DR-Prolog:A System for Reasoning with Rules and Ontologies on the Semantic Web

Antonis Bikakis and Grigoris Antoniou

Computer Science Department, University of Crete, Greece
Institute of Computer Science, FORTH, Greece
14 M. Kalivianis Str., Heraklio, Crete, Tel: +306945894785
{bikakis,antoniou}@ics.forth.gr

Introduction

The development of the Semantic Web [Berners Lee *et al*., 2001] proceeds in layers, each layer being on top of the others. At present, the highest layer that has reached sufficient maturity is the ontology layer in the form of the description logic based languages, DAML+OIL and OWL.

The next step in the development of the Semantic Web will be the logic and proof layers. Rule systems can play a twofold role in the Semantic Web initiative: (a) they can serve as extensions of, or alternatives to, description logic based ontology languages; and (b) they can be used to develop declarative systems on top (using) ontologies.

Defeasible reasoning is a simple rule-based approach to reasoning with incomplete and inconsistent information. It can represent facts, rules, and priorities among rules. Its main advantage is the combination of enhanced representational capabilities allowing one to reason with incomplete and contradictory information, coupled with low computational complexity compared to mainstream nonmonotonic reasoning .

In this paper we report on the implementation of a defeasible reasoning system for reasoning on the Web. Its main characteristics are the following: (a) It is compatible with RuleML (details in full paper[1]); (b) It is based on Prolog. The core of the system consists of a well-studied translation [Antoniou *et. al*., 2001] of defeasible knowledge into logic programs under Well-Founded Semantics [van Gelder *et al.*, 1991]; (c) The main focus is on flexibility. Strict and defeasible rules and priorities are part of the interface and the implementation. Also, a number of variants are implemented (ambiguity blocking/propagation, conflicting literals); (d) The system can reason with rules and ontological knowledge, through the transformation of the RDFS constructs and many OWL constructs into rules.

Basic Characteristics of Defeasible Logics

The root of defeasible logics lies on research in knowledge representation, and in particular on inheritance networks. Defeasible logics can be seen as inheritance networks expressed in a logical rules language. In fact, they are the first nonmonotonic reasoning approach designed from its beginning to be implementable.

Being nonmonotonic, defeasible logics deal with potential conflicts (inconsistencies) among knowledge items. Thus they contain classical negation, contrary to usual logic programming systems. They can also deal with negation as failure, the other type of negation typical of nonmonotonic logic programming systems. In defeasible logics, often it is assumed that NAF is not included in the object language. However, as Antoniou *et al.* [2000a] show, it can be easily simulated when necessary.

Conflicts among rules are indicated by a conflict between their conclusions. These conflicts are of local nature. The simpler case is that one conclusion is the negation of the other. The more complex case arises when the conclusions are declared to be mutually exclusive. Priorities on rules may be used to resolve some conflicts among rules.

Defeasible logics are skeptical in the sense that conflicting rules do not fire. Thus consistency of drawn conclusions is preserved.

Translation into Logic Programs

Translation of Defeasible Theories

The translation of a defeasible theory D into a logic program $P(D)$ has a certain goal: to show that

p is defeasibly provable in $D \Leftrightarrow$
p is included in the Well-Founded Model of $P(D)$

Two different translations have been so far been proposed, sharing the same basic structure:

- The translation of [Antoniou et al., 2000b; Maher et al., 2001] where a meta-program was used.
- The translation of [Antoniou and Maher, 2002], which makes use of control literals.

In DR-Prolog, we have adopted the second approach, mainly because of its better computational efficiency,. The system uses two different metaprograms (the ambiguity

[1] http://www.csd.uoc.gr/~bikakis/DR-Prolog.pdf

blocking and the ambiguity propagation metaprograms), which consist of logical clauses that define the definite and defeasible provability of DL theories.

Translation of RDF(S) and parts of OWL ontologies

In order to support reasoning with RDF/S and OWL ontologies, we translate RDF data into logical facts, and RDFS and OWL statements into logical facts and rules.

The RDF data are transformed into logical facts of the form: *Predicate(Subject,Object)*. To capture the semantics of RDF Schema constructs, we create the following rules.

a*: C(X):- rdf:type(X,C).*
b*: C(X):- rdfs:subClassOf(Sc,C),Sc(X).*
c*: P(X,Y):- rdfs:subPropertyOf(Sp,P),Sp(X,Y).*
d*: D(X):- rdfs:domain(P,D),P(X,Z).*
e*: R(Z):- rdfs:range(P,R),P(X,Z).*

Parts of OWL ontologies can also be translated using logical rules, which capture the semantics of some OWL constructs. For example the following rules capture the semantics of transitive and symmetric properties.

o1*: P(X,Z):- P(X,Y), P(Y,Z),*
rdf:type(P,owl:TransitiveProperty).
o2*: P(X,Y):- P(Y,X), rdf:type(P,owl:SymmetricProperty).*

Other OWL constructs which are also supported by our system are: *owl:equivalentClass, owl:equivalentProperty, owl:sameIndividualAs, owl:sameAs,* property restrictions , owl collections etc.

Implementation

DR-Prolog, in accordance with the general philosophy of logic programming, is designed to answer queries. There are two kinds of queries, depending on which strength of proof we are interested in: definite / defeasible provability.

In Figure 1 we present the overall architecture of our system. The system works in the following way: The user imports defeasible theories, either using the syntax of defeasible logic, or in the RuleML syntax. The theories are checked by the DL Parser, and if syntactically correct, they are passed to the Logic Translator, which translates them into logic programs. The RuleML theories are checked by the RuleML Parser and translated into defeasible theories, which are also passed to the Logic Translator. The Reasoning Engine compiles the logic programs and the metaprogram, and evalutes the answers to the user's queries. The logic programming system that we use is XSB, basically because: (a) it supports the well-founded semantics of logic programs through the use of tabled predicates and its *sk_not* negation operator; and (b) it offers an easy and efficient way to communicate with the other parts of the system. The RDF&OWL Translator is used to translate the RDF/S and OWL data into logical facts & rules, which can be processed by the defeasible rules, provided by the user.

There exist several previous implementations of defeasible logic, as *d-Prolog* [Covington *et al,* .2002],

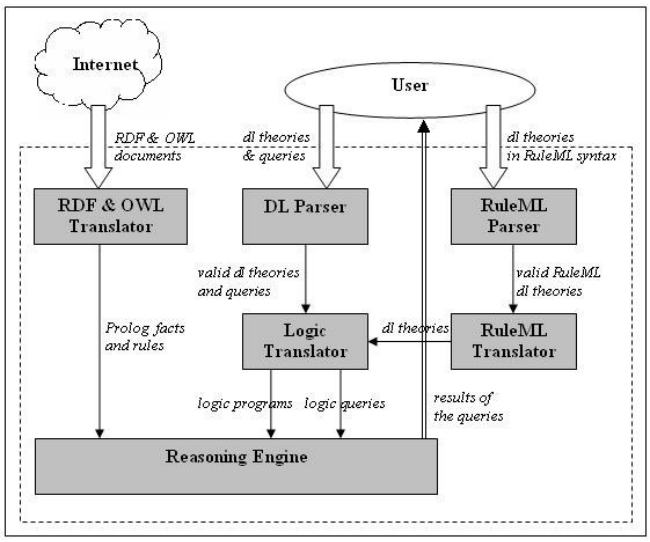

Figure 1: The overall architecture of DR-Prolog

Deimos and *Delores* [Maher *et al*., 2001], and *SweetJess* [Grosof *et al.*, 2002]. Comparing to our system the former three do not integrate with Semantic Web, and *SweetJess* implements only one reasoning variant of defeasible logic.

References

[Antoniou *et* al., 2000a] G. Antoniou, M. J. Maher and D. Billington (2000). Defeasible Logic versus Logic Programming without Negation as Failure. *Journal of Logic Programming* 41,1 (2000): 45-57

[Antoniou *et* al., 2000b] G. Antoniou, D. Billington, G. Governatori, M. J. Maher: A Flexible Framework for Defeasible Logics. In *Proc. AAAI' 2000*, 405-410

[Antoniou *et* al., 2001] G. Antoniou, D. Billington, G. Governatori and M.J. Maher (2001). Representation results for defeasible logic. *ACM Transactions on Computational Logic* 2, 2 (2001): 255 - 287

[Antoniou and Maher, 2002] G. Antoniou, M.J. Maher (2002). Embedding Defeasible Logic into Logic Programs. In *Proc. ICLP 2002,* 393-404

[Berners-Lee *et al*., 2001]T. Berners-Lee, J. Hendler, and O. Lassila (2001). The Semantic Web. *Scientific American*, 284, 5 (2001): 34-43

[Covington *et al*., 1997] M. A. Covington, D. Nute and A. Vellino (1997). *Prolog Programming in Depth*, 2nd ed. Prentice-Hall

[van Gelder *et al*., 1991] A. van Gelder, K. Ross and J. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM* 38 (1991): 620—650

[Grosof *et al*., 2002] B. N. Grosof, M. D. Gandhe and T. W. Finin: SweetJess: Translating DAMLRuleML to JESS. RuleML 2002. In: *Proc. International Workshop on Rule Markup Languages for Business Rules on the SW*

[Maher *et al*., 2001]M. J. Maher, A. Rock, G. Antoniou, D. Billington and T. Miller (2001). Efficient Defeasible Reasoning Systems. *International Journal of Tools with Artificial Intelligence* 10,4 (2001): 483--501