change to apply other in-



**Fig. 1.** *MDR* process

formation should be given. For example, the new collection of the enumeration



in case of an enumeration change, the new name of an entity in case of a rename entity change or the new range of a property in case of a range change. . . . In the following, the general template of a change pattern (see Fig. 1).

When the user launch the consistency maintenance plug-in within WODM, the modelled changes (*MDROntology* individuals) are loaded into the working memory and shown to the user through a user interface. When the user select one or more changes to apply, the IDRs are fired and detect the inconsistencies that will be caused. A general template of the inconsistency pattern is given below :

```
IF change.changeObject = changeObject
&& change.type = changeType
&& changeConstraint.satisfied = false
THEN change.inconsistency = inconsistency;
```

After the inconsistency detection and depending on the change to apply, the RRs are forced and one or more repair are proposed to the user, who will choose the repair to apply. The chosen repair will be automatically applied after verifying that it will not causes other inconsistencies. A template of the repair pattern is given below :

```
IF change.changeObject = changeObject
&& change.inconsistency = inconsistency
THEN inconsistency.repair = inconsistencyRepair;
```

### 3 Discussion

In section 1, we introduced the challenges we faced. In this section we discuss the challenges we resolved and those that we are trying to resolve.

To enable business users to deal with their business knowledge formalized using OWL Ontologies, we proposed an approach that consists of importing OWL ontologies into WODM. This approach enables authoring, in a natural controlled language, and executing rules over ontologies. Thus, business users are able to use the domain entities defined in the ontology to define business decisions using rules.

To import OWL ontologies into WODM, we performed an OWL to BOM mapping. Thus, when the users import an OWL ontology into WODM, the BOM is automatically generated and all the functionalities offered by the BRMS can be used.

WODM, or more specifically the BOM, is an Object Model. We cannot import all the expressiveness provided by OWL into such a model. Some constructs, such as `rdfs:subClassOf`, `owl:allValuesFrom`, `owl:inverseOf`. . . are mapped. Some others, such as `owl:someValuesFrom`, `owl:SymmetricProperty`, `owl:TransitiveProperty` cannot be mapped into the BOM but they are processed at runtime (see Example 2 in section 2.1). Other constructs are neither mapped into the BOM nor processed at runtime such as `owl:minCardinality`

(see Example 3 in section 2.1), `owl:maxCardinality`, `owl:complementOf...` A complete description of the mapping can be found in [1].

Ontologies evolve during their life cycle. Rules are authored over the ontology entities and depend on them; this is why an ontology evolution may make the rule set inconsistent. To make the rule set evolve with respect to the ontology while maintaining its consistency, we developed the *MDR* approach, which is a pattern based approach. The general idea of this approach is that the user models the ontology change he wants to apply using the change pattern. Then, using the inconsistency patterns, inconsistencies that may be caused by the change are detected automatically. Finally, repairs that resolve the inconsistencies are proposed automatically thanks to the repair Pattern. Nevertheless, in the actual state of the work, the inconsistency patterns detect only two types of inconsistencies from six. A definition of business rules inconsistencies is done in [3].

There are other challenges to be taken up; how to bring all the power of expressiveness of OWL to business users without losing information? In the *MDR* approach the inconsistency and repair patterns are defined manually which is a costly and not an easy task. Is it possible to automatically generate these patterns depending on the change to apply in a way that we will be able to detect all the inconsistencies that could impact business rules.

## References

1. A. Chniti, S. Dehors, P. Albert, and J. Charlet. Authoring business rules grounded in owl ontologies. In M. Dean et al. (Eds.), editor, RuleML 2010 : The 4th International Web Rule Symposium: Research Based and Industry Focused. LNCS 6403, Springer-Verlag Berlin Heidelberg 2010, 2010.
2. R. Djedidi and M.A. Aufaure. ONTO-EVO<sup>A</sup>L an ontology evolution approach guided by pattern modelling and quality evaluation. Proceedings of the Sixth International Symposium on Foundations of Information and Knowledge Systems (FoIKS 2010), 2010.
3. M. Fink, A. El Ghali, A. Chniti, R. Korf, A. Schwichtenberg, F. Lévy, J. Pührer, and T. Eiter. D2.6 consistency maintenance. final report. ONTORULE Deliverable, <http://ontorule-project.eu/deliverables>., 2011.

# Diamond Debugger Demo: Rete-Based Processing of Linked Data

Daniel P. Miranker, Rodolfo K. Depena, Hyunjoon Jung,  
Juan F. Sequeda, and Carlos Reyna

Department of Computer Science  
University of Texas at Austin  
{miranker, jsequeda}@cs.utexas.edu  
{rudyp.depena,polaris79, creynam89}@gmail.com

**Abstract.** Diamond is a Rete match based system that evaluates SPARQL queries on Linked Data. The evaluation of SPARQL query predicates is a useful intermediate milestone for a system ultimately intended to support full rule-based inference on Linked Data. A byproduct is the integrated graphical rule debugging environment is a first of its kind debug environment for SPARQL queries.

**Keywords:** Rete, SPARQL, Linked Data, Semantic Web

## 1 Background

The Linked Data model is an emerging component of the Semantic Web. The base layer of the Semantic Web is a representation of a directed labeled graph, expressed using resource description framework (RDF). Each edge of such a graph is commonly known as a triple. A triple is composed of a subject, a predicate and an object. The predicate is the edge label. The subject and object are vertex labels. Each constituent may be a URI. The intention is that labels form global unique ids and overlay DNS services to identify a particular server that may provide additional details (semantics) for the URI in the form of additional triples. The object of a triple may contain a literal. Thus, an RDF graph can represent complex data, spanning an arbitrary set of Internet servers [6].

Formal semantics for the Linked Data model are still emerging [4, 5]. The base principles mimic the behavior of hyperlinks in html documents [6]. That is, like a URL, dereferencing a URI instigates a response from a particular server. However, in lieu of an HTML document that may contain both text and an embedded set of URL-based hyperlinks, the server simply returns a set of triples.

One can expect that Linked Data crawlers will be intelligent. To date, all Linked Data specifications and most related work is limited to RDF. There is no explicit connection to the schema, ontology and rule layers, (RDFS, OWL, RIF), of the Semantic Web technology stack. Thus, in Linked Data, any semantic entailment will necessarily be implemented by inference processes associated with the processes that initiate and control the Linked Data crawlers. For example,

SPARQL 1.1 allows triples to be updated. SPARQL 1.1 also includes entailment rules that define closure over subclass hierarchies. Unless a system admits to limiting query and inference to a potentially inconsistent cache, it is necessarily the case that inference entail freshly collected data.

Architecturally, the intrinsic, incremental behavior of the Rete Match aligns well with the web crawling aspects of the Linked Data model. This is true if one is evaluating rule predicates, or just a single predicate. When a Linked Data URI is dereferenced it returns a set of triples. The values of the triples may include additional URIs that have not yet been dereferenced. This operational behavior is identical to the algorithmic behavior of the Rete match per its original context, the incremental evaluation of changes to working memory in forward-chaining rule systems. Since an RDF graph is arbitrarily large and dynamic, even if an implementation references a local cache of prefetched triples, one can anticipate that any formal semantics will have to be consistent with an evaluation method that, operationally, crawls the web of Linked Data and reports results prior to reaching all reachable vertices. I.e. crawling can be paused at any time, and the system evaluated.

Motivated, in part, by the anticipation intelligent Linked Data agents, we have first built a SPARQL query engine based on the Rete match and architected the integration of a Rete-based system with link-crawling and caching components [3]. In Diamond, there is a Rete network object. Each rule predicate is compiled as an instance of the Rete network object<sup>1</sup>.

Serendipitously our implementation of a graphical rule debugger is also a SPARQL query debugger. Those already familiar with graphical debugging environments for Rete-based inference engines will already be familiar with the operation and concomitant rendering of the Rete network and its content. This is not the case for most developers in the SPARQL community.

We anticipate the development of a SPARQL query debugger will, further, be welcomed by that community as SPARQL queries can be expansive, even larger than comparable SQL queries. To support this claim, a query from the Berlin SPARQL Benchmark Suite is reproduced in Figure 1. This benchmark is distinguished as it provides semantically equivalent queries in SQL, as shown in Figure 2. The definition of a set of Rete operators for SPARQL follows from long standing connections made between relational algebra and rule predicates, and results that prove an expressive equivalence between SPARQL, DatalogNeg without recursion and relational algebra. [1, 7]

## 2 The System

The Diamond architecture is illustrated in Figure 3. The Rete network is created, dynamically, from a runtime library of Rete network object definitions [7]. The URI dereferencing object is static. A critical design component is the pair of

<sup>1</sup> Optimizations based on sharing Rete network subtrees is anticipated by more sophisticated compilation techniques and providing for the composition of Rete network object instances.

## Diamond Debugger Demo: Rete-Based Processing of Linked Data

```
PREFIX bsbm-inst: <http://www4.wiwiiss.fu-berlin.de/bizer/bsbm/v01/instances/>
PREFIX bsbm: <http://www4.wiwiiss.fu-berlin.de/bizer/bsbm/v01/vocabulary/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema>
PREFIX dc: <http://purl.org/dc/elements/1.1/>

SELECT ?label ?comment ?producer ?productFeature
      ?propertyTextual1 ?propertyTextual2 ?propertyTextual3
      ?propertyNumeric1 ?propertyNumeric2
      ?propertyTextual4 ?propertyTextual5 ?propertyNumeric4
WHERE
  bsbm-inst:ProductXYZ rdfs:label ?label .
  bsbm-inst:ProductXYZ rdfs:comment ?comment .
  bsbm-inst:ProductXYZ bsbm:producer ?p .
  ?p rdfs:label ?producer .
  bsbm-inst:ProductXYZ dc:publisher ?p .
  bsbm-inst:ProductXYZ bsbm:productFeature ?f .
  ?f rdfs:label ?productFeature .
  bsbm-inst:ProductXYZ bsbm:productPropertyTextual1 ?propertyTextual1 .
  bsbm-inst:ProductXYZ bsbm:productPropertyTextual2 ?propertyTextual2 .
  bsbm-inst:ProductXYZ bsbm:productPropertyTextual3 ?propertyTextual3 .
  bsbm-inst:ProductXYZ bsbm:productPropertyNumeric1 ?propertyNumeric1 .
  bsbm-inst:ProductXYZ bsbm:productPropertyNumeric2 ?propertyNumeric2 .
  OPTIONAL bsbm-inst:ProductXYZ bsbm:productPropertyTextual4
    ?propertyTextual4
  OPTIONAL bsbm-inst:ProductXYZ bsbm:productPropertyTextual5
    ?propertyTextual5
  OPTIONAL bsbm-inst:ProductXYZ bsbm:productPropertyNumeric4
    ?propertyNumeric4
```

**Fig. 1.** BSBM SPARQL Query 2

```
SELECT pt.label, pt.comment, pt.producer, productFeature, propertyTex1,
      propertyTex2, propertyTex3, propertyNum1, propertyNum2,
      propertyTex4, propertyTex5, propertyNum4
FROM product pt, producer pr, productfeatureproduct pfp
WHERE pt.nr=XYZ AND pt.nr=pfp.product AND pt.producer=pr.nr
```

**Fig. 2.** BSBM SQL Query 2

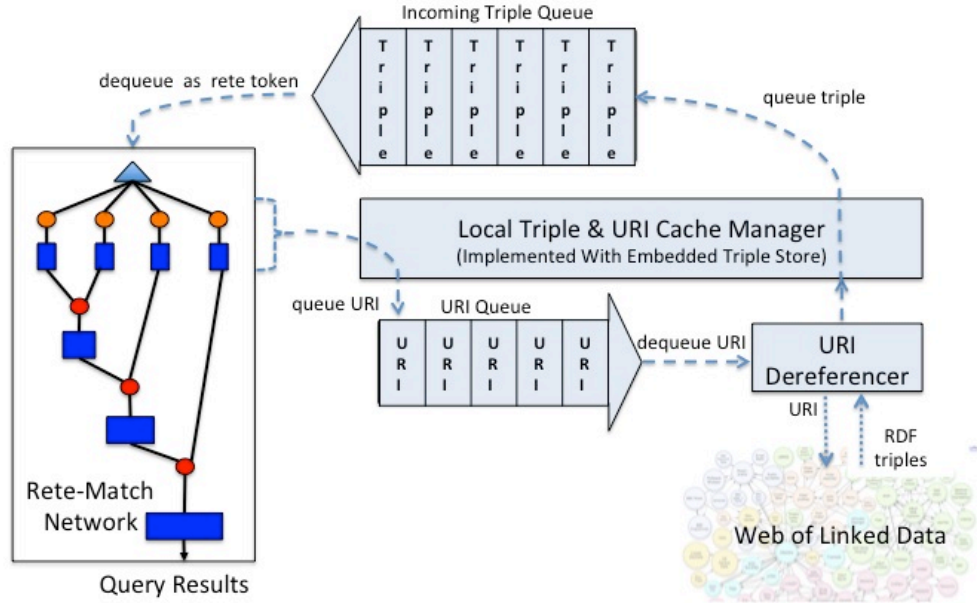


Fig. 3. Diamond Architecture Diagram

queues. The queues are intended to enable parallel, asynchronous execution of query evaluation and URI dereferencing. The queues are actively managed to avoid redundant dereferencing of URIs or redundant processing of triples by the Rete network. The queue manager is implemented by accumulating triple in an embedded copy of the Sesame<sup>2</sup> triplestore.

### 3 Demonstration

For demo purposes we use the benchmark query illustrated in Figure 1. For pedagogical purposes the screen shots are created by choosing an illustrative subset of 5 triple patterns. Video can be found at <http://ribs.csres.utexas.edu/diamond/>. Figure 4 is a screen shot of the debugger when there are triples that satisfy 4 of the 5 triple patterns. No triples that satisfy the third triple pattern.

Given the potential for a large number of both triples (data) and triple patterns (SPARQL clauses) the Rete network is rendered separately from the data and the contents of the memory nodes. The contents of a memory node is viewed by clicking on the node in the Rete network, which opens a new window. Users may open, resize and move such windows anywhere on the screen. To save space, the figure shows three memory node windows overlayed on the window of the Rete network. We show, contents of one alpha-memory, that there is a repre-

<sup>2</sup> <http://www.openrdf.org/>

## Diamond Debugger Demo: Rete-Based Processing of Linked Data

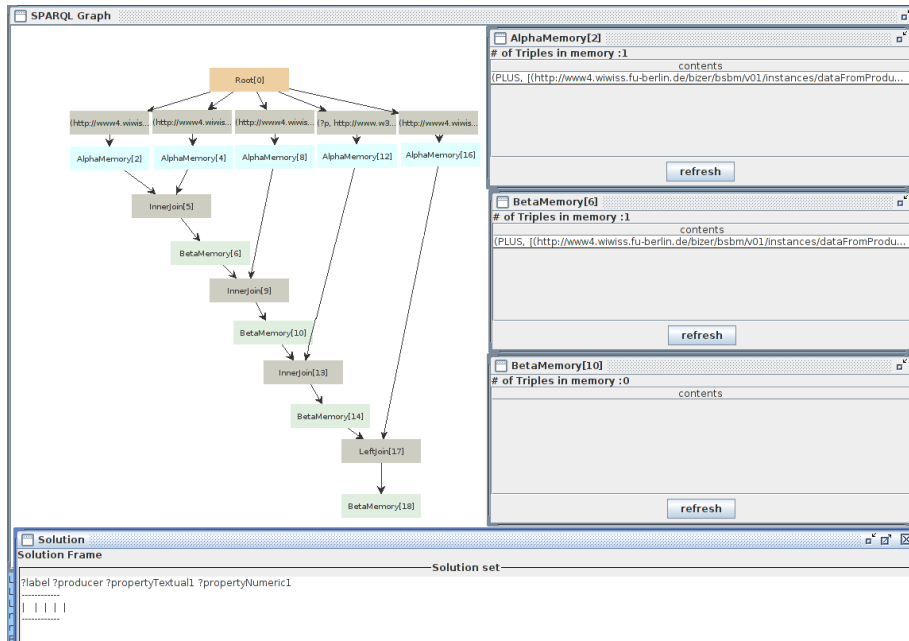


Fig. 4. Screenshot of before

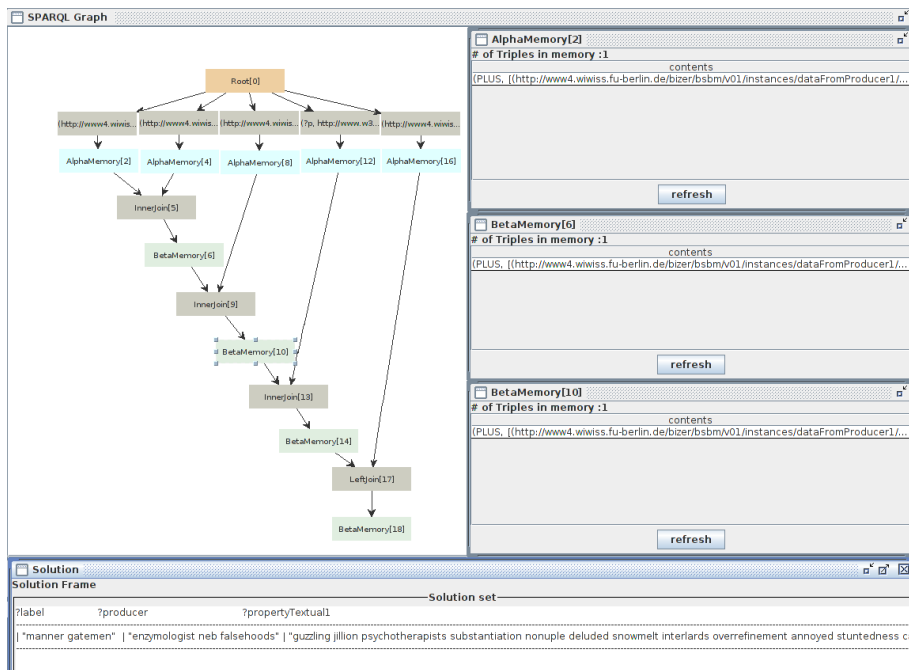


Fig. 5. Screenshot of after

sensation of a successful join in the first beta-memory. The remainder of the network is empty.

Upon the arrival of a triple that satisfies the third triple pattern the query becomes satisfied. Figure 5 shows the subsequent state of the Rete network. Each of the beta memories now has content.

## 4 Discussion

Although Diamond currently treats a SPARQL query as the predicate of a single rule, the system is easily extended to process a set of forward-chaining rules. The overlap of operator level equivalence between SPARQL query predicate evaluation and rule system evaluation is well established in the literature [1, 2, 8, 9]. The extensibility of the implementation is a byproduct of object-oriented design principles. Although we have no immediate plans per the investigation of parallel evaluation of rule systems, we note that the coordination of Rete network evaluation with the Linked Data crawlers is by means of asynchronous queues. Thus, the mechanisms for asynchronous concurrent rule evaluation are in place as well.

**Acknowledgments.** This research is supported by an NSF grant IIS-1018554. Juan F. Sequeda was supported by an NSF Graduate Research Fellowship.

## References

1. Renzo Angles and Claudio Gutierrez, ‘The expressive power of sparql’, in *International Semantic Web Conference*, pp. 114–129, (2008).
2. François Bry, Tim Furche, Bruno Marnette, Clemens Ley, Benedikt Linse, and Olga Poppe, ‘Sparqllog: Sparql with rules and quantification’, in *Semantic Web Information Management*, 341–370, (2009).
3. Charles Forgy, ‘Rete: A fast algorithm for the many patterns/many objects match problem’, *Artif. Intell.*, **19**(1), 17–37, (1982).
4. Olaf Hartig, ‘Sparql for a web of linked data: Semantics and computability’, in *ESWC*, pp. 8–23, (2012).
5. Olaf Hartig, Christian Bizer, and Johann Christoph Freytag, ‘Executing sparql queries over the web of linked data’, in *Proceedings of the 8th International Semantic Web Conference*, pp. 293–309, (2009).
6. Tom Heath and Christian Bizer, *Linked Data: Evolving the Web into a Global Data Space*, Synthesis Lectures on the Semantic Web, Morgan & Claypool Pub., 2011.
7. Daniel P. Miranker, Rodolfo K. Depena, Hyunjoon Jung, Juan F. Sequeda, and Carlos Reyna, ‘Diamond: A sparql query engine, for linked data based on the rete match’, in *Proc. of the Artificial Intelligence meets the Web of Data Workshop at ECAI12*, (2012).
8. Axel Polleres, ‘From sparql to rules (and back)’, in *Proc. of the 16th int. conf. on World Wide Web*, WWW ’07, pp. 787–796, New York, NY, (2007). ACM.
9. Simon Schenk and Steffen Staab, ‘Networked graphs: a declarative mechanism for sparql rules, sparql views and rdf data integration on the web’, in *Proc. of the 17th int. conf. on World Wide Web*, WWW ’08, pp. 585–594, New York, (2008). ACM.



# Monitoring BPMN-Processes with Rules in a Distributed Environment

Lothar Hotz<sup>1</sup>, Stephanie von Riegen<sup>1</sup>, Lars Braubach<sup>2</sup>, Alexander Pokahr<sup>2</sup>, and  
Torsten Schwinghammer<sup>3</sup>

<sup>1</sup> HITeC e.V. c/o Fachbereich Informatik, Universität Hamburg, Germany {hotz, svriegen}@informatik.uni-hamburg.de

<sup>2</sup> VSIS, Fachbereich Informatik, Universität Hamburg, Germany {braubach, pokahr}@informatik.uni-hamburg.de

<sup>3</sup> Uniiue AG, Hamburg, Germany  
Torsten.Schwinghammer@UniiueAG.com

**Abstract.** In this paper, we demonstrate an application of rules in a business process scenario. As business processes, we consider data-intensive applications which need to move huge data files from server to server. By using the Business Process Model and Notation (BPMN) in our application, we enable clearly and hierarchically represented business processes. Such modeled processes can automatically be executed in a distributed environment with the Jadex open source middleware. Furthermore, the process execution is monitored with declarative rules, also implemented with Jadex. The demonstration shows the start of BPMN-modeled processes and their execution monitoring through logs and rules.

**Keywords:** Distributed systems, BPMN, rule-based systems, monitoring

## 1 Introduction

In business intelligence scenarios a huge amount of continuously growing data files has to be processed in distributed environments. Examples are log file, database, and campaign management as they are common in banking or telecommunication organizations. For such tasks, organizations use data integration approaches. However, (Friedman et al., 2008) points out that the "...commitment for implementing and supporting custom-coded or semi-manual data integration approaches is no longer reasonable" caused by the need for cost control. Consequently, organizations do already use specific data integration applications, however, for controlling data flows on distributed systems, manual activities or individual processes are still needed.

The basic principle in such applications consists of an extraction of data files from an operative system (like a web server), transformation of the data (on a staging server), and storing it in an analytical platform (like a data warehouse), see Figure 1. Data integration tools already handle diverse data file formats, however, they blank out that organizational data primary exist as decentralized, distributed data files. In our approach, we enable a declarative representation of business processes with the Business Process Model and Notation (BPMN) (OMG, 2006). With this notation, a user models business

processes on the basis of a predefined application component library. Those components implement basic tasks needed for file manipulation or similar activities. Such modeled processes can directly (i.e. without further coding) be executed in a distributed environment, such that subprocesses or tasks run on different servers.

In this paper, we focus on the monitoring of business process execution. This monitoring task has the goal to identify interesting occurrences during process execution. Such occurrences can be normal process execution as well as failing executions like not finished processes or exceeded average runtime of processes. For this process observation tasks, we apply a rule-based approach (see (Barringer et al., 2010) for a similar approach).

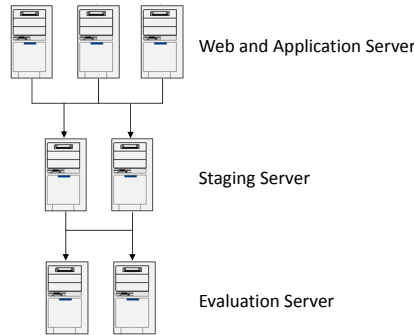


Fig. 1: Example Architecture

We are currently developing a system for automating business processes for gaining flexibility in process arrangement, quality improvements of data processing, and cost savings. The system consists of distributed, autonomously acting components, which are centrally controlled. Besides distributed quality assurance processes and integrative information life cycle strategy, the system has a process modeling component based on the BPMN (see Section 2) and a monitoring component (see Section 4), which are presented in this paper. We use the Jadex system<sup>4</sup> as a infrastructure supporting distributed agents and rules (see Section 3). In Section 5 and with a video<sup>5</sup>, we demonstrate the processing of agents and rules with a data staging example.

## 2 BPMN Example

The de facto standard to graphically model applicable business processes is the Business Process Model and Notation (BPMN) (OMG, 2006). This notation is suited to formulate e.g. organizational structures or data models with elements such as events, activities, gateways, data objects, and transactions arranged in a pool and lanes. We use BPMN in the business intelligence context for distributed management of processes and data.

<sup>4</sup> <http://jadex-agents.informatik.uni-hamburg.de>

<sup>5</sup> <http://monitoringrules.dyndns.org/>

In the following, we introduce a simple example use case. Figure 1 depicts an exemplary server setting where the first stage of server outputs large amounts of data (for example customer informations collected by a publicity campaign) which will be processed by the staging server. The staging server filters the input data according filters like e.g. correct addresses. The quality analysis of data provided by the evaluation server brings the setting to a close.

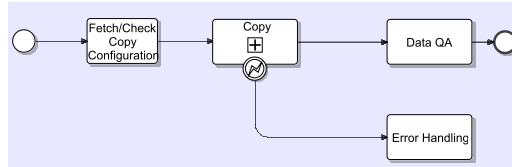


Fig. 2: BPMN model for data staging example

The notational model of the data processing is described in Figure 2. Before processing the collected data, the configuration data for the upcoming copy process has to be fetched and checked, possible configuration contents might be access credentials. The copy task is a collapsed subprocess (readily identifiable by the plus), in case of errors within this subprocess the error event leads to the error handling task. The expanded copy process is shown in Figure 3. After the data processing, the analyzing of quality (QA) step follows.

The expanded copy subprocess contains the following tasks: First the connection to a specific server has to be established, and before copying the data the existence is checked. Since of some data only one version is allowed to exist, a delete task is integrated via the exclusive gateway. Each task is bonded with an error catching event leading to the error handling task. Because of server-side connection problems, some errors might be corrected by a simple retry after a couple of minutes, see the timer event in Figure 3.

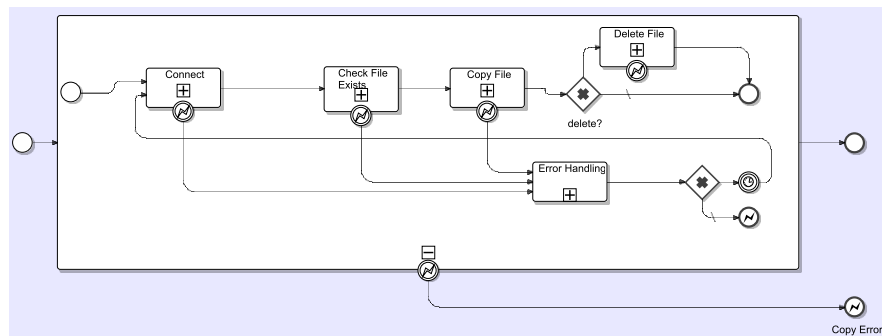


Fig. 3: Inner copy model of staging example

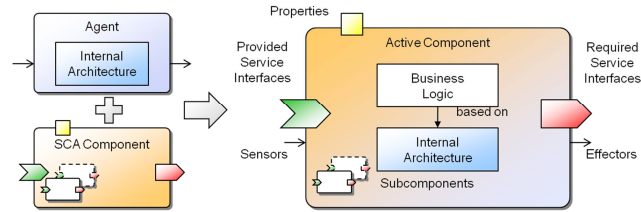


Fig. 4: Active Component

### 3 Jadex

The implementation platform used in this project is the open source platform Jadex (Braubach and Pokahr, 2011), which is a middleware for distributed systems. Jadex uses the new notion of active components for programming distributed systems. An active component or instance represents a unification of the component with agent and service concepts (cf. Fig. 4). The basic idea consists in having autonomously acting and loosely coupled entities in the sense of the actor model, i.e. communication among entities should be asynchronous and data objects should belong only to one actor to avoid data inconsistencies caused by concurrent accesses. This world model is combined with component and service ideas to strengthen the software engineering mechanisms that can be used to build software systems. The beneficial combination of component and service approaches has recently been put forward by the service component architecture (SCA) (Marino and Rowley, 2009), which introduces a component based architecture description framework for service oriented applications. The SCA component model defines a component with provided and required service interfaces to clearly state offered functionalities and dependencies. Furthermore, such interfaces allow for hierarchical (de)composition of complex software systems and foster reuse of software by building composites from readily available components. An active component further combines these SCA component concepts with agent characteristics, mainly it adds an internal architecture to a component. This internal architecture determines the type of an active component and the way its behavior has to be programmed. In this way very different kinds of active components such as BPMN workflows and even cognitive belief-desire-intention (Rao and Georgeff, 1995) agents can interact seamlessly because they share the same black-box view of each other.

#### 3.1 The Runtime Platform

The Jadex runtime environment consists of the component container called platform and an additional tool set mainly used for administration, debugging and testing purposes. The Jadex platform facilitates the development of distributed systems in the following ways:

- Distribution transparency is established between components, i.e. a component can invoke a service of another component without having to know if this component is local or remote.

- An overlay network is automatically built by platform awareness. This means that Jadex platforms automatically find each other in local as well as distributed networks employing multiple different techniques such as IP broadcasts for local detection. In this way services of new remote platforms can be found and used as soon as a new platform has been discovered. Of course, the network building can be customized or disabled for specific customer settings.
- Platform security is assured. On the one hand Jadex introduces security mechanisms to protect the privacy of user data and services by separating awareness from communication means, i.e. platforms may detect each other but communication between them is restricted with respect to security settings. On the other hand application services can be declaratively equipped with security features so that authentication, confidentiality and integrity of communication partners is ensured. This is achieved by relying on established security protocols such as SSL.

### 3.2 Workflows and Rule Support

BPMN workflow support for Jadex consists of a visual editor based on the open source eclipse stp editor and the workflow engine that is able to execute modeled workflows. The editor mainly extends the stp version with new input fields for annotating implementation data that is necessary for executing the workflow. Such modeled workflows can be directly loaded and executed within Jadex using the BPMN kernel, which enacts each workflow as a separate active component instance. A workflow can spawn new subworkflows either locally or on remote platforms and monitor their execution. Furthermore, as workflows are components, they can invoke services of other workflows or components via their provided service interfaces. Rule support is based on a typical forward chaining rule engine called Jadex Rules that is similar to JESS and Drools, but targeted towards a very lightweight engine solution that can be integrated with other application parts. One reason for such a lightweight solution was the requirement to be able to execute a rule engine as part of an active component, i.e. due to the actual number of such components many rule engines have to run concurrently.

## 4 Monitoring

The duty of the monitoring component is to observe process execution and signal successful or failure execution. This monitoring can depend on application specific attributes like duration of process execution time or transferred file size. As common for a rule-based approach, working memory elements based on templates and rules can be used for representing the involved data and knowledge. Templates describe via fields structured data objects. Rules consist of a condition and action part. If some working memory elements fulfill the condition part of a rule, the rule system executes its action part.

In our application, while tasks and processes are executed, *logs* are created within application components. We differentiate between effective and failure logs. The effective logs are grouped by BPMN process, micro agent, rule, and task start and end logs. Every time an agent, BPMN process, or a task is started or will terminate shortly after,

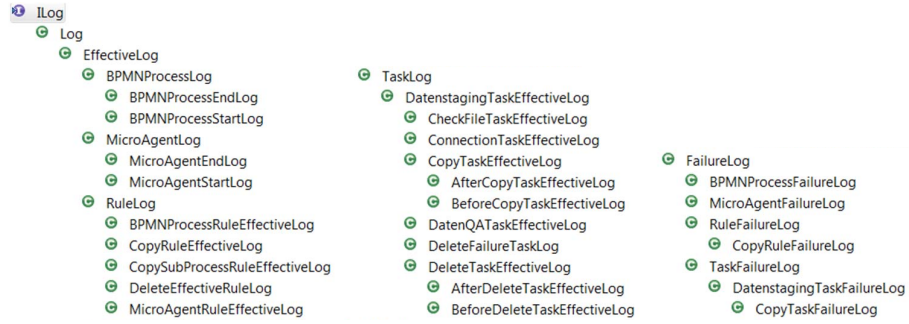


Fig. 5: Template hierarchy

a log will be created. In case of a failure within a task, rule, agent, or BPMN process a failure log is created. The basic layout of a log consists of the creation time, id, type, message, and parent id, but each log subtype extends this layout. For an overview of the currently used log types (implemented as Java classes), see Figure 5.

Working memory elements represent logs in the rule system. Consequently, in the condition part of a rule, types of logs and their fields can be tested. If a certain combination of logs was created, i.e. corresponding components were executed, a rule can fire and, thus, signal its execution with a further log. The rule depicted in Figure 6 creates a log of type *CopyRuleEffectiveLog* if two logs of type *BeforeCopyTaskEffectiveLog* and *AfterCopyTaskEffectiveLog* of the same process exist. Logs created through rule firing can be used in further rules for hierarchically testing log structures.

Thus, the application component implementor can model logs of a related type and rules describing certain effective or failure situations of an application component.

```
(defrule CopyTaskRuleEffective
;; Matches log-objects of type BeforeCopyTaskEffectiveLog
;; which deal with files of size not equal 0.
?ctlog1 <- (BeforeCopyTaskEffectiveLog
(taskStartTime ?tst)
(sourceFilesize ?sfs)
(processID ?pid1)
(test(= ?sfs 0)))
;; Matches log objects of type AfterCopyTaskEffectiveLog
;; which deal with of size not equal 0 and has the same
;; processID as above.
?ctlog2 <- (AfterCopyTaskEffectiveLog
(processID ?pid1)
(targetFilesize ?tfs)
(taskEndTime ?tet))
;; Task start time must be less task end time
(test (< ?tst ?tet))
(test (!= ?tfs 0))
=>
;; Creation of a combined working memory element representing
;; the firing of the rule.
(assert (CopyRuleEffectiveLog (logs ?ctlog1 ?ctlog2))))
```

Fig. 6: Example for a rule, written in CLIPS syntax, monitoring the copy task.

The monitoring component itself is implemented as an agent which continuously receives logs from the executing application agents (see Figure 7). This happens via so called *LocalMonitoringRepositories* and one central *MonitoringRepository* that store

the logs for later use. Thus, the monitoring component observes the execution of distributed acting agents in a central way. It further combines the results of the agents' activities through firing rules.

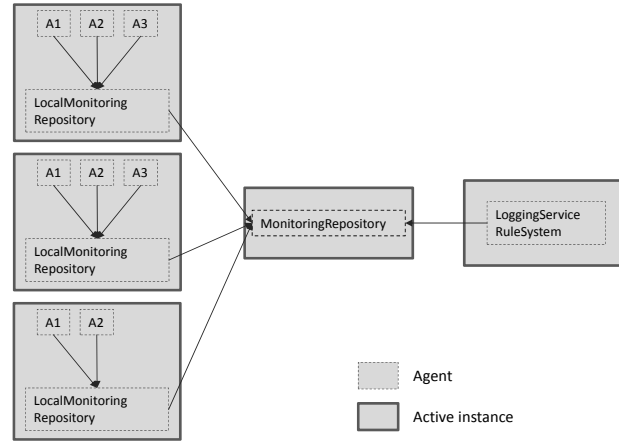


Fig. 7: Collecting logs from distributed agents running on different active instances

The monitoring component is part of a distributed business process execution application, which is currently under development. The application is implemented with JAVA and the extension Jadex for distributed agents.

## 5 Demonstrator

In the demonstrating example, we show the execution of an BPMN-modeled process for file transfer. For this task, following steps are executed:

- Start of the *Logging Service* agent that initializes the rule engine and waits for incoming logs.
- The specific working memory element *Clock* is initially created for representing the current time and, thus, enabling time-dependent rule firing.
- The BPMN-process is started by the user. Internally, new agents are created which follow the process model and execute the related basic component implementation. During the process execution specific logs are created like *BPMNProcessStartLog* and *BeforeCopyEffectiveLog*.
- Created logs and activated rules can be examined in a rule monitor window.
- Each fired rule creates new logs like the *CopyRuleEffectiveLog*.
- In a further run-through of the demonstrator another rule fires indicating that a copy task needs too much time to be processed.

## 6 Discussion and Summary

Other data integration tools already handle diverse data file formats, however, they blank out that organizational data primary exist as decentralized, distributed data files. User

of such systems are forced to manually move the data files to appropriate places or to develop scripts and programs that move them around in distributed systems. To the best of our knowledge, no other business intelligence software is focusing on the process execution monitoring via rules. Hereby, processes report their execution via logs (e.g. start and end logs) and rules observe those for identifying interesting monitoring occurrences. This rule-based approach has following advantages. First, we decouple the execution of the actual processes from their monitoring, i.e. we separate application logic from monitoring logic. If new interesting occurrences shall be recognized during execution the process component implementation has not to be changed but only rules have to be added. Thus, maintenance shall be simplified. Furthermore, by using rules, we allow a declarative representation of such interesting situations which can be modeled by domain experts, in the ideal case, e.g. when domain specific languages are introduced for rule modeling (see (Laun, 2011)). Such models (rules) can reflect on single processes as well as combinations of different processes or subprocesses. Similarly, results of rule firing can be aggregated through rule chaining. Thus, beside the typical procedural representation of process' behavior in BPMN diagrams, rules provide a declarative representation of the expected outcome of process execution. When rules are fulfilled, such informations can again be stored in repositories or communicated to user interfaces that present actual states of process execution (e.g. in a business process browser). The data-driven character of rule-based approaches enables a direct reaction and evaluation of current situations, like daemons who react actively on data occurrences. Contrarily, a database approach would need to actively apply queries on a database.

In this demonstration paper, we present a combination of process modeling based on BPMN, process execution in a distributed environment, and process execution monitoring with rules. The demonstrator shows how these technologies can successfully be combined to monitor process execution in a distributed environment.

## References

- Barringer, H., Rydeheard, D. E., and Havelund, K. (2010). Rule Systems for Run-time Monitoring: from Eagle to RuleR. *J. Log. Comput.*, 20(3):675–706.
- Braubach, L. and Pokahr, A. (2011). Addressing Challenges of Distributed Systems Using Active Components. In Brazier, F., Nieuwenhuis, K., Pavlin, G., Warnier, M., and Badica, C., editors, *Intelligent Distributed Computing V - Proceedings of the 5th International Symposium on Intelligent Distributed Computing (IDC 2011)*, pages 141–151. Springer.
- Friedman, T., Beyer, Mark, A., and Bitterer, A. (2008). Magic Quadrant for Data Integration Tools. Technical report, Gartner.
- Laun, W. (2011). Domain Specific Languages: Notation for Experts. In *International Conference on Reasoning Technologies (Rules Fest)*, San Francisco.
- Marino, J. and Rowley, M. (2009). *Understanding SCA (Service Component Architecture)*. Addison-Wesley Professional, 1st edition.
- OMG (2006). *Business Process Modeling Notation (BPMN) Specification, Final Adopted Specification*.
- Rao, A. and Georgeff, M. (1995). BDI Agents: from Theory to Practice. In Lesser, V., editor, *Proceedings of the 1st International Conference on Multi-Agent Systems (ICMAS 1995)*, pages 312–319. MIT Press.



# Loosely-Coupled and Event-Messaged Interactions with Reaction RuleML 1.0 in Rule Responder

Zhili Zhao<sup>1</sup>, Kia Teymourian<sup>1</sup>, Adrian Paschke<sup>1</sup>, Harold Boley<sup>2</sup>, Tara Athan<sup>3</sup>

<sup>1</sup> Freie Universität Berlin, Germany

{paschke, zhili, teymourian} AT inf.fu-berlin.de

<sup>2</sup> Information and Communications Technologies, National Research Council Canada  
Fredericton, NB, Canada

harold.bole AT nrc.gc.ca

<sup>3</sup> Athan Services, W Lafayette, IN, USA

taraathan AT gmail.com

**Abstract.** Reaction RuleML is one of the two major subfamilies of RuleML and acts as an interchange format for reactive rules and rule-based event-processing languages. Exemplified with a recent instantiation of Rule Responder, a rule-based inference agent middleware, we demonstrate the event messaging features of Reaction RuleML, which supports loosely-coupled interface-based interaction using rule signatures and decoupled communication via event messages.

## 1 Introduction

As one of the two major subfamilies of RuleML<sup>1</sup>, Reaction RuleML<sup>2</sup> presents a general compact rule interchange format for reaction rules, which are used to declaratively specify the reactive and behavioral logic of distributed systems and dynamic (Web-based) environments [17]. RuleML has broad coverage and is designed as an interchange language for the major kinds of (Web) rules. The RuleML family’s top-level distinction is *Deliberation rules* vs. *Reaction rules* [3]. *Deliberation rules* permit knowledge derivation and subsume further languages such as Hornlog (hence Datalog), which (syntactically) specialize to condition-less *Fact* and conclusion-less *Query* languages (the latter subsuming *Integrity Constraint (IC)* languages). On the other hand, *Reaction rules* focus on event-driven (re)actions in distributed and dynamic environments.

Reaction RuleML is intended as a common standard for representing reactive rules and rule-based complex event processing (CEP) in a platform independent XML markup language. It provides several layers of expressiveness for adequately representing reactive logic and for interchanging events (queries, actions, event data) and rules. As a whole, Reaction RuleML is characterized by the following features:

---

<sup>1</sup> <http://ruleml.org/>

<sup>2</sup> <http://reaction.ruleml.org/>

1. Reaction RuleML Metamodel, Semantic Types and Data Queries. Reaction RuleML is based on a metamodel and 'pluggable' ontologies and defines general concepts such as space, time, event, action situation, process, and agent in a modularized ontological top-level structure, with a left to right vertical order in the top-level ontologies. Therefore, it is possible for Reaction RuleML to support distributed and modularized knowledge bases through direct coupling via key references within a KB, `iri` pointers, and support for query languages.
2. Rule Interface Descriptions with Semantic Profiles and Signatures. Reaction RuleML separates the interface of a rule from its implementation. The interface describes the functional and non-functional (semantic) properties of a rule. The implementation, on the other hand, requires more flexibility and can be modified without any change of its interface.
3. Reaction RuleML Messaging. The interface description language of Reaction RuleML allows for loosely-coupled interaction with distributed inference services and agent KBs. Based on event messaging, Reaction RuleML also supports decoupled communication via event messages that are produced and published as Reaction RuleML serializations, e.g. on event streams or event clouds.

In this paper, exemplified with a recent instantiation of Rule Responder<sup>3</sup> [16, 15, 2], we demonstrate the distributed event-messaging interactions of Reaction RuleML 1.0 in loosely-coupled and de-coupled distributed rule-based agents. Reaction RuleML acts as a standardized interface description language and interchange format between these semantic agents which run their own platform specific rule engines and rule-based knowledge base (KB) at their core. The rest of the paper is organized as follows: Section 2 introduces Reaction RuleML and its reference application Rule Responder. In Section 3 we present the semantic interpretation and translation between Reaction RuleML as a standardized rule interchange language and several platform specific rule languages as well as the platform independent controlled English ACE. Section 4 presents how distributed event messaging supports loosely-coupled interaction with inference services/agents. Section 5 deals with decoupled communication via event messages. Finally, we conclude the paper with a summary in Section 6.

## 2 Reaction RuleML 1.0

Reaction rules are concerned with the invocation of actions in response to events and actionable situations [14]. They state the conditions under which actions must be taken and describe the effects of action executions. In the last decades various reaction rule languages and rule-based event processing approaches have been developed, which for the most part have been advanced separately. The Reaction RuleML standard<sup>4</sup> addresses four major reaction rule types: *Produc-*

---

<sup>3</sup> <http://responder.ruleml.org>

<sup>4</sup> <http://reaction.ruleml.org/>

*tion Rules* (Condition-Action rules), *Event-Condition-Action (ECA) rules*, *Rule-based Complex Event Processing (CEP)* (CEP reaction rules, (distributed) event messaging reaction rules, query reaction rules etc.), *Knowledge Representation (KR)* Event/Action/Situation Transition/Process Logics and Calculi

Reaction rules are defined by a general **Rule** element which can be specialized in the different Reaction RuleML branches to the four major types of reaction rules (and variants of these types). The following example shows the most general rule syntax of RuleML with of focus on Reaction RuleML. We use 1- or 2-letter indicators for syntax from Deliberation (D), Reaction (R), or Deliberation+Reaction (DR) RuleML.

```
<Rule @key @keyref @style>

  <!-- rule info and life cycle management, modularization -->

  <meta>    <!-- DR: (semantic) metadata of the rule -->          </meta>
  <scope>    <!-- R: scope of the rule e.g. a rule module -->      </scope>

  <!-- rule interface description -->

  <evaluation> <!-- R: intended semantic profiles -->              </evaluation>
  <signature> <!-- R: rule interface signature and modes -->      </signature>

  <!-- rule implementation -->

  <qualification> <!-- R: e.g. qualifying rule declarations, e.g.
                    priorities, validity, strategy -->              </qualification>
  <quantification> <!-- DR: quantifying rule declarations,
                    e.g. variable bindings -->                      </quantification>

  <on>         <!-- R: event part -->                                </on>
  <if>         <!-- DR: condition part -->                            </if>
  <then>       <!-- D: (logical) conclusion part -->                 </then>
  <do>         <!-- R: action part -->                                </do>
  <after>      <!-- R: postcondition part after action,
                    e.g. to check effects of execution -->          </after>
  <else>       <!-- DR: (logical) else conclusion -->                </else>
  <elsedo>     <!-- R: alternative/else action,
                    e.g. for default, exception handling -->        </elsedo>

</Rule>
```

*Rule Responder*<sup>5</sup> [16, 15, 2] is a reference application of Reaction RuleML. It is supporting distributed semantic multi-agent systems and rule-based inference services that run rule engines at their core and communicate using (Reaction) RuleML as a standardized rule interchange format. The Rule Responder Technical Group of RuleML is focused on implementing use cases that require the interchange of rule sets and support querying the distributed rule inference services. To implement different distributed system/agent topologies and semiotic structures with their negotiation/coordination mechanisms, Rule Responder instantiations employ three core classes of agents - Organizational Agents (OA), Personal Agents (PAs), and External Agents (EAs). An OA represents goals and strategies shared by its virtual organization (of agents) as a whole, using a rule base that describes its policies, regulations, opportunities, etc. OAs hence might act as centralized nodes in star-like distributed coordination networks.

<sup>5</sup> <http://ruleml.org/RuleResponder/>

They often follow an orchestration style execution logic where the OA is a centralized authority which orchestrates the other PAs. A PA assists a group or person/agent of the organization, semi-autonomously acting on their behalf by using a local knowledge base of rules defined by the entity. In decentralized distributed networks the PAs itself might communicate with each other following e.g. a choreography style coordination, e.g. for distributed problem solving. EAs can communicate with the virtual organization by sending messages to the public interfaces of the OA. EAs can be human users using, e.g., Web forms or can be automated services/tools sending messages via the multitude of transport protocols of the underlying enterprise service bus (ESB) middleware of Rule Responder. The agents employ ontologies in their rule-based knowledge bases to represent semantic domain vocabularies, normative pragmatics and pragmatic context of conversations and actions, as well as the organizational semiotics.

Since the Rule Responder framework has been conceived [16], many instantiations have been developed such as the Health Care and Life Sciences eScience infrastructure [11], Rule-based IT Service Level Management and the Rule Based Service Level Agreement (RBSLA) language [13], Semantic Business Process Management (BPM) [18, 12], WellnessRules(2) [1], PatientSupporter, and SymposiumPlanner systems<sup>6</sup>.

In this paper, we will employ the SymposiumPlanner 2011 to demonstrate the distributed event-messaging interactions in Rule Responder. SymposiumPlanner is a series of Rule Responder instantiations for the Questions&Answers (Q&A) sections of the websites of the RuleML Symposia since 2007.

### 3 Translator Service Framework

The design of Rule Responder follows the spirit of the OMG's Model Driven Architecture (MDA) approach [11, 15]:

1. On the computational independent level rules are engineered in a Rule Manager user interface in a natural controlled English language using blueprint templates and user-defined vocabularies and domain-specific translation rules.
2. The rules are mapped and serialized in Reaction RuleML which is used as platform independent rule interchange format to interchange rules between Rule Responder inference services (agents) and arbitrary other rule execution environments.
3. The Reaction RuleML rules are translated into the platform specific rule language for execution.

Rule Responder provides a translator service framework with Web form interfaces accepting controlled natural language inputs or predefined selection-based rule templates for the communication with external (human) agents on the computational independent level, as well as HTTP Rest and Web service interfaces,

---

<sup>6</sup> <http://ruleml.org/SymposiumPlanner/>

which can be used for translation into and from Reaction RuleML. In Rule Responder SymposiumPlanner 2011<sup>7</sup>, we also implemented a user client supporting queries in Attempto Controlled English (ACE) [5], which is a rich subset of controlled English designed to serve as a knowledge representation language. The demonstration of the SymposiumPlanner 2011 user client can be found at<sup>8</sup>. Before sending them to Rule Responder, the queries are translated into a discourse representation structure (DRS) by the Attempto Parsing Engine (APE)<sup>9</sup>. It is then fed into an XML parser which translates it into Reaction RuleML by an ACE2RML translator, which makes use of domain specific semantic vocabularies and domain rules [21].

On the platform-independent and platform specific level, Reaction RuleML can be translated or mapped into several domain specific reaction rule languages, which are run by platform specific rule engines, such as: Prova<sup>10</sup>, OO jDREW<sup>11</sup>, Emerald<sup>12</sup>, Euler, etc. The translator services are using different translation technologies such as XSLT stylesheet, JAXB, etc. to translate from and to Reaction RuleML and are configured in the transport channels of the inbound and outbound links of the deployed rule engines on the ESB. That is, incoming Reaction RuleML messages (receive) are translated into platform-specific rule bases which can be executed by different platform specific rule engines, e.g. Prova, and outgoing rule bases (send) are translated into Reaction RuleML in the outbound channels before they are transferred via a selected transport protocol such as HTTP or JMS, etc.

For example, a user query in ACE format: "Which papers are full and accepted?", which is used to get all full papers accepted by RuleML2011@IJCAI<sup>13</sup> is firstly translated into Reaction RuleML:

```
<?xml version="1.0" encoding="GBK"?>
<RuleML xmlns="http://www.ruleml.org/1.0/xsd"
  xsi:schemaLocation="http://www.ruleml.org/reaction/1.0/xsd
    http://ibis.in.tum.de/research/ReactionRuleML/1.0/rr.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <oid>
    <Ind>Generated message from ACE text "Which papers are full and accepted?".</Ind>
  </oid>
  <Message directive="query-sync">
    <oid>
      <Ind>RuleML-2011-IJCAI</Ind>
    </oid>
    <protocol>
      <Ind>esb</Ind>
    </protocol>
    <sender>
      <Ind>User</Ind>
    </sender>
    <receiver>
```

<sup>7</sup> <http://ruleml.org/SymposiumPlanner/documentation.html>

<sup>8</sup> <http://de.dbpedia.org/redirects/ruleml/ACE2ReactionRuleML/index.jsp>

<sup>9</sup> <http://attempto.ifi.uzh.ch/site/>

<sup>10</sup> <http://www.prova.ws/>

<sup>11</sup> <http://www.jdrew.org/ojdrew/>

<sup>12</sup> <http://lpis.csd.auth.gr/systems/emerald/>

<sup>13</sup> <http://www.defeasible.org/ruleml2011/>

```

<Ind>RuleML-2011-IJCAI</Ind>
</receiver>
<content>
  <Atom>
    <Rel>getPapers</Rel>
    <Ind>full</Ind>
    <Ind>accepted</Ind>
    <Var>B</Var>
  </Atom>
</content>
</Message>
</RuleML>

```

This example above also indicates the general message syntax of a Reaction Message [17]. In Reaction RuleML 1.0, each event message (the **Message** element) consists of a conversation identifier (the **oid** element), a pragmatic context description (the **directive** attribute), a transport protocol (the **protocol** element), such as HTTP, JMS, SOAP, etc., a sender (the **sender** element)/receiver (the **receiver** element) agent of the message and a message payload (the **content** element). When a message is sent from an External Agent, Rule Responder picks up the message, translates into a domain specific rule language and then sends it to a target agent. For example, the message of Reaction RuleML mentioned above is translated into a Prova message via XSLT sheet in SymposiumPlanner 2011, shown as follows:

```
[httpEndpoint:3,esb,httpEndpoint,query,[getPapers,full,accepted,<2901>]].
```

Each Prova message describes the messages which are received and sent by Prova agents and consists of constants, variables, or lists. For more information, see the Prova 3.0 Users Guide<sup>14</sup>. After the above Prova message is processed in the Prova rule engine, the resulting answer will be translated to Reaction RuleML before sending it to other agents.

Rule Responder's translation framework also supports the elementary translation between Drools<sup>15</sup> and Reaction RuleML. Drools is a business rule management system (BRMS) with a forward chaining production rule engine [20]. The production rule pattern of "when-then" in Drools can be represented by the pattern of "if-do" in Reaction RuleML, as shown in Figure 1. For more implementation details of the translation see [6].

## 4 Loosely-Coupled Interaction

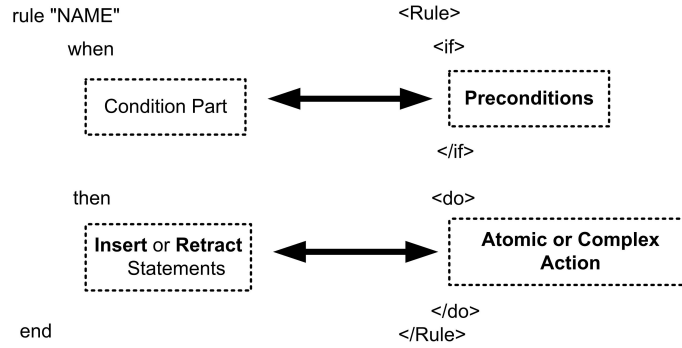
Reaction RuleML allows distributed event messaging interactions in loosely-coupled and decoupled distributed rule-based systems such as Web inference services and semantic agents. In this Section we will demonstrate how event messaging interaction plays an important role in Rule Responder.

The loosely-coupled interaction leads to a resilient relationship between distributed agents with some kind of exchange relationship. Each agent makes its

<sup>14</sup> <http://www.prova.ws/index.html?page=documentation.php>

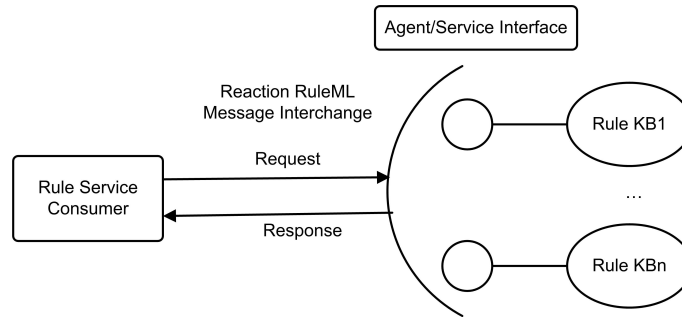
<sup>15</sup> <http://www.jboss.org/drools>

## Interactions with Reaction RuleML 1.0 in Rule Responder



**Fig. 1.** The Mappings between Drools and Reaction RuleML

requirements explicit and makes use of the public interface definitions of other agents for communicating with them, i.e., an agent publishes an interface definition (containing the public rule signatures), which can be accessed in one or many concrete ways by other agents - typically by a query to the agent using one of its public rule interface signatures. Instead of queries and answers, also an interchange of complete rules and rule bases as mobile rule code to an agent is possible. Their loosely-coupled dependency and their intended interpretation and execution semantics is specified by the interface and brings flexibility that a change in the underlying rule implementation does not necessarily require a change in the rule signature, except if the rule signature itself changes. Moreover, while the interfaces might be published publicly and can be queried by requesting agents, the concrete implementation of the rule base might be hidden and privately encapsulated in the knowledge base of the agent. Figure 2 demonstrates the loosely-coupled interaction in Rule Responder.



**Fig. 2.** Loosely-Coupled Communication via Messages to Agent Interface

Reaction RuleML 1.0 employs the Reaction RuleML Interface Description Language (RuleML IDL) [16] for describing functional and non-functional prop-

erties of a rule inference service and its rule-based KB. The functional description among others contains the signatures of public rule functions together with their term modes (input, output or arbitrary terms) and type declarations. For example, the signature of the aforementioned query of "getPapers" of SymposiumPlanner 2011 can be described as follows:

```
<signature>
  <Atom>
    <Rel>getPapers</Rel>
    <Var type="java://java.lang.String" mode="+"/>
    <Var type="java://java.lang.String" mode="+"/>
    <Var type="java://java.lang.String" mode="-"/>
  </Atom>
</signature>
```

Reaction RuleML distinguishes between the interface of a rule base or rule and its implementation. The signatures are defined in the interface either directly together with the implementation in one **<Rule>** or for better modularization and information hiding separated from the implementation of the rule on the level of a RuleML rule base **<Rulebase>** and asserted rule module **<Assert>**. The following example illustrates the use of such signature declarations in the interface descriptions of rules and distinguishes the interface from the implementation referring from the interface to the implementation via an XML key-keyref connection.

```
<!-- rule interface with two alternative interpretation semantics and a signature.
  The interface references the implementation identified by the corresponding key -->
<Rule keyref="r1">
  <evaluation index="1">
    <!-- WFS semantic profile define in the metamodel -->
    <Profile type="ruleml:Well-Founded-Semantics" direction="backward"/>
  </evaluation>
  <evaluation index="2">
    <!-- alternative ASS semantic profile define in the metamodel -->
    <Profile type="ruleml:Answer-Set-Semantics" direction="backward"/>
  </evaluation>
  <!-- the signature defines the queryable head of the backward-reasoning rule -->
  <signature>
    <Atom><Rel>getPapers</Rel><Var mode="+"/><Var mode="+"/><Var mode="-"/></Atom>
  </signature>
</Rule>

<!-- implementation of rule 1 which is interpreted either by WFS or by ASS semantics
  and onyl allows queries according to it's signature definition. -->
<Rule key="r1" style="reasoning">
  <if>... </if>
  <then>
    <Atom><Rel>getPapers</Rel><Var>Type</Var><Var>Status</Var><Var>Papers</Var></Atom>
  </then>
</Rule>
```

The signatures can be also defined or just referred to via key-keyref in the **<signature>** of a **<Rulebase>**.

This enables a loosely-coupled interaction with the inference service / agent, where queries can be posed against the public interface **signature** and interpreted with the intended semantics **evaluation**. Therefore, the interface also defines the applicable evaluation semantics, which in the example uses predefined semantic **Profiles** from the RuleML metamodel. This is in particular



useful for mobile code, i.e. rule bases which are uploaded to an inference service, since the underlying rule engine needs to support the intended semantics. It is also useful for verification and validation [8, 10, 7, 4], explanations, and proofs of answers to queries which are dependent on the applied semantics.

During the communication, Rule Responder represents the interactions between distributed agents via constructs for asynchronously sending and receiving event messages. Therefore it uses Reaction RuleML’s support for messaging in the CEP Reaction RuleML branch. For sending and receiving (event) messages, Reaction RuleML 1.0 supports serial messaging CEP reaction rules that `<Receive>` and `<Send>` events in arbitrary combinations. A serial (messaging) reaction rule starts with a receiving event (`<on>`) followed by any combination of conditions (`<if>`), events (`<Receive>`), and actions (`<Send>`) in the body of the rule for expressing complex event processing logic. This flexibility with support for modularization and aspect-oriented weaving of reactive rule code is in particular useful in distributed systems where event processing agents communicate and form a distributed event processing network, as e.g. in the following example:

```
<Rule style="active">
  <on><Receive> receive event from agent 1 </Receive></on>
  <do><Send> query agent 2 for regular products in a new sub-conversation </Send></do>
  <on><Receive> receive results from sub conversation with agent 2 </Receive></on>
  <if> prove some conditions, e.g. make decisions on the received data </if>
  <do><Send> reply to agent 1 by sending results received from agent 2 </Send></do>
</Rule>
```

These Reaction RuleML messaging constructs can directly map to the messaging reaction rules in Prova with: *sendMsg* predicates to send messages, reaction *rcvMsg* rules which react to inbound messages, and *rcvMsg* or *rcvMult* inline reactions in the body of messaging reaction rules to receive one or more context-dependent multiple inbound event messages, shown as follows:

```
sendMsg(XID,Protocol,Agent,Performative,Payload |Context)
rcvMsg(XID,Protocol,From,Performative,Paylod|Context)
rcvMult(XID,Protocol,From,Performative,Paylod|Context)
```

where *XID* is the conversation identifier. *Protocol* defines the communication protocol. *From* denotes the source of the message. *Performative* describes the pragmatic context in which the message is sent. And *Payload—Context* denotes the actual content of the event message.

The event messages between distributed agents conversation invoke the rule functions of the receiving agents if there exists a matching rule interface. For instance, the example given in Section 3 indicates that the receiver agent “RuleML-2011-IJCAI” needs to specify an appropriate signature for “getPapers” queries. In SymposiumPlanner 2011, the receiver “RuleML-2011-IJCAI” agent is a Prova engine, which implements the interface definition via its platform specific rule syntax: `interface(getPapers(Type, Status, Papers),getPapers(" +", "+", "-"), "return related papers of RuleML-2011@IJCAI.")`. This public interface can be queried in backward-reasoning style in a Prova engine and a “no\_further\_answers” message will be sent to the sender if there is no suitable public interface is found:

```
% look-up interface
processMessage(XID,From,Primitive,[X|Args]):-
  not(interface([X|Args],ModeDeclarations,Description)),
  sendMsg(XID,esb,From,"answer", noPublicInterface(interface([X|Args]))),
  sendMsg(XID,esb,From,"no_further_answers", [X|Args]),
  fail().
```

The implementation of a rule interface can be implemented by arbitrary rule agents, which might have different levels of expressiveness. For example, the implementation of the interface "getPapers" in Prova is shown as follows:

```
getPapers(XID, Type, Status, Papers):-
  sysTime(CT),

  @paperType(Type)
  getAcceptedPapers(Papers)[validate(CT)].

validate(CT) :-
  compare(CT,'> ',datetime(2011,5,31,0,0,0)).

@paperType(full)
getAcceptedPapers(Papers) :-
  QueryString = '
    PREFIX foaf: <http://xmlns.com/foaf/0.1/>
    PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
    PREFIX dc: <http://purl.org/dc/elements/1.1/>
    PREFIX swrc: <http://swrc.ontoware.org/ontology#>
    SELECT ?paper ?title
    FROM <http://de.dbpedia.org/redirects/ruleml/ruleml2011.rdfs>
    WHERE {
      ?paper a ?type .
      ?paper dc:title ?title .
      FILTER (?type = <http://ruleml.org/ontology#FullPaper> ) .
    }
  ',
  sparql_select(QueryString,[title(Papers)]).

@paperType(short)
getAcceptedPapers(Papers) :-
  QueryString = '
    PREFIX foaf: <http://xmlns.com/foaf/0.1/>
    PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
    PREFIX dc: <http://purl.org/dc/elements/1.1/>
    PREFIX swrc: <http://swrc.ontoware.org/ontology#>
    SELECT ?paper ?title
    FROM <http://de.dbpedia.org/redirects/ruleml/ruleml2011.rdfs>
    WHERE {
      ?paper a ?type .
      ?paper dc:title ?title .
      FILTER (?type = <http://ruleml.org/ontology#ShortPaper> ) .
    }
  ',
  sparql_select(QueryString,[title(Papers)]).

...
```

Prova supports modularization of its knowledge base and allows constructing metadata based views on the knowledge base, so called scopes. For example, the annotation "@paperType(Type)" on the followed goal literal "getAcceptedPapers(Papers)" is a scope constraint which applies the goal literal only on the target rule with matching metadata ("@paperType(full)", "@paperType(short)", etc.) during unification, i.e. there must be a match between the value given for the annotation @paperType and the value listed for the key in the target rule

of `getAcceptedPapers`. In the example, it would bind the metadata annotation values "full", "short", etc. to the variable "Type". The metadata can act as an explicit scope for constructive queries (creating a view) on the knowledge base and enables scoped (meta) reasoning with the semantic annotations. Besides, Prova supports literal guards which act as additional precondition constraints. In the above example, the goal literal is only available "after 31st, May, 2011", which is defined by the guard "[validate(CT)]" and its implementation as a rule "validate(CT):- compare(CT,'i',datetime(2011,5,31,0,0,0)).".

Reaction RuleML 1.0 provides corresponding expressiveness for metadata annotations `<meta>`, scope definitions `<scope>` and guards `<guard>`, which can be defined on the global level of a rule module and rule base as well as on the level of rules and literals. Scopes defined on the level of rule bases/modules set the context in which the knowledge of the rule base/module is applied, i.e. all queries and goal literals automatically apply within the scope. Nested scopes can be defined which override and specialize the outer (global) scopes, e.g. a scope within a rule `<Rule>` and on a particular goal literals `<Atom>` within the body of a rule. Scopes are e.g. useful to implement and distinguish different (behavioral) roles of a rule-based agent as scoped rule modules in the agent's knowledge base. Scopes are also useful to implement reactive workflow logics and (transactional) update logics [9].

## 5 Decoupled Interaction

The event messaging in Rule Responder also enables completely decoupled interaction via standardized Reaction RuleML event messages. Here some agents are event producers which publish events, e.g. in an event stream or in an event cloud / data source, irrespective of the event consumers. Other agents are consumers which try to detect and consume relevant events on those streams applying rule-based complex event processing techniques. That is, in difference to the loosely-coupled interaction, where the events are sent directly to other agents and the interaction with them takes place in a loosely-coupled way according to their interface definitions, the events in the decoupled scenario are just published, but there is no direct interaction with the consumers of those events.

For the decoupled interaction the message content itself is an event. Like for rules, the generic syntax pattern for an **Event** again distinguishes between the general event information, the event interface with the signature defining the event pattern (event type) and the concrete implementation in terms of an event instance.

```
<Event @key @keyref @iri @type>
  <!-- event info and life cycle management, modularization -->
  <oid>    <!-- R: event instance object id -->          </oid>
  <meta>   <!-- R: (semantic) metadata of the event -->  </meta>
  <scope>  <!-- R: scope of the event -->                </scope>
  <!-- event pattern description -->
  <evaluation> <!-- R: semantics: selection, consumption policies --> </evaluation>
  <signature> <!-- R: event pattern declaration -->      </signature>
  <!-- event instance -->
  <qualification> <!-- R: e.g. qualifying event declarations, e.g.
```

```

                                priorities, validity, strategy -->          </qualification>
<quantification> <!-- R: quantifying rule declarations -->          </quantification>

                                <!-- R: event instance content -->        </content>
</Event>

```

Reaction RuleML 1.0 provides the support for rule-based event processing and semantic complex event processing. With its typed logic, RuleML provides the support for (re)using external temporal, spatial, situation, event, and action ontologies and a metamodel which can be applied in the definition of semantic event/action types and temporal and spatial relations [3, 17]. Reaction RuleML defines a library of typical event, action, interval algebra operators and generic elements such as **Event**, **Action**, **Situation**, **Time**, **Location**, **Interval**, **Operator**. The type of these generic elements can be defined by an **@type** reference to external ontologies, e.g. to the Reaction RuleML metamodel (see [17]). For instance, `<Operator type="ruleml:Sequence">` instead of `<Sequence>`. The following example shows a complex event pattern definition:

```

<Event key="ce2" type="ruleml:ComplexEvent">
  <signature> <!-- pattern signature definition -->
    <Sequence>
      <!-- atomic event -->
      <signature>
        <Event type="ruleml:SimpleEvent">
          <signature><Atom>...event_A...</Atom></signature>
        </Event>
      </signature>
      <!-- nested complex event referenced by @keyref -->
      <signature><Event type="ruleml:ComplexEvent" keyref="ce1"/></signature>
      <!-- Common Base event selected via xpointer/xpath query in iri attribute -->
      <signature>
        <Event type="cbe:CommonBaseEvent" iri="cbe.xml#xpointer(//CommonBaseEvent)"/>
      </signature>
    </Sequence>
  </signature>
</Event>

<Event key="ce1">
  <signature> <!-- event pattern signature -->
    <Concurrent>
      <Event><meta><Time>...t3</Time></meta><signature>...event_B</signature></Event>
      <Event><meta><Time>...t3</Time></meta><signature>...event_C</signature></Event>
    </Concurrent>
  </signature>
</Event>

```

Such a complex event pattern definition can be used for event detection in the `<on>` part of a reaction rule of a rule-based event consuming agent:

```

<Rule style="active">
  <on><Event keyref="ce2"/></on>
  ...
  <do> ... </do>
</Rule>

```

These Reaction RuleML rules for Complex Event Processing (CEP) can be translated and executed e.g. in Prova. For an overview on typical (complex) event pattern functions and their implementations see [19]<sup>16</sup>.

<sup>16</sup> slides at <http://goo.gl/E30Vu>

In our SymposiumPlanner demo scenario we consume and process the events of the symposium, such as the news from the Twitter feed, calendar events (deadlines etc.), etc. We apply a typical publish-subscribe approach where users can subscribe their information needs in terms of (complex) event pattern to the rule-based semantic event processing agents. The agents actively inform the subscribers if they detect the relevant event patterns by continuously processing the published events on the news feeds.

## 6 Summary

In this paper, we presented how the standardized Reaction RuleML 1.0 interchange format supports loosely-coupled and de-coupled event-messaged interactions in the rule-based semantic multi-agent system Rule Responder. We demonstrated several expressiveness features of Reaction RuleML 1.0 on the example of the Symposium Planner use case. We also showed how the computational independent (natural) language Attempto Controlled English (ACE) is used to construct user queries against rule-based KBs in distributed Rule Responder agents (inference services), which are using Reaction RuleML as an intermediary platform-independent language between the computational independent user interface language (ACE) and the platform-specific execution languages (Prova, OO jDrew, Drools, ...).

## References

1. Harold Boley, Taylor Osmun, and Benjamin Craig. Social Semantic Rule Sharing and Querying in Wellness Communities. In Asuncin Gmez-Prez, Yong Yu, and Ying Ding, editors, *The Semantic Web*, volume 5926 of *Lecture Notes in Computer Science*, pages 347–361. Springer Berlin / Heidelberg, 2009.
2. Harold Boley and Adrian Paschke. Rule Responder Agents Framework and Instantiations. In Atilla Eli, MamadouTadiou Kon, and MehmetA. Orgun, editors, *Semantic Agent Systems*, volume 344 of *Studies in Computational Intelligence*, pages 3–23. Springer Berlin Heidelberg, 2011.
3. Harold Boley, Adrian Paschke, and Omair Shafiq. RuleML 1.0: The Overarching Specification of Web Rules. In *RuleML*, pages 162–178, 2010.
4. Jens Dietrich and Adrian Paschke. On the Test-Driven Development and Validation of Business Rules. In *ISTA*, pages 31–48, 2005.
5. Norbert E. Fuchs, Kaarel Kaljurand, and Gerold Schneider. Attempto Controlled English Meets the Challenges of Knowledge Representation, Reasoning, Interoperability and User Interfaces. In Geoff Sutcliffe and Randy Goebel, editors, *FLAIRS Conference*, pages 664–669. AAAI Press, 2006.
6. Tichomir Jabarski. Design and Development of A Translator Framework for Rule Languages Based on RuleML, Master Thesis. Master’s thesis, Free University Berlin, 2012.
7. A. Paschke, J. Dietrich, A. Giurca, G. Wagner, and S. Lukichev. On Self-Validating Rule Bases. In *Int. Semantic Web Enabled Software Engineering Workshop (SWESE’06)*, 2006.

8. Adrian Paschke. The ContractLog Approach Towards Test-driven Verification and Validation of Rule Bases - A Homogeneous Integration of Test Cases and Integrity Constraints into Dynamic Update Logic Programs and Rule Markup Languages (RuleML). In *IBIS, TUM, Technical Report 10/05*, 2005.
9. Adrian Paschke. ECA-RuleML: An Approach Combining ECA Rules with Temporal Interval-based KR Event/Action Logics and Transactional Update Logics. *CoRR*, abs/cs/0610167, 2006.
10. Adrian Paschke. Verification, Validation and Integrity of Distributed and Interchanged Rule Based Policies and Contracts in The Semantic Web. In *In Second International Semantic Web Policy Workshop (SWPW06)*, pages 2–16, 2006.
11. Adrian Paschke. Rule Responder HCLS eScience Infrastructure. In *Proceedings of the 3rd International Conference on the Pragmatic Web: Innovating the Interactive Society*, ICPW '08, pages 59–67, New York, NY, USA, 2008. ACM.
12. Adrian Paschke. A Semantic Rule and Event Driven Approach for Agile Decision-Centric Business Process Management - (Invited Paper). In *ServiceWave*, pages 254–267, 2011.
13. Adrian Paschke and Martin Bichler. Knowledge Representation Concepts for Automated SLA Management. *Decision Support Systems*, 46(1):187–205, 2008.
14. Adrian Paschke and Harold Boley. Rules Capturing Events and Reactivity. In Adrian Giurca, Dragan Gasevic, and Kuldar Taveter, editors, *Handbook of Research on Emerging Rule-Based Languages and Technologies: Open Solutions and Approaches*, pages 215–252. IGI Publishing, May 2009.
15. Adrian Paschke and Harold Boley. Rule Responder: Rule-Based Agents for The Semantic-Pragmatic Web. *International Journal on Artificial Intelligence Tools*, 20(6):1043–1081, 2011.
16. Adrian Paschke, Harold Boley, Alexander Kozlenkov, and Benjamin Larry Craig. Rule responder: RuleML-based Agents for Distributed Collaboration on The Pragmatic Web. In *ICPW*, pages 17–28, 2007.
17. Adrian Paschke, Harold Boley, Zhili Zhao, Kia Teymourian, and Tara Athan. Reaction RuleML 1.0: Standardized Semantic Reaction Rules. In *Proceedings of RuleML 2012*, 2012.
18. Adrian Paschke and Alexander Kozlenkov. A Rule-based Middleware for Business Process Execution. In *Multikonferenz Wirtschaftsinformatik*, 2008.
19. Adrian Paschke, Paul Vincent, Alexandre Alves, and Catherine Moxey. Tutorial on Advanced Design Patterns in Event Processing. In *DEBS*, pages 324–334, 2012.
20. L.M. Surhone, M.T. Tennoe, and S.F. Henssonow. *Drools*. VDM Verlag Dr. Mueller AG & Co. Kg, 2010.
21. Zhili Zhao, Adrian Paschke, Chaudhry Usman Ali, and Harold Boley. Principles of The SymposiumPlanner Instantiations of Rule Responder. In *Proceedings of The 5th International Conference on Rule-based Modeling and Computing on The Semantic Web*, RuleML'11, pages 97–111, Berlin, Heidelberg, 2011. Springer-Verlag.

# Graph-based rule editor

Maciej Nowak, Jaroslaw Bak, 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 rules. This tool uses a graph-based Palantir tool environment as a user interface to model rule conditions and conclusions. It is also used to visualize data and results of reasoning. We present a process of converting graph models stored in an XML format file into the Jess knowledge base and rules. Results obtained in the reasoning process are presented to the user in the same form as source data.

**Keywords.** graphical rule representation, Jess, Palantir, reasoning

## 1 Introduction

Rule engines are becoming one of the most commonly used technologies in business and engineering projects. The usage of rule engines enables the reasoning process which enriches gathered data and searching methods, utilizing pattern matching applied in rules. Rules and rule engines are successfully used in: expert systems, business processes, data integration and transformation, and in other applications requiring intelligent data processing. Despite the clear advantages of rule-based technologies, there are many software application areas where they occur in a relatively simple form, e.g. by using filters. In the area of criminal analysis, the addition of rule engines to investigation systems would enable analysts to discover very complex criminal schemes, totally beyond the capacity of traditional systems.

A wide range of rule environments have been proposed, each with its own syntax and semantics for a rule engine and interface. The process of acquiring the knowledge can be simplified with the use of a graphical representation of rules and a user-friendly interface.

The main aim of this demo paper is to present a graph-based tool, in which an untrained analyst is able to construct a set of simple rules and use them in order to obtain new (inferred) information. The rules constitute the expert's knowledge of a given domain, while facts represent data. Both rules and facts are expressed graphically in the form of directed graphs. Rules can be applied to facts using a reasoning engine. After the inference process, a user gets the result which can be a graph with:

- the addition of new objects and/or relations,
- the modification of existing objects and/or relations,

- the lack of objects and/or relations that were deleted.

The paper is organized as follows. Section 2 presents the main overview of the proposed approach and related work. Section 3 describes a prototypical implementation and applied tools. Section 4 demonstrates an example which reflects a fragment of analysis of the real-world crime case. Section 5 contains concluding remarks.

## 2 Graph-based Rule Representation

### 2.1 Existing methods

Graphical rule representation and creation has been the subject of research conducted by many investigators and companies. Some efforts have sought to standardize graphical notations, for example: Unified Modeling Language/Object Constraints Language (UML/OCL) [5], UML-based Rule Modeling Language (URML) [6] or Object Role Modeling (ORM) [7]. Among them, the ORM language is the most intuitive and easy to use for people who are not familiar with the complex syntax of rules and UML/OCL. The ORM concepts were adopted [8] also in the SBVR (Semantics of Business Vocabulary and Business Rules) standard [8]. Our ideas are based on ORM and graph-based representation. Other popular rule representations include: decision tables, decision trees and eXtended Tabular Trees [10].

Tools that implement graphical rules representation are:

- Visual Rules [11] – supports building flow rules and decision tables using rich and intuitive graphical editors.
- Drools Guvnor [12] – provides many ways of representing rules: guided editor (easy to use, but not graphical), guided decision tables creation, rule flows (which represent the flow of logic).
- VisiRule [13] – is an extension to Win-Prolog which supports building decision models using a graphical paradigm. It offers graphical representation of forward chaining rules, with access to Prolog.
- OntoStudio Graphical Rule Editor [14] – is based on ObjectLogic [15] internally. It supports drawing rule diagrams, which consist of concepts, attributes of these concepts and relations between them. It does not allow the comparison of variables (only comparisons between values and variables are allowed).

In this work we give only a short overview of the main differences. Detailed comparisons among the mentioned standards will be presented elsewhere.

In most of the current approaches, rules are created to control data workflows and making decisions, while we apply rules to discover new information and to process data. Accordingly, one rule (LHS and RHS respectively) is represented by two graphs. Tools like Visual Rules, Drools Guvnor etc. are rule authoring frameworks while our approach is only an attempt to integrate rules and data in one graph-based form and perform reasoning. Such work, to the best of our knowledge, has not yet been done for the Jess engine.



## 2.2 Overview of the approach

The main goal of this paper is to present the graph-based tool, in which a user can: import data, construct rules, perform reasoning and obtain results. Rules and data, represented graphically, can be more easily understood by an untrained analysts and by engineers without intensive training. Our aim is to provide an easy-to-use and easy-to-understand analytical tool which can be used in many domains where rules and graphs can be employed to support a user's work.

The process of rule creation consists of creating two graphs which will later serve as sides of the constructed rule: the LHS (left hand side, called the body) and the RHS (right hand side, called the head). In our approach rules should be understood as *if LHS then RHS* statements. These rules (expressed in the Jess language) can be used to infer new information in a given rule-based knowledge base.

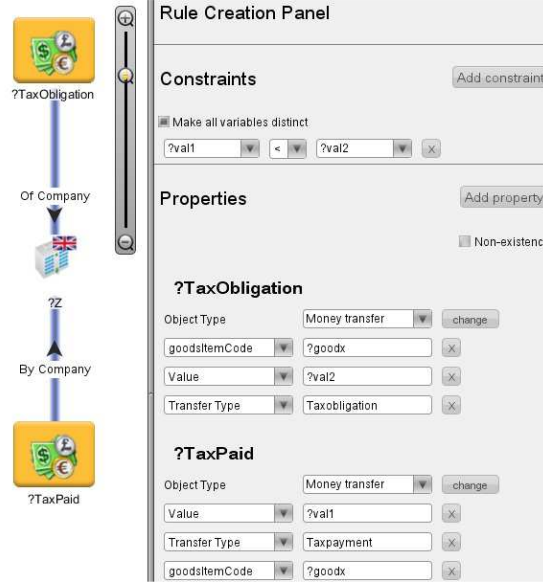
The LHS is built from condition elements (patterns) that need to be fulfilled in order to execute instructions written in the RHS. There are two types of conditions: the (non-) existence of a fact in the knowledge base with specified attributes, and the relationship between two attributes of existing facts. The execution of the rule may cause one of the following results: modification or removal of an existing fact, or addition of a new one. These operations are defined in the RHS of a rule.

It is possible to represent conditions from the body of a rule in a graphical form, more precisely in a graph. The graph consists of nodes and edges. The nodes are graphical representation of objects from the Palantir ontology (see Section 3.1), and the edges are the relations between them. Objects can have many properties; the type of an object is the most important one. Relations do not possess properties other than a type. The presence of an object in the graph means that a representing fact should exist in the knowledge base with attributes equal to the properties of the object. The presence of an object or a relation on a red background means that these artefacts should not exist in the knowledge base. The red colour on the graph expresses the negation of existence of objects or relations.

The construction of a rule is made with the following steps (within the Palantir environment):

1. A user creates a graph which constitutes the body of the rule, the conditions. The user creates objects and relations between them. Values of objects' properties, variables and constraints are specified in the Rule Creation Panel (RCP).
2. The user creates another graph, a modification of the first one which constitutes the head of a rule. The user adds/removes/modifies objects or relations of this graph. Conclusions - changes in the knowledge base after the application of the rule - can be modelled as the difference between two graphs.

Such an approach allows modelling of rules depending on object types, relations between them and values of objects' properties. It allows comparing attributes' values with each other, which is a significant advantage over some other tools (e.g. OntoStudio). There is no graphical way of presenting the comparison on the graph, so the only solution is to present it in the corresponding panel. For this purpose, we use a simple tab called Rule Creation Panel, which is presented in Figure 1.



**Figure 1.** Attributes and relationships of selected objects (highlighted in yellow).

Created rules need to be applied to the working memory built from facts. In this paper we present a converter (see Section 3.3), which transforms rules and data to the Jess engine according to the structure expressed in the Palantir ontology. After the reasoning process, a user obtains results presented on a new graph (in comparison to the source data graph).









In the Jess language, we represent objects and relations from the graph as *triples* (as in RDF) [16] in order to express dependencies between objects and their attributes. The triple consists of subject, predicate and object. Each relation (edge) from the graph is mapped as the predicate, with its starting point as the subject and the ending point as the object of a triple. Each subject (node) has a set of properties, where: the *id* of the node constitutes the subject; property name corresponds to the predicate; and the value constitutes the object (in triple-based representation). Such an approach can be compared to the OWL Web Ontology Language, where ObjectProperties represent relations between objects and DatatypeProperties represent links from individuals (objects) to data values. Employing the triple-based representation we are able to apply OWL ontology in the future.

We have defined mapping for a bidirectional transformation between Palantir and Jess, executed by the XMLtoJess converter. Table 1 presents available expressions, with examples in the Jess language and graph elements.

The authors of this paper have successfully used rule engines in the past [16, 18, 19]. They were used during investigations of a number of cases. For some economic crime, the complete model of a crime investigation was constructed. That allowed achieving a result in a fully automatic way, but each time the rule set was made by a programmer experienced in the Jess language after consultation with a business specialist. We want to shorten this process with the help of the proposed system, and to increase analytical flexibility by including new elements of crime schemes.

The introduction of the rule engine offers not only the possibility of reasoning about complex dependencies, but also to performing queries. Any graph containing nodes and edges can be entered as a search phrase. Rule engine will search the whole knowledge base for a given set of conditions, and return all objects that meet the specified requirements.

**Table 1.** Representations of main elements, using a Graph and RCP panel, in the Jess code.

Element	Graph and RCP panel representation	Jess code
Object	 ?Y	<code>(triple (subject ?Y) (predicate "Object Type") (object "GBOrganization"))</code>
Relation	 By Company  ?TaxPaid ?Y	<code>(triple (subject ?TaxPaid) (predicate "relation-By Com- pany") (object ?Y))</code>
Attribute Value	 Value <input type="text" value=""/> ?paidval ?TaxPaid	<code>(triple (subject ?TaxPaid) (predicate "property-Value") (object ?paidval))</code>
Comparison of attribute values	<input type="text" value=""/> ?oblval <input type="text" value=""/> > <input type="text" value=""/> ?paidval	<code>(test (&gt; ?oblval ?paidval))</code>
Declaration of non-existence (red background)	 ?TaxPaid	<code>(not (triple (subject ?TaxPaid) (predicate "Object Type") (object "MoneyTransfer")))</code>
Distinction of Variables	 ?X  ?Y  Constraints Make all variables distinct	<code>(test (neq ?X ?Y))</code>
Addition of an object/relation/attribute	New object/relation/attribute on the RHS (We add a new object/relation/attribute to a graph.)	<code>(assert (triple (subject ?TaxObligation) (predicate "Object Type") (object "MoneyTransfer")))</code>
Modification of existing object/relation/attribute	Modified object/relation/attribute on the RHS (We modify an object/relation/attribute in a graph.)	<code>(modify ?f (object "DefaultingTrader"))</code>
Removal of object/relation/attribute	Lack of object/relation/attribute on the RHS (We delete object/relation/attribute from a graph.)	<code>(retract ?f)</code>

### 3 Technologies Used

#### 3.1 Palantir Government Graph Application

Palantir Government [1] is a Java-based platform for analysing and visualizing data. It is widely used by financial (Palantir Finance) and government agencies. It is

capable of importing data structured in many various formats (such as Excel), and, due to the Palantir Dynamic Ontology (PDO) [2], objects inside the platform possess some semantic background meaning, which can be easily transformed into rules. The PDO is very simple; it only indicates that two objects are connected with a certain relation (represented then on a graph by an icon or relation).

Graph is the most sophisticated part of the Palantir Platform. It provides visualization of input data, with the structure defined in the given ontology. Properties of each object are not visible directly on the graph; they are reachable under the "Browser" tab. It is possible to export information from the graph into an external XML file, which is an essential element of the integration with a rule engine.

### 3.2 The Jess Rule Engine

Jess [3] is a rule engine and rule-based environment for building expert systems. It uses an enhanced version of the Rete [4] algorithm, which processes rules and facts in a very efficient way. Jess supports forward and backward chaining, working memory queries and many other useful features. Jess is provided as a library written in the Java language. It can easily be embedded into other Java applications. We applied Jess and its forward reasoning as extensions to the Palantir Government tool.

### 3.3 XMLtoJess Converter

XML is used as the interchange format between Jess and Palantir modules. Rule engines require input knowledge in form of facts, and that is why XMLtoJess converter is an essential part of the presented method.

The XMLtoJess converter is used to extract objects and relations stored in a Palantir XML (pXML) document generated from Palantir and create the Jess knowledge base. pXML format is the default output structure of the Palantir Platform. It holds information about objects in the Graph and all properties related to selected objects.

## 4 Example

In this section, we provide an example which reflects part of the analysis of a real-world crime case, the VAT carousel crime, also called the Missing Trader Intra-Community crime (MTIC). It is a sophisticated international fraud exploiting Value Added Tax (VAT) evasion, in order to create large amounts of unpaid VAT liabilities and VAT repayment claims connected with them. More information can be found on the demo site [17] and in [18].

Figure 2 depicts a graphical representation of the rule presented on the next page (where letters are used as shortcuts: *s* – subject, *p* – predicate, *o* – object).

```
(defrule VATFraudsterRule
  ?f <- (triple (s ?Z) (p "Object Type") (o "GBOrganization"))
  (triple (s ?TaxObligation) (p "Object Type") (o "MoneyTransfer"))
  (triple (s ?TaxObligation) (p "property-Transfer Type") (o "Taxobligation"))
  (triple (s ?TaxObligation) (p "property-goodsItemCode") (o ?good1))
```

```

(triple (s ?TaxObligation) (p "relation-Of Company") (o ?Z))
(triple (s ?TaxObligation) (p "property-Value") (o ?oblval))
(triple (s ?TaxPaid) (p "Object Type") (o "MoneyTransfer"))
(triple (s ?TaxPaid) (p "property-Transfer Type") (o "Taxpayment"))
(triple (s ?TaxPaid) (p "property-goodsItemCode") (o ?good1))
(triple (s ?TaxPaid) (p "relation-by Company") (o ?Z))
(triple (s ?TaxPaid) (p "property-Value") (o ?paidval))
(test (> ?oblval ?paidval))
(test (neq ?TaxObligation ?TaxPaid))
=>
(modify ?f (object "VATFraudster"))

```

This rule modifies the icon of the company which pays obligatory tax, less than it should be. As a result, this company is called a VAT fraudster. Unfortunately, the Palantir Government tool limits objects to one type (some additional types can be deduced only by an engineer according to the given Palantir Dynamic Ontology).

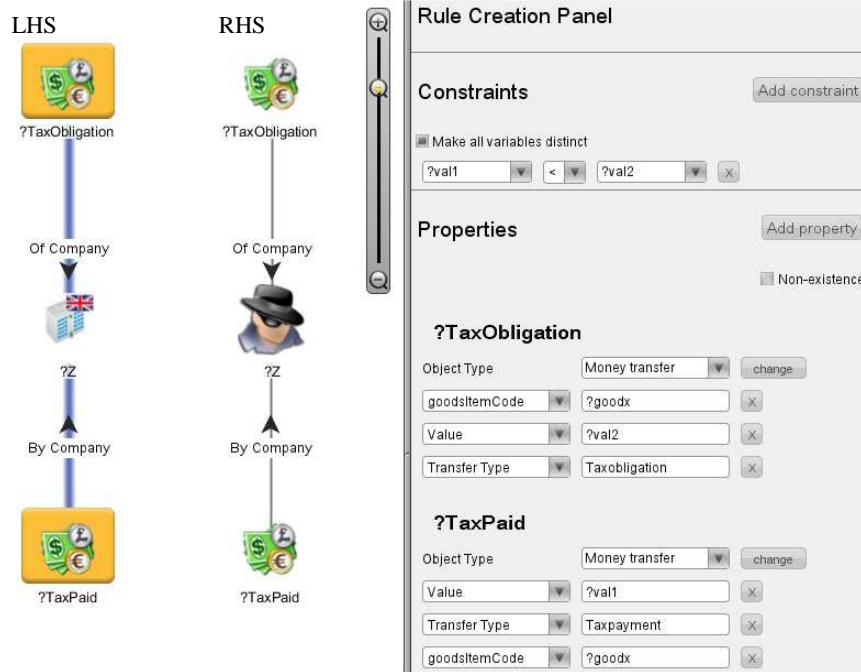


Figure 2. An exemplary VAT fraudster rule.

## 5 Conclusions

In this paper we have demonstrated a tool which supports graph-based creation of rules for the Jess engine. The tool integrates data and rules in the Palantir Government tool. Graph-based representation is convenient and intuitive for an untrained analysts. Such tool can be used in many domains where rules and graphs can be employed to support a user in her/his work.

Because of copyright issues connected with the Palantir Government application, we provide only the presentation which contains screenshots of executing the example

concerning an analysis of a real-world criminal case. The presentation with a more detailed description is available on the demo site [17].

**Acknowledgement.** This work was supported by DS-MK 45-102/12 and 45-085/11 DS-PB grants.

## References

1. Palantir Government Platform, <http://palantir.com/government>
2. Palantir Dynamic Ontology, <https://wiki.palantir.com/pgdz/palantir-dynamic-ontology-properties.html>
3. Jess (Java Expert System Shell), <http://jessrules.com/>
4. Forgy C., Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem, *Artificial Intelligence*, 19, pp. 17-37, 1982.
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-il/?q=URML>
7. Halpin T.: Object-Role Modeling: an overview, 2001, <http://www.orm.net/pdf/ORMwhitePaper.pdf>
8. 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
9. Semantics of Business Vocabulary and Business Rules, <http://www.omg.org/spec/SBVR/1.0/>
10. 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.
11. Visual Rules, <http://www.visual-rules.com/business-rules-management-software-rules-engine.html>
12. Drools Guvnor Rules Authoring, <http://docs.jboss.org/drools/release/5.4.0.Final/drools-guvnor-docs/html/ch04.html>
13. VisiRule, <http://www.lpa.co.uk/vsr.htm>
14. OntoStudio Graphical Rule Editor, [http://ontorule-project.eu/showcase/OntoStudio\\_Graphical\\_Rule\\_Editor](http://ontorule-project.eu/showcase/OntoStudio_Graphical_Rule_Editor)
15. Michael Kifer, Georg Lausen, and James Wu. Logical foundations of object oriented and frame-based languages. *J. ACM*, 42(4):741–843, 1995.
16. Bak J., Jedrzejek C., Falkowski M.: Usage of the Jess Engine, Rules and Ontology to Query a Relational Database. In *Proceedings of the 2009 International Symposium on Rule Interchange and Applications (RuleML '09)*, Guido Governatori, John Hall, and Adrian Paschke (Eds.). Springer-Verlag, Berlin, Heidelberg, 216-230.
17. Demo site: <http://draco.kari.put.poznan.pl/ruleml2012/>
18. Jedrzejek C., Bak J., Falkowski M., Cybulka J., Nowak M., On the Detection and Analysis of VAT Carousel Crime, in: *Frontiers in Artificial Intelligence Applications*, vol. 235, *Proceedings of JURIX 2011: The Twenty-Fourth Annual Conference Legal Knowledge and Information Systems*, pp. 130 – 134, IOS Press, 2011
19. Nowak M., Jedrzejek C., Bak J., Szwabe A., A rule-based expert system for building evidence in VAT-carousel, *Proceedings MISSI'12, Multimedia and Internet Systems: New Solutions*, in print.

# RuleTheWeb!: Rule-based Adaptive User Experience <sup>★</sup>

Adrian Giurca<sup>1,2</sup>, Matthias Tylkowski<sup>2</sup> and Martin Müller<sup>2</sup>

<sup>1</sup> Dept. of Databases and Information Technology,

<sup>2</sup> Binary Park

Brandenburg University of Technology,  
P.O. 101344, 03013 Cottbus, Germany

**Abstract.** During the last years the business rules industry proliferated as rules were recognized as a practical tool for solving real-world problems. Nowadays, many research communities develop rule languages and rule systems as well as rule markup languages and interoperability tools. However, due to the necessary high level of knowledge and complexity of tools, rules are yet designed only inside of narrow and high-skilled communities. After more than a decade of research on Semantic Web, and after initiatives of the main industry players, the Web is fast evolving into a world of objects as content creators started enriching their pages with semantic annotations. This paper presents an application using simple rules to enrich the user navigation experience on the web. We show a demo of adaptive user experience based on semantic data and reaction rules aiming to enable social rules designed and shared by web users.

## 1 Background

Five years ago, in a blog posting [12], Ora Lassila was pointing out that Semantic Web may not be only about data but also there is significant work to do with respect of *"systems that work on users' behalf"*:

For a long time (longer than I have worked on the Semantic Web) I have wanted to build *systems that work on users' behalf*. Semantic Web is one of the enabling technologies, a means to an end, and not the end itself. Every time I look critically at the current use of (information) technology, I cannot help but wonder how it is possible to actually get away with the approach taken today (where substantial burden is placed on the users).

The Semantic Web community developed an amazing set of knowledge representation languages such as Resource Description Framework (RDF) (See [23] for a hub of resources), and Web Ontology Language (OWL)[22], query languages such as SPARQL [26], and thousand of well established tools. However,

---

<sup>★</sup> This research is supported by ESF-EXIST grant No. 03EGSBB066, CatchThis: Conjoint-Analyse in Sozialen Netzwerken

most of the applications were centered on creating and querying Linked Data, i.e., to connect related data that wasn't previously linked using URIs and RDF. There is a little publicly available work with respect of building Semantic Web applications which use business intelligence to connect various web documents according with user preferences.

## 1.1 Application Vocabularies

Ontology experts developed a large amount of web vocabularies such as FOAF [18], DOAP [17] GoodRelations [19] just to mention some of them. There are many projects aiming to process large amount of semantic data (big data projects). Recently, initiatives such as Web Data Commons<sup>3</sup> published extracted semantic data from several billion web pages<sup>4</sup>.

However, one of the main difficulties to use this data comes from the large number of vocabularies that are involved, as SPARQL queries must be aware of vocabularies. Along with the Facebook Open graph Initiative <https://developers.facebook.com/docs/opengraph/>, in June 2011, Google, Bing and Yahoo! launched a common initiative, <http://schema.org> towards *a unique web vocabulary* to be used in semantic annotations:

A shared markup vocabulary makes easier for webmasters to decide on a markup schema and get the maximum benefit for their efforts. So, in the spirit of sitemaps.org, search engines have come together to provide a shared collection of schemas that webmasters can use.

Initiatives such as <http://getSchema.org> already report large amount of web sites using this vocabulary. We expect that, due to the increasing revenues of the content creators when using Schema.org annotations, this vocabulary will spread very fast on the Web content. Therefore our application focuses on this vocabulary although only little change would be needed to support other vocabularies too. Recently Microsoft and others announced submission to standardization of the Open Data Protocol <http://www.odata.org/>.

## 1.2 Business Rules

Some of our previous work (see [6]) reported on rule-based processing of semantic data annotations of HTML pages by considering annotation languages such as RDFaLite [15]. We emphasized that using rules one can significantly enrich the user interaction experiences on the Web. In addition, by offering new information in ways not originally planned, such application contributes to creation of linked data too. Specifically business rules can be successful involved when it comes to capture user's interaction with web pages towards running various business processes such as:

---

<sup>3</sup> <http://webdatacommons.org/>

<sup>4</sup> Notice that the extracted data does not come in standard RDF as they use an RDF triple extension, N-Quads. Basically they augment the RDF triple with another component which is the URI from where the triple was extracted



- Developing groups of navigational items that are meaningful to users. This includes the development of the most sensible set of navigational menu items, e.g., *Whenever the user clicks more than 3 times a menu item add this item to the fast access menu items.*
- Giving concepts from a vocabulary (Schema.org) on a page, show related linked data, e.g. *If the user loads financial news, then offer him a three months subscription to Financial Times.*
- Showing concepts visually, e.g. *When the user loads specific news about weather forecast, then deliver him a map of related weather events at the location.*
- Allow user to add calendar events related to visited pages, e.g. *If the user loads a conference web site then ask him to add event to calendar and show him travel opportunities.*
- Allow user to annotate the page, e.g., *If the page is loaded more than 5 minutes then on close ask user to annotate the page with tags/ratings.*

It is easily to see that such kind of rules may also involve events that occurs in the web browser, therefore RuleTheWeb! application uses both production rules and reaction rules. The readers may notice that when the rules are based on a unique vocabulary they can be far *fast shared between various actors*.

## 2 The Challenge

Enriching user navigation experience is not a novel paradigm. Web publishers can use various available tools such as Outbrain<sup>5</sup> or Linkwithin<sup>6</sup>, just to mention some, to embed related content in their web pages. However, this experience is related to the publishers and not the readers of the web content. These application **do not include user preferences** as they embed related content only by processing the web site content and, as they are commercial products, there is no information on the models and the technology they use to create related data. However there is also research work on similar web pages mostly featuring machine learning concepts and using similarity measures (metric-based, feature-based, or probabilistic). By contrary, RuleTheWeb! employs user preferences, Semantic Web annotations and behavioral targeting to create the best related content towards a semantic navigation on the web. In addition, because the semantic annotations are extracted on the fly and rules are always up to date there is no inconsistency between cached data (such as existent crawled summaries or raw data on the server side of other solutions) and the actual status. When content creators update their web sites and the user visit them, of course, RuleTheWeb! delivers **immediate and up to date related content**. RuleTheWeb! is enriching the reader experience by considering the semantic of the visited page and user's own preferences encoded as rules.

<sup>5</sup> <https://www.outbrain.com/>

<sup>6</sup> <http://www.linkwithin.com/>

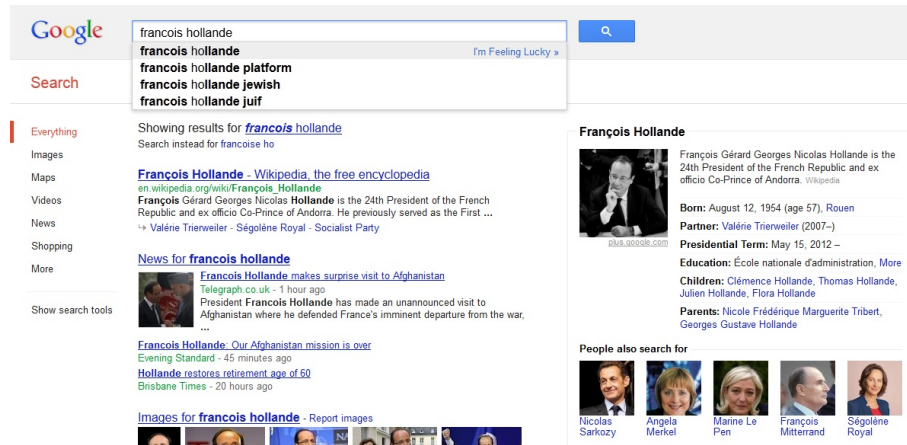


Fig. 1. Google Enriching Search Results

Big players such as Google already come up with enriching the search related content as depicted in Figure 1. This results may be related to Schema.org initiative or may not.

### 3 RuleTheWeb! - The Application

RuleTheWeb! is related to the W3C use cases of Linked Data Incubator [21] basically to the generic case of social recommendations<sup>7</sup> allowing users to benefit on the linked data recommendations with respect of the web sites they visit and the activities they perform. The related data to be offered is real-time computed by the application.

The actual implemented demo scenario included the second use case described in Section 1.2 i.e., when the current loaded page contains specific Schema.org product annotations<sup>8</sup>, the application will suggest related product offers and related reviews, from various service providers.

Basically, when a user navigates the web using a browser employing RuleTheWeb!, they will receive recommendations as soon as the page information matches one of their rulesets. The rulesets are automatically loaded from the rule repository and the user is able to choose between various rulesets. The application uses two main categories of rulesets:

1. Rule-based user preferences. Rules are computed on top of userpreferences via logic-based conjoint analysis, [25], [7], [8], [9].
2. Social Web Rules users can create/generate/share rules. Social Web Rules forms an application field of social rules theory [4] being a basic form of

<sup>7</sup> [http://www.w3.org/2005/Incubator/lld/wiki/Use\\_Case\\_Social\\_Recommendations](http://www.w3.org/2005/Incubator/lld/wiki/Use_Case_Social_Recommendations)

<sup>8</sup> See <http://schema.org/Product> and <http://getschema.org> for more examples

human interaction. Users are always motivated to (1) use publicly available rules meeting their goals - public rules are powerful because we tend to believe our friends before believing a marketing message from a brand. (2) create their own private rules and (3) share rules with the community. People like to share because (a) it brings valuable and entertaining content to others; (b) is a way of self definition; (c) is a source of growing their relationships in the community. A work in progress is a rulestore API allowing consumers to manage web rules.

### 3.1 The Rule Language

The rule repository stores RuleML, [24] rules while the rules in the secondary storage are JSON rules [5]. Developed in 2012, JSON rules version 2, uses document object model (DOM) event types [14] as underlining events vocabulary and a condition language build on the HTML5 DOM Core [10]. This version features five types of conditions:

1. *JavaScript Boolean conditions* - to capture any experience that can be induced by running JavaScript code in the browser as rule condition. For example, `document.getElementById('id').value=="container"` is a JavaScript Boolean condition evaluating `true` if the current document has an element with `id="container"`.
2. *Descriptions* - to offer a simple format to express conditions with respect of the current document structure. For example the description:

```
{
  "type": "input",
  "context": "$E",
  "constraints": [
    {
      "propertyName" : "id",
      "operator" : "EQ",
      "restriction" : { "type": "String", "value" : "
        postalCode" }
    },
    {
      "propertyName" : "nodeValue",
      "operator" : "MATCH",
      "restriction" : { "type": "Regex", "value" : "/^\d
        {5}$/" }
    },
    {
      "bind" : "$V",
      "propertyName" : "nodeValue"
    }
  ]
}
```

will bound variable `$E` to the specific `input` element, if such element exists and its value encodes a postal code following a specific structure described by a regular expression (its value is bound to variable `$V`). The language keywords include names such as `tagName`, `nodeValue`, `id`, `class`, `about`, `property`, `vocab`, `typeof`, `itemscope`, `itemtype`, `itemprop` to address the corresponding DOM and HTML5 (including RDFa 1.1 Lite and Microdata) attributes.

3. *XPath conditions* - to offer fast access to any content of the current document. For example, `$X in html/table[1]/tr[2]` will bound the variable `$X` to a collection of table data, the second row of the first table in the current document.
4. *Equality*. The traditional equality between two logical terms.
5. *Built-in predicates*. Built-in predicates do not follow any specific schema, they are simple Boolean JavaScript expressions. Failure to evaluate such a JavaScript expression is interpreted as logical `false`.

JSON Rules actions are close to JavaScript function calls as such there is very much freedom on implementing both state change actions and environment change actions. This solution covers the RIF Production Rule Dialect [13] standard actions too:

- State change actions:
  1. We experience an *assert fact action* when create a new element/attribute
  2. A *retract fact action* when we delete a DOM element/attribute
  3. A *retract all slot values* when we delete specified all attributes of a DOM Element
  4. A *retract object* action when deleting an attribute of a DOM Element
- Environment change actions:
  1. An *execute* action when we run JavaScript code.

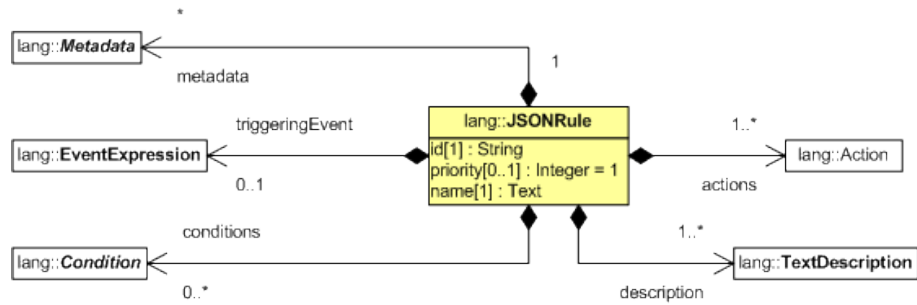
The reader may notice that while RuleML is a large family of rule languages allowing rules to be defined in top of any vocabulary, JSON rules are defined using a specific vocabulary based on Schema.org, the Facebook Open Graph and the Document Object Model (DOM). As DOM is an universal specification for all web pages the main benefit is that such rules can be immediately shared between users. However, JSON-Rules does not aim to offer standard actions (allowing for any JavaScript function call) and its actual implementation sticks to only a set of predefined possible actions.

This way we keep close to the approach of RDF rules [2] as well on some principles of publishing rules online [3]. The JSON rule model is depicted by Figure 2.

### 3.2 A Simple Scenario

When users like to enrich their experience on visiting web sites discussing movies they can use RuleTheWeb! and load a specific ruleset from the rules repository (either their private ruleset or a public ruleset). For example, such a ruleset may contain rules implementing cases like below:

Whene the visited page contains Schema.org Movie annotations, then



**Fig. 2.** RuleTheWeb: The Core Data Model

1. show related movies by the same genre.
2. show related films with the same genre and created in the same year.
3. show a trailer of the movie and background information. Also offer the sound-track, the latest news about the movie as well as statistical information.
4. show film suggestions from the same director.
5. show related film directed by the same director, in the same year.
6. show background information about the producer.
7. show background information about the music creator and offer related music composed by the same person.

The rule repository returns JSON Rules. The Example 1 shows a possible rule describing a part of the above scenario:

*Example 1 (A JSON Rule).*

```

{
  "id": 3,
  conditions: [
    {
      "type": "Element",
      "context": "$T",
      "constraints": [
        {
          "propertyName": "itemscope",
          "operator": "NEQ",
          "restriction": { "type": "String", "value": "null" }
        },
        {
          "propertyName": "itemtype",
          "operator": "EQ",
          "restriction": { "type": "URL", "value": "http://
                        schema.org/Movie" }
        }
      ]
    }
  ]
}

```

```

    "type": "Element",
    "context": "$_",
    "constraints": [
    {
        "propertyName" : "itemprop",
        "operator" : "EQ",
        "restriction" : { "type": "String", "value" : "name"}
    },
    {
        "propertyName" : "parentNode",
        "operator" : "EQ",
        "restriction" : "$T"
    },
    {
        "bind" : "$Y",
        "propertyName" : "nodeValue"
    }
    ],
    "actions": [
        "invoke('youtube', $Y + ' trailer')",
        "invoke('imdb', $Y )",
        "invoke('amazon', $Y + ' soundtrack')",
        "invoke('googlenews', $Y)",
        "invoke('wolframalpha', $Y)"
    ]
}

```

Action **invoke** is an environment change action (included in the standard execute action of RIF-PRD). The conditions of the rule are related to the DOM content and, as usual, must be satisfied to execute the intended actions.

The actions are invoked sequentially but the final result, including possible state change i.e. changes into the current DOM (the working memory), will take place at the end of all action calls and the environmental effect may be a *sum* of all actions to be executed.

When elements annotated with `http://schema.org/Movie` (bound to `$T`) have a child annotated with the property `name` (`$Y` is bound to the text node "Francis Ford Coppola") then the conditions are satisfied. For example when the DOM contains the below fragment all rule conditions are satisfied:

```

...
<div itemscope itemtype="http://schema.org/Movie">
...
  <span itemprop="name">Francis Ford Coppola</span>
...
</div>
...

```

While the above example uses HTML5 Microdata annotations [16], JSON-Rules language also supports RDFa 1.1. Lite annotations [15]. This is quite

simple, as RDFa 1.1. lite is very close to Microdata, basically by using `typeof` instead of `itemtype` and `property` instead of `itemprop`.

### 3.3 On Rule Execution

RuleTheWeb! does not employ a rule engine. Rather, the rules are directly compiled to executable JavaScript code. Also, the way the actions are executed is part of the compiling technology which is in development.

### 3.4 How to use it

To install the application please visit <https://ruletheweb.org>. After a successful installation one must see an explanation page offering a brief introduction on how to use the extension.

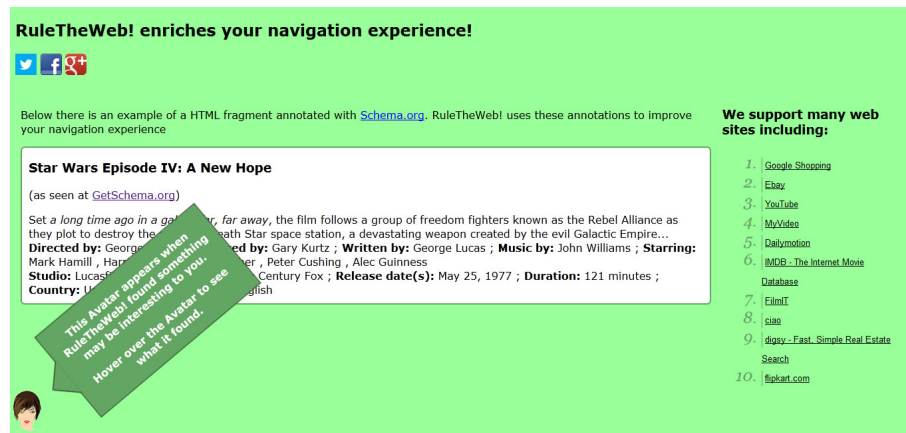
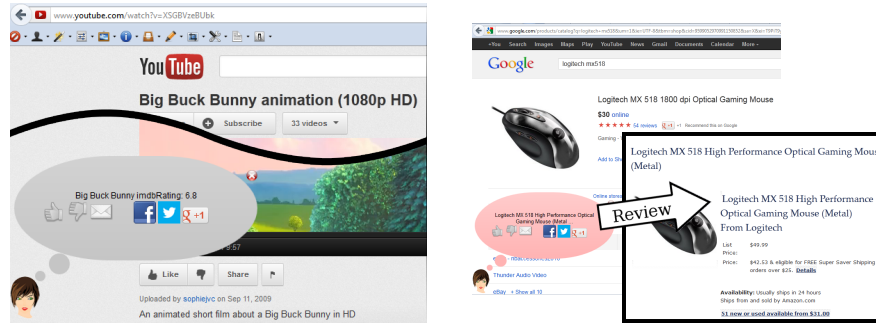


Fig. 3. RuleTheWeb! introduction page

The Figure 4 illustrates the application by showing activities on web sites such as Google Shopping and YouTube. When watching a movie or a trailer on YouTube one can receive additional information of that movie from imdb.com. When navigating on Google Shopping and search for a desired product, RuleTheWeb! offers additional information like reviews or specifications.

### 3.5 Architecture

The application architecture is depicted by Figure 5. All rulesets are loaded by the application from a rule repository based on RuleML serializations.



**Fig. 4.** Related news on YouTube pages and helping on buying products

**The Client** The client components are depicted by Figure 5.

*Browser current window.* The current DOM, containing semantic annotations, acts as a facts provider: the semantic data is extracted and these are the facts to be matched with rule conditions.

*RuleTheWeb.* It compiles rules to JavaScript code and execute them under usual conditions. The rule execution result is sent to the Service Layer towards executing the actions. The execution result is used by the Formatter module to create the desired presentation.

*Secondary Storage.* The secondary storage combines two different kinds of rules: Shared rules from the rule repository and private rules that the user created himself. The user can decide to upload his custom created rules to the rule repository to publicly share them with other users.

**The server side** The server side has two main components: (a) A rule repository infrastructure with the main role of serving user-defined rulesets and (b) a service processor with the main role to process rule actions.

### 3.6 Notes on Implementation

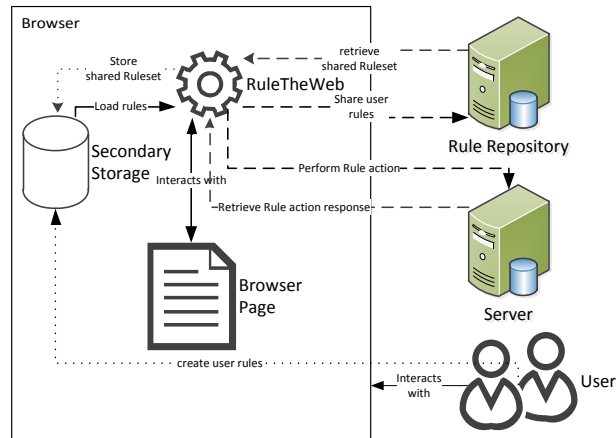
RuleTheWeb! is implemented as a standard client-server application invoking a server service from a web based application (the client) under the same specific session. A sequence diagram, depicted by Figure 6, describes the basic execution process. The client-server communication is Ajax based.

The **Ruleset** object is serialized to the browser secondary storage. As usual, a ruleset is a collection of rules designed to fulfill a goal.

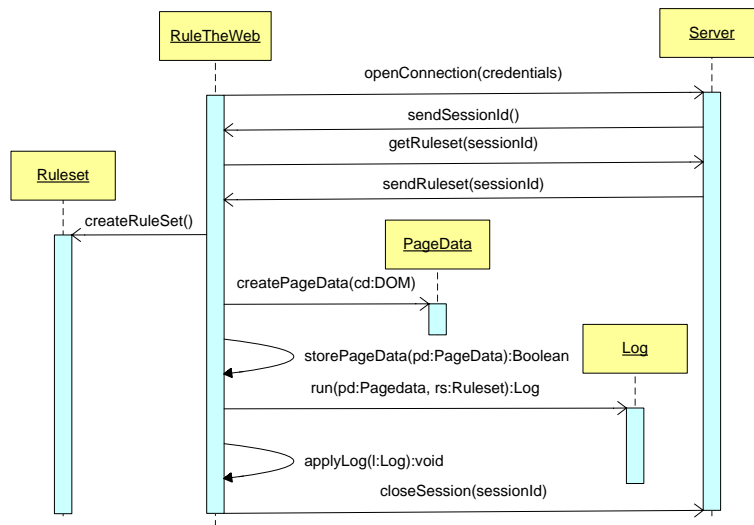
The **PageData** object implements the logic of rules working memory, i.e. it manages the facts on which the rules are matched.

The **Log** object implements the logic of actions to be executed. Basically, when a ruleset is active, the Log object stores all actions effects to be performed. Notice that the actions will not be executed immediately when they are fired by some rule but at the end of the entire ruleset execution. Therefore this object will also deal with the order of action effects.





**Fig. 5.** RuleTheWeb! Architecture



**Fig. 6.** RuleTheWeb: Basic RuleSet Execution

The RuleTheWeb! Firefox demo uses the storage only available for extensions<sup>9</sup>. In addition, there are two other kinds of storages that the application is using: (a) The session storage<sup>10</sup>, used to store the state of the application (disabled or enabled). This storage remains valid over the browser session and settings are restored when the browser is restarted, and (b) the DOM Storage<sup>11</sup>, persistent as long as the actor stays on the same page.

## 4 Conclusion and Future Work

This demo as a proof-of-concept gave insight how rules can be used together with semantic annotated content on web pages to enrich the user web surfing experience. There is an ongoing work to define a complete data model of capturing user preferences by investigating the capabilities of data collection offered by modern communication tools such as online media and social networking. Such model aims to capture most of widely accepted preference properties with respect to behavioral economics concepts [11] such as heuristic, i.e. consumers make decisions based on approximate rules and not strict logic (see also [20] for an interesting use case).

Because this application uses rulesets created by third parties according with the user profile they store, future work will offer users to change and store their own rules. While writing rules typically requires professional expertise, our goal is to allow users to write simple rules while experts may contribute to complex rules as well as to rule curation.

In addition, for the content creators, the application will be offered as a standalone JavaScript application to be added to the web pages whereas a browser extension with respect of these pages will no longer be necessary.

## Acknowledgements

We would like to gratefully acknowledge Prof. Daniel Baier, head of department of Marketing and Innovation and Prof. Ingo Schmitt, head of department of Databases and Information Technology for their useful insights.

## References

1. Y. Aytar, M. Shah, J. Luo. Utilizing Semantic Word Similarity Measures for Video Retrieval, IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2008), 24-26 June 2008, Anchorage, Alaska, USA.
2. T. Berners-Lee, D. Connolly, E. Prud'homeaux, Y. Scharf. Experience with N3 rules, W3C Rules language Workshop April 2005, <http://www.w3.org/2004/12/rules-ws/paper/94/>.

<sup>9</sup> <https://developer.mozilla.org/en/Storage>

<sup>10</sup> [https://developer.mozilla.org/en/Session\\_store\\_API](https://developer.mozilla.org/en/Session_store_API)

<sup>11</sup> <https://developer.mozilla.org/en/DOM/Storage>

3. H. Boley: Are Your Rules Online? Four Web Rule Essentials. Advances in Rule Interchange and Applications, International Symposium, RuleML 2007, Orlando, Florida, October 25-26, 2007, pp. 7-24, <http://www.cs.unb.ca/~boley/papers/RuleEssentials.pdf>.
4. T.R. Burns, and T. Dietz (1992) Cultural Evolution: Social Rule Systems, Selection, and Human Agency. International Sociology 7:250-283.
5. A. Giurca and E. Pascalau (2008). JSON Rules. In G. J. Nalepa and J. Baumeister (Eds.) Proceedings of 4th Knowledge Engineering and Software Engineering, KESE 2008, collocated with KI 2008, Spetember 23, 2008, Kaiserlautern, Germany, CEUR Workshop Proceedings Vol 425.
6. A. Giurca, E. Pascalau (2009). Building Intelligent Mashups. Tutorial at 32nd Annual Conference on Artificial Intelligence (KI 2009), September 15-18, 2009, Paderborn, Germany, [https://docs.google.com/View?id=dcff8ncf\\_181gxb3ss65](https://docs.google.com/View?id=dcff8ncf_181gxb3ss65).
7. A. Giurca, I. Schmitt, and D. Baier. Performing Conjoint Analysis within a Logic-based Framework. Proc of IEEE Federated Conference on Computer Science and Information Systems, (FedCSIS2011), Szczecin, Poland, 18-21 September, 2011.
8. A. Giurca, I. Schmitt, and D. Baier. Can Adaptive Conjoint Analysis perform in a Preference Logic Framework? The 8th workshop on Knowledge Engineering and Software Engineering (KESE8) at the ECAI 2012 Montpellier, France, August 27-31, 2012.
9. A. Giurca, I. Schmitt, and D. Baier. Adaptive Conjoint Analysis. Training Data: Knowledge or Beliefs? A Logical Perspective of Preferences as Beliefs. Proc of IEEE Federated Conference on Computer Science and Information Systems, (FedCSIS2012), Wroclaw, Poland, 9 - 12 September, 2012.
10. A. Le Hors, P. Le Hgaret, L. Wood, G. Nicol, J. Robie, M. Champion, S. Byrne. Document Object Model (DOM) Level 3 Core Specification, Version 1.0, W3C Recommendation 07 April 2004, <http://www.w3.org/TR/DOM-Level-3-Core/>.
11. D. Kahneman, and A. Tversky (1979). Prospect theory: An analysis of decisions under risk. Econometrica 47 (2): 263-291.
12. O. Lassila. Semantic Web Soul Searching, Blog posting , March 19, 2007. [http://www.lassila.org/blog/archive/2007/03/semantic\\_web\\_so\\_1.html](http://www.lassila.org/blog/archive/2007/03/semantic_web_so_1.html) last retrieved: June 10, 2012.
13. C. de Sainte Marie, G. Hallmark, A. Paschke. RIF Production Rule Dialect, W3C Recommendation 22 June 2010, <http://www.w3.org/TR/rif-prd/>.
14. D. Schepers, J. Rossi, B. Höhrmann, P. Le Hgaret, T. Pixley. Document Object Model (DOM) Level 3 Events Specification, W3C Working Draft 31 May 2011, <http://www.w3.org/TR/DOM-Level-3-Events/>.
15. Manu Sporny. RDFa 1.1. Lite, W3C Candidate Recommendation, <http://www.w3.org/TR/rdfa-lite/>.
16. \* \* \* HTML5. A vocabulary and associated APIs for HTML and XHTML, <http://www.whatwg.org/specs/web-apps/current-work/multipage/microdata.html>. last retrieved: June 20, 2012.
17. \* \* \* DOAP: Description of a Project, <https://github.com/edumbill/doap/wiki> last retrieved: June 10, 2012.
18. \* \* \* Friend Of A Friend, <http://semanticweb.org/wiki/FOAF>, last retrieved: June 20, 2012.
19. Good Relations: a Web vocabulary for e-commerce, <http://www.goodrelations-vocabulary.org/>, last retrieved: June 20, 2012.
20. L. Lee, S. Frederick and D. Ariely (2006), Try It, Youll Like It: The Influence of Expectation, Consumption, and Revelation on Preferences for Beer. Psychological Science. Vol. 17, No. 12: 10541058.

21. Library Linked Data Incubator Group: Use Cases. <http://www.w3.org/2005/Incubator/lld/wiki/UseCases>, last retrieved: June 20, 2012.
22. \* \* \* Ontology Web Language, W3C, <http://www.w3.org/OWL/>, last retrieved: June 10, 2012.
23. \* \* \* Resource Description Framework, W3C, <http://www.w3.org/RDF/>, last retrieved: June 10, 2012.
24. \* \* \* The RuleML Initiative, <http://ruleml.org>, last retrieved: June 10, 2012.
25. I. Schmitt, and D. Baier. Logic Based Conjoint Analysis using the Commuting Quantum Query Language, Proc. of Conference of the German Classification Society (GfKI2011), August 31 to September 2, 2011, Frankfurt am Main, Germany.
26. \* \* \* SPARQL 1.1 Query Language, W3C Working Draft 05 January 2012, <http://www.w3.org/TR/sparql11-query/>, last retrieved: June 15, 2012.

# PLIS+: A Rule-Based Personalized Location Information System

Iosif Viktoratos<sup>1</sup>, Athanasios Tsadiras<sup>1</sup>, Nick Bassiliades<sup>2</sup>,

<sup>1</sup>Department of Economics, <sup>2</sup>Department of Informatics,  
Aristotle University of Thessaloniki  
GR-54124 Thessaloniki, Greece

{viktorat, tsadiras, nbassili}@auth.gr

**Abstract.** In this paper, the idea of providing personalized, location-based information services via rule-based policies is demonstrated. After a short introduction, an innovative Personalized Location Information System (PLIS+) is designed and implemented. PLIS+ delivers personalized and contextualized information to users according to rule-based policies. More specifically, many categories of points of interest (for example shops, restaurants) have rule-based policies to expose and deploy their marketing strategy on special offers, discounts, etc. PLIS+ evaluates these rules on-the-fly and delivers personalized information according to the user's context and the corresponding rules fired within this context. After discussing the design and the implementation of PLIS+, illustrative examples of PLIS+ functionality are presented. As a result, PLIS+ proves that combining contextual data and rules can lead to powerful personalized information services.

**Keywords:** RuleML, Rules, Location Based Services, Context, Points of Interest, Jess.

## 1 Introduction

### 1.1 Rule-based Information Services and related work

Latest information services adopt rule based approaches so as to enable higher quality context perception. Rule-based systems are more autonomous because they are capable of understanding context changes and responding accordingly without user intervention [1].

As a result, up-to-date Location Based Services (LBS) combine semantics (ontologies, rules) with smartphone's capabilities (GPS, sensors) to deliver contextualized up-to-date information [2-5] to users. Thus, LBS have become a popular sector of everyday life and they are used consistently by millions of people for navigation, tracking, information, even in emergency situations [6].

## 1.2 Motivation-Overview

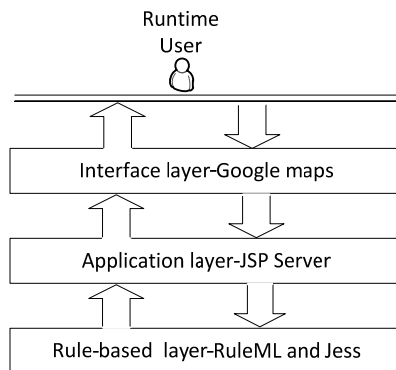
The aim of the presented work is to combine semantics with location information services to deliver personalized and contextualized information services to users. A system called “Personalized Location Information System” or PLIS+ was implemented for this purpose. PLIS+ is an extended version of the PLIS system that is presented in [7]. It can be accessed at <http://tinyurl.com/ca42fwj>

A rule-based approach was followed for PLIS+ implementation, based on discussion in section 1.1 Core component of PLIS+ is RuleML, a powerful markup language (XML with a predefined Schema) which supports various types of rules such as deductive, reactive and normative. As an XML-based language, RuleML addresses the issues of interoperability and flexibility among different systems on the web, by allowing rules to be encoded in a standard way [8]. Last but not least, because of the fact that PLIS+ users are capable of adding rules at run-time, an xml-based user friendly language is desirable.

PLIS+ could easily be combined with most of existing approaches and differs by enabling a dynamic rule base that offers users the option to add rules at run time. A detailed demonstration of the system is included in the following sections.

## 2 Design, Implementation and technical details

In everyday life, in order to deploy their specific business strategy, many points of interest (as businesses) adopt a rule-based policy (e.g. special offers). The general idea is to combine POI’s policies with user’s context to deliver ‘qualitative’ information. A general interface for connection between POI owners and potential customers-users is provided. Every time a user logs into the system to search for a point of interest, PLIS+ gets user’s context (profile, location, day, time), evaluates the rules associated with nearby POIs and delivers personalized information to user, depending on the rules fired. Users can also become owners of various POIs and after that they are capable of inserting their own rule base policy for those POIs. The general idea is illustrated in Figure 1.



**Fig. 1** General design of system

To be more specific, PLIS+ offers the following functionalities illustrate in Figure 2:

**A: User's Registration:**

- A1. User registers to the system by completing a registration form so as PLIS+ to build a profile (registration time user).
- A2. User profile data such as first name, last name, occupation, gender, age, city, state, etc are stored in the database.

**B: Insertion of Points of Interest:**

- B0. After a user has logged in, the system obtains user position and retrieves nearest POI's from external sources such as Google Places API. If information of a POI is already in the system, PLIS+ updates its related data with the latest information.
- B1. User is capable (by becoming POI owner) of attaching to existing places attributes and explicit rules relevant to those attributes. In detail, user is able to assert a rule base which contains any attribute related to his POIs and then assert rules concerning these attributes. For example a restaurant owner asserts a policy which contains data (related to his business) such as Pizza 10€, Spaghetti 8€, Minimum order 5€ and along with above data, relevant rules such as "*if a person is a student and hour is after 18:00, then Pizza price is 8€*". Furthermore, a user is of course capable of inserting his/her own POIs accompanied by their own rule based policy. Alternatively, place owners can upload their rule base to their website and insert the relevant link. The editing of the corresponding rule base is authorized only to the POI's owner.
- B2. All data and rules concerning the POI are saved to the corresponding database. Except from this, files containing rule bases are kept to the server.

**C: Presentation of Personalized information.** To present the personalized information to the end user, the following steps are made:

- *Step C1:*
  - a. After registration, user is able to log into the system by entering his/her username and password.
  - b. System checks user profile database for authentication.
- *Step C2:* Java Server Pages (JSP) collects user context (profile, location, time, day) attribute values (run time user).
- *Step C3:*
  - a. For every POI, rules (if any) are being fetched (by JSP), along with relevant attribute values (for example price, etc).
  - b. Rules (after being transformed to a machine understandable language), POI data and user context attribute values are asserted to the Jess rule engine.
- *Step C4:* Jess rule engine evaluates rules using the asserted facts and updates POIs' data according to the rules fired depending on user's context. The new data are fetched by JSP.
- *Step C5:* Finally, data transfer to client is performed for visualization and presentation of personalized information. It's worth mentioning that a user-oriented interface has been implemented so as the run-time user to become capable of understanding the general idea of PLIS+. First of all, different markers are applied for better illustration. In detail, except from the standard red marker for POIs, a) a yellow marker

indicates that the place contains a rule base but no rule fired for current user, b) a green marker indicates that the place contains a rule base and rules were fired for the current user, c) a crown over the marker indicates that the current user is also the POI owner of this place. Moreover, when a person clicks on a place additional information appears in a message explaining which rules were fired and why or in the case that no rule was fired for the specific POI, which rules exist for the place (if any).

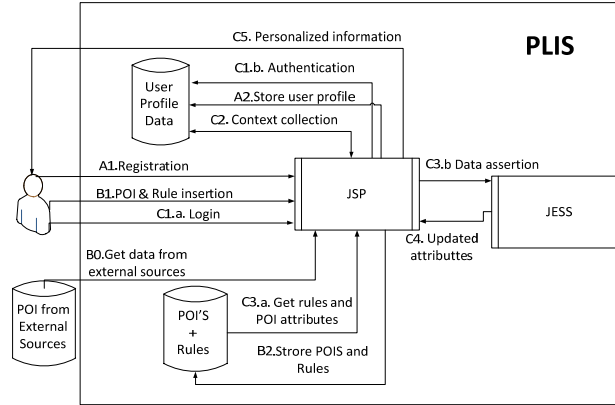
Basic component of PLIS+ is Reaction RuleML, a subcategory of RuleML. It is used for rule representation. This subcategory was chosen because such kinds of policies are usually represented by production rules and Reaction RuleML is suitable for that task [9]. Reaction RuleML also supports both deductive rules, i.e. rules which derive data at their RHSs, and event-based rules, i.e. rules which are activated by specific events, such as user actions or temporal events. Therefore, Reaction RuleML could be easily used to express a multitude of rule-based calculations and data processing for business strategies.

On the other hand, to implement a system like PLIS+, an inference engine is needed in order rules to be executed by a machine. Jess was chosen to implement the core of PLIS+, because of the fact that it is a lightweight rule engine and connects well with web technologies, which were needed for PLIS+ system implementation [10]. So for example, after the translation from RuleML to Jess, the rulebase for the POI with the following characteristics: a) pizza: 10€, b) spaghetti: 8€, c) minimum order: 5€ and d) a rule considering “pizza price decreased to 8€ after 18:00 for students” is represented as shown below:

```
(bind ?fact (assert (place (pizza 10) (Spaghetti 8)
(minimum_order 5))))
(defrule decreased_pizza_price
  (declare (salience 10))
  (person (hour ?d) (occupation student))(test (> ?d 18))
=>(modify ?fact (pizza 8) )
(store EXPLANATION "Pizza price decreased to 8€ after 18:00 for students" ) )
```

Rules in RuleML format transformed to Jess rules by using XSLT [11]. XSL technology is used massively to transform XML documents. Another core technology of PLIS+ is Java Server Pages (JSP). The vast majority of rule engines such as Jess, Drools, Prova [12] are not only server-oriented (for security issues) but also java-based. JSP implementation fulfills both criteria. Moreover, JSP can be easily embedded into html documents and is heavily used along with client-oriented technologies such as JavaScript for visualization.





**Fig. 2** PLIS+ functionalities

### 3 Demonstration of PLIS+

This section includes a demonstration of the PLIS+ system. A random user profile snapshot is used as an example (Table 1). On the other hand two random places from the database selected for testing (Table 2). Table 2 shows the attributes and also the rules that were attached to these places.

**Table 1.** A random user profile.

Profile					Environment		
	Name	Occupation	Gender	Age	Time	Day	Location
User A	Susan	Student	Female	26	20:12	Friday	LocationA

**Table 2.** Two random places.

	Name	Asserted Data	Rule 1	Rule 2
Place A	Cafe Delmundo	Coffee:3€ Ice-cream: 5€	Coffee price reduced to 2€ for students which are closer than 200m after 22:00	Half prices for coffee and ice-cream (1.5 and 2.5€) for women on Fridays after 18:00 o'clock
Place B	Verona pizza	pizza:10€ fish:12€ minimum order:5€	40% discount (6€) into pizza price for students	special prices for pizza (8€) and a decreased minimum order (4€) for women under 35 years old which are closer than 500m

As it was previously referred, PLIS+ gets user profile, evaluates rules and displays personalized information. When user A ("Susan") logs into the system she is able to click on places displayed by green markers (places containing rules that fired according to current user profile), so as to understand which rules fired for her and why

(Figure 3). Taking Place A (Table 2) as an example, rule 2 is fired (because she is a woman, the day is Friday and current time is after 18:00). PLIS+ updates attribute values according rule 2 and delivers contextualized information to Susan (Figure 4). Coffee and ice-cream price for Susan are 1.5 and 2.5€ after rule 2 execution. She is also capable of understanding why a rule fired with the rule explanation field.



**Fig. 3.** PLIS+ Starting screen

Similarly, both rules were fired for place B. Susan is a student (Rule 1 criterion) and she is a woman under 35, closer than 500m from place B (Rule 2 criteria, assuming that location A is closer than 500m). According to Rule 2, minimum order for current user is 4€, but there is a conflict for pizza price (pizza price is 6€ according to rule 1 and 8€ according to the second). PLIS+ handles rule confliction problems by applying priorities according to assertion turn. A Rule which was inserted first has a higher priority. Taking those under consideration pizza price for Susan is 6€ and related information displayed as in Figure 5.

This example illustrates how the delivered information is displayed to the end user and the capabilities of PLIS+ concerning a) gender, b) day, c) location, d) occupation, e) time, f) a non-applicable rule case and g) rule confliction.

#### Cafe Delmundo

Category:cafe  
coffee:1.5  
ice\_cream:2.5

Rules Explanation:Half prices for coffee and ice-cream (1.5 and 2.5 Euro) for women on Fridays after 18:00 o'clock  
[Become POI owner](#)



**Fig. 4.** Information for Susan concerning Place A.

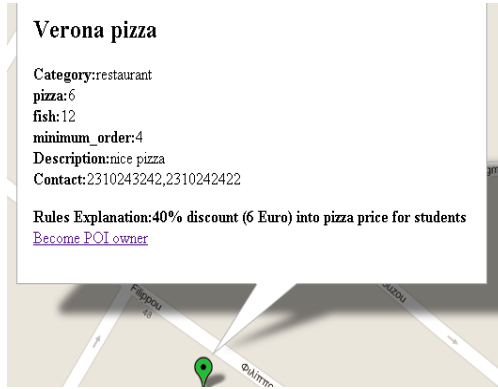


Fig. 5. Information for Susan concerning place B

#### 4 Limitations and comparison with related works

A limitation of the proposed system is the fact that users can add data and rules concerning these data only in textual form. Furthermore, no rule sharing between place owners or a rule recommendation operation is supported. Moreover, there are some security issues concerning place owners operations. For example, an authentication process validating that the user who is becoming POI owner is the actual place owner would be desirable. In addition to this, a rule validation operation would be useful.

Concerning comparison with related approaches described in section 1 PLIS+ differs by letting users adding and editing rules at run time. It is a fully dynamical service where users can add an unlimited number of data and rules, and consequently it becomes more and more intelligent and autonomous as soon as new data and rules are asserted to the system. On the other hand, many rule based approaches adopt more complex rule bases in relation to PLIS+. Last but not least, by using RDF and OWL such kind of services can be more flexible.

#### 5 Conclusions and Future Work

Location-Based Services are currently popular, enjoying both commercial and scientific interest. An important part of LBS is the points of interest, and as mentioned, most of them can have their explicit rule policies. Taking these facts under consideration, embedding rules dynamically to location-based information systems can offer a boost to the quality of delivered information. PLIS+ combines location-based technologies with rule-based technologies to demonstrate the viability of this idea.

PLIS+ implementation can evolve in various ways. First of all, because of the fact that POI owners are unfamiliar with RuleML, a user-friendlier environment has to be implemented. Either a form-based interface will be implemented, or otherwise a ruleml editor [13] will be embedded, so as place owners to become capable of adding and editing rules without much effort. Additionally, system should propose rules that

other owners have added previously and run time owners-users should be capable of choosing between existing rule sets.

Furthermore, in our future plans is to use OWL and/or RDF data (as in linked data) to represent user profiles and POI related information, for greater flexibility. Moreover, recommendation algorithms depending on the retrieved information (after the rules fired) would be useful. Additionally, future work will also focus on security and interoperability issues. Besides these, a mobile application e.g. for a smartphone, can be implemented and integrated with the native context sensing devices (e.g. GPS, compass, etc.). In addition, experimental testing of the system is in progress by making PLIS+ public.

## References

1. Lassila, O. Applying Semantic Web in Mobile and Ubiquitous Computing: Will Policy-Awareness Help? Semantic Web and Policy Workshop (2005)
2. D.Serrano, R. Hervás, J. Bravo. Telemaco: Context-aware System for Tourism Guiding based on Web 3.0 Technology. International Workshop On Contextual Computing and Ambient Intelligence in Tourism (2011)
3. N.Tryfona, D. Pfoser: Data Semantics in Location-based Services. Journal on Data Semantics III, vol. 3534, pp. 168–195 (2005)
4. Iliam V. Woensel, S. Casteleyn and O. Troyer. Applying semantic web technology in a mobile setting: the person matcher,” Web Engineering, pp. 506- 509(2010)
5. Van Woensel, W., Casteleyn, S., and Troyer, O. A framework for decentralized, context aware mobile applications using semantic web technology. In Proc. of the Confederated International Workshops and Posters On the Move to Meaningful Internet Systems. Springer, 88-97. (2009)
6. S. Steiniger, Moritz N., Alistair E.: Foundation of Location Based Services. Lecture Notes on LBS (2006)
7. Viktoratos I., Tsadiras A. and Bassiliades N. *Personalizing Location Information through Rule-Based Policies* accepted for presentation to RuleML 2012
8. Yuh-Jong Hu, Ching-Long Y., Wolfgang L.: Challenges for Rule Systems on the Web. RuleML '09 Proc. of the 2009 International Symposium on Rule Interchange and Applications (2009)
9. Paschke A., Kozlenkov A., Boley H.: A Homogenous Reaction Rule Language for Complex Event Processing, 2nd International Workshop on Event Drive Architecture and Event Processing Systems (EDA-PS 2007), Vienna, Austria, (2007)
10. Ernest Friedman-Hill: Jess in Action. Rule-Based Systems in Java. Manning Publications. ISBN-10: 1930110898, pp. 32-33(2003)
11. Gregory Sherman: A Critical Analysis of XSLT Technology for XML Transformation. Senior Technical Report. (2007)
12. S. Liang, P. Fodor, H. Wan, M. Kifer: OpenRuleBench: An Analysis of the Performance of Rule Engines. WWW 2009 Madrid(2009)
13. E. Kontopoulos, N. Bassiliades, G. Antoniou, A. Seridou: Visual Modeling of Defeasible Logic Rules with Dr-VisMo. Intern. Journal on Artificial Intelligence Tools, 17(5), pp. 903–924, 2008.