Semantically-Enhanced Authoring of Defeasible Logic Rule Bases in the Semantic Web

Efstratios Kontopoulos
School of Science & Technology,
International Hellenic University
GR-57001, Thessaloniki, Greece
Dept. of Informatics,
Aristotle University of Thessaloniki
GR-54124, Thessaloniki, Greece
+302310998418
e.kontopoulos@ihu.edu.gr

Thetida Zetta
Dept. of Informatics,
Aristotle University of Thessaloniki
GR-54124, Thessaloniki, Greece
+302310998418
School of Computer Science
& Informatics,
Cardiff University, Cardiff, UK
thzetta@gmail.com

Nick Bassiliades
Dept. of Informatics,
Aristotle University of Thessaloniki
GR-54124, Thessaloniki, Greece
+302310997913
nbassili@csd.auth.gr

ABSTRACT

The Semantic Web represents an initiative to improve the current Web, by augmenting content with semantics and encouraging cooperation among human and software agents. The development of the logic and proof layers of the Semantic Web is currently concentrating the related research effort and is vital, since these layers allow systems to infer new knowledge from existing information, assisting them in explaining their actions and, ultimately, increasing user trust towards the Semantic Web. However, there is a lack of applications that could contribute towards developing logic-based applications. Consequently, users resort to inadequate tools that offer syntactic support, without being able to support the user semantically as well. This work presents S^{2}DRREd, a software tool that introduces a supplementary level of semantic assistance during rule base development. The tool allows creating meta-models of the main notions of the loaded rule sets and assists the user in authoring rule bases, independently of the explicitly chosen rule language syntax. The domain of application is defeasible logic, a type of logic that allows reasoning with incomplete and conflicting information and, as such, it can play an increasingly important role in a drastically dynamic environment like the Web.

Categories and Subject Descriptors

H.5.2 [**User Interfaces**]: Graphical User Interfaces

General Terms

Design, Languages.

Keywords

Semantic Web, RuleML, Rule Bases, Defeasible Logic, Non-monotonic Reasoning.

1. INTRODUCTION

The *Semantic Web* [2] represents an initiative to improve the current Web, by augmenting Web content with semantics and facilitating cooperation among human and software agents. Its basic infrastructure has reached sufficient maturity and research efforts are now shifting towards the higher layers of *logic* and *proofs*. These layers are critical, since they will allow systems to infer new knowledge from existing information, assisting them in explaining their actions and, eventually, increasing user trust towards the Semantic Web. Researchers are focusing mainly on the integration of rules and ontologies (*OWL2 RL* [13]) and on rule representation standardization (*SWRL* [8], *RuleML* [4]).

Unfortunately, there is often a narrow and domain-specific variety of appropriately focused software tools that could assist towards the development of rule-based applications, like e.g. generic and adaptable rule editors. Consequently, users resort to existing applications for purposes that exceed their given array of functionality. For instance, in order to deploy a RuleML rule set, one could make use of an XML editor that indeed offers syntactic support, but there is an evident lack of parallel semantic support. This lack involves on one hand the rule structure itself, but also the data model, to which these rules refer to. And the problem becomes more apparent, when moving “upwards” to the domain model, with rules containing class and attribute names.

This paper presents *S^{2}DRREd*, a software tool for authoring defeasible theories. A supplementary level of semantic support during rule base development is added, which cooperates with established techniques for syntactic support. The application domain is *defeasible logic*, a member of the non-monotonic logics family that allows reasoning with incomplete and conflicting information [10]. Since the Web comprises a drastically dynamic environment, the role of non-monotonic logics becomes even more significant. Via its *Semantic Tag Mapping* (*STM*) functionality, the tool permits creating meta-models of the main defeasible logic theory notions, like “rule”, “rule type”, “rule head” etc., independently of the chosen rule language syntax.

2. DEFEASIBLE LOGIC

A *defeasible theory D* is a knowledge base or a program in defeasible logic that consists of three basic components: a set of *facts* (*F*), a set of *rules* (*R*) and a *superiority relationship* (>). Therefore, *D* can be represented by the triple (*F*, *R*, >).

Defeasible logic features three distinct types of rules: *Strict rules* are denoted by $A \rightarrow p$ and are interpreted in the typical sense. An example of a strict rule is: "*Novels are books*", which, written formally, would become: r_1: novel(X) $\rightarrow$ book(X). Contrary to strict rules, *defeasible rules* can be defeated by contrary evidence and are denoted by $A \Rightarrow p$. Two examples are: r_2: book(X) $\Rightarrow$ hardcover(X) ("*Books are typically hard-covered*") and r_3: novel(X) $\Rightarrow$ ¬hardcover(X) ("*Novels are typically not hard-covered*"). *Defeaters*, denoted by $A \rightsquigarrow p$, are rules that do not actively support conclusions, but can defeat defeasible conclusions, by producing evidence to the contrary. A defeater example is: r_2': cheap(X) $\rightsquigarrow$ ¬hardcover(X) ("*Cheap books are not hard-covered*"), which can defeat e.g. rule r_2 mentioned above.

The *superiority relationship* is an acyclic relation > among the rule set *R*, which is used, in order to resolve conflicts among rules. For example, defeasible rules r_2 and r_3 contradict each other, but if the superiority relationship $r_3 > r_2$ is introduced, then r_3 overrides r_2 and we can indeed conclude that the novel is not hardcover. In this case rule r_3 is called *superior* to r_2.

Defeasible reasoning encompasses additional sophisticated features, like *conflicting literals* and *negation-as-failure* (*NAF*), the description of which is omitted here, due to space limitations.

3. S^{2}DRREd

S^{2}DRREd (*Syntactic-Semantic Defeasible Reasoning Rule Editor*) is a software tool, dedicated to authoring defeasible logic theories expressed in the *DR-RuleML* syntax [1], a rule language based on Object-Oriented RuleML [3]. Figure 1 displays a DR-RuleML fragment example (v.0.91) and, more specifically, defeasible rule r_2 from the previous secton.

The primary contribution of the system is the additional level of semantic assistance offered during the development of rule bases, making the software ideal not only for managing DR-RuleML files, but any other XML-based rule representation as well.

3.1 Architecture and Functionality

Figure 2 displays the overall architecture of the system. The main module is the *Rulebase Editor*, which directly interacts with the end-user. The module is tightly integrated with the *Semantic Tag Mapping* (*STM*) module, which is described later and comprises the main means for semantic support, as well as with the other secondary modules that offer additional syntactic/semantic support during the development of defeasible logic rule bases.

The *Syntax Loader* is the module responsible for loading external schemas that define allowed vocabularies and subsequently feeds them to the parser. The latter detects all elements and attributes and temporarily stores them in memory, so that they can be utilized later on (e.g. for syntax highlighting and for the STM functionality). During detection, the system also copes with `<include>` and `<redefine>`, following the corresponding URLs and collecting the detected elements and attributes.

Besides loading custom external schemas, there is also the option of loading one of three predefined DR-RuleML schemas, v.0.89, v.0.9 and v.0.91, which are defeasible logic extensions to the respective versions of the core RuleML vocabularies. One of the imminent goals for the future is to develop a DR-RuleML vocabulary for the recently released RuleML v.1.0. The corresponding structures that contain the collected elements and attributes for the predefined schemas are stored in respective files. When loading one of the three schemas, this feature saves time for users, as they don't have to reinitiate the detection process. The application commences each time having as default schema the one that was loaded during the last time S^{2}DRREd was executed.

```
<Implies ruletype="defeasiblerule">
  <oid> <Ind uri="r2">r2</Ind> </oid>
  <head>
    <Atom>
      <op> <Rel>hardcover</Rel> </op>
      <slot>
        <Ind>name</Ind>
        <Var>x</Var>
      </slot>
    </Atom>
  </head>
  <body>
    <Atom>
      <op> <Rel>book</Rel> </op>
      <slot>
        <Ind>name</Ind>
        <Var>x</Var>
      </slot>
    </Atom>
  </body>
</Implies>
```

Figure 1. DR-RuleML fragment (v.0.91).

The main window of the program is displayed in Figure 3. The top menu bar consists of various menus. The *File* menu contains options for managing projects and rule bases. The notion of a project contains the rule base (DR-RuleML file), along with additional files that contain custom, project-specific properties and are discussed subsequently. Each project is assigned a distinct folder space in memory. The *Properties* menu contains the more sophisticated functionalities for syntactic and semantic support. Users can load schema files for the rule base under development, they can check the well-formedness and validity of the RuleML file, while users can also launch the Semantic Tag Mapping (STM) module that comprises the backbone of the system's semantic support and is described subsequently.

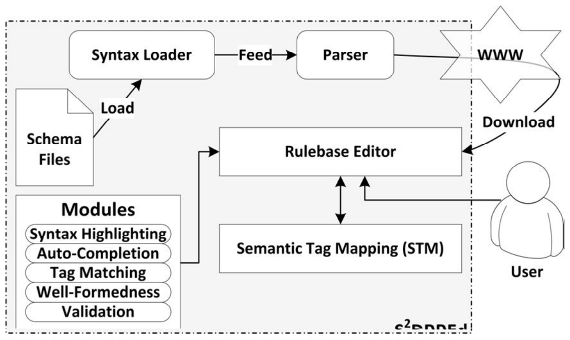
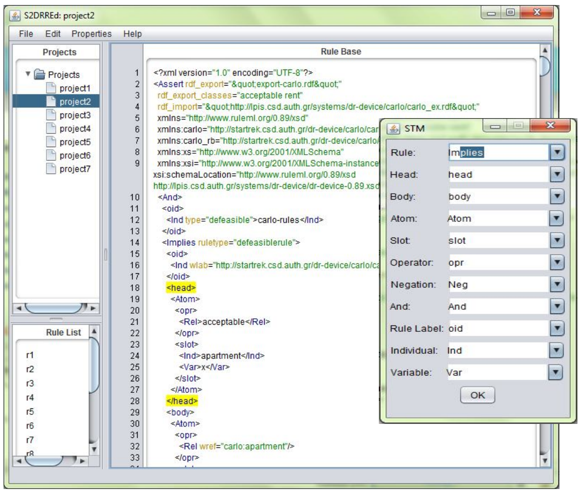


Figure 2. S^{2}DRREd Overall Architecture.

The bulk of the main window is occupied by the rule base editor, while the two side panels are the *Projects Tree* and the *Rules List*. The former displays the tree of available S^{2}DRREd projects, while the latter features a list of all rules contained in the loaded rule base. By clicking on a rule name, the cursor moves to the corresponding rule in text area. As new rules are added, the rule list is automatically refreshed with the new rule names.

3.2 Semantic Tag Mapping

After loading a schema S^{2}DRREd displays the *STM* (*Semantic Tag Mapping*) window (Figure 3), where the user can match the tags of the loaded vocabulary to the rule base concepts. In essence, STM provides a *meta-modelling facility* for generating schemas over various language versions. A *meta-model* provides a schema for semantic data, specifying what elements may be contained in the model and how they relate to one another. In reality, each meta-model is a specification of a domain-specific modelling language.



```
<?xml version="1.0" encoding="UTF-8"?>
<Assert rdf_export="&quot;export-carlo.rdf&quot;"
 rdf_export_classes="acceptable rent"
 rdf_import="&quot;http://lpis.csd.auth.gr/systems/dr-device/carlo/carlo_ex.rdf&quot;"
 xmlns="http://www.ruleml.org/0.89/xsd"
 xmlns:carlo="http://startrek.csd.auth.gr/dr-device/carlo/car
 xmlns:carlo_rb="http://startrek.csd.auth.gr/dr-device/carlo/
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance
xsi:schemaLocation="http://www.ruleml.org/0.89/xsd
http://lpis.csd.auth.gr/systems/dr-device/dr-device-0.89.xsd
 <And>
  <oid>
   <Ind type="defeasible">carlo-rules</Ind>
  </oid>
  <Implies ruletype="defeasiblerule">
   <oid>
    <Ind wlab="http://startrek.csd.auth.gr/dr-device/carlo/ca
   </oid>
   <head>
    <Atom>
     <opr>
      <Rel>acceptable</Rel>
     </opr>
     <slot>
      <Ind>apartment</Ind>
      <Var>x</Var>
     </slot>
    </Atom>
   </head>
   <body>
    <Atom>
     <opr>
      <Rel wref="carlo:apartment"/>
     </opr>
```

Rule: Implies
Head: head
Body: body
Atom: Atom
Slot: slot
Operator: opr
Negation: Neg
And: And
Rule Label: oid
Individual: Ind
Variable: Var

Figure 3. S^{2}DRREd Main and STM windows.

An STM meta-model is a collection of notions that form the vocabulary, which provides an understanding of the given domain. It provides a communication means between the end user and S^{2}DRREd, by facilitating common grounds of understanding over the notions of "rule", "rule type" "rule head", "rule body" etc. STM is the primary, novel contribution of this work that can handle a wide range of notions, with the list being easily customizable as well.

There is the option of choosing the tag/element name considered more suitable from a list that contains all the elements and attributes of the loaded schema. The latter list promotes user-friendliness, by also providing auto-completion features during typing.

For the schemas already included (v.0.89, v.0.9 and v.0.91), the STM table is already filled; the process of manually matching tags and concepts is necessary only when loading a custom schema and has to be performed only once, the first time a new language schema is loaded to the system.

The following subsections indicate how STM semantically supports some of the secondary editor functionalities.

3.2.1 Syntax Highlighting

S^{2}DRREd provides syntax highlighting for all loaded XML-based languages. When loading a schema, highlighting is supported for the elements and attributes detected in the schema and defined by the current sublanguage. It is easily understood, that this functionality of the software is not restricted to defeasible logic RuleML-like languages only, but to any XSD-based XML sublanguage, making S^{2}DRREd a highly versatile system.

3.2.2 Code Completion (Auto-complete)

Upon typing "<" (i.e. inserting a new tag in the rule base), a popup menu with the suitable choices shows up, which are (semantically) defined from STM. For example, the rule contains a head and a body, thus, when opening a new tag inside a rule tag (element `<Implies>` for v.0.91 of the language), the two choices ("head" and "body") will appear. **Σφάλμα! Το αρχείο προέλευσης της αναφοράς δεν βρέθηκε.** demonstrates the auto-complete feature.

3.2.3 Tag Matching

This feature highlights matching sets of tags. It assists in navigating the rule base and spotting potentially improper matching. Figure 3 illustrates an example of utilizing this feature, where the opening and closing tags of element `<head>` are highlighted in the main window of the program.

4. RELATED WORK

There are various free and commercial software tools for deploying rule bases. For example, the *SWRL Editor* [11] is a Protégé-OWL plug-in for creating, authoring and editing documents containing SWRL rules that also allows interoperation with third-party inference engines through a Java API. The software can save only syntactically valid rules and only allows saving of rules relating to currently loaded OWL entities. It offers elementary semantic checking (e.g. no variables that were not referred to in the antecedent can be used in a rule consequent) and auto-completion. Detailed logic checking, on the other hand, is not performed (e.g. a rule could contradict OWL constraints). SWRL rules are saved as OWL individuals, along with the associated OWL file. Together with the SWRL Editor, the SWRL API is also provided that offers mechanisms for creating and manipulating SWRL rules in an OWL knowledge base. Finally, the tool also provides inference capabilities through the Jess rule engine, which is integrated into Protégé-OWL.

DLRule [7] is another plug-in for Protégé that allows the rewriting of specific rules into DL axioms using OWL 1.1. If a rule isn't rewritable, it is added to the ontology as an SWRL rule. The rules that are rewritten as DL axioms enable additional inference avoiding the DL-safety limitation. Non-rewritable rules (added as SWRL) can be handled by a reasoner that supports DL-safe rules. The main features provided by the plug-in are a graphical representation of the rules and a validation tool that presents the effect of the rule on the ontology. A command-line tool for processing the ontologies is also offered.

Ontology Rule Editor (*ORE*) [12] is a similar tool for editing, testing, debugging and validating ontology rules. It provides an

API that can be accessed by third-party software to manage ontologies and perform rule-based reasoning. It also provides a graphical environment for defining rules, by browsing the domain elements in the ontology and dragging and dropping them to the corresponding slots in the rule. The platform that performs inference process is based on Jena and Pellet reasoning engines, but can be easily extended to include more engines. Debugging and validation are performed with syntactic and semantic checking of rule definitions.

Finally, a commercial software paradigm is *Rule Manager*[1] by Acumen Business. The tool allows business users to develop, animate, validate and visualize business rules, using simple, English-based textual constructs, without the need of inserting technical code. The main features are: vocabulary management, animation through an interactive rule map, rule validation and verification, missing rules discovery, full text search, report generation, rule graph and export to Windows Workflow Foundation. Example target rule engines are Microsoft BizTalk and Microsoft Windows Workflow Foundation. Developing rules with Rule Manager is based on Reaction RuleML and the software also integrates a corresponding adapter for exporting business rule policies in this format. Other examples of commercial (business) rule editors include Iovation's software[2] and the *Business Rule Composer*, a graphic tool used for authoring, versioning, and deploying policies and vocabularies.

Compared to the software tools presented above, S^{2}DRREd offers a significantly higher degree of semantic support during the development of rule bases. More specifically, contrary to the rest of the systems that explicitly comply with respective rule language syntaxes (e.g. SWRL, RuleML etc), S^{2}DRREd is versatile enough to adapt to any XML-based rule language. The only requirement on behalf of the user is to manually create (via STM) a meta-model corresponding each time to the specific XML-based rule language (e.g. DR-RuleML etc) that describes the key notions of the rule base. This procedure has to be performed only once, during the first time a new (sub)language schema is loaded to the system.

5. CONCLUSIONS AND FUTURE WORK

The paper presents S^{2}DRREd, a software tool for authoring defeasible theories. The software adds a supplementary level of semantic assistance during rule base development that cooperates with established techniques for syntactic support. The Semantic Tag Mapping module of the system creates meta-models of the main defeasible logic theory notions, like "rule", "rule type" etc, independently of the explicitly chosen rule language syntax.

As for our future plans for improving the software, it would be interesting to take advantage of S^{2}DRREd's modular architecture and integrate the tool with a reasoning engine, like e.g. *OO-JDrew* [6]. The latter currently possesses a simple text editor for authoring rule bases and it would definitely improve its functionality to cooperate with a RuleML-aware editor instead. Another appealing improvement would be the integration of *d-POSL* (defeasible POSL) into S^{2}DRREd. d-POSL [9] is a defeasible extension to *POSL* [5], an ASCII language that integrates Prolog's positional and F-logic's slotted syntaxes for representing knowledge (facts and rules) in the Semantic Web.

[1] http://www.acumenbusiness.com/

[2] http://www.iovation.com/business-rules-editor.html

6. REFERENCES

[1] Bassiliades, N., Antoniou, G., & Vlahavas, I. 2006. A Defeasible Logic Reasoner for the Semantic Web. *Int. J. on Semantic Web and Information Systems* 2, 1, Sheth, A., & Lytras, M. D. (Eds.), IDEA Group Publishing, 1-41.

[2] Berners-Lee, T., Hendler J., & Lassila O. 2001. The Semantic Web. *Scientific American* 284, 5, 34-43.

[3] Boley, H. 2003. Object-Oriented RuleML: User-Level Roles, URI-Grounded Clauses, and Order-Sorted Terms. *Proc. Rules and Rule Markup Languages for the Semantic Web* (*RuleML-2003*). Sanibel Island, Florida, LNCS 2876, Springer-Verlag, 1-16.

[4] Boley, H. 2006. The RuleML Family of Web Rule Languages. In *Principles and Practice of Semantic Web Reasoning*. Springer, 1-17.

[5] Boley, H. 2010. Integrating Positional and Slotted Knowledge on the Semantic Web. *Journal of Emerging Technologies in Web Intelligence* 2, 4, 343-353.

[6] Craig, B. 2007. The OO jDREW Engine of Rule Responder: Naf Hornlog RuleML Query Answering. *Proc. Int. Conf. on Advances in Rule Interchange and Applications (RuleML'07)*, Paschke, A., & Biletskiy, Y. (Eds.). Springer-Verlag, Berlin, Heidelberg, 149-154.

[7] Gasse, F., & Haarslev, V. 2008. DLRule: A Rule Editor Plug-in for Protégé. *Proc. 4th Int. Workshop on OWL: Experiences and Directions (OWLED 2008)*. Washington, DC, USA, April 1-2.

[8] Horrocks, I., & Patel-Schneider, P.F. 2004. A Proposal for an OWL Rules Language. *Proc. 13th Int. Conf. on World Wide Web* (WWW '04). ACM, NY, USA, 723-731.

[9] Kontopoulos, E., Bassiliades, N., & Antoniou, G. 2011. Visualizing Semantic Web Proofs of Defeasible Logic in the DR-DEVICE System. *Knowledge-based Systems* 24, 3, Elsevier, 406-419.

[10] Nute, D. 1994. Defeasible Logic. *Handbook of Logic in Artificial Intelligence and Logic Programming*, Gabbay, D.M., Hogger, C.J., and Robinson, J. A. (Eds.). Vol. 3, Oxford University Press, 353-395.

[11] O'Connor, M.J., Knublauch, H., Tu, S.W., Grossof, B., Dean, M., Grosso, W.E., & Musen, M.A. 2005. Supporting Rule System Interoperability on the Semantic Web with SWRL. *Proc. 4th Int. Semantic Web Conference (ISWC)*, Galway, Ireland, Springer Verlag, LNCS 3729, 974-986.

[12] Ortega, A.M., Alcaraz Calero, J.M., Botía Blaya, J.A., Martínez Pérez, G., & García Clemente, F.J. 2010. Knowledge Authoring with ORE: Testing, Debugging and Validating Knowledge Rules in a Semantic Web Framework. *J. UCS* 16, 2, 1234-1266.

[13] Reynolds, D. 2010. OWL 2 RL in RIF. *W3C Working Group Note 22 June 2010*. Available at: http://www.w3.org/TR/rif-owl-rl/#Background_OWL_2_RL, last access: November 2011.