Requirements for an Expressive Rule Language on the Semantic Web

Michael Kifer
Department of Computer Science
State University of New York at Stony Brook
Stony Brook, NY 11794-4400
kifer@cs.stonybrook.edu

Position paper for the W3C Workshop on Rule Languages, Washington D.C., April 2005

Abstract

This position paper is an attempt to enumerate and substantiate a number of key features, which, we believe, should be part of an expressive rule-based language to support the Semantic Web. This list of desiderata is based on our experience with the implementation and use of the FLORA-2 system [11, 31] as well as on our work on the languages for Semantic Web Services as part of the SWSL (*http://www.daml.org/services/swsl*) and WSMO (*http://www.wsmo.org*) projects.

1 Introduction

Semantic Web has become one of the main foci of research in AI and knowledge representation and we have already seen the first steps towards standardization of the representation of shareable knowledge on the Web. The most notable of these standardized languages are RDF [20] and OWL [26]. At the same time, there is growing sentiment that RDF and OWL are not the end of the story and the next big thing will be a Web rule language. SWRL, a rule language extending OWL has recently been proposed [16], a rule language for RDF, N3 [3], has been around for several years, and two rules languages, WSML-Rules [10] and SWSL-Rules [2], both based on F-logic [18], have just been proposed as possible bases for Semantic Web Services.

Rule-based languages have been around for a long time, and for about seven years, in the 1980s, they had their moment of glory, and Prolog was all the rage. Unfortunately, this first wave of heightened interest was replaced with gloom and skepticism when the unrealistically high expectations of the breakthrough due to the use of Prolog did not materialize. Our contention is that the root of the problem was not performance, as some people outside of the field tend to think, but the fact that Prolog is neither declarative (despite all the claims) nor high-level. In fact, as a rule language, it is relatively featureless—an assembler of the rule languages of the world. As a result, it was very hard for an average programmer to learn how to write Prolog programs correctly and even harder to squeeze performance out of them.

In this position paper we discuss progress that was made in the almost twenty years since Prolog fell out of favor—progress that gives hope that the future rule language for the Semantic Web will be met more favorably. We discuss the semantic and syntactic developments that can make a rule language more declarative and better suited for knowledge representation. We do not discuss XML serialization of such a language—a complimentary issue that is being addressed under the umbrella of the RuleML initiative [23].

2 Fixing the Semantics

From the very beginning, Prolog has been plagued with two semantic problems: its control strategy did not match its presumed logical semantics and its treatment of negation as failure was ad hoc and logically flawed. Both of these problems were solved in the late 80's. The OLD resolution method [27] made the top-down evaluation strategy logically complete and the Magic Sets method [1] made the bottom-up strategy more efficient. In the 1990's both of these methods were implemented in various systems. XSB [24] is the best-known such system, which uses the modified OLD resolution in a top-down evaluation strategy. At least one other high-performance system, the YAP Prolog[1] has a partial implementation of the same strategy. Unfortunately, the Magic Sets-based bottom-up engines developed during the 1990's, LDL, Coral, and Aditi, are no longer supported.

The problem with negation as failure was also solved in late 1990's. The two widely used solutions are based on the so-called well-founded semantics for negation [29] and on the stable model semantics [12]. The well-founded semantics has a nice property that it always has a unique model, and it is favored in logic programming systems such as XSB and YAP. However, stable model semantics is more suitable for some other reasoning tasks.

Another important related development was the introduction of classical negation alongside negation as failure [13, 14]. This gives the advantage of being able to specify negative information directly in the knowledge base, which can greatly simplify the specifications.

3 Frames: Making the Logic Object-Oriented

One of the widely recognized deficiency of logic-based languages (dating back to Minsky [22]) is the fact that these languages are based on predicate calculus and thus on an essentially flat data model. To counter that,

[1] http://yap.sourceforge.net

Minsky has argued that knowledge should be represented using *frames* [22] and, since (in his view) frames are inherently non-logical, logic is inappropriate as a knowledge representation language. The same conclusion was reached by Ullman later, although his argument was different [28].

Nevertheless, suitable logical formalisms have soon been proposed. One of the most popular such formalisms is F-logic [18]. F-logic has been advocated as a logical way to provide a reasoning component on top of RDF, for example, in the TRIPLE system [25] and in [30]. The upcoming member submissions from WSMO and SWSI also use F-logic as the basis for their rule languages.

The basic logical formulas in F-logic are frames of the form `mary[name->'Mary Jones', children->{bob,kathy}]` and statements about class membership, such as `mary:employee` and subclass relationship, such as `employee::person`. These statements can be combined using logical connectives and the rules sublanguage can be formed in the usual way. There are also facilities to make statements about types and more.

4 Meta-reasoning

One of the most attractive features of F-logic is that it has higher-order flavor and yet retains tractability. The higher-order flavor comes from treating classes as objects and due to the ability to place variables virtually anywhere. In this way, object data and meta-data can be queried uniformly. This seamless integration of object data and meta-data paves the way for declarative meta-reasoning.

HiLog [7, 8] is a further extension in the direction of meta-reasoning. Developed in 1989, HiLog was independently rediscovered in 1991 under the name of SKIF [15], which is a testimony for the robustness of the idea. HiLog allows variables not only over method names or classes, but also over function and predicate symbols, and over atomic formulas. For instance, the well-known built-in of Prolog, `call`, can be defined completely logically in HiLog as follows:

```
call(?X) :- ?X.
```

In this way, HiLog supports *reification* of atomic formulas. This idea was applied to F-logic and further extended in [30]. HiLog goes even further by supporting parameterized families of predicates. For example, this makes it possible to define the generic transitive closure of an arbitrary predicate (represented by the variable `?P`) :

```
closure(?P)(?X,?Y) :- ?P(?X,?Y).
closure(?P)(?X,?Y) :- ?P(?X,?Z) and closure(?P)(?Z,?Y).
```

HiLog and F-logic are seamlessly integrated in the FLORA-2 system [11, 31].

5 State Changes, Reactive Rules and Procedural Attachments

We believe that an expressive rule language needs to support at least the following three features, and various researchers proposed partial solutions to some of them:

- *Event-condition-action rules* (triggers, reactive rules)
- *Procedural attachments*
- *State-changing actions*

Limitation of space does not allow us to survey the various approaches, so we will mention only what we believe is the most promising approach: *Transaction Logic* [4, 5]. Our reasons have to do with the fact that this logic provides a solution to all the above desiderata in a uniform and declarative way. Transaction Logic was designed specifically for the rule-based paradigm and was intended to provide a declarative account for state-changing actions within such languages.

The basic idea behind Transaction Logic is introduction of a new logical connective and a corresponding path-based model theory. These two ideas enable the logic to talk about sequences of state-changing and state-querying actions. Elementary state changes are modeled through so-called transition oracles, which is precisely what is needed in order to support procedural attachments. Various works have shown that this logic is natural enough to support the concept of triggers in databases, process modeling, and even behavioral aspects of contracting for Semantic Web Services [6, 9]. Transaction Logic is implemented as part of the FLORA-2 system and another implementation is also available at *http://www.cs.toronto.edu/∼bonner/ctr/*.

6 Paraconsistency and Approximate Reasoning

One would be wise to take what one finds on the Web with a grain of salt, and there is no reason to give a different treatment to Web information that is specified formally. An intelligent Web agent should be able to assign certainty factors to statements it comes across in various ontologies, and it should not break down when some of these statements contradict each other. Tolerance to such contradictions is called *paraconsistency*. There is vast literature on the subject [21], and limitation of space does not permit us spending any time on it. We would only like to mention one proposal, which was designed for and was developed in the framework of a traditional rule-based language: the *Annotated Logic* [19].

In an annotated logic, each fact has a certainty factor, and rules derive new facts also with only some degree of confidence. Contradictory facts lead to local inconsistency, which affects only the information that is directly dependent on the contradictory facts. The knowledge base remains globally consistent, and useful inferences can be drawn from the parts that are not tainted. Annotated logic has been implemented and successfully used in the HERMES system.[2]

7 Other Features

In this section we will briefly mention some other important features of an expressive rule language, which we cannot elaborate on due to space limitations.

Database style constraints. Constraints in the database style are used to ensure that, after each update, the state of the knowledge base is consistent with a set of formulas that are specified separately from the knowledge base. Unlike what is called *restrictions* in languages such as OWL, database constraints rely on the closed world assumption and are not (and cannot be) used to make additional inferences. (In the Web environment the scope within which the closure of the knowledge is taken should always be made clear.)

Constraint programming. Constraint programming [17] (not to be confused with constraints of the previous paragraph) have been recognized as a very useful problem solving paradigm. We believe that some Web applications (e.g., scheduling, service discovery) could benefit from this feature.

Modularity. Since knowledge on the Web is inherently decentralized, clashes involving naming will be common. It is widely recognized that this problem can be solved through the namespace mechanism. A more subtle kind of a clash arises when different ontologies make different (perhaps compatible, perhaps not) statements about the same concepts or individuals. This problem can be solved through the mechanism of *modules* of the kind that exists in FLORA-2 [31] and TRIPLE [25]. WSML, the specification language of WSMO, has also adopted this concept. Unfortunately, modules have often been confused with namespaces, and there is insufficient awareness of this concept and of the fact that modules and namespaces are orthogonal issues.

8 Summary

In summary, our position is that we should learn from the mistakes made by the logic programming community in the 1980's and from the advancements made in 1990's. These lessons include:

[2] *http://www.cs.umd.edu/projects/hermes/*

1. A rule language for the Web should be feature*full*, not feature*less*.
2. It should be declarative as much as possible.
3. It should support the object-oriented paradigm and meta-reasoning.
4. The language must support triggers, constraints, procedural attachments, and state-changing actions.
5. It must be tolerant to inconsistency and be able to represent uncertain information.
6. It should support not only namespaces, but also modularization a la FLORA-2 and TRIPLE.

References

[1] C. Beeri and R. Ramakrishnan. On the power of magic. *Journal of Logic Programming*, 10:255–300, April 1991.

[2] D. Berardi, H. Boley, B. Grosof, M. Gruninger, R. Hull, M. Kifer, D. Martin, S. McIlraith, J. Su, and S. Tabet. SWSL: Semantic Web Services Language. Technical report, Semantic Web Services Initiative, April 2005. http://www.daml.org/services/swsl/.

[3] T. Berners-Lee. Primer: Getting into RDF & Semantic Web using N3, 2004. http://www.w3.org/2000/10/swap/Primer.html.

[4] A.J. Bonner and M. Kifer. Transaction logic programming. In *Int'l Conference on Logic Programming*, pages 257–282, Budapest, Hungary, June 1993. MIT Press.

[5] A.J. Bonner and M. Kifer. A logic for programming database transactions. In J. Chomicki and G. Saake, editors, *Logics for Databases and Information Systems*, chapter 5, pages 117–166. Kluwer Academic Publishers, March 1998.

[6] A.J. Bonner, M. Kifer, and M. Consens. Database programming in transaction logic. In A. Ohori C. Beeri and D.E. Shasha, editors, *Proceedings of the International Workshop on Database Programming Languages*, Workshops in Computing, pages 309–337. Springer-Verlag, February 1994. Workshop held on Aug 30–Sept 1, 1993, New York City, NY.

[7] W. Chen, M. Kifer, and D.S. Warren. HiLog: A first-order semantics for higher-order logic programming constructs. In *North American Conference on Logic Programming*, Cambridge, MA, October 1989. MIT Press.

[8] W. Chen, M. Kifer, and D.S. Warren. HiLog: A foundation for higher-order logic programming. *Journal of Logic Programming*, 15(3):187–230, February 1993.

[9] H. Davulcu, M. Kifer, and I.V. Ramakrishnan. CTR-S: A logic for specifying contracts in Semantic Web Services. In *13th International World Wide Web Conference (WWW2004)*, May 2004.

[10] J. de Bruijn, H. Lausen, R. Krummenacher, A. Polleres, L. Predoiu, and D. Fensel. The WSML family of representation languages. Technical report, DERI, March 2005. http://www.wsmo.org/TR/d16/d16.1/.

[11] FLORA-2. The FLORA-2 Web site. http://flora.sourceforge.net.

[12] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *Logic Programming: Proceedings of the Fifth Conference and Symposium*, pages 1070–1080, 1988.

[13] M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9(3/4):365–386, 1991.

[14] B.N. Grosof. A courteous compiler from generalized courteous logic programs to ordinary logic programs. Technical Report RC 21472, IBM, July 1999.

[15] P. Hayes and C. Menzel. A semantics for the knowledge interchange format. In *IJCAI Workshop on the IEEE Standard Upper Ontology*, August 2001. http://reliant.teknowledge.com/IJCAI01/HayesMenzel-SKIF-IJCAI2001.pdf.

[16] I. Horrocks, P.F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. W3C Member Submission, May 2004. http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/.

[17] J. Jaffar and J.-L. Lassez. Constraint logic programming. In *ACM Symposium on Principles of Programming Languages*, Munich, W. Germany, 1987.

[18] M. Kifer, G. Lausen, and J. Wu. Logical foundations of object-oriented and frame-based languages. *Journal of ACM*, 42:741–843, July 1995.

[19] M. Kifer and V.S. Subrahmanian. Theory of generalized annotated logic programming and its applications. *Journal of Logic Programming*, 12(4):335–368, April 1992.

[20] O. Lasilla and R.R. Swick (editors). Resource description framework (RDF) model and syntax specification. Technical report, W3C, February 1999. http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/.

[21] James J. Lu, Lawrence J. Henschen, V. S. Subrahmanian, and Newton C. A. da Costa. Reasoning in paraconsistent logics. In *Automated Reasoning: Essays in Honor of Woody Bledsoe*, pages 181–210, 1991.

[22] M. Minsky. A framework for representing knowledge. In J. Haugeland, editor, *Mind design*, pages 95–128. MIT Press, Cambridge, MA, 1981.

[23] RuleML. Rule markup language initiative, 2004. Available at `http://www.ruleml.org`.

[24] K. Sagonas, T. Swift, and D.S. Warren. XSB as an efficient deductive database engine. In *ACM SIGMOD Conference on Management of Data*, pages 442–453, New York, May 1994. ACM.

[25] M. Sintek and S. Decker. TRIPLE – A query, inference, and transformation language for the Semantic Web. In *International Semantic Web Conference (ISWC)*, June 2002.

[26] M.K. Smith, C. Welty, and D.L. McGuinness. OWL Web Ontology Language Guide. W3C Candidate Recommendation, 18 August 2003. Available at `http://www.w3.org/TR/owl-guide/`.

[27] H. Tamaki and T. Sato. OLD resolution with tabulation. In *Int'l Conference on Logic Programming*, pages 84–98, Cambridge, MA, 1986. MIT Press.

[28] J.D. Ullman. Database theory: Past and future. In *ACM Symposium on Principles of Database Systems*, pages 1–10, New York, 1987. ACM.

[29] A. Van Gelder, K.A. Ross, and J.S. Schlipf. The well-founded semantics for general logic programs. In *ACM Symposium on Principles of Database Systems*, pages 221–230, New York, 1988. ACM.

[30] G. Yang and M. Kifer. Reasoning about anonymous resources and meta statements on the Semantic Web. *Journal on Data Semantics, LNCS 2800*, 1:69–98, September 2003.

[31] G. Yang, M. Kifer, and C. Zhao. FLORA-2: A rule-based knowledge representation and inference infrastructure for the Semantic Web. In *International Conference on Ontologies, Databases and Applications of Semantics (ODBASE-2003)*, November 2003.