ASP — PROLOG: Composition and Interoperation of Rules

C. Baral[1], E. Pontelli[2], and T.C. Son[2]

[1] Arizona State University
Department of Computer Science
chitta@asu.edu
[2] New Mexico State University
Department of Computer Science
{epontell,tson}@cs.nmsu.edu

1 Introduction

One of the main goals of the Semantic Web initiative [3] is to extend the current Web technology to allow for the development of intelligent agents, which can *automatically* and *unambiguously* process the information available on millions of web pages. It has been recognized very early in the development of the Semantic Web that rules are essential for the Web[3] and for Semantic Web applications—e.g., description of semantic web services, rules interchange for e-business applications.

The RuleML initiative is a response to the need of a shared rule markup language using XML markup, which has a precisely defined semantics and efficient implementations. In recent years, a significant amount of work has been devoted to develop knowledge representation languages suitable for the task and a variety of languages for rule markup has been proposed. The initial design [4] included a distinction (in terms of distinct DTDs) between *reaction rules* and *derivation rules*. The first type of rules is used for the encoding of event-condition-action (ECA) rules while the second is meant for the encoding of implicational/inference rules. Despite the fact that many different proposals for ECA rules encoding have appeared the work on ECA rules is still very vague. The most recent modularized description of RuleML [6] reports this area (indicated as *PR RuleML* in that document) as work in progress.

The derivation rules component of the RuleML initiative has originated a family of languages.[4], Datalog plays the role of a core language, with simplified versions (unary and binary Datalog) developed for combining RuleML with OWL (as in SWRL). Various sublanguages have been created to include features like explicit equality (e.g., `fologeq`), negation as failure (e.g., `naffolog`), and Hilog layers (e.g., `hohornlog`). Various authors [7] have argued that any realistic architecture for the Semantic Web must be based on various independent but interoperable languages, including logic programming languages with and without negation-as-failure. The need for these languages and their interaction have been discussed (e.g., [8, 7]). It is also of

[3] `http://www.w3.org/DesignIssues/Rules.html`
[4] `www.ruleml.org/modularization/ruleml_m12n_089_uml_05-06-01.png`.

note that many of the sublanguages of RuleML have been implemented either through translators (e.g., GEDCOM, which translates to XSB and JESS) or engines (e.g., j-DREW, a top-down engine for RuleML, DR-Device, an engine supporting defeasible logic and both strong and default negation, and CommonRules, a bottom-up engine for the Datalog sublanguage).

In this work, we propose a general framework to address the problem of *(i)* inter-operation between knowledge bases encoded using different RuleML languages, and *(ii)* development and integration of different components that reason about RuleML knowledge bases. The approach adopted in this work relies on using a core logic programming framework to address the issues of integration and inter-operation. In particular, the spirit of our approach relies on the following beliefs:

- the natural semantics of various levels of the RuleML deduction rules hierarchy can be captured by different flavors of logic programming;
- modern logic programming systems are provided with foreign interfaces that allow declarative interfacing to other paradigms.

The idea is to combine the ASP-Prolog framework of [5] with the notations for modularization of answer set programming of [2]. The result is a logic programming framework, where modules responding to different logic programming semantics (e.g., Herbrand minimal model, well-founded semantics, answer set semantics) can co-exist and interoperate.

The framework provides a natural answer to the problems of use and inter-operation of RuleML knowledge. Most of the emphasis is on using answer set programming to handle some of the sublanguages (e.g., datalog, ur-datalog, nafdatalog and negdatalog), though the core framework will naturally support most of the languages (e.g., hornlog, hohornlog).

A thorough presentation of the framework can be found in [1].

References

[1] C. Baral et al. A Framework for Composition and Interoperation of Rules in the Semantic Web. *RuleML*, Springer Verlag, pp. 39-50, 2006.

[2] C. Baral et al. Macros, macro calls, and use of ensembles in modular answer set programming. *ICLP*, Springer, 2006.

[3] T. Berners-Lee et al. The semantic web. In Scientific American, May 2001.

[4] H. Boley et al. RuleML Design, Version 0.8. 2002-09-03, 2002.

[5] O. Elkhatib et al. A Tool for Knowledge Base Integration and Querying. *AAAI Spring Symp.*, AAAI Press, 2006.

[6] D. Hirtle, H. Boley. The Modularization of RuleML. 2005-12-15, 2005.

[7] M. Kifer et al. A Realistic Architecture for the Semantic Web. *RuleML*, Springer Verlag, pp. 17–29, 2005.

[8] W. May et al. Active Rules in the Semantic Web: Dealing with Language Heterogeneity. *RuleML*, Springer, 2005.