Applying Prolog to Semantic Web Ontologies & Rules Moving Toward Description Logic Programs

K. Samuel[1], L. Obrst[1], S. Stoutenburg[2], K. Fox[2], P. Franklin[2], A. Johnson[2], K. Laskey[2], D. Nichols[1], S. Lopez[2], J. Peterson[2]

The MITRE Corporation*
[1] 7525 Colshire Drive, McLean, VA 22102-7508
[2] 1155 Academy Park Loop, Colorado Springs, CO 80910-3716
{samuel, lobrst, suzette, kfox, pfranklin, abjohnson, klaskey, dlnichols, slopez, jasonp} @mitre.org

We are developing SWORIER (Semantic Web Ontologies and Rules for Interoperability with Efficient Reasoning), which is a system that uses Logic Programming to reason about ontologies and rules in order to answer queries. The system expects a human developer to create ontologies in the formalisms of OWL-DL (Web Ontology Language for Description Logic) along with rules in SWRL (the Semantic Web Rule Language) or RuleML (the Rule Markup Language). Then, at compile time, this information is translated into Prolog code using XSLTs (Extensible Stylesheet Language Transformations). In addition, a Prolog program called 'General Rules', which is meant to capture the semantics of OWL's primitives, is appended to the XSLT output to form a complete Prolog program. We then use knowledge compilation techniques to create an efficient version of the program. At run time, the system can answer queries and assimilate dynamic changes by reasoning over the given information.

For our prototype, we have developed ontologies and rules in a military command and control domain in which a supply convoy moves through an unsecured area. New information can become available at any time, such as an approaching sandstorm or the discovery of a new hostile theater object. Rules trigger alerts and recommendations to assist the commander in making decisions. For example, if an enemy unit is within the convoy's region of interest, the system reports that and recommends a new route.

Recent research has addressed issues similar to ours concerning combining logic programming with Semantic Web ontologies and rule technologies. Related work includes Description Logic Programming [2, 4, 6], answer set programming [1, 5, 7], and courteous logic programs [3]. In particular, we are building on the groundbreaking work of [8], addressing a number of problems that the paper suggested were unsolvable.

For example, Prolog typically has negation as (finite) failure, while OWL uses logical negation. So we created a new Prolog predicate called **`logicNot`**. Also, to

* The authors' affiliation with The MITRE Corporation is provided for identification purposes only, and is not intended to convey or imply MITRE's concurrence with, or support for, the positions, opinions or viewpoints expressed by the authors. We note that the views expressed in this paper are those of the authors alone and do not reflect the official policy or position of any other organization or individual.

address the fact that Prolog does not allow disjunction in the head, we will create another Prolog predicate, **`or`**. The **`logicNot`** predicate enables SWORIER to represent OWL's open world assumption and to reason about complementary and disjoint classes. And with the **`or`** predicate, SWORIER can handle enumerated classes (the **`owl:oneOf`** primitive).

We created a Prolog predicate for each OWL primitive, unlike [8], who made the ontology's (object-level) classes and properties into Prolog predicates. Our syntax makes it easier to enforce substitutivity of equivalent classes and to handle inconsistent cardinality restrictions.

Most inconsistencies can be addressed in multiple ways, such as by sending an error message to the developer or by creating a new unnamed individual to satisfy a constraint. In this way, we address constraints such as those imposed by cardinality and existential quantification.

At run time, SWORIER can reason about queries, switch from one rule set to another, and assimilate assertions and deletions of individuals.

We ran experiments with the convoy use case described above. SWORIER was too slow for practical use until we implemented three techniques: extensionalization, avoiding reanalysis, and code minimization. Now SWORIER's knowledge compilation phase takes less than seven hours, and at runtime, SWORIER can answer a query in less than a second and assimilate a dynamic change in a few milliseconds.

In the future, we intend to test our ideas that address issues involving disjunction, inconsistencies, enumerated classes, cardinality, etc. We will also implement more OWL primitives and enable SWORIER to handle more kinds of dynamic changes.

1. Eiter, T., Lukasiewicz, T., Schindlauer, R., Tompits, H.: Combining Answer Set Programming with Description Logics for the Semantic Web. In: The Proceedings of the 9th International Conference on Principles of Knowledge Representation and Reasoning (2004).
2. Eiter, T., Ianni, G., Schindlauer, R., Tompits, H.: DLV-HEX: Dealing with Semantic Web under Answer-Set Programming. In: The Proceedings of the 4th International Semantic Web Conference (2005).
3. Grosof, B., Labrou, Y., Chan, H. Y.: A Declarative Approach to Business Rules in Contracts: Courteous Logic Programs in XML. In: The Proceedings of the 1st ACM Conference on Electronic Commerce (1999).
4. Grosof, B., Horrocks, I., Volz, R., Decker, S.: Description Logic Programs: Combining Logic Programs with Description Logic. In: The Proceedings of the 12th International Conference on the World Wide Web (2003).
5. Heymans, S., Vermeir, D.: Integrating Description Logics and Answer Set Programming. In: The Proceedings of the International Workshop on Principles and Practice of Semantic Web Reasoning. Springer LNCS 2901 (2003), 146-159.
6. Motik, B., Rosati, R.: Closing Semantic Web Ontologies. University of Karlsruhe Technical Report. www.cs.man.ac.uk/~bmotik/publications/paper.pdf (2006).
7. Niemel, I., Simons, P.: Smodels — An Implementation of the Stable Model and Well-Founded Semantics for Normal Logic Programs. In: The Proceedings of the 4th International Conference on Logic Programming and Non-Monotonic Reasoning (1997), 420-429.
8. Volz, R., Decker, S., Oberle, D.: Bubo — - Implementing OWL in Rule-Based Systems. www.daml.org/listarchive/joint-committee/att-1254/01-bubo.pdf (2003).