Editor: Steffen Staab
University of Karlsruhe
sst@aifb.uni-karlsruhe.de

Where Are the Rules?

Web Ontology Language is now the W3C's candidate recommendation,[1] which makes me think that the promises of the Semantic Web will come closer to being realities.[2] Right? A close reading of the famous *Scientific American* article and comparison with OWL reveals, however, that OWL cannot account for rules such as "If a city code is associated with a state code, and an address uses that city code, then that address has the associated state code." OWL doesn't even respond to trivial conversion between measures in different systems (say meters versus feet).

At this point, the famous Semantic Web layer cake Tim Berners-Lee developed comes in handy (www.w3.org/2001/09/06-ecdl/slide17-0.html). Beyond a solid foundation of RDF, RDFS, and OWL, we are missing rule standardization. This, in turn, triggers the question that seems to be paralyzing further standardization—namely, "Which type of rules do we actually need?" Some favorite types include event-condition-action rules (such as triggers in databases), first-order Horn logic axioms, semantic translation rules, and so on.

In this issue, you'll find a range of answers, from "Do we really need rules?" (Ian Horrocks) to "We need all different type of rules at once!" (Gerd Wagner). Because no canonical answer seems to exist, I guess the promises of the Semantic Web will have to wait. But, let's think about what the next step should be.

—Steffen Staab

References

1. D. McGuinness and F. van Harmelen, eds., *OWL Web Ontology Language Overview*, W3C Candidate Recommendation, 18 Aug. 2003; www.w3.org/TR/2003ICR-owl-features-20030818.

2. T. Berners-Lee, J.Hendler, and O. Lassila, "The Semantic Web," *Scientific Am.*, vol. 284, no. 5, May 2001, pp. 35–43; www.sciam.com/article.cfm?articleID=00048144-10D2-IC70-84A9809EC588EF21.

Rules for the Semantic Web

Ian Horrocks, *University of Manchester*

Adding rules to the Semantic Web is certainly a controversial trend. Even establishing a consensus as to what should be understood (in this context) by *rules* has, in my experience, proven extremely difficult. The answer seems to range somewhere between "a subset of first-order predicate calculus" and "a knowledge-based application-development paradigm." Here I'll explain why I subscribe to the former view and what benefits I believe would accrue from this approach.

Using semantic markup

The Semantic Web rests on the fundamental idea that Web resources should be annotated with semantic markup that captures information about their meaning. Establishing standards for semantic markup languages will be crucial in developing the Semantic Web, just as establishing the HTML standard for rendering markup was crucial for developing the current "syntactic Web." W3C standardization efforts addressing this requirement have already produced RDF, RDFS, and OWL—a family of ontology languages designed to represent knowledge and add ontology-based markup to Web resources. (See the sidebar for acronym definitions.)

These languages say nothing, however, about how semantic markup should be used. There does not seem to be any good reason to agree on how applications should exploit the information available on the Web—or even to agree on a common application development environment. Rather, there are good reasons to encourage heterogeneity, as more and varied Semantic Web applications would surely be a "good thing." It therefore seems sensible, at least in the context of the Semantic Web, to focus on using rules as a knowledge representation language or paradigm rather than in application development.

Having established that (Semantic Web) rules should be viewed as a KR language, we can consider whether we need them and, if so, what they should look like and what their relationship to existing Semantic Web KR languages (such as RDFS and OWL) should be.

Let's first consider their relationship to existing languages. Current proposals for a Semantic Web architecture envisage a layering of languages. Great effort has gone into maximizing interoperability (both syntactic and semantic) between the various language layers—in particular, between RDFS and OWL. To maximize both the utility and relevance of rules languages, as well as to avoid fragmenting the Semantic Web, any rules language layer should also integrate well with these existing ontology

1094-7167/03/$17.00 © 2003 IEEE
Published by the IEEE Computer Society

languages. We can best achieve this by giving it a first-order style semantics compatible with those of RDF and OWL (and not the minimal Herbrand model semantics often employed by rules languages). That is, a rule of the form $A(x) \leftarrow B(x)$ should be interpreted as equivalent to the first-order predicate calculus (FOPC) sentence $\forall x$ $B(x) \rightarrow A(x)$. Not only would this maximize compatibility with existing ontology languages, but it would also facilitate further compatible extensions up to and including full FOPC.

Under such semantics, we can view rules simply as another subset of FOPC—in fact, one closely corresponding with Horn logic. We can easily show that such a language has a substantial overlap with OWL-DL[1] and that it includes the whole "first-order subset" of RDFS.[1,2] Classes and properties would then be equivalent to unary and binary predicates, respectively, and a rule of the form $A(x) \leftarrow B(x)$ would obviously be equivalent to the RDFS/OWL-DL axiom SubClassOf(A B).[3] We can also transform more complex rules containing conjunctions and negations of unary predicates, as well as some rules containing binary predicates, into equivalent OWL-DL axioms. For example, the rule $D(y) \leftarrow C(x) \wedge R(x, y)$ is equivalent to the OWL-DL axiom

```
SubClassOf(C restriction
(R allValuesFrom(D))).
```

What rules should look like is clearly of far less importance than their meaning. An XML-based syntax as specified in RuleML would be a perfectly reasonable choice. An RDF-based syntax would also be possible, although as with OWL this would no doubt prove rather cumbersome and lead to similar problems if we sought correspondence between the semantics of the rules themselves and of the RDF triples used to encode their syntax.[4]

Do we need a rules language?

Having established that the semantics of rules should be that of Horn logic and that their syntax is relatively unimportant, let's now look at the first question I raised: Does the Semantic Web need a rules language at all? As we have seen, we can easily transform many rules into OWL-DL axioms, so perhaps we should simply view rules as an alternative syntax for OWL-DL—one that we could easily implement in suitable user interfaces. While this approach would be useful in helping those familiar with rules to use languages like OWL-DL, rules clearly capture information that we *cannot* represent in OWL-DL.

Even apart from their obvious ability to deal with higher arity predicates, rules provide more expressive power with respect to binary predicates (properties)—for example, allowing one property to be inferred from a composition of others. A well-known example is the assertion that the composition of "parent" and "brother" should imply "uncle"—that is, uncle(x, z) $\leftarrow$ parent(x,y) $\wedge$ brother(y, z), a relationship that can't be captured using OWL. This kind of relationship between properties (binary predicates) is quite common and would certainly be useful in applications as varied as medical terminologies and Web service descriptions. Unfortunately, adding this kind of rule to OWL-DL would lead to the undecidability of key inference problems, such as consistency (although recent research has shown how we can extend DL in this direction while still retaining decidability[5]).

To sum up, if the Semantic Web needs rules at all, it needs them as a KR language that is semantically compatible with existing ontology languages. Although OWL-DL can already capture many kinds of rule (and vice versa), a full Horn clause rules language would add useful expressive power. The combination of Horn rules and OWL-DL leads to undecidability, but we might be willing to pay this price in some applications. Although undecidable, the combined language might still exhibit better computational behavior than full FOPC.

Common Terms

CLIPS	C Language Integrated Production System
DAML+OIL	DARPA Agent Markup Language (ontology inference layer)
DL	description logic
FOL	first-order logic
JESS	Java Expert Systems Shell
KR	knowledge representation
OWL	Web Ontology Language
RDF	Resource Description Framework
RDFS	RDF Schema
SQL	Structured Query Language

References

1. B.N. Grosof et al., "Description Logic Programs: Combining Logic Programs with Description Logic," *Proc. 12th Int'l World Wide Web Conf.* (WWW 2003), ACM Press, 2003, pp. 48–57; http://www2003.org/cdrom/papers/refereed/p117/p117-grosof.html.
2. I. Horrocks and P.F. Patel-Schneider, "Three Theses of Representation in the Semantic Web," *Proc. 12th Int'l World Wide Web Conf.* (WWW 2003), ACM Press, 2003, pp. 39–47; http://www2003.org/cdrom/papers/refereed/p050/p50-horrocks.html.
3. P.F. Patel-Schneider, P. Hayes, and I. Horrocks, "OWL Web Ontology Language Semantics and Abstract Syntax," Mar. 2003; www.w3.org/TR/owl-semantics.
4. P.F. Patel-Schneider and D. Fensel, "Layering the Semantic Web: Problems and Directions," *Proc. 1st Int'l Semantic Web Conf.*, Springer-Verlag, 2002, pp. 16–29.
5. I. Horrocks and U. Sattler, "Decidability of SHIQ with Complex Role Inclusion Axioms," *Proc. 18th Int'l Joint Conf. Artificial Intelligence* (IJCAI 03), AAAI Press, 2003, pp. 343–348.

It Rules!

Jürgen Angele, *Ontoprise GmbH*

The World Wide Web is the largest knowledge base ever built; you can find all kinds of information on it. Making even parts of the Web interpretable by computers would drastically revolutionize current business processes and even create new business models. This is the goal of the next-generation Web: the Semantic Web.

To make information interpretable by computers, an appropriate KR formalism must represent it. (See the sidebar for a definition of this and other acronyms.) For this purpose, so-called ontologies represent a special domain's knowledge. Emerging standards for ontology languages are RDF and OWL. RDFS is a simple, conceptual language that supports classes with binary relationships to other classes, hierarchies of

classes and relationships with attribute inheritance, and instances of classes. OWL adds DL primitives to RDFS.[1]

In all our ontology projects, numerous instances play a crucial role. In contrast to the schema information ontologies provide, instances represent the real content of information. This is similar to databases where the schema only provides information about the data's structure and constraints, but the content of tables in the relational case gives the important information. Most domains require relations between objects that are much more complex than simple binary relations can express. In the Halo project (www.projecthalo.com), my colleagues and I recently developed an ontology for chemistry that represents information such as "If, in an aqueous solution, ion A and ion B recombine to a new substance C, and C isn't soluble in water, then the chemical reaction looks like A + B → "C." The basis for this complex relation between A, B, and C is numerous different chemical substances and ions and their properties. So far, it's impossible to represent such complex relations in OWL, and we don't foresee being able to infer such complex relations using OWL extensions with a reasonable performance. On the other hand, rules might easily express complex relations such as the example just given. For example, our Halo ontology contains some 400 rules about such chemical interactions, and there are evaluation engines such as XSB (http://xsb.sourceforge.net) or OntoBroker[2] that very efficiently evaluate rules.

What properties does such a rule language require? It must have a clear and declarative semantics. Experience in expert systems shows that production rule systems have obscure behavior and so are hard to maintain. Predicate logics provide a good basis for such a rule language. While Prolog is an often-cited basis for predicate logics, primitives such as assert, cut, fail, and so forth make Prolog more like a production rule system than a declarative KR language.

The rule language's expressive power must be sufficient. The language has to support function symbols and negation. For an object-oriented representation function in particular, symbols are mandatory because of their ability to give object identities a logical semantics.

A rule language must seamlessly integrate into the OO paradigm provided by RDFS or OWL. This means that there must be an OO syntax to access the concepts and their relationships and instances and their relations. The syntax of pure predicate logic syntax is inadequate because flat predicates alone represent all relations.

For a rule language, constructing high-performance reasoning engines must be feasible given the state-of-the-art knowledge about evaluating ontologies with rules. So, reasoning's performance on numerous instances is really crucial because, as I already mentioned, it's the most frequent necessity in enterprise-scale, ontology-based applications such as Web services.

Some languages and systems that satisfy these requirements are already available. The first system to integrate RDFS and rules was SILRI (Simple Logic-based RDF Interpreter). The SILRI engine adds F-logic rules[3] to RDFS ontologies. F-logic has an OO syntax and is based on a declarative semantics. OntoBroker and FLORA[4] are examples of efficient F-logic implementations. TRIPLE is a recent variant of F-logic that has simpler syntax but still satisfies the mentioned requirements.[5]

A combination of OWL and rules is in discussion. Most primitives of OWL are translatable to a Horn rule-based representation.[6] In ontologies translated from OWL to Horn, logic reasoners based on Horn logics are much faster than OWL reasoners on the original model.[6] On the other hand, some primitives remain based on subsumption in OWL that might be important for tasks—such as ontology aligning and ontology integration—that can't easily be translated into Horn logics. The challenge for standardizing an ontology language is to combine the advantages of both paradigms with a common understandable syntax and a clear declarative semantics.

References

1. F. Baader et al., *The Description Logic Handbook*, Cambridge Univ. Press, 2001.
2. S. Decker et al., "OntoBroker: Ontology Based Access to Distributed and Semi-Structured Information," *Database Semantics: Semantic Issues in Multimedia Systems*, Kluwer Academic, 1998, pp. 351–369.
3. M. Kifer, G. Lausen, and J. Wu, "Logical Foundations of Object-Oriented and Frame-Based Languages," *J. ACM*, vol. 42, no. 4, July 1995, pp. 741–843.
4. G. Yang and M. Kifer, "FLORA: Implementing an Efficient DOOD System Using a Tabling Logic Engine," *Computational Logic—CL 2000*, LNAI 1861, Springer-Verlag, 2000, pp. 1078–1093.
5. S. Decker and M. Sintek, "TRIPLE—A Query, Inference, and Transformation Language for the Semantic Web," *The Semantic Web— ISWC 2002*, LNCS 2,342, Springer-Verlag, 2002, pp. 364-378.
6. B.N. Grosof et al., "Description Logic Programs: Combining Logic Programs with Description Logic," *Proc. 12th Int'l World Wide Web Conf.* (WWW 03), Int'l World Wide Web Conf. Committee, 2003, pp. 48–57.

Semantic Rules

Stefan Decker, *Information Sciences Institute*

Motivations

Not surprisingly, everyone perceives the Semantic Web in a different way. One view is that it's about semantics, and semantics is about KR. (See the sidebar for a definition of this and other acronyms.) So, you could believe that all that's required to make the Semantic Web a reality is rehashing the existing work on KR and providing it with a fashionable syntax (XML suits it just fine), and—voilà—the Semantic Web is born. The problem with this view is that it's hard to see why everybody should suddenly start doing KR.

Another view—maybe a more practical one—is that the Semantic Web is about overcoming the syntax of data so that users and developers can concentrate on the semantics of information. This means that languages and tools for the Semantic Web must be oriented on practical problems rather than generic KR tasks. That is, they should make it easier and cheaper to publish, understand, use, and reuse data and services on the Web in an interoperable, scalable way. Languages that help define how different data sets and vocabularies relate to each other are necessary; they provide the glue between (distributed) information systems and data sets.

This view also has consequences for designing rule languages for the Semantic Web: no magic bullet exists for driving down cost to establish interoperability. Currently it's still an art that usually involves a programmer and a lot of time. Although establishing interoperability by writing code works fine on a small scale (for example, to establish interoperability between two companies), it doesn't scale to the Semantic Web, which requires dynamic interoperation among many information providers.

To attack that problem, writing rules to

establish interoperation is usually faster and cheaper than writing program code (for example, in Java) because rules aren't burdened with the details of general programming languages. Rules provide benefits over a software product's life cycle; they are simpler to write than code, more concise, and easier to understand, share, and maintain.

Standardizing such a rule language has several benefits. The standardization makes it feasible for consumers as well as vendors of rule engines to invest into the infrastructure. It also enables competition to drive innovation because you can compare different rule engines implementing the same standard. And last but not least, it simplifies interoperation by rule sharing. Because rule engine users can now exchange the rules on how to achieve interoperability, they can also share in the development (and thus cost) of interoperation rules.

Implications

I can derive several requirements and implications from the scenario that a rule language should serve as a data transformation and glue language.

A rule language for mission-critical tasks needs a defined semantics (as a basis for implementation) and efficient evaluation mechanisms. Deductive databases and logic programming provide a solid, application-driven, performance-oriented background and numerous techniques and mechanisms.

A data transformation rule language must be based on the lowest common denominator. In the Semantic Web's case it's RDF, which means a rule language has to support RDF querying, derivation, transformation, and generation. Support for RDF implies more requirements, such as support for namespaces, XML querying, and the ability to reason with distributed data and thus contexts in which the data appears.

Expressing declarative statements about how data must be transformed requires KR mechanisms, because defining these rules is essentially capturing knowledge about how to achieve interoperability. The logic programming community has investigated these mechanisms. Applicable mechanisms include nonmonotonic negation,[1] explicit negation,[2] Lloyd-Topor transformation,[3] object orientation (F-logic,[4] for example), and preference mechanisms. The need to apply KR principles also implies that SQL isn't an ideal basis for such a query and rule language: a KR language must be understandable to not only computers but also humans. It should be have a concise, appealing syntax. Any SQL-based syntax for RDF rules makes it cumbersome to express all but the most trivial relationships and dependencies between data items.

Different modeling languages are already or will be defined on top of RDF, such as OWL, RDF-S, UML, and Topic Maps. A rule language must be able to query and reason not only with plain RDF but also with the semantics of modeling languages. Because a fixed set of modeling languages doesn't exist (and new modeling languages are invented every day), a rule language must be able to cope flexibly with the semantics of modeling languages. An RDF query and rule language lacking this ability needs a language-specific query and inference system for each modeling, shifting the effort from integrating data to integrating query and inference systems.

Realization

Realizing the rule language I've outlined requires different efforts. A working group must define the language's syntax and semantics in such a way that convinces open-source and commercial developers to invest their time and money. My colleagues and I have worked toward such a language by developing TRIPLE (an RDF query, interface, and transformation language),[5] which fulfills many of the requirements mentioned earlier. More developments are necessary, and efforts such as FORUM (http://forum.semanticweb.org) can be the next step and deliver a crucial piece of infrastructure for the Semantic Web.

Usage

The existence of a fairly easy, deployable, and usable rule language implementation will likely have an effect comparable to the first Web browser. It offered immediate gratification by delivering nicely formatted HTML pages created for the Web, providing an incentive to create pages others could view. By creating and exploiting a network effect, the rule language could potentially get people to use and integrate Web data in their applications, which in turn provides an incentive for information providers to create and publish data.

References

1. G. Brewka and J. Dix, "Knowledge Representation with Logic Programs," *Logic Programming and Knowledge Representation*, LNAI 1471, Springer-Verlag, 1998, pp. 1–51.
2. C. Baral and M. Gefond, "Logic Programming and Knowledge Representation," *J. Logic Programming*, vols. 19–20, 1994, pp. 73–148.
3. J.W. Lloyd and R.W Topor, "Making Prolog More Expressive," *J. Logic Programming*, vol. 3, 1984, pp. 225–240.
4. M. Kifer, G. Lausen, and J. Wu, "Logical Foundations of Object-Oriented and Frame-Based Languages," *J. ACM*, vol. 42, no. 4, July 1995, pp. 741–843.
5. M. Sintek and S. Decker, "TRIPLE—A Query, Inference, and Transformation Language for the Semantic Web," *The Semantic Web–ISWC 2002*, LNCS 2342, Springer-Verlag, pp. 364–378.

The Semantic Web Needs a Logic for Rules and Objects

Michael Kifer, *Stony Brook University*

OWL specifies basic semantic information in a standardized, machine-processible way. (See the sidebar for a definition of this and other acronyms.) However, the key here is the word "basic." OWL is mainly the language of facts and constraints, and the main type of inference it supports is what is provided by DL—namely, subsumption. If the Semantic Web were all about subsumption, then I would say that it is much ado about nothing. For the Web to exhibit intelligence that draws on the semantic information encoded in ontologies, more powerful declarative languages are needed. Otherwise, most of the so-called knowledge will stay encoded in Java or C programs, and the would-be promise of the Semantic Web will be wasted.

To realize the Semantic Web's full potential, we need another semantic layer, which, in my mind, should be a rule language. What kind of rule language? Triggers? Forward-chaining systems based on the RETE algorithm, such as OPS5 or CLIPS? I don't think so. The logical semantics of these languages is suspect. Even operationally, forward-firing rules are hard to control, and this paradigm is not readily amenable to modular, top-down program development. Although these types of rules are widely used, the reason, I believe, has more to do with their wide availability, easy implementation, and good integration with Java.

My own analysis of several applications built using forward-firing systems shows that programmers often try to simulate Prolog-style top-down inference through ingenious techniques for controlling rule firing—unfortunately, at the expense of clarity. I remember seeing a sample program in an ancient (mid 80s) CLIPS manual, which contained (for that particular example) the main elements of the well-known Magic Sets method.[1] The author tried to show readers how to efficiently solve the problem at hand. For all I know, this example might be older than the first incarnation of the Magic Sets method. This imitation of the workings of Prolog is a good example of the effort that might be required to make productive use of forward-firing systems.

If programmers of the forward-firing systems are trying to imitate Prolog, why not use Prolog in the first place? "Pure" Prolog is firmly based on FOL and provides a solid paradigm for top-down program development. However, several key problems were responsible for Prolog's lack of success, including these:

- The control structure in Prolog programs is only marginally better than that of OPS5-like systems. At the same time, Prolog's language is more complex, and, to take advantage of its power, you must resort to numerous nonlogical features.
- Prolog lacks native support for complex objects and object-oriented programming. The support for metaprogramming (an important requirement on the Web) is powerful but ad hoc and nonlogical.

Fortunately, significant progress has been made on both of these issues. First, cross-pollination between the research in deductive databases and logic programming gave birth to the Magic Sets method and, perhaps more important, to its top-down counterpart, the tabled evaluation techniques for logic programs.[2] I believe that the invention of the tabling method was more important, because it offered a way to capitalize on the vast experience accumulated by the implementers of WAM-based Prolog engines. XSB (http://xsb.sourceforge.net) was the first, most complete, and best-known system of this kind. This system largely solves the first problem, as it enables declarative programming using logic without undue worry about the operational semantics of the underlying engine.

The development of F-logic solved the second problem. F-logic extends classical predicate logic with complex objects and most of the typical elements of the OO paradigm.[3] F-logic is related to logic-based OO modeling and programming in the same way that predicate calculus is to standard logic programming. Recent systems such as FLORA-2 (http://flora.sourceforge. net), Triple (http://triple.semanticweb.org), and the commercial OntoBroker (http://www.ontoprise.de) attest to the viability of F-logic-based languages.

These solutions make Prolog-style rule languages more accessible to the average user. In the future, I envision that this access will be mediated by a variety of tools that would target different categories of users: naive users (who will use only simple rules), nonprogrammer knowledge engineers (who will be exposed to simple rules as well as to F-logic's OO features), and, finally, programmers (who would have full access to the rule language). Eventually, various extensions will be added to this rule language, such as those needed to support contradictory and probabilistic knowledge. This vision is consistent with the ongoing effort around RuleML (www.ruleml.org), which tries to develop a single, extensible XML-based syntactic framework for capturing different varieties of rules.

References

1. F. Bancilhon et al., "Magic Sets and Other Strange Ways to Implement Logic Programs," *Proc. 5th ACM SIGACT-SIGMOD Symp. Principles of Database Systems*, ACM Press, 1986, pp. 1–16.

2. H. Tamaki and T. Sato, "OLD Resolution with Tabulation," *Proc. 3rd Int'l Conf. Logic Programming* (ICLP 86), LNCS 225, Springer-Verlag, pp. 84–98.

3. M. Kifer, G. Lausen, and J. Wu, "Logical Foundations of Object-Oriented and Frame-Based Languages," *J. ACM*, vol. 42, no. 4, July 1995, pp. 741–843.

Rules and RuleML in the Semantic Web

Benjamin Grosof, *MIT Sloan School of Management*

Over the last two and a half years, the Semantic Web community has reached a broad consensus that the Semantic Web vision includes not only ontologies in the manner of W3C's OWL, but also rules. (See the sidebar for a definition of this and other acronyms.)

The leading approach to Semantic Web rules

RuleML, an XML markup language for rules based on declarative logic programs, has emerged as the leading standardization approach to rules for the Semantic Web. I cofounded and cochair the RuleML Initiative, which aims to enable Web-based interoperability between heterogeneous rule systems and applications. I also cochair the closely related DAML Rules effort, which is currently the main focus of the Joint Committee that developed DAML+OIL, OWL's close predecessor.

Declarative KR

Standards should be founded on techniques that are well established at a research level. The only approach to making the Semantic Web be "semantic" that is well understood—and well accepted—from a substantial body of previous research is to found it on *declarative KR*. RuleML, like OWL and RDF, shares this approach. Each of these languages starts with an underlying declarative KR, which has an associated semantics, then adds *Webized* syntax.

I mean declarative here in the sense of (AI) KR theory. A given set of premises entails a set of sanctioned conclusions, independent of inferencing control strategy or procedural aspects. For example, those sanctioned conclusions are independent of whether the inferencing direction is backward (goal-directed query answering) or forward (data driven). Employing declarative KR greatly facilitates reuse, and multiple kinds of uses, of knowledge. It lets a knowledge-based application (that is, agent) anticipate precisely and completely the meaning another agent will draw from the communicated (or shared) knowledge. In particular, declarative rules, as compared to general-purpose program code, provide a relatively high level of conceptual abstraction that helps nonprogrammers understand, specify, and dynamically modify and merge them. They are executable, but you can manage them as data, separate from code.

Webized here means using the Web's overall open spirit and its standards suite—particularly XML, URIs (uniform resource identifiers), and namespaces—to support multiauthored, widely distributed knowledge modules. RuleML initially empha-

sized XML as the syntax form to facilitate building tools for translation and inferencing. RuleML has since added an RDF syntax and a human-oriented string syntax (as well as an abstract syntax that bridges all these). However, for rules, as for ontologies, the most crucial design choice is that of underlying fundamental KR.

Commercially important rule systems

Four families of rule systems are the most currently commercially important (CCI) ones: SQL relational databases, Prolog, production rules—such as OPS5, CLIPS, and JESS—and event-condition-action (ECA) rules. You'll often find these kinds of rules embedded in systems built using object-oriented programming languages (such as C++ and Java). These rule systems are often used for business process connectors or workflow, and they've achieved growing commercial presence since the 1980s, with manifold diverse e-business applications today. Hundreds of thousands of developers and millions of IT users are familiar with one or more of these CCI families—especially SQL and, to a lesser extent, Prolog, including through academic training. (Likewise, many are familiar with Horn FOL.)

Declarative logic programs as shared KR

RuleML takes as a prime requirement that a Semantic Web rules language must support interoperability among rule-using applications (agents) that use heterogeneous members of these CCI rule systems. A key observation underpinning RuleML's design rationale is that these four families all have a common core abstraction: the declarative *logic programs* (LP) KR. RuleML has, accordingly, started with this KR.

In particular, the Datalog Horn case of LP is RuleML's kernel expressiveness. A relatively simple core that all four CCI families share, this case is the most well-studied LP subset and is logically monotonic. It's a subset of classical FOL (and a moderate weakening of Datalog Horn FOL: conclusions are essentially restricted to ground atoms). Datalog Horn LP is also the heart of SQL—that is, of relational algebra. It is computationally tractable (polynomial-time inferencing, given a bounded number of logical variables per rule), which enables practical scalability and inferential completeness. Datalog here means that logical functions (beyond zero arity) are prohibited.

Nonmonotonicity and procedural attachments

To this kernel, RuleML evolutionarily adds a family of extensions for various expressive features and restrictions—according to established research and driven by applications needs. Extensions that enable nonmonotonicity and procedural attachments are especially important. *Negation-as-failure* is a nonmonotonic feature that exists in all the CCI families and is heavily used in all of them except SQL. LP with NAF ("ordinary" or "normal" LP) has been well studied. I have proposed the design of two other major features, both heavily used in all the CCI families, which the RuleML Initiative is currently discussing. The first is the nonmonotonic feature of *prioritized conflict handling*. Examples include priority between rules in Prolog based on static rule sequence, dynamically computed priorities among rules in production rule and ECA rule systems, inheritance with exceptions, and updating in databases (where more recent assertions override previous ones).

The second feature is procedural attachments, to perform *actions* triggered by drawing conclusions and to perform *queries* when testing rule antecedent conditions. The *Situated Courteous* extension of LP (SCLP),[1] which I have developed, includes all three features (NAF, prioritized conflict handling, procedural attachments). It's declarative and preserves tractability. However, it's not as well studied as ordinary LP. RuleML also defines several other expressive extensions and expressive restrictions.

Classical logic is not enough

The first major attempt at standardizing a declarative KR for knowledge interchange was the Knowledge Interchange Format. Used primarily for research systems, KIF basically preceded the Web. KIF and its close successor (Simple) CommonLogic—an early phase standards effort—are based on the underlying KR of FOL, which is popular among mathematicians. However, FOL fundamentally lacks the ability to express nonmonotonicity or procedural attachments, and (beyond LP) has thus not become widely deployed for commercial applications—partly owing also to its intractability.

Supporting LP's overlap with classical logic

LP and FOL overlap, but not completely. So do LP and OWL's underlying DL KR, which is a subset of FOL. It's important to support non-LP FOL (for example, material implications between complex formulas) as an additional direction of expressiveness for rules. For example, this is useful in the Description Logic Programs approach[2] that my colleagues and I have developed to combine the semantics of LP and DL, where RuleML LP rules refer to or import ontological knowledge from OWL DL (for example, class definitions or property predicates). The Lloyd-Topor transformation provides a rich expressive LP extension that is tractably reducible to ordinary LP.[3] Limited classical ("strong") negation is another such extension.[1] CommonLogic and LP RuleML can and should share syntax as well as semantics to a great extent. RuleML has led the development of such Webized syntax. Indeed, it would be reasonable to treat, and develop, CommonLogic as part of the RuleML family.

Challenges and exciting opportunities in services

Rules promise to be useful in various new Semantic Web services applications—for example, our work on e-contracts, financial knowledge integration, and travel packages. Much more than ontologies, rules can actually do stuff. However, achieving the full vision of Semantic Web rules will require more research. We must further understand how LP relates to DL and FOL and how to reconceive rule KR as essentially highly distributed, especially when featuring nonmonotonicity, procedural attachments, and events. We need more KRs closely related to rules—notably, probabilistic, inductive, and constraint-based—for the full Semantic Web vision, yet few have addressed these in standardization efforts.

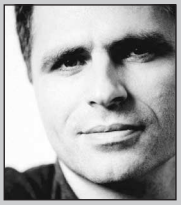
References

1. B. Grosof, Y. Labrou, and H. Chan, "A Declarative Approach to Business Rules in Contracts: Courteous Logic Programs in XML," *Proc. 1st ACM Conf. Electronic Commerce* (EC 99), ACM Press, 1999.

2. B. Grosof et al., "Description Logic Programs: Combining Logic Programs with Description Logic," *Proc. 12th Int'l World Wide Web Conf.* (WWW 2003), ACM Press, 2003.

3. J. Lloyd, *Foundations of Logic Programming*, Springer-Verlag, 1987.

Ian Horrocks is a professor of computer science at the University of Manchester. Contact him at horrocks@cs.man.ac.uk or www.cs.man.ac.uk/~horrocks.



Jürgen Angele is the chief executive officer of Ontoprise GmbH, a semantic technologies provider. Contact him at angele@ontoprise.de.

Stefan Decker is senior research fellow and research cluster leader of the Semantic Web Cluster at the Digital Enterprise Research Institute, National University of Ireland, Galway and a research assistant professor at the University of Southern California Information Sciences Institute. Contact him at stefan@isi.edu or stefan@deri.ie.

Michael Kifer is a professor with the department of computer science, State University of New York at Stony Brook. He has a PhD in Computer Science from the Hebrew University of Jersualem. His research interests include database systems, knowledge representation, and Web information systems. Contact him at kifer@cs.stonybrook.edu.

Benjamin Grosof is a Douglas Drane assistant professor in information technology at the MIT Sloan School of Management. Contact him at bgrosof@mit.edu.

Gerd Wagner is an assistant professor at the faculty of technology management, Eindhoven University of Technology, The Netherlands. He received a PhD in philosophy from the Free University of Berlin. His research interests include rule-based systems, business rules, foundational ontologies, agent-oriented modeling, and agent-based simulation. Contact him at g.wagner@tm.tue.nl; http:tmitwww.tm.tue.nl/staff/gwagner.

Seven Golden Rules for a Web Rule Language

Gerd Wagner, *Eindhoven University of Technology*

Yes, we do need a Web rule language. Rules and rule markup languages such as RuleML will play an important role in the Semantic Web's success. Rule expressions will be used in Web applications for defining derived terms based on a taxonomy, specifying validation constraints, representing organizational policies and business rules, specifying a software agent's behavior, and many other purposes (see http://tmitwww.tm.tue.nl/staff/gwagner/myRuleML/What_is_a_rule.html). Rule markup languages will be the vehicle for using rules on the Web. They will let users deploy, execute, publish, and communicate rules, and also serve as a lingua franca for exchanging rules between different systems and tools.

In a narrow sense, a Web rule language is a concrete (XML-based) rule syntax. In a broader sense, it should have an abstract syntax as a common basis for defining various concrete sublanguages serving different purposes.

RuleML, in its current version 0.8, is insufficient as a general Web rule language. However, with some syntactic simplifications, it is a good starting point.

A Web rule language should have a formal semantics. However, there might be language constructs, which don't have a formal semantics based on classical FOL but are needed to deal with certain practical problems. We should avoid two dangers related to this trade-off:

- Adopting practical language constructs (such as procedural attachments), that seem to be important and have some intuitive (but not formal) semantics, even though alternatives with formal (but nonstandard) semantics exist
- Paying too much attention to theoretical issues of standard FOL (see the sidebar for a definition of this and other acronyms), such as computational (asymptotic worst- case) complexity, decidability, and compactness

A Web rule language standardization effort should pay special attention to the concerns of the users of SQL and Prolog and to production rules (CLIPS, JESS, ILOG, and so on). It must let these users map their language's most important constructs. My general advice about such an effort is summarized in seven golden rules (GRs).

GR 1: Relational databases are more important than FOL

Many KR formalisms strictly (or blindly?) follow classical FOL and ignore the nonclassical inference features and rule concepts, which have proved essential in relational databases such as three-valued connectives, nonmonotonic queries, and (state-changing) trigger rules. A Web rule language cannot afford to ignore these fundamental information-processing concepts, which require abandoning classical logic.

GR 2: UML is more important than OWL

UML represents a larger body of information, knowledge-modeling experience, and expertise than OWL does. UML includes an expressive language for integrity constraints. These constraint expressions also form a kind of rule (an integrity rule) and should be covered by a Web rule language. Remarkably, UML also provides more support for advanced ontological constructs. For example, it supports part-whole relationships (with aggregation and composition) and *powertypes* as classes whose instances are subclasses of another class (BiologicalSpecies and PassengerAircraft are examples of powertypes).

GR 3: Rules are not implications

Although an implication is an expression of a logical formula language, typically possessing a truth value, a derivation rule does not possess a truth value; instead, it generates derived sentences. Logics exist that don't have an implication connective

but do have a derivation rule concept. In standard logics (such as classical and intuitionistic logic), a close relationship exists between a derivation rule (also called a *sequent*) and the corresponding implicational formula—they both have the same models. For nonmonotonic rules (for example, with negation-as-failure), this is no longer the case. The intended models of such a rule are, in general, not the same as the intended models of the corresponding implication.

GR 4: Web rules are not just Horn clauses

This golden rule is a corollary of GR 3. Because Horn clauses are a limited type of implication, and rules are not implications (according to GR 3), it follows that Web rules are not just Horn clauses. Web rules are rule expressions used in Web documents and in Web applications. They must be much more expressive than Horn clauses (see http://lists.w3.org/Archives/Public/www-rdf-rules/2001Sep/0079.html).

GR 5: Web rules should be able to express database rules

In Web applications, we should expect similar uses of rules as in databases. This consideration suggests that a Web rule language must accommodate

- *SQL assertions*: integrity rules
- *SQL views*: nonmonotonic derivation rules with three-valued connectives and open and closed predicates
- *SQL triggers*: reaction rules, which are limited to update events

GR 6: A Web rule language should let users express and implement business rules

Business rules refer to the hundreds, if not thousands, of policies, procedures, and definitions that govern how a company operates and interacts with its customers and partners. The literature has identified three basic types of business rules:[1]

- *Integrity rules*: assertions that must hold in all evolving states and state transition histories of an enterprise viewed as a discrete, dynamic system. Example: "The driver of a rental car must be at least 25 years old."
- *Derivation rules*: statements of knowledge that is derived from other knowledge by an inference or a mathematical calculation. Example: "A gold customer is a customer with more than $1 million on deposit."
- *Reaction rules*: expressions of policies specifying actions in response to events. Example: "When a share price drops by more than five percent and the investment is exempt from profit tax, then sell it."

GR 7: A Web rule language should allow for multiple purposes, multiple languages, and multiple semantics

The Web is a pluralistic world, no matter if it is semantic or not. There will be multiple purposes, multiple languages, and multiple semantics for Web rules. The real challenge is to dvelop an integrated metamodel, or abstract syntax, that supports this plurality. ❒

Reference

1. K. Taveter and G. Wagner, "Agent-Oriented Enterprise Modeling Based on Business Rules," *Proc. 20th Int'l Conf. Conceptual Modeling* (ER 2001), LNCS 2224, Springer-Verlag, 2001, pp. 527–540.

IEEE Intelligent Systems

Advertiser/Product Index
September/October 2003

	Page No.
Singapore Management University	13

NEXT ISSUE

November/December 2003

Agents & Markets

Advertising Sales Offices

Sandy Brown
10662 Los Vaqueros Circle, Los Alamitos, CA 90720-1314; phone +1 714 821 8380; fax +1 714 821 4010; sbrown@computer.org.

Advertising Contact: Marian Anderson, 10662 Los Vaqueros Circle, Los Alamitos, CA 90720-1314; phone +1 714 821 8380; fax +1 714 821 4010; manderson@computer.org.

For production information, and conference and classified advertising, contact Marian Anderson, *IEEE Intelligent Systems*, 10662 Los Vaqueros Circle, Los Alamitos, CA 90720-1314; phone +1 714 821 8380; fax +1 714 821 4010; manderson@computer.org; http://computer.org.