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

Ontologies' KISSES in Standardization

> The Semantic Web is about increasingly machine-readable Web content and other data. The underlying technologies are not explicitly new, but the upcoming scale and scope of deployment offers great potential—as well as challenges.
>
> —*Alexander Linden*[1]

> Lack of interoperability threatens the future of the Internet. The Internet will extend far beyond people using computers and handheld gadgets. By 2007, billions of devices and bits of software will sense and act on the physical world—no humans required. The result: Everything from tiny sensors to mammoth business apps will exchange data. But products won't interoperate if they have incompatible data formats. This will hurt manufacturers and customers, and jeopardize the future of the Internet.[2]

With the Semantic Web proliferating through support from academic funding agencies and with businesses now hopping onto the train,[1] you can imagine the excitement that nourishes Semantic Web researchers and entrepreneurs alike.

Before the research and deployment of exciting, high-impact technologies, there comes standardization.[2] The resource description framework and RDF Schema constitute a data model and a typing system, respectively, and are in a near-standard status. However, a Web ontology language that concisely specifies terms is currently under heavy controversy.

The setting

The World Wide Web Consortium puts working groups in place for recommending standards. The Web Ontology Working Group (WOWG) is composed of people from W3C member institutes as well as invited experts. The goal of the W3C WebOnt Working Group (www.w3.org/2001/sw/WebOnt) is a difficult one. The collection of use cases implicating the yet-to-be-defined Web ontology language turned up many of those properties that have motivated AI and database research over several decades. Therefore, the working group will inevitably have to adhere to the common 90–10 rule: you can do 90 percent of the work with 10 percent of applicable mechanisms, but the remaining 10 percent of benefits requires a 90 percent effort. This also entails that a corresponding standard should be boringly simple.

Analogous cases

Analogous cases for such a standardization process abound, although probably none of them were as relevant for intelligent systems as the Web ontology language. XML (Extensible Markup Language) standardization, for example, is a famous success story. (For a list of abbreviations, see the "Glossary" sidebar.) Simplifying SGML in favor of an easily understandable and implementable XML model, together with support from W3C and major software vendors, turned a small SGML community into a huge market for research on and software support for XML. (In the model, simplicity is measured by the number of pages of standardization documents.)

In contrast, many other standardization efforts have failed. For instance, workflow management has had problems with standardization—to the detriment of research significance and software products alike. The Business Process Markup Language might be successful, but this has yet to be determined.

Standardization aims to resolve controversy. A Web ontology language is a tool that many people want to use for various purposes. It's like a vehicle—there's not one right vehicle for transporting things. When moving, you want a truck; when in a hurry, a Porsche would do; to cheaply get to work, you might take the bus. In terms of standards, you might not fancy the big, fast, or cheap; you want something that lives up to the 90–10 rule of offering benefits in most cases.

Seven dimensions for a Web ontology language

So, what are the minimal requirements for our vehicle? Depending on whom you ask, the answer will vary along the following dimensions:

- *Layering*: Should we end up with one ontology language or a hierarchy of languages with clear commitments and boundaries between them? Typically, layering would also correspond to efficiency and boundaries of classes of decidability.
- *Subsumption*: Discussing efficiency, is subsumption reasoning necessary, or should we just explicitly specify taxonomies?
- *Size*: Should we include a minimum or maximum set of language features?
- *Research*: Do you want to be at the cutting edge of research or have something more mature for your Web ontology language?
- *Outlook*: The WOWG's charter is to explicitly delegate rules and queries to another, later working group. However, to what extent must we foresee a rule and query language that integrate rules and queries in a later standard?
- *Social*: Some of the success factors for standards depend not so much on the technicalities but rather on the social factors. How many people can quickly provide tools for the Web ontology language? Given their daily business constraints, how many people can actually use the Web ontology language?
- *Business*: What is the expected benefit, and how much time (in terms of

1094-7167/02/$17.00 © 2002 IEEE

training and implementation) will it take to achieve it?

Contributions

Eight people from different communities, including applications, description logics, and frame-logic express their view of a Web ontology language. I gave them complete freedom in forming their statements, but I tried to have them focus on these core questions:

- What is the single most important property of a Web ontology language?
- Taking a formal semantics for granted, should a Web ontology language be based on description logics such as OIL or DAML+OIL?
- How should we communicate a Web ontology language to developers?

Having said so much about the dire nature of a Web ontology language's standardization, where lies its beauty and its excitements? The answer in my eyes is to *k*eep *i*t *s*traight, *s*imple, and *e*xtensible. There are, of course, different opinions. So what does this boil down to from the technical view? Let the following contributors tell you how ontologies' *kisses* move them.

—Steffen Staab

Acknowledgments

I am indebted to all contributors who positively responded to my request for a statement within two weeks, sharing their thoughts on recent W3C WOWG discussions.[2]

References

1. A. Linden, *The Semantic Web: Trying to Link the World*, Gartner Group Research Note, Aug. 2001.

2. D. Truog et al., "How The X Internet Will Communicate," *Forrester Report*, Dec. 2001.

The Complexity of the Web Ontology Language

Frank van Harmelen, *Vrije Universiteit, Amsterdam*

The W3C's Web Ontology Working Group's charter (see www.w3.org/2001/sw/WebOnt/charter) explicitly states that DAML+OIL must be taken as the starting point when designing a Web ontology language (see www.w3.org/Submission/2001/12). So have DAML+OIL designers succeeded? As one of the designers, I feel entitled to take a critical look at the DAML+OIL design and make recommendations for OWL's design. (The intended W3C recommendation is likely to be called OWL [the Ontology Web Language, http://lists.w3.org/Archives/Public/www-webont-wg/2002Jan/0033.html], so I will use this name to refer to whatever language the WebOnt Group produces.)

There are many ways to measure a language's complexity. Do we mean the computational complexity of various algorithms for DAML+OIL? If so, which algorithm? Do we mean the computational complexity of parsing it or of the general inference in DAML+OIL? Or is it the computational complexity of consistency checking a specific inference task or of subsumption reasoning (that is, calculating the subclass hierarchy)? Do we care about the traditional asymptotic worst-case complexity class or something vaguer such as the "average case" complexity?

We could also focus on technical complexity. How hard is it for tool builders to implement services for OWL (such as editors, storage, inference, and so forth)?

Alternatively, we might focus on OWL's conceptual complexity. Of course, this is a much vaguer notion than computational complexity. How hard is it for the average intended OWL user to learn and use the language?

Here I discuss how DAML+OIL scores on each of these issues.

Computational complexity

Computational efficiency is the least interesting of the three, especially when interpreted as asymptotic worst-case complexity results. Even *decidability* as a required property is debatable. Decidability (for example, of consistency) simply means that no algorithm exists that will decide consistency for arbitrary OWL theories. However, algorithms might well exist that work in many if not all practical cases. Even when we can't formally guarantee the correctness of such answers, the practical reliability of them might be sufficient.

Nevertheless, this computational complexity has been one of the guiding lights of the DAML+OIL design—probably too much so.

Technical complexity

Clearly, any viable W3C standard must take technical complexity very seriously. Rapidly and widely available cheap technological support is essential for any W3C standard's success. DAML+OIL seems to have struck the balance fairly well in this respect. In its rather short lifespan of less than two years, a surprisingly large set of tools and technology has been developed to support DAML+OIL usage (www.daml.org/tools): editors, inference engines, crawlers, browsers, APIs, storage devices, and so forth. This rapid development of technology has no doubt been possible

Glossary

DAML	DARPA Agent Markup Language
DAML-S	DAML Service
DL	Description logic
DTD	Document Type Definition
KIF	Knowledge Interchange Format
EER	Enhanced Entity-Relationships
OIL	Ontology Inference Layer
OWL	Ontology Web Language
RDF	Resource description framework
RDFS	RDF Schema
SGML	Standard Generalized Markup Language
UML	Universal Modeling Language
W3C	World Wide Web Consortium
WOWG	Web Ontology Working Group (WebOnt Group)
WSDL	Web Service Definition Language
XML	Extensible Markup Language

because of the reuse of much of XML- and RDF-based technology. Clearly, the decision to design DAML+OIL on top of XML and RDF has paid off. Although this decision is currently controversial for OWL (see Peter Patel-Schneider's contribution), we must seriously consider it, because it is not just a matter of language syntax and semantics but also of the technological base on which the language is built.

Conceptual complexity

The area where DAML+OIL is most off balance is on its conceptual complexity. Of course, this notion is rather informal and fuzzy. How should we measure how hard it is for the average user to learn DAML+OIL or use it? How can we measure how well DAML+OIL fits with the background and established practices of the intended user community? These are inherently vague notions, but that does not make them less important. The only way to get an impression of how well DAML+OIL has done in this respect is to look at the ontologies that have been built with it.

The first thing to notice is the limited subset of DAML+OIL that users have been exploiting. Roughly speaking, most of the DAML+OIL that I've seen written in my two years with the language consists of the RDF Schema primitives, plus:

- toClass (equals local range restrictions)
- Cardinality restrictions (typically with values 1 or 2)
- Disjointness statements (both DisjointWith and DisjointUnionOf)
- inverseOf and TransitiveProperty statements on properties
- Enumerated classes (oneOf)
- UniqueProperty

This means that language constructions such as negation, disjunction, the hasValue and hasClass restrictions, and all qualified number restrictions are almost never used in any DAML+OIL ontologies I've seen so far. This also holds (although to a somewhat lesser extent) for intersectionOf, unionOf, and UnambiguousProperty. To use an often-heard maxim: In retrospect, I'm convinced that we could have obtained 80 percent of the usage of DAML+OIL with 20 percent of the language constructs. I know that many of these constructs are interdefinable, so this reduction won't change the language's computational complexity, but the conceptual complexity (whatever it means precisely) will decrease—it has to.

Another important aspect of conceptual complexity is modeling style. Most of the ontology-modeling that I have seen (and this covers academic work as well as industrial applications in widely varying areas such as engineering, financial services, human resource management, and medical knowledge) are firmly rooted in a frame-based modeling style:

- All classes of interest are explicitly named, so the use of class expressions is practically zero.
- All subclass relations are explicitly stated, so subsumption-style reasoning is only used for verification (and not for online use).
- Properties only apply to a class when they have been stated as such. This is opposed to the approach in description logics, where properties apply to any class unless a restriction forbids it.

The Web is a pluralistic world, and we should expect the Semantic Web to be so, too. This makes any attempt to create a standard Web ontology language a huge challenge.

All these three aspects of the frame-based modeling style directly conflict with the description logics style of DAML+OIL. Clearly, OWL will need a formal semantics. This is undisputed in the WebOnt Group. DLs are the best candidate for providing the formal foundations of ontology, but this does not imply that OWL itself should have a DL's form. OIL, one of DAML+OIL's predecessors (see www.ontoknowledge.org/oil), showed that it is possible to have a language with a DL for its foundation but frame-based in its appearance.

OWL must significantly lower DAML+OIL's conceptual complexity by removing half of the language and hiding its DL semantics behind a strongly frame-based syntax based on the resource description framework and RDF Schema.

The Many Faces of the Semantic Web

Peter Clark and Mike Uschold, *Boeing Engineering and Information Technology*

The Web is a pluralistic world, and we should expect the Semantic Web to be so, too. This makes any attempt to create a standard Web ontology language a huge challenge, due to application developers' differing needs and goals. Although people generally agree that the Semantic Web is about making Web content accessible to machines as well as humans, there are many interpretations about what this means in practice and the corresponding role that ontologies and formal representations should play. This is reflected in the confusing myriad of different languages being circulated—XML, RDF, OIL, DAML-OIL, XMLSchema, RDFS, WebOnto, and so forth.

An ontology's roles

For a user building a Web-based document repository, an ontology's role might simply be to provide a standard, conceptual vocabulary for labeling documents. For example, the ontology might provide a standard set of terms for filling in the dc:subject field when describing a Web document's subject in its metadata. The formal requirements for defining this ontology are minimal; ideally, the ontology could be expressed in a standardized format so that it could be downloaded, exchanged, maintained, and browsed.

For a user wanting to publish or exchange database-style information on the Web (for example, a price list of goods for sale), his or her interest will be in having an agreed set of concepts and a syntax for making statements using them (say, XML). Again, in many cases an informal, textual characterization of each concept's meaning might be adequate for characterizing the concept space, and an XML document type definition (DTD) might suffice for defining the syntax.

Finally, there are other uses that demand a more expressive and formal language.

One example is the automatic semantic integration of information between software agents performing their required tasks. Consider a tax assistant agent that requires access to machine-sensible IRS tax rules for a given year. To be sure that the rules are used in the correct way, the underlying ontology language must have some characterization of what inferences are valid—that is, what its semantics are. Those semantics could be those of pure first-order logic or a subset of it (as is done in OIL), but this creates a serious tension between the idealized worlds, which are easily expressible, and the messy world in which we live, where almost any universal statement has exceptions. From a practitioner's viewpoint, it seems that a modified semantics or the ability to state inference assumptions (for example, the closed-world assumption, or use of default logic or of some procedurally defined semantics) is more desirable to allow the irregularities of the world to be easily expressed, despite the theoretical difficulties this poses.

Requirements for designing a Web ontology language

So, how should these considerations affect the design of a standardized Web ontology language? First and most important, if a single language is going to serve all these different communities, it needs a layered design in which a simple core can accommodate simple taxonomies and relationships, while additional layers of expressivity, functionality, and complexity can be added for groups requiring more expressive power. In fact, this layering principle has been fundamental in current standardization efforts—OIL and DAML+OIL extend a core RDF layer. As per our examples above, we imagine that there will be a suite of different kinds of applications, each requiring different levels of formality and sophistication—for example, search, semantic integration, or travel-planning softbots. In the near term, informal or lightweight ontologies will go a long way. We anticipate that the more complex the ontology, and the greater the need for formality, the fewer and less mainstream will be the applications that use the complexity. Eventually, there will be sufficiently large niche markets to justify this complexity and expense.

Second, the language (or at least its core) needs a simple, accessible syntax if it is to have a realistic chance of widespread acceptance. From a formalist's viewpoint, the syntax might seem largely irrelevant, but from a pragmatic, sociological viewpoint, this is highly important. A key feature contributing to the Web's emergence was the simplicity and accessibility of HTML, and this should not be overlooked for designing a Web ontology language. Recent efforts such as the N3 notation for resource description framework (RDF) directly address this issue.

Third, if a standardized ontology language is to take off, we need software tools to support ontology development, use, and maintenance. This will include tools that let users create ontologies in their familiar languages, such as UML, and export them to the standard language. In addition, software developers need guidance in deciding what level of complexity and formality is appropriate for a given type of application.

A key feature contributing to the Web's emergence was the simplicity and accessibility of HTML, and this should not be overlooked for designing a Web ontology language.

Finally, and perhaps most important, there must be some large-scale applications developed and in common use that demonstrate the value of a standardized ontology language, demonstrate the value of formal semantics, and help define and convey how the different layers of the Semantic Web must play together, and how these technologies relate to the more available XML technologies (for example, the relationship of DAML Service to the Web Service Definition Language (WSDL). Again, the Web Ontology Working Group is pursuing this goal vigorously with use-case analyses, and the RDF Core working group is resolving many minor compatibility issues between the RDF layer and the ontological layer.

The Web is a multifaceted world, and the Semantic Web will be too, with a wide variety of different requirements that a standardized ontology language should support. This will make such a language challenging to develop, but the process seems to be moving in the right direction.

Acknowledgments

Thanks to Dave Jones and John Thompson for contributions and comments.

Ontologies on the Semantic Web

James Hendler, *University of Maryland*

To understand this short essay, I need you to use the following visualization technique. Spread your hands about one foot wide. Imagine your right hand is the traditional AI knowledge representation view of formal ontologies. Think of moving left as moving from carefully constructed large ontologies to smaller component ontologies, created in less formal ways. About the time you hit your left hand, you're at the edge of the traditional AI view of ontologies and into something much less formal and somewhat less well-defined. Now turn your head left and squint into the distance—that's where you'll find my view of ontologies. Ontologies are going to change the world, but only if we change our view of them.

A tidal wave is coming

The field of knowledge representation is about to get shaken up and bent out of recognition in the way that the fields concerned with online documents were a few years back. The text markup and hypertext communities spent years working on expressive text markup languages and complicated programs designed to allow detailed analysis of online documents coupled by two-way links between the pages. Great care was taken to guarantee that users would never hit a link that didn't work. Guaranteeing consistency in the links required centralized repositories and complex protocols to make sure if a document moved, the links moved with it. The community knew that the death of their systems would arise if links broke.

However, this assumption was blown out of the water by a small program called the World Wide Web, which was supported by a simple protocol called the Hypertext Trans-

Semantic Error 409—Ontology Not Found

You've encountered an "Ontology Not Found" error while trying to access a semantic term grounded on the University of Maryland Computer Science Department Semantic Web server.

fer Protocol. Its creator, Tim Berners-Lee, realized that the hypertext systems his peers envisioned could never scale—if person A needed permission or (even worse) had to pay person B to link to a text, the system wouldn't grow. So, he developed a scheme that challenged the hypertext community's fundamental assumptions—and its success has been without parallel. As a result, the research communities doing markup and hypertext were profoundly changed, and current meetings have few of the old-timers involved as major research participants.

What is the tidal wave threatening to swamp our community? The Semantic Web effort, led by a combination of researchers from government, industry, and academia, wants to take ontologies (and other *knowledge representation* technologies) and bring them to the Web—at a Web scale. When these things start to proliferate, it will challenge some of our basic KR notions, and our community will lose if we try to kick in our heels and stay firm. For example, on the Web there is no way to guarantee consistency. It also contains information that is inconsistent, incorrect, lacking reliable sources, combined with other information without author approval, and much more. Even worse, if I point at terms in your ontology, and then you change it (or move it), my representation becomes ungrounded. KR has never before had to deal with the AI equivalent of an Error 404!

Putting the "web" into the Semantic Web

What will this brave new world of ontological information look like? Consider the current Web and how it works. Documents of many sizes and shapes are available all over the place, with some sites developed just to index others (some of the most popular sites on the Web). These documents are linked in many different ways, and they refer to things on other pages with little or no effort required. Web ontologies will grow in this same way. People who know that ontologies exist (the Web masters of the Semantic Web) will help define ontologies for areas of interest to them. These ontologies will use terms from other ontologies and so on. In addition, products written for end users, powered by access to these ontologies, will also allow for the extension of ontological terms (for example, a form to be filled out might have a field called "other" that, when named, asserts a new property of the class being extended).

With this in mind, let's look at ontology languages. First and foremost, a Web ontology language must be anchored on the Web. This means that each symbol must be grounded at a Universal Resource Indicator. Thus, on the Web, there is no such thing as a generic *person* filed—rather, you must refer to a particular definition such as www.cs.umd.edu/projects/plus/personontology.daml#person, which is in turn anchored to other Web locations. This notion of Web embedding is absolutely crucial to the Semantic Web.

On top of this is the ability to relate these grounded terms one to another. Thus, we can create classes and subclasses, use descriptions or classification, and so forth. However, the Web is huge: No reasoner can work by assuming it knows everything and has found all possible classes. Instead, there must be a way to assert what concepts a document is closed with respect to (a particular document or site, the collection of facts found by a Semantic Web crawler, and so forth). Furthermore, this is the Web! Different people will use the same semantic assertions for different uses, and thus a Web language's exact form might not matter as much as the fact that everyone uses it.

In fact, the Semantic Web will not be powered solely by reasoning systems but by a wide variety of tools and techniques that use ontologies as their interlingua. These very different agents must be able to trace terms found on different pages and find common referents—the content will be distributed, inconsistent, and preferably (ready for this?) not very expressive. On the Web, expressivity is the kiss of death—simpler languages and solutions go further than complex ones. After all, HTML, a watered-down version of the much more expressive SGML, took over the world. SGML developers are still wandering around telling people how everyone else is doing it wrong, but strangely no one seems to listen. And that is the lesson we in the KR community must take to heart.

Come join the party

Web-based ontologies are poised to move out to the Web and become embedded in Web tools. Most users will not even know they exist as they take advantage of the ontological content included in shrink-wrapped programs ranging from specialized process-modeling tools such as UML to Web page creation tools bought over the counter to business-related applications such as spread sheets and word processors. When this happens, KR researchers will have a choice—move with the times or become the old ignored grouches complaining about how the world simply doesn't understand what this KR stuff is all about.

An Ontology Language for the Semantic Web

Ian Horrocks, *University of Manchester*

The Semantic Web dream is of a Web where resources are machine understandable and where both automated agents and humans can exchange and process information. The proposed mechanism for realizing this dream is to add semantic markup to Web resources that describes their content and functionality. Ontologies will define the vocabulary for such markup; they consist of sets of statements describing the structure of the domain of interest and giving meaning to the terms therein.

The need for formal semantics

A Web ontology language's most important property is a well-defined formal semantics. This requirement is now widely accepted, but it was not always so. Doubts have been expressed about the wisdom of having such semantics. Some researchers have argued that more people would use the language if they could

interpret its meaning in ways that suited their own application requirements and that formal (logical) semantics would always lead to contradictions. The Web is so large that if a fact P is asserted somewhere, a denial of P is sure to be asserted elsewhere.

The problem with this laissez-faire approach is that it defeats the whole purpose of using ontologies: providing a shared understanding. If two agents can freely interpret the information in an ontology, then they could both arrive at different conclusions as to the meaning of the information they are exchanging. Moreover, there would be no obvious mechanism for resolving disputes—such as between a service provider and consumer—because it would be impossible to say whose interpretation was correct.

Another concern is that implementing the Semantic Web will result in the entire contents of the Web being treated as a single knowledge base. The Web is, however, much too large and dynamic for this to be a realistic prospect. Instead, Semantic Web users will work with a very small subset of the total available information, using only that which is relevant to the task at hand and deemed to be trustworthy. In particular, when two agents (humans or machines) exchange information, they can agree on the use of one or more ontologies to provide a vocabulary for their communication—the meaning of which both sides understand. It is useful to be able to check that these ontologies are logically consistent (if not, we might wish to revise our opinion as to their trustworthiness) and that the information being exchanged is consistent with respect to the ontologies.

Knowledge representation

Ontologies are intended to capture knowledge about the world, and the question naturally arises as to how this knowledge should be represented. One natural approach to divide the world into classes of objects with common properties, identifying some classes as specializations of others, inheriting all the properties of the more general class and (possibly) adding new properties of their own. This methodology has been studied since Aristotle's time, and can be seen in many modern applications, such as object-oriented databases, semantic networks, and frame systems.

When this methodology is formalized by giving well-defined semantics to the descriptions of class properties (as in OIL and DAML+OIL), we derive the added benefit of being able to check that descriptions are logically consistent, compute class specialization and generalization relationships (often called subsumption relationships), and infer from the description of an individual whether it is a member of a given class. This allows, for example, two agents exchanging information about antique furniture, who have agreed to use a furniture ontology containing the information that furniture made in Britain between 1760 and 1811 is late Georgian, to agree that a table made in Britain in 1795 is a late Georgian table. Without some ability to exploit semantics in this way, there seems little point in providing it in a machine-readable form.

> The Semantic Web dream is of a Web where resources are machine understandable and where both automated agents and humans can exchange and process information.

A class- and property-based KR methodology should facilitate its use by software developers and modelers, because many of them are already familiar with the object-oriented paradigm and the use of modeling languages such as Enhanced Entity-Relationship (EER) and UML. However, extending these methodologies with logic-based semantics and inference mechanisms increases complexity and might require some adaptation on the their part. One way to minimize this is to provide tools that look familiar to the target users, do not intimidate them, and let them adapt their practices rather than revolutionize them. One difficulty with this approach, however, is that it can fool users into believing that they understand what they are implementing or modeling, when, in fact, they do not.

An example of this is the common confusion between integrity constraints and inference. Many users of traditional software engineering and modeling tools expect that if the ontology states that all computer science students must take a math course, and if we find a computer science student without a listed math course, the system will indicate that there is an error in that student's data. Instead, a logical ontology language will simply infer that one of the student's courses really is a math course (without that fact having been explicitly stated) or that she is taking another unspecified course that is in fact a math course.

Many difficult problems remain to be tackled—trust, time, and identity, to name but a few. However, an ontology language that is powerful enough to capture most desired information, while still being simple enough to allow agents to perform inferences based on the facts at their disposal at any particular moment, will represent a major contribution to the Semantic Web's development.

Building the Web's Ontology Layer

Peter F. Patel-Schneider, *Bell Labs Research*

The Web ontology language must be able to describe and organize knowledge on the Web. Fine idea, but what does it mean?

Requirements

Starting at the beginning, the Web ontology language is part of the World Wide Web, not just the Semantic Web.[1] Thus, it should be able to describe and organize knowledge expressed in XML (www.w3.org/XML), not just expressed in Semantic Web languages such as the resource description framework and RDF Schema.

Unfortunately, this first requirement is already problematic, because there is a gulf between XML and RDF.[2] Will the Web ontology language work with knowledge expressed in XML and bypass RDF, or will it work with knowledge expressed in RDF and ignore XML? As there is vastly more knowledge expressed in XML than in RDF, the answer to this should be easy—work with XML and bypass RDF.

Knowledge on the Web certainly includes data, such as the data that is cur-

rently expressed in XML documents. However, knowledge in the Web also includes things that cannot be expressed in XML or related languages such as XML Schema, such as many kinds of uncertain or vague information. For example, the Web ontology language should be able to deal with disjunctive information (such as one of John's siblings is either Mary or Bill) as well as information about unidentified objects (such as John has at least two siblings or John's siblings are all doctors). The ability to deal with this non-XML information is why the XML Schema is not a candidate for the Web ontology language.

Organizing knowledge

Organizing knowledge on the Web means a multitude of things. Because an ontology language is about defining categories or groups of objects, the Web ontology language must include at least this facility. Defining categories is generally done by creating concepts, frames, classes, or descriptions that identify categories of objects, such as people, and provide properties for objects that belong to the category, such as people having siblings that are also people.

Organizing knowledge has many other aspects. Taken to the extreme, it could require the facilities of a general-purpose logical formalism that can deal with temporal, spatial, epistemic, and inferential aspects of knowledge. The Web ontology language is certainly not all that, but there are other aspects of knowledge that are in the province of the Web ontology language.

For example, the concepts in an ontology are related to one another—in particular, through generalization relationships, as *student* is related to *person*. The Web ontology language should be able to represent these generalization relationships. Furthermore, concepts in an ontology can have complete definitions, such as the definition of a student as a person who is enrolled in an educational institution. The Web ontology language should be able to give complete definitions for its concepts.

When concepts have complete definitions it is possible to infer generalization relationships, such as inferring that people who are enrolled in universities are students. Such inferences are a necessary component of a system that can deal with the many disparate sources of knowledge on the Web—for example, to determine that people who are enrolled in "hochschules" are university students. The Web ontology language thus must be able to perform these and other related inferences. Implementations of the Web ontology language must provide these inference services, making them much more than just simple data storage and retrieval systems.

Finally, the Web ontology language must be able to describe the knowledge it is concerned with in a way that precisely determines what it is about. There are several ways to produce this determination, but the best way is some sort of denotational or model-theoretic semantics. What is model-theoretic semantics and why should Web programmers care about it? It is really nothing more than a generalization of data models, such as the relational data model or semistructured data, that can deal with identifiable objects and uncertain and vague information.

> The Web's decentralized nature makes it inevitable that communities of users or software developers will use their own ontologies to describe their data or services.

The Web ontology language

All these things make the Web ontology language a powerful formalism. Fortunately, a vast amount of research into such formalisms already exists. The Web ontology language is a sort of logic, and thus the facilities of formal logics are available for its design and analysis. In particular, the Web ontology language will be a sort of description logic, very similar to expressive DLs such as OIL.[3]

The Web ontology language will not be just a simple frame system, where only simple aspects of concepts can be defined and the only relationships between concepts are the ones explicitly stated in ontology documents. Instead, the Web ontology language will define powerful inferential services that Web ontology servers will have to implement, much as database languages define powerful querying services that database management servers must implement.

Programmers will thus interact with knowledge in Web ontologies just as they interact with data in a database. The interface to knowledge in Web ontologies will be through a Web ontology server that provides powerful services with a full-featured application programming interface. It would not be not via simple data-access tools, such as current RDF systems. Users of a Web ontology server will have access to many powerful services that store and analyze the knowledge available on the Web and make inferences from this knowledge, providing to programmers much more than the data explicitly carried in XML documents.

References

1. T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web," *Scientific American*, vol. 284, no. 5, May 2001, pp. 34–43.
2. P. Patel-Schneider and J. Siméon, "The Yin/Yang Web: XML Syntax and RDF Semantics," to be published in *Proc. 11th World Wide Web Conf.* (WWW11), ACM Press, New York, 2002.
3. D. Fensel et al., "OIL: Ontology Infrastructure to Enable the Semantic Web," *IEEE Intelligent Systems*, vol. 16, no. 2, Mar./Apr. 2001, pp. 38–45.

The Semantic Web Needs Languages for Representing (Complex) Mappings Between (Simple) Ontologies

Marie-Christine Rousset, *University of Paris Sud*

There is a general agreement that ontologies are likely to play a key role for making the promising vision of the Semantic Web a reality. The Semantic Web envisions a worldwide distributed architecture where highly distributed and possibly heterogeneous data or computational resources will easily interoperate to coordinate complex tasks such as answering queries or global computing. Semantic marking up of Web resources using ontologies is expected to provide the necessary glue for making this vision work.

However, the discussion is still open on several important issues related to the kinds of ontologies that are useful for the Semantic Web, the languages for representing them, and their standardization.

Ontologies

The purpose of ontologies in the Semantic Web is to provide a kind of semantic typing for the data distributed all over the Web to facilitate their interrogation by users through search or query engines, and more generally their use as input or output of Web services.

Building a global and general ontology (or using an existing one, such as CYC[1]) to serve as a reference set of semantic tags for marking up the data of the Web is, at worst a utopia, at best, an enormous enterprise that would eventually be useless in practice for the Semantic Web.

The Web's decentralized nature makes it inevitable that communities of users or software developers will use their own ontologies to describe their data or services. In this vision, ontologies are distributed and can only be exchanged, merged, mapped, and extended.

Ontologies are structured vocabularies shared by communities of users or practitioners who are not knowledge engineers or logicians. Therefore, they are simple: trees of terms, in the spirit of Yahoo's categories or taxonomies of classes described by their names and their attributes, are the kinds of ontologies that I see as the future flesh of the Semantic Web.

The fact that they are simple does not mean that they do not have a well-defined formal semantics: it is precisely because they are simple that their formal semantics can fit nicely with their intuitive meaning. Having a formal semantics is indeed a necessary property for the ontologies serving for typing the Web data, to enable a precise and rigorous characterization of the operations that are to be performed on those data. For example, to be rigorously defined (and thus proved), the correctness of a query answering algorithm only makes sense with respect to a given formal semantics of the data and of the query language that is used for querying them.

In this vision of the Semantic Web, based on simple but distributed ontologies, the key point is the mediation between data, services, and users, using mappings between ontologies.

Mappings

Complex mappings and reasoning about those mappings are necessary for comparing and combining ontologies and for integrating data or services described using different ontologies. For example, the fact that what is denoted in ontology O1 to be *restaurants* whose attribute *category* is filled with *music-hall* show corresponds to what is denoted in ontology O2 as *cabarets* whose attribute *food service* is filled with *dinner* might be exploited to answer a user's query about having dinner and seeing a show in the same place.

Existing data integration systems (such as TSIMMIS,[2] Information Manifold,[3] Infomaster,[4] Picsel,[5] Momis,[6] and Xyleme[7]) are centralized systems of mediation between users and distributed data that exploit mappings between a single mediated schema and schemas of data sources. Those mappings are modeled as views (over the mediated schema in the local-as-view approach, or over the sources schemas in the global-as-view approach), expressed using languages that vary from one system to another but that all have a formal semantics.

The main issue for a Web ontology language is less the choice of a language for describing ontologies than the choice of a language for representing mappings between ontologies.

For scaling up to the Web, this centralized approach of mediation is probably not flexible enough, and distributed systems of mediation are more appropriate. For an easy deployment of distributed data management systems at the scale of the Web, it will be essential to use expressive and declarative languages for describing semantic relationships between ontologies serving as schemas of distributed data or services.[8] Therefore, the main issue for a Web ontology language is less the choice of a language for describing ontologies than the choice of a language for representing mappings between ontologies.

Standardization

Regarding standardization, if a standard syntax for a language is desirable, I don't think that standardizing syntactical notations is an essential issue. The important point is to have a well-accepted formal semantics for Web ontology languages. Formal semantics is the necessary foundation for reasoning tasks, complex query processing, formal verification, optimization, and ontology learning. These services are useful for building, in a well-founded manner, distributed architectures for sharing the future Semantic Web's data and services.

References

1. D. Lenat, "Cyc: A Large-Scale Investment in Knowledge Infrastructure," *Comm. ACM*, vol. 38, no. 11, Nov. 1995, pp. 32–38.
2. H. Garcia-Molina et al., "The TSIMMIS Project: Integration of Heterogeneous Information Sources," *J. Intelligent Information Systems*, vol. 8, no. 2, Mar. 1997, pp. 117–132.
3. A. Levy, A. Rajaraman and J. Ordille, "Querying Heterogeneous Information Sources Using Source Description," *Proc. 22nd Int'l Conf. Very Large Data Bases,* 1996, pp. 251–262.
4. M. Genesereth, A. Keller, and O. Duschka, "Infomaster: An Information Integration System," *Proc. SIGMOD*, ACM Press, New York, 1997, pp. 539–542.
5. F. Goasdoué, V. Lattès, and M.C. Rousset, "The Use of Carin Language and Algorithms for Information Integration: The PICSEL System," *Int'l J. Cooperative Information Systems*, vol. 9, no. 4, Dec. 2000, pp. 383–401.
6. D. Beneventano et al., "Information Integration: The MOMIS Project Demonstration," *Proc. 22nd Int'l Conf. Very Large Data Bases*, Morgan Kaufmann, San Francisco, 2000, pp. 611–614.
7. L. Xyleme, "A Dynamic Warehouse for XML Data of the Web," *IEEE Data Eng. Bulletin*, vol. 24, no. 2, 2001, pp. 40–47.
8. A. Halevy, Z. Ives, and D. Suciu, *Schema Mediation in Peer Data Management Systems*, tech. report, Computer Science Dept., Univ. of Washington, Seattle, 2002.

The Web Is Not Well-Formed

Guus Schreiber, *University of Amsterdam*

The debate about what a Web ontology language should look like is reminiscent of past neat–scruffy struggles. Knowledge modelers want expressiveness, logicians stress decidability. The main difference is that the Semantic Web actually forces us to make some choices: there is a strong need for real-world knowledge representation.

Expressivity requirements

The people arguing for an expressive language have a strong case. For example, suppose a person wants to find images of red apes on the Web. Most photos of orangutans (which generally have a red-orange color) will satisfy this query. However, you can't expect the indexer of every orangutan photo to explicitly state the ape's color (this also leads to unwanted inter-indexer variability). You really want them to annotate the photo with the class *orangutan*, and possibly only specify the color if it is not red or orange (old animals can be brown or gray, some orangutans are albino, and so forth). This requires the ability to express default knowledge, a language property logicians are not very fond of because it requires nonmonotonic reasoning. However, if specification of default color values is disallowed, we will not be able to retrieve a significant number of relevant photos. Actually, making this match is what the Semantic Web is all about (as it enables a match between a query *red ape* and a photograph that is annotated with neither of these terms). Also note that from the search perspective, 100-percent correctness (precision) is not needed, just a sufficiently high percentage.

In the same animal domain, we find another example of a frequently occurring form of knowledge that is difficult to capture in description logic—namely, the fact that there is in practice no hard borderline between instances and classes. If we look in a biology book at the definition of an orangutan, we will find something like this:

Orangutan

Latin name:	*Pongo pygmaeus*
kingdom:	Animalia
phylum:	Chordate
class:	Mammalia
order:	Primates
family:	Hominidae
genus:	Pongo

From the viewpoint of the biological taxonomy of species, an orangutan is an instance of a species class, while at the same time it represents a collection of animal instances. *Orangutan* can thus be considered both a class and an instance. Note that specifying orangutan as a subclass of species (and defining the values above as slot-value restrictions) is incorrect. An individual orangutan is not an instance of species (it is not an animal type).

> The methodological guidelines should encourage user groups to agree on a small set of metaclasses to handle their particular expressivity requirements, which fall outside the DL core.

This notion of classes as instances of (meta)classes comes up during conceptual modeling of almost any domain with some degree of complexity. Another example is a Boeing 747, which denotes both a collection of individual aircrafts but also is itself a member of a collection of aircraft types, with other members such as Airbus 310. Both interpretations are needed to semantically annotate Web pages of aircraft industry.

The metaclass mechanism

An ontology language on top of RDF Schema only makes sense if it introduces some formal semantics to the ontology definitions. Undoubtedly, description logic provides a well-researched basis for such a semantics. Subsumption, which forms the basis of description logic, is a natural way for people to express domain knowledge. However, if we just ignore expressivity requirements, people will simply not use the Web ontology language, and the whole effort will become a failure. Logic is an ideal, well-formed world—the Web is not.

To cater to this, the language should have an extendible metalevel, which enables users to describe additional interpretations of classes and properties. The *class-as-instance* mechanism can actually fulfill this role (as it does in RDF Schema). In addition, the Web ontology language should provide methodological guidelines for using the base language plus the metalevel mechanism in an appropriate manner. For example, one can solve the default problem by defining a class *orangutan* with necessary property restrictions and, in addition, a subclass such as *archetypical orangutan* to define default property restrictions. This leads to correct logical interpretations by DL reasoners. Subsequently, we can define a metaclass such as *archetype* and make the subclass an instance of this class. This enables index and search programs to treat this class in a special manner.

Of course, the metaclass mechanism could potentially open a can of worms. If used only in an ad hoc fashion, it will lead to messy ontologies. The methodological guidelines should encourage user groups to agree on a small set of metaclasses to handle their particular expressivity requirements, which fall outside the description logic core. In the future, we might even want to define standards for metaclasses to be used. ❐

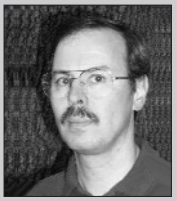
Acknowledgments

This essay benefited from discussions with other WebOnt Group participants and with various colleagues in Amsterdam. The author is supported by the ICES-KIS project Multimedia Information Analysis, funded by the Dutch government and by the IST Project IBROW funded by the European Union.

Frank van Harmelen is a professor in the AI Department at the Vrije Universiteit in Amsterdam. His research interests include knowledge representation, approximate and anytime reasoning, and applications of these on the Web and in medical applications. He received a PhD in artificial intelligence from the University of Edinburgh. Contact him at Dept. of AI, Faculty of Sciences, Vrije Universiteit Amsterdam, de Boelelaan 1081a, 1081HV Amsterdam, Netherlands; frank.van.harmelen@cs.vu.nl; www.cs.vu.nl/~frankh.

Ian Horrocks is a senior lecturer in computer science with the Information Management Group at the University of Manchester, UK. His research interests include knowledge representation, automated reasoning, optimizing reasoning systems, and ontological engineering, with particular emphasis on the application of these techniques to the World Wide Web. He received a PhD in computer science from the University of Manchester. He is a member of the OIL steering committee and the Joint EU/US Committee on Agent Markup Languages, and is coeditor of the DAML+OIL language specification. Contact him at the Dept. of Computer Science, Univ. of Manchester, Oxford Rd., Manchester, M13 9PL, UK; horrocks@cs.man.ac; www.cs. man.ac.uk/~horrocks.

Peter Clark is a research scientist in the Knowledge Systems group within Boeing's Engineering and Information Technology organization. His interests are in knowledge representation, the construction and application of large-scale knowledge-bases and ontologies, and the role of knowledge-bases in data interpretation and natural language processing. He received his MSc in artificial intelligence from Edinburgh University and his PhD in computer science from Strathclyde University. Contact him at The Boeing Company, m/c 7L66, PO Box 3707, Seattle, WA 98124; peter.e.clark@boeing.com.



Peter F. Patel-Schneider is a member of the technical staff at Bell Labs Research. His research interests center on the properties and use of description logics. He designed and implemented large sections of CLASSIC, a DL-based knowledge representation system. He is also interested in rule-based systems, including more standard systems derived from OPS as well as newer formalisms such as R++. He received his PhD from the University of Toronto. He is involved with the World Wide Web Consortium's Web Ontology Working Group, helping to design a new language for representing ontologies in the Web. Contact him at Bell Labs Research, 600 Mountain Ave., Murray Hill, N.J. 07974; pfps@research.bell-labs.com; www.bell-labs.com/user/pfps.

Mike Uschold is a research scientist in Boeing's Engineering and Information Technology organization. His interests include the development and application of ontologies. He is on the industrial advisory board of a UK research consortium on Advanced Knowledge Technologies (www.aktors.org) and is on the steering committee of Ontology.Org (www.ontology.org). He received his BS in mathematics and physics at Canisis College, New York, MS in computer science from Rutgers University, and PhD in artificial intelligence from The University of Edinburgh. Contact him at The Boeing Company, m/c 7L40, PO Box 3707, Seattle, WA 98124; michael.f.uschold@boeing.com.

Marie-Christine Rousset is a professor and the leader of the Artificial Intelligence and Inference Systems Group in the Laboratory of Computer Science at the University of Paris-Sud. Her research topics are knowledge representation and information integration—in particular, description logics, hybrid knowledge representation languages, query rewriting using views, and automatic classification and clustering of semistructured data (such as XML documents). She received a best paper award from AAAI in 1996 for a joint work with Alon Levy. Contact her at LRI, Building 490, Univ. of Paris-Sud, 91405 Orsay cedex, France; mcr@lri.fr; www.lri.fr/~mcr.

James Hendler is a professor of computer science at the University of Maryland and the Director of Semantic Web and Agent Technologies at the Maryland Information and Network Dynamics Laboratory. He received his PhD in artificial intelligence from Brown University in 1986. He received a 1995 Fulbright Foundation Fellowship, is a fellow of the AAAI, is a member of the US Air Force Science Advisory Board, and is the former Chief Scientist for Information Systems at DARPA. He chairs the W3C's Web Ontology Working Group. Contact him at hendler@cs.umd.edu; www.cs.umd.edu/~hendler.

Guus Schreiber is an associate professor of knowledge technology at the University of Amsterdam. His research interests lie mainly in the area of knowledge engineering, knowledge-system development, and semantic markup of multimedia collections. He holds an MSc in medicine from the University of Utrecht and a PhD in artificial intelligence form the University of Amsterdam. He is a member of the Belgian-Dutch Artificial Intelligence Association BNVKI. He is currently chairing W3C's Web Ontology Working Group. Contact him at Social Science Informatics, Univ. of Amsterdam, Netherlands; schreiber@swi.psy.uva.nl; www.swi.psy.uva.nl/usr/Schreiber/home.html.