Three Theses of Representation in the Semantic Web

Ian Horrocks
University of Manchester
Manchester, UK
horrocks@cs.man.ac.uk

Peter F. Patel-Schneider
Bell Labs Research
Murray Hill, NJ, USA
pfps@research.bell-labs.com

ABSTRACT

The Sematic Web is vitally dependant on a formal meaning for the constructs of its languages. For Semantic Web languages to work well together their formal meanings must employ a common view (or thesis) of representation, otherwise it will not be possible to reconcile documents written in different languages. The thesis of representation underlying RDF and RDFS is particularly troublesome in this regard, as it has several unusual aspects, both semantic and syntactic. A more-standard thesis of representation would result in the ability to reuse existing results and tools in the Semantic Web.

Categories and Subject Descriptors

I.2.4 [**Knowledge Representation Formalisms and Methods**]: Representation languages; F.4.1 [**Mathematical Logic**]: Model theory.

General Terms

Languages, Standardization, Theory

Keywords

Semantic Web, representation, model-theoretic semantics

1. INTRODUCTION

In the short span of its existence, the World Wide Web has resulted in a revolution in the way information is transferred between computer applications. It is no longer necessary for humans to set up channels for inter-application information transfer; this is handled by TCP/IP and related protocols. It is also no longer necessary for humans to define the syntax and build parsers used for each kind of information transfer; this is handled by HTML, XML and related standards. However, it is still not possible for applications to interoperate with other applications without some pre-existing, human-created, and outside-of-the-web agreements as to the meaning of the information being transferred.

The next generation of the Web aims to alleviate this problem—making Web resources more readily accessible to automated processes by adding information that describes Web content in a machine-accessible and manipulable fashion. This coincides with the vision that Tim Berners-Lee calls the Semantic Web in his recent book "Weaving the Web" [5].

If such information (often called *meta-data*) is to make resources more accessible to automated agents, it is essential that its meaning can be understood by such agents. For this to be the case there must be some common way of providing meaning for meta-data in the Semantic Web, or at least a common underpinning for providing such meaning. Otherwise agents will not be able to combine meta-data from different sources in the Semantic Web.

WWW2003, May 20–24, 2003, Budapest, Hungary.
ACM 1-58113-680-3/03/0005.xxx.

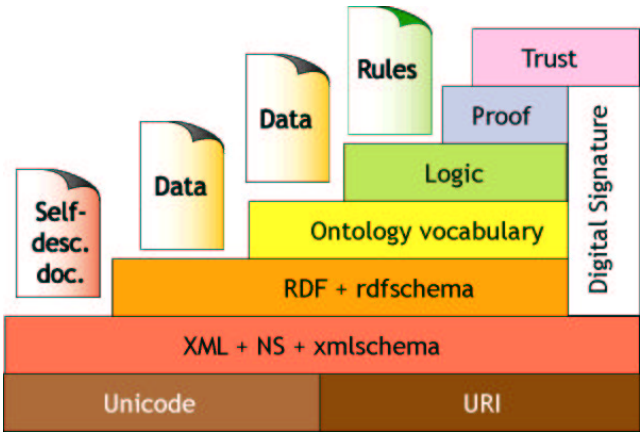

Figure 1: Semantic Web architecture `http://www.w3.org/2000/Talks/1206-xml2k-tbl/`

A common underpinning is especially important for the Semantic Web as it is envisioned to contain several languages, as in Tim Berners-Lee's "layer cake" diagram (Figure 1) first presented at XML 2000. The diagram depicts a Semantic Web Architecture in which languages of increasing power are layered one on top of the other. Unfortunately, the relationships between adjacent layers are not specified, either with respect to syntax or semantics.

The basis of a particular way of providing meaning for meta-data is embodied in the model theory for RDF [22], the language at the base of the Semantic Web. However, this basis is built on an unusual thesis of representation, one that is different from the representation thesis built into most logical languages. Moreover, the RDF thesis of representation has other unusual aspects that make its use as the foundation of representation in the Semantic Web difficult at best. In particular, RDF has a very limited collection of syntactic constructs, and these are treated in a very uniform manner in the semantics of RDF. The RDF thesis requires that no other syntactic constructs are to be used and that the uniform semantic treatment of syntactic constructs cannot be changed, only augmented.

We argue that it would be better, at least in the beginnings of the Semantic Web, to employ a more-standard thesis of representation. This would permit the use of much existing work on (knowledge)

representation and allow the direct employment of existing reasoning tools.

1.1 Ontology Languages

A current focus of activity in the Semantic Web is the construction of an ontology language for the Semantic Web. An ontology typically consists of a hierarchical description of important concepts in a domain, along with descriptions of the properties of (the instances of) each concept. Ontologies will play a pivotal role in the Semantic Web by providing a source of shared and precisely defined terms that can be used in meta-data. The degree of formality employed in capturing these descriptions can be quite variable, ranging from natural language to logical formalisms, but increased formality and regularity clearly facilitates machine understanding.

The recognition of the key role that ontologies are likely to play in the future of the web has led to the extension of web markup languages in order to facilitate content description and the development of web based ontologies, e.g., XML Schema [12], RDF (Resource Description Framework) [26], and RDF Schema (RDFS) [7]. RDFS in particular is recognisable as an ontology/knowledge representation language: it talks about classes and properties (binary relations), range and domain constraints (on properties), and subclass and subproperty (subsumption) relations.

RDFS is, however, a very limited language (at least in some respects), and more expressive power is clearly both necessary and desirable in order to describe resources in sufficient detail. For example, it is useful to be able to state that a property is functional or transitive and it is extremely useful to be able to describe classes in terms of the properties of the individuals that belong to them. Moreover, such descriptions should be amenable to *automated reasoning* if they are to be used effectively by automated processes, e.g., to determine the semantic relationship between syntactically different terms.

A recognition of the limitations of RDFS led to the development of new web ontology languages such as OIL [13, 14], DAML-ONT [24] and DAML+OIL [35]. DAML+OIL has now been submitted to W3C,[1] and is being used by the W3C Web Ontology Working Group[2] as the basis of a new W3C web ontology language called the OWL Web Ontology Language [10].

As a second language for the Semantic Web, a decision has to be made as to whether OWL is to use the thesis of representation employed by RDFS or whether OWL should use a different thesis of representation. Some kind of language layering certainly seems to be a reasonable goal for RDFS and OWL, given that they serve similar functions and share many features (e.g., the ability to describe a hierarchy of classes based on the sub-class relationship). In persuance of this objective, OWL (like DAML+OIL) uses the same syntax as RDF (and RDFS) to represent ontologies. What this means in practice is that OWL uses RDFS resources directly where the required functionality already exists in RDFS, e.g., OWL uses rdfs:subClassOf to assert sub-class relationships, and uses OWL-specific classes and properties to extend RDFS functionality, e.g., the owl:complementOf property is used to add complementation of classes to the language. Moreover, all OWL-specific syntactic constructs are written as combinations of RDF syntactic constructs.

Although this solution satisfies the layering objective from a syntactic perspective, the *semantic* layering of the two languages is more problematical. The difficulty stems from the fact that OWL (like DAML+OIL) is largely based on a Description Logic [2], the semantics of which would normally be given by a "classical" first-order model theory in which individuals are interpreted as elements of some domain (a set), classes are interpreted as subsets of the domain and properties are interpreted as binary relations on the domain. The semantics of RDFS, on the other hand, are given by a non-standard model theory, where individuals, classes and properties are all elements in the domain, property elements have extensions which are binary relations on the domain, and class extensions are only implicitly defined by the extension of the rdf:type property. Moreover, RDFS supports reflection on its own syntax: it is defined in terms of classes and properties which are interpreted in the same way as other classes and properties and whose meaning can be extended by statements in the language. As has been discovered in the OWL effort, language layering is much more complex when different layers subscribe to these two different approaches.

A third approach, which lies somewhere between classical first-order and RDFS, is taken by a group of (relatively) new languages including SKIF [23], L_{base} [20] and Common Logic [8]. Like RDFS, these languages have a non-standard model theory, with predicates being interpreted as elements of the domain. Unlike RDFS, however, classes are treated as unary predicates whose extensions are subsets of the domain, and reflection on language syntax is not supported. It has been suggested that this approach could solve the layering problem by providing a common semantic basis for all Semantic Web languages [20].

In the remainder of this paper we will study these three approaches in more detail, discuss their advantages and disadvantages, present different visions for language layering based on the various approaches, and illustrate the consequences of each choice, both for OWL and for further layers that may come to be added on top of OWL. In particular, we will demonstrate some of the problems associated with the RDFS limited-syntax and limited-semantics approach to the Semantic Web, illustrate how it complicates the semantics of OWL, and show how some of these problems can be partially overcome.

We will take the first-order subset of SKIF (i.e., no row variables, and no variably polyadic predicates) as our exemplar of the third approach as its syntax and semantics have been clearly defined in [23]. For convenience, we will refer to the three approaches as the FOL approach, the RDFS approach and the SKIF approach respectively.

2. BACKGROUND

Representation, by which we mean the relationship between constructs in a language and entities in the world, is an inherently difficult notion. Philosophers, and others, have wrestled with this notion for millenia, shedding much heat, and some light, on the subject. Although the entire notion of representation *is* a fundamental question for the Semantic Web, we will only concern ourselves here with part of the subject, namely the relationship between syntactic constructs in a formal language and objects in some other formal system that is supposed to be an analogue of (part of) the world.

For many languages this representation relationship is handled by means of data models, such as the the XQuery Data Model [15] for XML. Here the meaning of an XML document is a labelled tree, which is supposed to have some closer relationship to some portion of the world than the document itself.

Data models adequately handle this part of the representation relationship for simple languages like XML. However, for languages with more expressive power it is not possible to have a single data model that captures the meaning of a document, or other syntactic construct. Instead the meaning of these languages is often captured by a model-theoretic semantics. In a model-theoretic seman-

[1] `http://www.w3.org/Submission/2001/12/`
[2] `http://www.w3c.org/2001/sw/WebOnt/`

tics there is a many-to-many relationship between documents (or whatever) and *interpretations*. An interpretation is similar to a data model in that it has some closer relationship to the world, but it is different from a data model in that it may have components that do not directly correspond to information in a document. Instead of the very close relationship between document and data model there is instead a much looser relationship between a document and an interpretation, where an interpretation is said to be a model of a document if it is compatable with the document, in some way formally specified by the model theory.

Our three theses of representation differ, in fundamental ways, on what the interpretations of their model-theoretic semantics contain and how these component interact. In addition, the RDFS thesis differs from the other two in how statements can be made.

2.1 DAML+OIL and OWL

Both DAML+OIL and OWL are closely related to very expressive Description Logics (DLs), with a DAML+OIL or OWL ontology corresponding to a DL terminology. DLs are built around individuals, which have membership in classes and are related to other individuals or data values via properties. Data values in DLs play a much more limited role than individuals, but do belong to datatypes. As in a DL, DAML+OIL and OWL classes can be names (URI references in the case of DAML+OIL and OWL) or *expressions*, and a variety of *constructors* are provided for building class expressions.

The meaning of such DLs (and the native meaning of DAML+OIL[3]) is given by a standard model-theoretic semantics [34]. An interpretation consists of a pair $\langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$, where $\Delta^{\mathcal{I}}$ is the domain of discourse (a set) and $\cdot^{\mathcal{I}}$ is an interpretation function. When datatypes are supported (as in DAML+OIL and OWL), the domain is divided into two disjoint sets, the "object domain" $\Delta^{\mathcal{I}}_{\mathcal{O}}$ and the "datatype domain" $\Delta^{\mathcal{I}}_{D}$ such that $\Delta^{\mathcal{I}} = \Delta^{\mathcal{I}}_{\mathcal{O}} \cup \Delta^{\mathcal{I}}_{D}$. The interpretation function $\mathcal{I}$ maps individuals into elements of the object domain, classes into subsets of the object domain, datatypes into subsets of the datatype domain and data values into elements of the datatype domain. In addition, two disjoint sets of properties are distinguished: object properties and datatype properties. The interpretation function maps the former into subsets of $\Delta^{\mathcal{I}}_{\mathcal{O}} \times \Delta^{\mathcal{I}}_{\mathcal{O}}$ and the latter into subsets of $\Delta^{\mathcal{I}}_{\mathcal{O}} \times \Delta^{\mathcal{I}}_{D}$. The interpretation function is extended to composite syntactic constructs much as is done in FOL, e.g., the extension of a conjunction of classes is given by the set intersection of the interpretations of the classes.

In this framework, individuals and data values correspond to FOL constants, classes and datatypes correspond to unary predicates, properties correspond to binary predicates and subclass/property relationships correspond to implication. As DLs include a notion of equality, they are only truely embeddable in FOLs with equality.[4] For example, assertions that John is of type Person, Mary is the mother of John, John is the same individual as Jack, and that Person is a subclass of Animal correspond, respectively, to the FOL sentences $\mathsf{Person}(\mathsf{John})$, $\mathsf{isMotherOf}(\mathsf{Mary}, \mathsf{John})$, $\mathsf{John} = \mathsf{Jack}$, and $\forall x \,.\, \mathsf{Person}(x) \rightarrow \mathsf{Animal}(x)$.

2.2 SKIF

SKIF [23] derives from efforts to formalise the KIF language [16]. Like KIF, SKIF uses a LISP-compatible syntax, but otherwise looks a lot like conventional FOL. In common with Common Logic [8], and $\mathrm{L}_{\mathrm{base}}$ [20], the core of SKIF is standard FOL syntax extended with the ability to use predicates and variables interchangeably. This means that it is possible to use variables in predicate position, e.g., to write sentences like $\exists x \,.\, x(\mathsf{John}) \wedge x(\mathsf{Mary})$ (there exists some predicate that is true of both John and Mary), and to use predicates in variable position, e.g., to write sentences like $P(\mathsf{John}) \wedge Q(P)$ (P is true of John and Q is true of P), or even $P(P)$ (P is true of P).

Like many other logics, the semantics of SKIF is based on interpretations. In SKIF, however, an interpretation is a triple $\langle D, ext, V \rangle$, where D is the domain (a set); V is a function that maps predicates, variables and constants to elements of D; and ext is a function that maps D to sets of tuples over D. The interpretation of predicate symbols is different in SKIF than in conventional FOLs. In conventional FOLs (and in DLs) there is a mapping directly from predicate symbols (class and property names in DLs) into a set of tuples. In SKIF, on the other hand, meaning is given to predicates as predicates by first mapping the predicate symbol into an element of the domain of discourse via V. The domain element is then mapped into sets of tuples via ext.

Because of this double mapping SKIF sentences do *not* have the same meaning as the corresponding FOL (or higher-order logic in the case that a variable occurs in predicate position). Hayes and Menzel [23] give a transformation from SKIF sentences into FOL sentences which, it is claimed, is truth preserving, i.e., the resulting FOL sentence should have an (FOL) interpretation iff the original SKIF sentence had a (SKIF) interpretation. The transformation uses a well known technique for embedding higher-order syntax in FOL. Each SKIF term of the form $P(x_1, \ldots, x_k)$ (and recursively its sub-terms) is replaced with a term $Holds_k(P, x_1, \ldots, x_k)$, where $Holds_0, \ldots, Holds_n$ are newly introduced predicates with arity $1, \ldots, n+1$ respectively.

It is easy to see, however, that the transformation is *not* truth preserving for FOL with equality: the sentence $\forall x, y \,.\, (x = y) \wedge P(x) \wedge \neg Q(x)$ is FOL satisfiable, but the transformed sentence $\forall x, y \,.\, (x = y) \wedge Holds_1(P, x) \wedge \neg Holds_1(Q, x)$ is *not* satisfiable (because $P = Q$, giving $Holds_1(P, x) \wedge \neg Holds_1(P, x)$.[5]

2.3 RDF, RDFS, and the RDF Model Theory

The Resource Description Framework (RDF) [26] and its schema extension, the RDF Schema Specification (RDFS) [7], form the lowest semantic layer of the Semantic Web. (The lower layers in the Semantic Web Architecture are only syntactic layers, in that any semantics provided therein is not carried over to the Semantic Web.)

Initially RDF and RDFS had no formal model theory, nor any formal meaning at all. This made them unlikely foundations for the Semantic Web, and even led to disagreements about the meaning of parts of the language, e.g., whether multiple range and domain constraints on a single property should be interpreted conjunctively or disjunctively. Recently, however, a model theory [22] has been proposed for RDF and RDFS as part of the workings of the W3C RDF Core Working group[6].

The model theory starts with RDF triples [19], an intermediate form (or abstract syntax) for RDF knowledge bases, where many of the syntactic peculiarities of the XML syntax for RDF have been eliminated. An RDF triple consists of a subject, a predicate, and an object. The subject is either a URI reference or a blank node—essentially either a constant or a globally existentially quantified

[3] OWL has two different but closely related semantics, one in this style and one in an RDF style.

[4] As equality is present in most representation languages we will assume the presence of a notion of equality in any semantic foundation.

[5] A more complex translation appears to solve these problems.

[6] `http://www.w3.org/2001/sw/RDFCore/`

variable. The predicate is also a URI reference—essentially a constant representing a binary predicate, called a property in RDF. The object is either a URI reference, a blank node, or a data value, e.g., an integer. RDF triples are generally written in subject, predicate, object order as in John loves Mary.

An interpretation in the RDF model theory is a triple $\langle IR, IEXT, IS\rangle$,[7] where IR is the domain (of resources); IS is a function that maps URI references to elements of IR; and $IEXT$ is a function from IR to $IR \times IR$. The interpretation of properties in RDF is similar to the interpretation of predicates in SKIF, as properties are first mapped to domain elements and only then to a set of tuples. However, the RDF model theory has only *binary* predicates, not predicates of varying arity, as does SKIF.

Because RDF has only triples in its syntax and thus only binary properties in its semantics, it has to encode many of what one might expect to be its logical constants and syntactic constructs. This shows up most vividly in RDFS.

RDFS is built around the notion of a collection of interrelated classes. Class membership in RDFS, which might be expected to be represented as unary predication, is encoded in triples whose predicate is the special URI references `rdf:type`, whose subject is the class member, and whose object is the class itself. Inclusion between classes, which might be expected to be represented as implication, is encoded in triples whose predicate is the special URI reference `rdfs:subClassOf`, whose subject is the smaller class, and whose object is the larger class. Supplying domains and ranges for RDFS properties is handled by triples whose predicate is `rdfs:domain` or `rdfs:range`, respectively.

To make all these encodings work, an extension to the basic RDF model theory is needed. This extension incorporates conditions on interpretations that force the meaning of the special URI references used to encode the above notions to have the appropriate characteristics. For example, $IEXT(IS($`rdfs:subClassOf`$))$ is required to be transitive in the model theory extension.

This trick of encoding works relatively well for RDF and RDFS, but, as we well see below, causes severe problems for more-expressive languages.

3. THE FOL THESIS

The layering of languages is not a new idea. FOL itself could be described as being layered on top of propositional logic: it extends propositional logic with new syntax for quantifiers, it extends propositional logic semantics with interpretations of quantified terms, and it preserves the meaning of propositional logic sentences.

The extension of propositional logic to FOL gives greatly increased expressive power, but at the cost of greatly increased computational cost: determining the satisfiability of a propositional logic sentence is known to be decidable, and to have NP-complete complexity with respect to the size of the input [9], whereas for FOL this problem is known to be undecidable[8] [32]. Many intermediate "layers" have been studied, however, with Horn clauses, the two variable fragment [18], the guarded fragment [1], and numerous description, modal and dynamic logics known to be decidable and to have complexities (for the satisfiability problem) ranging from Polynomial to NExpTime-complete [11, 29].

[7]Recent changes to the RDF specification have modified the technical details of the RDF model theory so this is no longer quite true. However, the points presented in this paper do not depend on these changes, so this old (and simpler) version is still used.

[8]It is impossible to devise an algorithm that will determine the truth value of arbitrary FOL sentences in a finite number of steps.

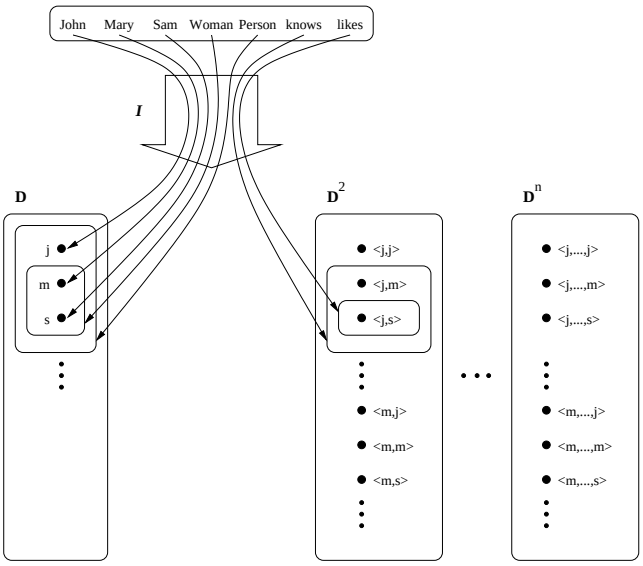

Figure 2: The FOL Universe

This established hierarchy of languages could be taken as a starting point for a layered architecture of Semantic Web languages. In this framework, classes and properties are always taken to correspond to unary and binary predicates as described in Section 2.1. The main difference between the various layers would be in the available syntax: higher layers would provide more (or less restricted) syntax, and increased expressive power, right up to full FOL. (It would even be possible to go further, e.g., to higher-order logic or other more-powerful logics that are extensions of FOL.) Languages in this framework could be given a semantics via a bespoke (FOL style) model theory, or could simply rely on their correspondence with FOL.

The basic thesis of this FOL vision of the Semantic Web is that individual names denote elements of a domain of discourse. Names used in other ways, i.e., as predicates including classes and properties, do *not* denote elements of the domain of discourse. Instead, such names denote sets of tuples over this domain. Thus, in the FOL thesis, the only elements of the domain of discourse are the individuals. This state of affairs is illustrated in Figure 2.

There would be no need for a total ordering of languages within this framework: e.g., Horn- and DL-based subsets of FOL could exist side-by-side, each less expressive (and more tractable) than full FOL, but neither contained within the other. All languages would, however, subscribe to the FOL model theory and be subsets of full FOL. In the case of two languages where one is contained within the other, sentences in the less expressive language could be transfered directly into the more expressive one; otherwise sentences in two languages could be compared in the least expressive language that included both of them, which, by definition, would always be a subset of FOL.

A big advantage with this framework is that a great deal is already known about FOL and its various sub-languages. For example, DLs define decidable fragments of FOL with complexities ranging from Polynomial (for CLASSIC [6]) to NExpTime-hard (for $\mathcal{SHOIQ}$ [33]). Moreover, highly optimised implementations are available for many of these languages (e.g., CLASSIC, FaCT [25] and Racer [21]); these implementations exploit the features of the particular languages and of typical ontology based reasoning problems in order to achieve better performance. Similar

claims can be made for "rule" languages, i.e., those based on Horn clauses [3].

Highly optimised implementations of full FOL are also available (e.g., Vampire [31], E-SETHEO [28]) and can be exploited in an effective manner due to the direct correspondence of ontologies with FOL sentences.

3.1 RDFS and FOL

One problem with the above architecture is that RDFS already falls outside the proposed framework: as we have seen in Section 2.3, it has a SKIF style model theory that differs from the FOL model theory. RDFS can, however, be transformed into FOL, using the standard trick of employing a *Holds* predicate. This trick, however, takes the translation of RDFS ontologies outside the decidable fragments mentioned above. From the FOL point of view, this is a serious design defect: although RDFS seems to be a very simple language (much too simple for many Semantic Web applications), its transformation into FOL is not included in any of the above mentioned decidable fragments.

It is interesting, however, to consider the "FOL subset" of RDFS. This language would correspond to RDFS with a reinterpretation of the meta-classes and other structural vocabulary of RDFS. Classes such as rdfs:Class and rdf:Property would be removed. Vocabulary such as rdf:type and rdfs:domain would simply be a way of stating the appropriate FOL statements and would no longer denote elements of the domain of discourse. The resulting language would be equivalent to a very simple description logic, and would also be expressible in Horn languages. This language would form an ideal base layer within the FOL framework and would, via progressive extensions of the syntax, allow for a full semantic layering of languages such as DAML+OIL and OWL.

3.2 Beyond FOL

For applications where even the full power of FOL is not enough, it would be natural to extend the framework to higher-order logics (HOL). This would also provide for some of the features seen in RDFS, in particular the use of predicates in variable position, while still maintaining the desirable syntactic and semantic layering characteristics of the framework: FOL sentences are syntactically valid and semantically equivalent HOL sentences. Moreover, although HOLs are computationally highly intractable, they are relatively well understood, and reasoning systems for HOLs do already exist, e.g., HOL [17] and Isabel [30].

4. THE SKIF THESIS

In the SKIF thesis, one of the SKIF style languages (e.g., $\mathrm{L}_{\mathrm{base}}$) is taken to be the semantic foundation for all other Semantic Web languages. In the same way as for the FOL architecture, languages within the SKIF framework could be given their own (SKIF style) model theoretic semantics, or could simply rely on their correspondence with SKIF. This thesis is being promoted by Hayes and Guha, who state

> The semantics of each [Semantic Web Language] L_i is defined by specifying how expressions in the L_i map into equivalent expressions in $\mathrm{L}_{\mathrm{base}}$ [20].

This thesis is very close to the FOL thesis of the previous section, but with standard FOL replaced by $\mathrm{L}_{\mathrm{base}}$.

The difference between the FOL and SKIF visions of the Semantic Web is thus constituted just of the differences between the fundamental assumptions in their model theories. In both model theories individual names denote elements of the domain of discourse. However, in SKIF the domain of discourse *also* includes elements

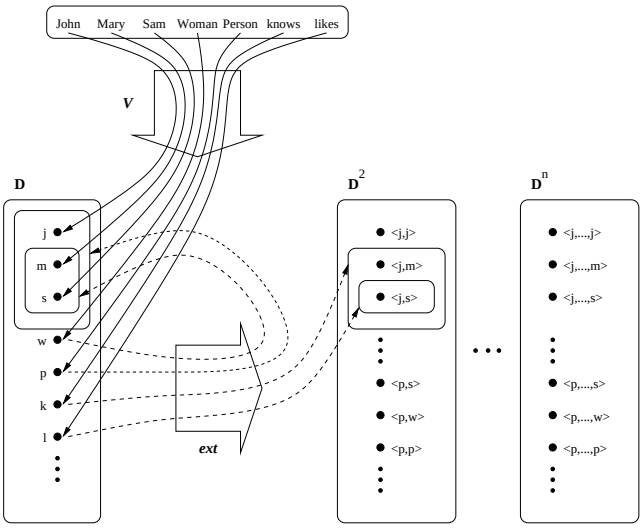

Figure 3: The SKIF Universe

that correspond to predicates, and predicate names are first mapped into these domain elements and only then mapped into sets of tuples. This state of affairs is illustrated in Figure 3.

One obvious advantage of using SKIF rather than FOL to provide meaning for the Semantic Web is that SKIF seems to be a better fit with RDFS in that it supports the use of predicates in variable position (e.g., classes as instances of other classes). However, as we will see below, RDFS does not really fit within SKIF.

One problem with using SKIF is that little is known about SKIF and even less is known about sub-languages of SKIF, except for those that completely correspond (not just in syntax, but also in semantics) to sub-languages of FOL. There has been little or no study of DL-based subsets of SKIF, for example.

Another problem with this framework is that there are no native reasoners for SKIF. It is true that using the transformation into FOL allows FOL reasoners to be (mostly correct) reasoners for SKIF, but this transformation may result in very slow reasoning as it interferes with many of the optimisations used in FOL reasoners. There are also no native reasoners for Horn or DL subsets of SKIF, so the only way of dealing with such sub-languages would be to employ a FOL reasoner, and, moreover, to use an optimisation-destroying transformation.

4.1 RDFS and SKIF

RDFS is even more "liberal" than SKIF in that it allows for (and even employs in its own specification) reflection on its own syntax. For example, in RDFS the type membership and subclass relationships are not distinguished from other predicates, and can occur as arguments of other predicates. This means that RDFS triples such as John `rdf:type` Person and Person `rdfs:subClassOf` Animal cannot be mapped into SKIF as $\mathsf{Person}(\mathsf{John})$ and $\forall x.\mathsf{Person}(x) \rightarrow \mathsf{Animal}(x)$, but instead must be mapped as $\texttt{rdf:type}(\mathsf{Person}, \mathsf{John})$ and $\texttt{rdfs:subClassOf}(\mathsf{Person}, \mathsf{Animal})$.

This feature of RDFS may seem relatively harmless at first sight, but in fact has serious repercussions. In particular, the meaning of predicates such as `rdfs:subClassOf` is not given directly by the mapping into SKIF (as logical implication), but must be explicitly stated, or *axiomatised*, so that they can themselves be the object of other predicates. E.g., the fact that `rdfs:subClassOf`

is transitive and reflexive must be stated in axioms such as:

$$\forall x, y, z \;.\; \texttt{rdfs:subClassOf}(x, y) \wedge \texttt{rdfs:subClassOf}(y, z) \rightarrow \texttt{rdfs:subClassOf}(x, z)$$
$$\forall x \;.\; \texttt{rdfs:subClassOf}(x, x)$$

while the relationship between `rdfs:subClassOf` and `rdf:type` must be captured in an axiom such as

$$\forall x, y \;.\; (\forall z. \texttt{rdf:type}(z, x) \rightarrow \texttt{rdf:type}(z, y)) \iff \texttt{rdfs:subClassOf}(x, y).$$

When layering a language such as DAML+OIL or OWL on top of RDFS, a similar mapping must be used if the semantics of DAML+OIL or OWL are to correspond with those of RDFS. As DAML+OIL and OWL are more expressive languages than RDFS, a larger and more complex axiomatisation is required in order to capture the meaning of its additional constructs (e.g., cardinality constraints and restrictions). It is notoriously difficult to get such axiomatisations right, and even more difficult to prove that they are right.[9]

While this is not strictly a problem with SKIF, so much as a problem with RDFS, the fact that the SKIF approach does not provide a direct solution to the problem of layering on top of RDFS does reduce its attractiveness.

5. THE RDFS THESIS

As indicated above, RDFS does not fit within either the FOL or the SKIF thesis. Instead RDFS has its own thesis of knowledge representation. This thesis can perhaps best be summed up as

> All syntax is RDF triples and all RDF triples are equal.

That is:

1. All Semantic Web languages must use only RDF syntax, either the encoding of RDF in XML known as RDF/XML [4] or RDF triples.

2. The meaning of RDF triples in all Semantic Web languages must be compatible with the meaning given to them in the RDFS model theory.

This thesis has dramatic consequences, both semantic and syntactic. The semantic consequences of the RDFS thesis include that all properties (predicates) are elements of the domain of discourse, as in the SKIF thesis, and all semantic relationships are reducible to properties. Part of this state of affairs is illustrated in Figure 4.

Figure 4 does not, however, capture the entirety of the semantic portion of the RDFS thesis. The semantic portion of the RDFS thesis includes the encoding of unary predicates in the binary rdf:type property and the encoding of any trinary or higher arity predicates as several (binary) properties.

Another portion of the RDFS thesis is that all syntax is (reducible to) RDF triples, and, moreover, that all RDF triples denote property relationships between elements of the domain of discourse. This part of the RDFS thesis is, of course, not much of a burden in RDF and RDFS, but it does cause severe problems in more expressive Semantic Web languages.

Even languages as simple as DAML+OIL or OWL have considerable problems with this requirement. To follow the RDFS thesis,

[9]FOL reasoners can be used to find obvious errors in such axiomatisations [36], but failure to find errors does not prove that an axiomatisation correctly captures the intended semantics (and, due to the incompleteness of FOL reasoners, does not even prove that no errors exist).

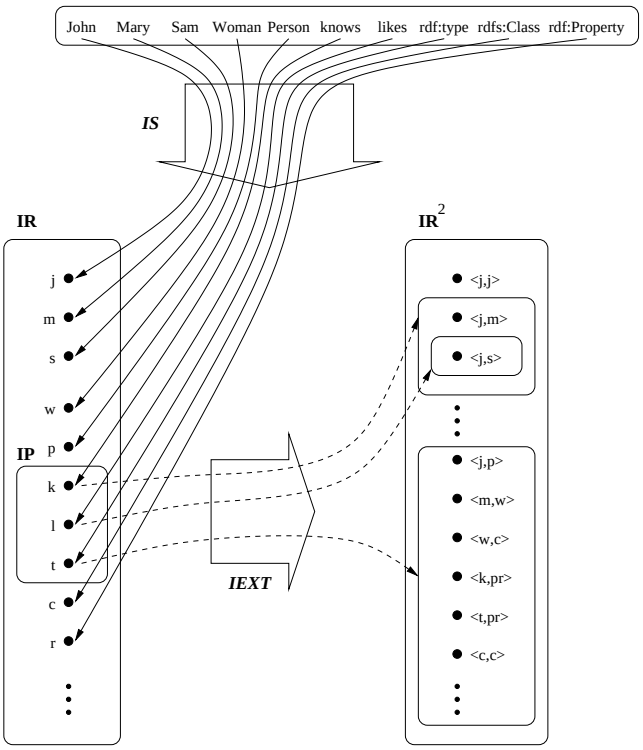

Figure 4: The RDFS Universe

the syntactic constructs of such languages have to be encoded in RDF triples and these syntactic RDF triples end up having semantic consequences. An RDFS-compatiable model theory for such a language (e.g., the RDFS-compatible model theory for OWL) has to treat these semantic consequences extremely carefully, resulting in a complex model theory. In an attempt to sidestep any negative consequences of the complex model theory there are two views of OWL in the RDFS-compatible model theory for OWL—one where the RDF triples that correspond to OWL syntax are outside of the purview of OWL and one where these RDF triples are in the purview of OWL. The first view of OWL corresponds to a more-standard stance, and has been shown to be equivalent to an expressive but standard description logic. In the second view of OWL, reasoning is undecidable and would have to be implemented by a general first-order reasoner (via an optimisation-destroying transformation).

Languages with greater expressive power than DAML+OIL or OWL would have even worse problems with the RDFS thesis. They would have to represent syntactic constructs such as disjunction as collections of RDF triples, resulting in a model theory that would have to account for the presence of such RDF triples in the semantics. Even worse would be the representation of quantification. Incompatibility with more expressive constructs would seem to be a fundamental, and possibly fatal flaw, in the RDFS thesis.

6. ANALYSIS

Having presented three different theses of (knowledge) representation, the question naturally arises as to which is most suitable as the basis for the Semantic Web.

The FOL thesis has many obvious advantages. Perhaps the most important of these is that it is based on a logical formalism that has been the subject of extensive and intensive study over the course of

nearly a century. There exists, as a result, a great wealth of theoretical knowledge and practical experience with respect to FOL.

From a theoretical point of view, it is well known that reasoning is FOL is undecidable. There are, however, many decidable subsets of FOL, and these have also been the subject of extensive investigations. As a result, not only is the theoretical (worst case) complexity known for many of these languages, but a variety of algorithms have been devised with different characteristics with respect to completeness, worst case performance and typical case performance. This knowledge is advantageous when it comes to language layering as it allows different layers to be designed so as to satisfy particular requirements with respect to, e.g., decidability and complexity of reasoning, and the availability of suitable reasoning algorithms. DAML+OIL, for example, was carefully designed so that (subsumption/satisfiability) reasoning was decidable.

From a practical point of view, many reasoning systems have been implemented for FOL and its sub-languages. Modern implementations are often highly optimised so that they perform well with a wide range of problems. This means that applications using FOL based languages can save very significant implementation effort by exploiting existing highly optimised reasoners. DAML+OIL applications, for example, can exploit DL reasoners such as FaCT or Racer, while more expressive languages would be able to exploit FOL reasoners such as Vampire or E-SETHEO.

Moreover, because all the languages within this framework subscribe to the FOL model theory, they can all be directly mapped into FOL, i.e., with individual names being mapped to constants, and other names (e.g., classes and properties) mapped to predicates. This means that the mapping into FOL provides complete semantic interoperability between languages within the framework. For example, a Horn clause based ontology might include a statement of the form

$$\mathsf{uncle}(x, z) \leftarrow \mathsf{father}(x, y) \wedge \mathsf{brother}(y, z),$$

which asserts that the composition of father and brother relationships implies an uncle relationship, while a DL based ontology might include a statement of the form

$$\mathsf{RichUncle} \equiv \exists \mathsf{uncle}.\mathsf{Rich},$$

which asserts that RichUncle is the class of objects having at least one uncle that is Rich. The mapping of both statements into FOL sentences gives

$$\forall x, y, z.\mathsf{father}(x, y) \wedge \mathsf{brother}(y, z) \rightarrow \mathsf{uncle}(x, z)$$
$$\forall x.\mathsf{RichUncle}(x) \iff \exists y.\mathsf{uncle}(x, y) \wedge \mathsf{Rich}(y),$$

which provides for complete semantic interoperability. For example, from $\mathsf{father}(\mathsf{Mary}, \mathsf{John})$, $\mathsf{brother}(\mathsf{John}, \mathsf{Peter})$ and $\mathsf{Rich}(\mathsf{Peter})$, we would be able to conclude $\mathsf{RichUncle}(\mathsf{Mary})$.

The obvious disadvantage of the FOL thesis is that treating classes like individuals (classes as instances) is not possible until the language is extended beyond FOL into HOL, and although HOL reasoners are available, their performance may not be acceptable in ontology applications.

Moreover, FOL is a relatively poor fit with RDFS due to their fundamentally different semantic foundations. Although RDFS can be mapped into FOL via a suitable axiomatisation, this does *not* lead to semantic interoperability with languages that have been directly mapped. For example, if the RDF triples Mary father John, John brother Peter and Peter `rdf:type` Rich, were mapped into FOL using a *holds* predicate (i.e., to give $holds(\mathsf{father}, \mathsf{Mary}, \mathsf{John})$ etc.), then we would obviously *not* be able to conclude $\mathsf{RichUncle}(\mathsf{Mary})$.

The most obvious advantage of the SKIF thesis is that the unrestricted mixing of classes and instances is a basic feature of the language, and does not require HOL extensions. This also seems to make SKIF a better fit with RDFS.

Like FOL, SKIF can be used as the basis for semantic interoperability with respect to a family of sub-languages. Little is known, however, about SKIF sub-languages. As we saw in Section 4.1, even RDFS cannot be directly mapped into SKIF, and requires some kind of axiomatisation. Moreover, although languages like DAML+OIL and OWL would seem to be amenable to a direct mapping into SKIF, such a mapping would *not* lead to semantic interoperability with RDFS for the same reason that direct mappings into FOL of its sub-languages would not lead to semantic interoperability with a *holds* style mapping of RDFS.

An obvious disadvantage with SKIF is that it is relatively new, and its properties are not nearly so well understood as those of FOL. For example, it was believed up until recently that SKIF sentences that were also syntactically valid as FOL had the same meaning whether interpreted as SKIF or as FOL. As we saw in section 2.2, this is not universally true.[10] The syntactic similarity of SKIF and FOL could even be seen as dangerously deceptive—users may be tempted to write SKIF believing that it has the same meaning as FOL.

From a practical point of view, there are no native reasoners for SKIF, and little or nothing is known about the computational properties of, or reasoning algorithms for, sub-languages of SKIF. Applications using SKIF-based languages would, therefore, either have to implement their own reasoners or use full FOL reasoners via a potentially performance-damaging mapping from SKIF.

The RDFS thesis is that interoperability between layered languages can best be achieved by using RDFS syntax. It is, however, a gross over-simplification to imagine that mappings into a single language automatically provide semantic interoperability. As we have seen above, this is not true when using either FOL or SKIF as the common language. The primitive nature of RDFS means that embedding more more expressive languages in RDFS will require more complex mappings, with a wide range of different and semantically incompatible mapping techniques being possible.

For example, both OIL and OWL have an RDF triple syntax. Completely different approaches were taken, however, to the mapping of the various logical constructs: sets of classes in a disjunction are represented in OIL by using a hasOperand property to link them directly to a single node of type `oil:OR`, while in OWL such sets are represented using a list syntax with `first` and `rest` properties. Mixing the RDF triples obtained via the two mappings would provide, at best, very limited semantic interoperability. Similarly, a mapping of typed predicate calculus into RDF triples using reification has been proposed [27] that is not semantically compatible with either the OIL or OWL mappings.

As discussed in Section 5, the requirement of the RDFS thesis that the syntax of all Semantic Web languages be encoded as RDF triples, and have the semantic consequences that RDFS gives to RDF triples, is basically incompatible with more-expressive languages. One possible benefit of this approach is that RDF tools can be used to parse all Semantic Web languages. It is, however, difficult to see any great benefit in the parsing of an *encoding* of complex syntactic constructs, let alone a benefit that would outweigh the problems inherent in this approach.

7. DISCUSSION

[10] Even if it is possible to repair this problem, its discovery is indicative of the relatively new and untested nature of the language.

As we have seen, the formal meaning of Semantic Web (ontology) languages is of crucial importance if automated agents are to exchange, understand and exploit Semantic Web metadata. Moreover, if the Semantic Web is to contain several language layers, as is envisioned, a common semantic underpinning will greatly facilitate interoperability between the different layers. In this paper we have studied three different approaches to providing such a common semantic underpinning: one based on conventional first order logic, one based on SKIF/L_{base}, and one based on RDF/RDFS.

Currently, it seems to be widely assumed that RDF/RDFS will provide such an underpinning: RDF is already a W3C Recommendation, and the RDF/RDFS based web architecture "layer cake" is ubiquitous. As we have seen, however, it is far from clear that this represents the best solution, and in fact it leads to serious problems when trying to layer more expressive languages on top of RDFS. Moreover, this approach does not lead directly to any "computational pathway", i.e., it is not clear if/how applications would be able to reason with languages layered on top of RDFS.

An alternative approach is to use conventional first order logic as the semantic underpinning. As we have seen, this approach has many advantages: FOL is well established and well understood; it naturally lends itself to the development of a family of languages based on various FOL subsets offering different tradeoffs with respect to expressive power, complexity and computability; and the direct mapping of such languages into (subsets of) FOL also provides immediate semantic interoperability (e.g., between Horn and DL based languages). The FOL approach also provides the most straightforward computational pathway: reasoning in FOL and its sublanguages is well understood, and applications would even be able to exploit existing highly optimised reasoners for FOL and many of its subset languages. This approach is not directly compatible with RDFS, but it is compatible with a simplified version of RDFS (the FOL subset of RDFS). This language would form an ideal base layer within the FOL framework and would allow for a full syntactic and semantic layering of languages such as DAML+OIL (and the DL fragment of OWL).

The third approach, based on SKIF/L_{base}, can be seen as a compromise between the FOL and RDF/RDFS approaches. It is similar to the FOL approach, with SKIF/L_{base} providing the semantic underpinning instead of FOL. SKIF style languages are, however, much less well understood than FOL, and little or nothing is known about SKIF sub-languages. SKIF does provide for the mixing of classes and instances, but this is not enough to allow a direct mapping of RDFS into SKIF. Therefore, in order to provide for semantic interoperability between languages layered on top of RDFS, either the base layer would have to be a SKIF subset of RDFS, or all languages would have to be mapped into SKIF using an RDFS compatible axiomatisation. Moreover, the computational pathway for SKIF is currently restricted to a potentially performance damaging mapping into FOL, a problem that would be further exacerbated if mappings into SKIF were via an axiomatisation (as would be the case for RDFS).

It seems clear that the FOL approach is in many respects the most attractive. It remains to be seen, however, if it is too late to issue a product recall on the Semantic Web bandwagon in order to carry out a safety modification on the RDF/RDFS component.

Acknowledgements

Thanks to Andrei Voronkov for helpful comments and discussions.

8. REFERENCES

[1] H. Andréka, J. van Benthem, and I. Németi. Modal languages and bounded fragments of predicate logic. *Journal of Philosophical Logic*, 27:217–274, 1998.

[2] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, implementation, and applications*. Cambridge University Press, to appear.

[3] C. Baral and M. Gelfond. Logic programming and knowledge representation. *Journal of Logic Programming*, 19,20:73–148, 1994.

[4] D. Beckett. Rdf/xml syntax specification (revised). W3C Working Draft, Mar. 2002. `http://www.w3.org/TR/rdf-syntax-grammar/`.

[5] T. Berners-Lee. *Weaving the Web*. Harper, San Francisco, 1999.

[6] A. Borgida and P. F. Patel-Schneider. A semantics and complete algorithm for subsumption in the CLASSIC description logic. *Journal of Artificial Intelligence Research*, 1:277–308, June 1994.

[7] D. Brinkley and R. V. Guha. Resource description framework (RDF) schema specification 1.0. W3C Candidate Recommentation, Mar. 2000. `http://www.w3.org/TR/2000/CR-rdf-schema-20000327/`.

[8] Common Logic Working Group. Common logic: Abstract syntax and semantics. `http://cl.tamu.edu/discuss/cl-syntax-semantics.html`, 2002.

[9] S. A. Cook. The complexity of theorem-proving procedures. In *Conference Record of Third Annual ACM Symposium on Theory of Computing*, pages 151–158, 1971.

[10] M. Dean, D. Connolly, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein. OWL web ontology language 1.0 reference, July 2002. `http://www.w3.org/TR/owl-ref/`.

[11] F. M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. *Information and Computation*, 134:1–58, 1997.

[12] D. C. Fallside. XML schema part 0: Primer. W3C Recommentation, May 2001. `http://www.w3.org/TR/xmlschema-0/`.

[13] D. Fensel, I. Horrocks, F. van Harmelen, S. Decker, M. Erdmann, and M. Klein. OIL in a nutshell. In R. Dieng, editor, *Proc. of the 12th European Workshop on Knowledge Acquisition, Modeling, and Management (EKAW 2000)*, number 1937 in Lecture Notes in Artificial Intelligence, pages 1–16. Springer, 2000.

[14] D. Fensel, F. van Harmelen, I. Horrocks, D. L. McGuinness, and P. F. Patel-Schneider. OIL: An ontology infrastructure for the semantic web. *IEEE Intelligent Systems*, 16(2):38–45, 2001.

[15] M. Fernández and J. Marsh. The XQuery 1.0 and XPath 2.0 data model. W3C Working Draft., June 2001. `http://www.w3.org/TR/query-datamodel/`.

[16] M. R. Genesereth and R. E. Fikes. Knowledge interchange format version 3.0 reference manual. Report Logic 92-1, Stanford Logic Group, June 1992. Also available at `http://logic.stanford.edu/sharing/papers/kif.ps`.

[17] M. J. C. Gordon and T. F. Melham, editors. *Introduction to HOL: A theorem proving environment for higher order logic*. Cambridge University Press, 1993.

[18] E. Grädel, P. G. Kolaitis, and M. Y. Vardi. On the decision

problem for two-variable first-order logic. *Bulletin of Symbolic Logic*, 3(1):53–69, 1997.

[19] J. Grant and D. Beckett. RDF test cases. W3C Working Draft, Apr. 2002. http://www.w3.org/TR/rdf-testcases/.

[20] R. V. Guha and P. Hayes. LBase: Semantics for languages of the semantic web. W3C NOT-A-Note, Aug. 2002. http://www.w3.org/2002/06/lbase/.

[21] V. Haarslev and R. Möller. RACER system description. In *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2001)*, volume 2083 of *Lecture Notes in Artificial Intelligence*, pages 701–705. Springer, 2001.

[22] P. Hayes. RDF model theory. W3C Working Draft, Apr. 2002. http://www.w3.org/TR/rdf-mt/.

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

[24] J. Hendler and D. L. McGuinness. The darpa agent markup language". *IEEE Intelligent Systems*, 15(6):67–73, 2000.

[25] I. Horrocks. Using an expressive description logic: FaCT or fiction? In *Proc. of the 6th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'98)*, pages 636–647, 1998.

[26] O. Lassila and R. R. Swick. Resource description framework (RDF) model and syntax specification. W3C Recommentation, Feb. 1999. http://www.w3.org/TR/REC-rdf-syntax/.

[27] D. McDermott and D. Dou. Representing disjunction and quantifiers in RDF. In *Proc. of the 2002 International Semantic Web Conference (ISWC 2002)*, number 2342 in LNCS, pages 250–263, 2002.

[28] M. Moser, O. Ibens, R. Letz, J. Steinbach, C. Goller, J. Schumann, and K. Mayr. SETHEO and e-SETHEO - the CADE-13 systems. *Journal of Automated Reasoning*, 18(2):237–246, 1997.

[29] L. Pacholski, W. Szwast, and L. Tendera. Complexity results for first-order two-variable logic with counting. *SIAM J. on Computing*, 29(4):1083–1117, 2000.

[30] L. C. Paulson. *Isabelle: A Generic Theorem Prover*. Number 828 in LNCS. Springer, 1994.

[31] A. Riazanov and A. Voronkov. Vampire. In *Proc. of CADE-16*, number 1632 in LNAI, 1999.

[32] U. Schoening. *Logic for Computer Scientists*. Birkhauser, Boston, 1989.

[33] S. Tobies. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*. PhD thesis, LuFG Theoretical Computer Science, RWTH-Aachen, Germany, 2001.

[34] F. van Harmelen, P. F. Patel-Schneider, and I. Horrocks. A model-theoretic semantics for DAML+OIL (March 2001), Mar. 2001. http://www.daml.org/2001/03/model-theoretic-semantics.html.

[35] F. van Harmelen, P. F. Patel-Schneider, and I. Horrocks. Reference description of the DAML+OIL (March 2001) ontology markup langauge, Mar. 2001. http://www.daml.org/2001/03/reference.html.

[36] R. Waldinger. [DAML+OIL] cardinality restriction axioms too weak, Aug. 2001. http://lists.w3.org/Archives/Public/www-rdf-logic/2001Aug/0000.html.