Reviewing the Design of DAML+OIL: An Ontology Language for 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, U.S.A.
pfps@research.bell-labs.com

Frank van Harmelen
Vrije Universiteit
Amsterdam, the Netherlands
Frank.van.Harmelen@cs.vu.nl

Abstract

In the current "Syntactic Web", uninterpreted syntactic constructs are given meaning only by private off-line agreements that are inaccessible to computers. In the Semantic Web vision, this is replaced by a web where both data and its semantic definition are accessible and manipulable by computer software. DAML+OIL is an ontology language specifically designed for this use in the Web; it exploits existing Web standards (XML and RDF), adding the familiar ontological primitives of object oriented and frame based systems, and the formal rigor of a very expressive description logic. The definition of DAML+OIL is now over a year old, and the language has been in fairly widespread use. In this paper, we review DAML+OIL's relation with its key ingredients (XML, RDF, OIL, DAML-ONT, Description Logics), we discuss the design decisions and trade-offs that were the basis for the language definition, and identify a number of implementation challenges posed by the current language. These issues are important for designers of other representation languages for the Semantic Web, be they competitors or successors of DAML+OIL, such as the language currently under definition by W3C.

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" (Berners-Lee 1999).

Eighteenth National Conference on Artificial Intelligence, Edmonton, Alberta, Canada, July 2002.

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. Ontologies will play a pivotal role here by providing a source of shared and precisely defined terms that can be used in such meta-data. An ontology typically consists of a hierarchical description of important concepts in a domain, along with descriptions of the properties of each concept. 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.

Ontologies can be profitably used, e.g., in e-commerce sites (McGuinness 1998), where they can facilitate machine-based communication between buyer and seller, enable vertical integration of markets, and allow descriptions to be reused in different marketplaces; in search engines, where they can help searching to go beyond the current keyword-based approach, and allow pages to be found that contain syntactically different, but semantically similar words/phrases; in Web services (McIlraith, Son, & Zeng 2001), where they can provide semantically richer service descriptions that can be more flexibly interpreted by intelligent agents.

DAML+OIL: An Ontology Language for the Semantic Web

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[1], RDF[2] (Resource Description Framework), and RDF Schema. (See (Decker *et al.* 2000) for the role of each of these on the Semantic Web). RDF Schema (RDFS) in particular is recognisable as an ontology language: it talks about classes and properties, range and domain constraints, and subclass and subproperty relations.

RDFS is, however, a very limited language and more expressive power is clearly both necessary and desirable in order to describe data in sufficient detail. Moreover, such descriptions should be amenable to *automated reasoning* if

[1] `http://www.w3.org/XML/Schema/`
[2] `http://www.w3c.org/RDF/`

they are to be used effectively by automated processes, e.g., to determine the semantic relationships between syntactically different terms.

DAML+OIL is the result of merging DAML-ONT (an early result of the DARPA Agent Markup Language (DAML) programme[3]) and OIL (the Ontology Inference Layer) (Fensel *et al.* 2001), developed by a group of (largely European) researchers, several of whom were members of the European-funded On-To-Knowledge consortium.[4]

Until recently, the development of DAML+OIL has been undertaken by a committee largely made up of members of the two language design teams (and rather grandly titled the Joint EU/US Committee on Agent Markup Languages).[5] More recently, DAML+OIL has been submitted to W3C as a proposal for the basis of the W3C Web Ontology language.[6]

As it is an ontology language, DAML+OIL is designed to describe the *structure* of a domain. DAML+OIL takes an object oriented approach, with the structure of the domain being described in terms of *classes* and *properties*. An ontology consists of a set of *axioms* that assert characteristics of these classes and properties. Asserting that resources are instances of DAML+OIL classes or that resources are related by properties is left to RDF, a task for which it is well suited.

Since the definition of DAML+OIL is available elsewhere,[7] we will not repeat it here. Instead, in the following sections, we will review a number of fundamental design choices that were made for DAML+OIL: foundations in Description Logic, XML datatypes, layering on top of RDFS, comparison with its predecessor OIL, and the role of inference for a Semantic Web ontology language.

Foundations in Description Logic

DAML+OIL is, in essence, equivalent to a very expressive Description Logic (DL), with a DAML+OIL ontology corresponding to a DL terminology. As in a DL, DAML+OIL classes can be names (URI's in the case of DAML+OIL) or *expressions*, and a variety of *constructors* are provided for building class expressions. The expressive power of the language is determined by the class (and property) constructors provided, and by the kinds of axioms allowed.

Figure 1 summarises the constructors in DAML+OIL. The standard DL syntax is used in this paper for compactness as the RDF syntax is rather verbose. In the RDF syntax, for example, Human $\sqcap$ Male would be written as

```
<daml:Class>
  <daml:intersectionOf
       rdf:parseType="daml:collection">
    <daml:Class rdf:about="#Human"/>
    <daml:Class rdf:about="#Male"/>
  </daml:intersectionOf>
</daml:Class>
```

[3] http://www.daml.org/

[4] http://www.ontoknowledge.org/oil

[5] http://www.daml.org/committee

[6] http://www.w3.org/Submission/2001/12/

[7] http://www.daml.org/2001/03/daml+oil-index.html

Constructor	DL Syntax	Example
intersectionOf	$C_1 \sqcap \ldots \sqcap C_n$	Human $\sqcap$ Male
unionOf	$C_1 \sqcup \ldots \sqcup C_n$	Doctor $\sqcup$ Lawyer
complementOf	$\neg C$	$\neg$Male
oneOf	$\{x_1 \ldots x_n\}$	{john, mary}
toClass	$\forall P.C$	$\forall$hasChild.Doctor
hasClass	$\exists P.C$	$\exists$hasChild.Lawyer
hasValue	$\exists P.\{x\}$	$\exists$citizenOf.{USA}
minCardinalityQ	$\geqslant n\, P.C$	$\geqslant$ 2 hasChild.Lawyer
maxCardinalityQ	$\leqslant n\, P.C$	$\leqslant$ 1 hasChild.Male
cardinalityQ	$= n\, P.C$	= 1 hasParent.Female

Figure 1: DAML+OIL class constructors

The meanings of the first three constructors from Figure 1 are just the standard boolean operators on classes. The oneOf constructor allows classes to be defined by enumerating their members.

The toClass and hasClass constructors correspond to slot constraints in a frame-based language. The class $\forall P.C$ is the class all of whose instances are related via the property P only to resources of type C, while the class $\exists P.C$ is the class all of whose instances are related via the property P to at least one resource of type C. The hasValue constructor is just shorthand for a combination of hasClass and oneOf.

The minCardinalityQ, maxCardinalityQ and cardinalityQ constructors (known in DLs as qualified number restrictions) are generalisations of the hasClass and hasValue constructors. The class $\geqslant n\, P.C$ ($\leqslant n\, P.C$, $= n\, P.C$) is the class all of whose instances are related via the property P to at least (at most, exactly) n *different* resources of type C. The emphasis on different is because there is no unique name assumption with respect to resource names (URIs): it is possible that many URIs could name the same resource.

Note that arbitrarily complex nesting of constructors is possible. The formal semantics of the class constructors is given by DAML+OIL's model-theoretic semantics[8] or can be derived from the specification of a suitably expressive DL (e.g., see (Horrocks & Sattler 2001)).

Figure 2 summarises the axioms allowed in DAML+OIL. These axioms make it possible to assert subsumption or equivalence with respect to classes or properties, the disjointness of classes, the equivalence or non-equivalence of individuals (resources), and various properties of properties.

A crucial feature of DAML+OIL is that subClassOf and sameClassAs axioms can be applied to arbitrary class expressions. This provides greatly increased expressive power with respect to standard frame-based languages where such axioms are invariably restricted to the form where the left hand side is a class name, there is only one such axiom per name, and there are no cycles (the class on the right hand side of an axiom cannot refer, either directly or indirectly, to the class name on the left hand side).

A consequence of this expressive power is that all of the class and individual axioms, as well as the uniqueProperty

[8] http://www.w3.org/TR/daml+oil-model

Axiom	DL Syntax	Example
subClassOf	$C_1 \sqsubseteq C_2$	Human $\sqsubseteq$ Animal $\sqcap$ Biped
sameClassAs	$C_1 \equiv C_2$	Man $\equiv$ Human $\sqcap$ Male
subPropertyOf	$P_1 \sqsubseteq P_2$	hasDaughter $\sqsubseteq$ hasChild
samePropertyAs	$P_1 \equiv P_2$	cost $\equiv$ price
disjointWith	$C_1 \sqsubseteq \neg C_2$	Male $\sqsubseteq$ $\neg$Female
sameIndividualAs	$\{x_1\} \equiv \{x_2\}$	{President_Bush} $\equiv$ {G_W_Bush}
differentIndividualFrom	$\{x_1\} \sqsubseteq \neg\{x_2\}$	{john} $\sqsubseteq$ $\neg${peter}
inverseOf	$P_1 \equiv P_2^-$	hasChild $\equiv$ hasParent$^-$
transitiveProperty	$P^+ \sqsubseteq P$	ancestor$^+$ $\sqsubseteq$ ancestor
uniqueProperty	$\top \sqsubseteq \leqslant 1\, P$	$\top \sqsubseteq \leqslant 1$ hasMother
unambiguousProperty	$\top \sqsubseteq \leqslant 1\, P^-$	$\top \sqsubseteq \leqslant 1$ isMotherOf$^-$

Figure 2: DAML+OIL axioms

and unambiguousProperty axioms, can be reduced to subClassOf and sameClassAs axioms (as can be seen from the DL syntax).

As we have seen, DAML+OIL also allows properties of properties to be asserted. It is possible to assert that a property is unique (i.e., functional) and unambiguous (i.e., its inverse is functional). It is also possible to use inverse properties and to assert that a property is transitive.

XML Datatypes in DAML+OIL

DAML+OIL supports the full range of datatypes in XML Schema: the so called primitive datatypes such as string, decimal or float, as well as more complex derived datatypes such as integer sub-ranges. This is facilitated by maintaining a clean separation between instances of "object" classes (defined using the ontology language) and instances of datatypes (defined using the XML Schema type system). In particular, the domain of interpretation of object classes is disjoint from the domain of interpretation of datatypes, so that an instance of an object class (e.g., the individual "Italy") can never have the same denotation as a value of a datatype (e.g., the integer 5), and that the set of object properties (which map individuals to individuals) is disjoint from the set of datatype properties (which map individuals to datatype values).

The disjointness of object and datatype domains was motivated by both philosophical and pragmatic considerations:

- Datatypes are considered to be already sufficiently structured by the built-in predicates, and it is, therefore, not appropriate to form new classes of datatype values using the ontology language (Hollunder & Baader 1991).
- The simplicity and compactness of the ontology language are not compromised: even enumerating all the XML Schema datatypes would add greatly to its complexity, while adding a logical theory for each datatype, even if it were possible, would lead to a language of monumental proportions.
- The semantic integrity of the language is not compromised—defining theories for all the XML Schema datatypes would be difficult or impossible without extending the language in directions whose semantics would be difficult to capture within the existing framework.
- The "implementability" of the language is not compromised—a hybrid reasoner can easily be implemented by combining a reasoner for the "object" language with one capable of deciding satisfiability questions with respect to conjunctions of (possibly negated) datatypes (Horrocks & Sattler 2001).

From a theoretical point of view, this design means that the ontology language can specify constraints on data values, but as data values can never be instances of object classes they cannot apply additional constraints to elements of the object domain. This allows the type system to be extended without having any impact on the ontology language, and vice versa. Similarly, the formal properties of hybrid reasoners are determined by those of the two components; in particular, the combined reasoner will be sound and complete if both components are sound and complete.

From a practical point of view, DAML+OIL implementations can choose to support some or all of the XML Schema datatypes. For supported datatypes, they can either implement their own type checker/validater or rely on some external component. The job of a type checker/validater is simply to take zero or more data values and one or more datatypes, and determine if there exists any data value that is equal to every one of the specified data values and is an instance of every one of the specified data types.

Extending RDF Schema

DAML+OIL is tightly integrated with RDFS: RDFS is used to express DAML+OIL's machine readable specification,[9] and RDFS provides the only serialisation for DAML+OIL. While the dependence on RDFS has some advantages in terms of the re-use of existing RDFS infrastructure and the portability of DAML+OIL ontologies, using RDFS to completely define the structure of DAML+OIL is quite difficult as, unlike XML, RDFS is not designed for the precise specification of syntactic structure. For example, there is no way in RDFS to state that a restriction (slot constraint) should consist of exactly one property (slot) and one class.

[9] `http://www.daml.org/2001/03/daml+oil.daml`

The solution to this problem adopted by DAML+OIL is to define the semantics of the language in such a way that they give a meaning to any (parts of) ontologies that conform to the RDFS specification, including "strange" constructs such as restrictions with multiple properties and classes. The meaning given to strange constructs may, however, include strange "side effects". For example, in the case of a restriction with multiple properties and classes, the semantics interpret this in the same way as a conjunction of all the constraints that would result from taking the cross product of the specified properties and classes, but with the added (and probably unexpected) effect that all these restrictions must have the same interpretation (i.e., are equivalent).

DAML+OIL's dependence on RDFS may also have consequences for the decidability of the language. Decidability is lost when cardinality constraints can be applied to properties that are transitive, or that have transitive sub-properties. (Horrocks, Sattler, & Tobies 1999). There is no way to formally capture this constraint in RDFS, so decidability in DAML+OIL depends on an informal prohibition of cardinality constraints on non-simple properties.

DAML+OIL vs. OIL

From the point of view of language constructs, the differences between OIL and DAML+OIL are relatively trivial. Although there is some difference in "keyword" vocabulary, there is usually a one to one mapping of constructors, and in the cases where the constructors are not completely equivalent, simple translations are possible.

OIL also uses RDFS for its serialisation (although it also provides a separate XML-based syntax). Consequently, OIL's RDFS based syntax would seem to be susceptible to the same difficulties as described above for DAML+OIL. However, in the case of OIL there does not seem to be an assumption that any ontology conforming to the RDFS meta-description should be a valid OIL ontology—presumably ontologies containing unexpected usages of the meta-properties would be rejected by OIL processors as the semantics do not specify how these could be translated into $\mathcal{SHIQ}(\mathbf{D})$. Thus, OIL and DAML+OIL take rather different positions with regard to the layering of languages on the Semantic Web.

Another effect of DAML+OIL's tight integration with RDFS is that the frame structure of OIL's syntax is much less evident: a DAML+OIL ontology is more DL-like in that it consists largely of a relatively unstructured collection of subsumption and equality axioms. This can make it more difficult to use DAML+OIL with frame based tools such as Protégé (Grosso *et al.* 1999) or OilEd (Bechhofer *et al.* 2001) because the axioms may be susceptible to many different frame-like groupings. (Bechhofer, Goble, & Horrocks 2001).

The treatment of individuals in OIL is also very different from that in DAML+OIL. In the first place, DAML+OIL relies wholly on RDF for assertions involving the type (class) of an individual or a relationship between a pair of objects. In the second place, DAML+OIL treats individuals occurring in the ontology (in `oneOf` constructs or `hasValue` restrictions) as true individuals (i.e., interpreted as single elements in the domain of discourse) and not as primitive concepts as is the case in OIL. This weak treatment of the `oneOf` construct is a well known technique for avoiding the reasoning problems that arise with existentially defined classes, and is also used, e.g., in the CLASSIC knowledge representation system (Borgida & Patel-Schneider 1994). Moreover, DAML+OIL makes no unique name assumption: it is possible to explicitly assert that two individuals are the same or different, or to leave their relationship unspecified.

This treatment of individuals is very powerful, and justifies intuitive inferences that would not be valid for OIL, e.g., that persons all of whose countries of residence are Italy are kinds of person that have at most one country of residence:

$$\mathsf{Person} \sqcap \forall \mathsf{residence}.\{\mathsf{Italy}\} \sqsubseteq \leqslant 1\,\mathsf{residence}$$

Inference in DAML+OIL

As we have seen, DAML+OIL is equivalent to a very expressive DL. More precisely, DAML+OIL is equivalent to the $\mathcal{SHIQ}$ DL (Horrocks, Sattler, & Tobies 1999) with the addition of existentially defined classes (i.e., the oneOf constructor) and *datatypes* (often called concrete domains in DLs (Baader & Hanschke 1991)). This equivalence allows DAML+OIL to exploit the considerable existing body of description logic research to define the semantics of the language and to understand its formal properties, in particular the decidability and complexity of key inference problems (Donini *et al.* 1997); as a source of sound and complete algorithms and optimised implementation techniques for deciding key inference problems (Horrocks, Sattler, & Tobies 1999; Horrocks & Sattler 2001); and to use implemented DL systems in order to provide (partial) reasoning support (Horrocks 1998a; Patel-Schneider 1998; Haarslev & Möller 2001).

A important consideration in the design of DAML+OIL was that key inference problems in the language, in particular class consistency/subsumption, to which most other inference problems can be reduced, should be decidable, as this facilitates the provision of reasoning services. Moreover, the correspondence with DLs facilitates the use of DL algorithms that are known to be amenable to optimised implementation and to behave well in realistic applications in spite of their high worst case complexity (Horrocks 1998b; Haarslev & Möller 2001).

Maintaining the decidability of the language requires certain constraints on its expressive power that may not be acceptable to all applications. However, the designers of the language decided that reasoning would be important if the full power of ontologies was to be realised, and that a powerful but still decidable ontology language would be a good starting point.

Reasoning can be useful at many stages during the design, maintenance and deployment of ontologies.

Reasoning can be used to support ontology design and to improve the quality of the resulting ontology. For example, class consistency and subsumption reasoning can be used to check for logically inconsistent classes and (possibly unexpected) implicit subsumption relationships (Bechhofer *et* *al.* 2001). This kind of support has been shown to be particularly important with large ontologies, which are often built and maintained over a long period by multiple authors. Other reasoning tasks, such as "matching" (Baader *et al.* 1999) and/or computing least common subsumers (Baader & Küsters 1998) could also be used to support "bottom up" ontology design, i.e., the identification and description of relevant classes from sets of example instances.

Like information integration (Calvanese *et al.* 1998), ontology integration can also be supported by reasoning. For example, integration can be performed using inter-ontology assertions specifying relationships between classes and properties, with reasoning being used to compute the integrated hierarchy and to highlight any problems/inconsistencies. Unlike some other integration techniques, this method has the advantage of being non-intrusive with respect to the original ontologies.

Reasoning with respect to deployed ontologies will enhance the power of "intelligent agents", allowing them to determine if a set of facts is consistent w.r.t. an ontology, to identify individuals that are implicitly members of a given class etc. A suitable service ontology could, for example, allow an agent seeking secure services to identify a service requiring a userid and password as a possible candidate.

Challenges

Class consistency/subsumption reasoning in DAML+OIL is known to be decidable (as it is contained in the C2 fragment of first order logic (Grädel, Otto, & Rosen 1997)), but many challenges remain for implementors of "practical" reasoning systems, i.e., systems that perform well with the kinds of reasoning problem generated by realistic applications.

Individuals Unfortunately, the combination of DAML+OIL individuals with inverse properties is so powerful that it pushes the worst case complexity of the class consistency problem from ExpTime (for $\mathcal{SHIQ}$/OIL) to NExpTime. No "practical" decision procedure is currently known for this logic, and there is no implemented system that can provide sound and complete reasoning for the whole DAML+OIL language. In the absence of inverse properties, however, a tableaux algorithm has been devised (Horrocks & Sattler 2001), and in the absence of individuals (in extensionally defined classes), DAML+OIL can exploit implemented DL systems via a translation into $\mathcal{SHIQ}$ (extended with datatypes) similar to the one used by OIL. It would, of course, also be possible to translate DAML+OIL ontologies into $\mathcal{SHIQ}$ using OIL's weak treatment of individuals, but in this case reasoning with individuals would not be complete with respect to the semantics of the language. This approach is taken by some existing applications, e.g., OilEd (Bechhofer *et al.* 2001)

Scalability Even without the oneOf constructor, class consistency reasoning is still a hard problem. Moreover, Web ontologies can be expected to grow very large, and with deployed ontologies it may also be desirable to reason w.r.t. a large numbers of class/property instances.

There is good evidence of empirical tractability and scalability for implemented DL systems (Horrocks 1998b; Haarslev & Möller 2001), but this is mostly w.r.t. logics that do not include inverse properties (e.g., $\mathcal{SHF}$ (Horrocks, Sattler, & Tobies 1999)). Adding inverse properties makes practical implementations more problematical as several important optimisation techniques become much less effective. Work is required in order to develop more highly optimised implementations supporting inverse properties, and to demonstrate that they can scale as well as $\mathcal{SHF}$ implementations. It is also unclear if existing techniques will be able to cope with large numbers of class/property instances (Horrocks, Sattler, & Tobies 2000).

Finally, it is an inevitable consequence of the high worst case complexity that some problems will be intractable, even for highly optimised implementations. It is conjectured that such problems rarely arise in practice, but the evidence for this conjecture is drawn from a relatively small number of applications, and it remains to be seen if a much wider range of Web application domains will demonstrate similar characteristics.

New Reasoning Tasks So far we have mainly discussed class consistency/subsumption reasoning, but this may not be the only reasoning problem that is of interest. Other tasks could include querying, explanation, matching, computing least common subsumers, etc. Querying in particular may be important in Semantic Web applications. Some work on query languages for DLs has already been done (Calvanese, De Giacomo, & Lenzerini 1999; Horrocks & Tessaris 2000), and work is underway on the design of a DAML+OIL query language, but the computational properties of such a language, either theoretical or empirical, have yet to be determined.

Explanation may also be an important problem, e.g., to help an ontology designer to rectify problems identified by reasoning support, or to explain to a user why an application behaved in an unexpected manner. As discussed above, reasoning problems such as matching and computing least common subsumers could also be important in ontology design.

Discussion

There are other concerns with respect to the place DAML+OIL has in the Semantic Web. After DAML+OIL was developed, the W3C RDF Core Working Group devised a model theory for RDF and RDFS[10], which is incompatible with the semantics of DAML+OIL, an undesirable state of affairs. Also, in late 2001 W3C initiated the Web Ontology working group[11], a group tasked with developing an ontology language for the Semantic Web. DAML+OIL has been submitted to this working group as a starting point for a W3C recommendation on ontology languages.

A W3C ontology language needs to fit in with other W3C recommendations even more than an independent DAML+OIL would. Work is thus needed to develop a semantic web ontology language, which the Web Ontology

[10] `http://www.w3.org/TR/rdf-mt/`

[11] `http://www.w3.org/2001/sw/WebOnt/`

working group has tentatively name OWL, that layers better on top of RDF and RDFS.

Unfortunately, the obvious layering (that is, using the same syntax as RDF and extending its semantics, just as RDFS does) is not possible. Such an extension results in semantic paradoxes—variants of the Russell paradox. These paradoxes arise from the status of all classes (including DAML+OIL restrictions) as individuals, which requires that many restrictions be present in all models; from the status of the class membership relationship as a regular property (rdf:type); from the ability to make contradictory statements; and from the ability to create restrictions that refer to themselves. In an RDFS-compliant version of DAML+OIL, a restriction that states that its instances have no rdf:type relationships to itself is not only possible to state, but exists in all models, resulting in an ill-formed logical formalism.

The obvious way around this problem, that of using non-RDF syntax for DAML+OIL restrictions, appears to be meeting with considerable resistance so either further education or some other solution is needed.

Conclusion

We have discussed a number of fundamental design decisions underlying the design of DAML+OIL, in particular its foundation in Description Logic, its use of datatypes from XML Schema, its sometimes problematic layering on top of RDF Schema, and its deviations from its predecessor OIL. We have also described how various aspects of the language are motivated by the desire for tractable reasoning facilities.

Although a number of challenges remain, DAML+OIL has considerable merits. In particular, the basic idea of having a formally-specified web language that can represent ontology information will go a long way towards allowing computer programs to interoperate without pre-existing, outside-of-the-web agreements. If this language also has an effective reasoning mechanism, then computer programs can manipulate this interoperability information themselves, and determine whether a common meaning for the information that they pass back and forth is present.

References

Baader, F., and Hanschke, P. 1991. A schema for integrating concrete domains into concept languages. In *Proc. of IJCAI-91*, 452–457.

Baader, F., and Küsters, R. 1998. Computing the least common subsumer and the most specific concept in the presence of cyclic $\mathcal{ALN}$-concept descriptions. In *Proc. of KI'98*, 129–140. Springer-Verlag.

Baader, F.; Küsters, R.; Borgida, A.; and McGuinness, D. L. 1999. Matching in description logics. *J. of Logic and Computation* 9(3):411–447.

Bechhofer, S.; Horrocks, I.; Goble, C.; and Stevens, R. 2001. OilEd: a reason-able ontology editor for the semantic web. In *Proc. of the Joint German/Austrian Conf. on Artificial Intelligence (KI 2001)*, 396–408. Springer-Verlag.

Bechhofer, S.; Goble, C.; and Horrocks, I. 2001. DAML+OIL is not enough. In *Proc. of the First Semantic Web Working Symposium (SWWS'01)*, 151–159. CEUR Electronic Workshop Proceedings, http://ceur-ws.org/.

Berners-Lee, T. 1999. *Weaving the Web*. San Francisco: Harper.

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

Calvanese, D.; De Giacomo, G.; Lenzerini, M.; Nardi, D.; and Rosati, R. 1998. Information integration: Conceptual modeling and reasoning support. In *Proc. of CoopIS'98*, 280–291.

Calvanese, D.; De Giacomo, G.; and Lenzerini, M. 1999. Answering queries using views in description logics. In *Proc. of DL'99*, 9–13. CEUR Electronic Workshop Proceedings, http://ceur-ws.org/Vol-22/.

Decker, S.; van Harmelen, F.; Broekstra, J.; Erdmann, M.; Fensel, D.; Horrocks, I.; Klein, M.; and Melnik, S. 2000. The semantic web: The roles of XML and RDF. *IEEE Internet Computing* 4(5).

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

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

Grädel, E.; Otto, M.; and Rosen, E. 1997. Two-variable logic with counting is decidable. In *Proc. of LICS-97*, 306–317. IEEE Computer Society Press.

Grosso, W. E.; Eriksson, H.; Fergerson, R. W.; Gennari, J. H.; Tu, S. W.; and Musen, M. A. 1999. Knowledge modelling at the millenium (the design and evolution of protégé-2000). In *Proc. of Knowledge acqusition workshop (KAW-99)*.

Haarslev, V., and Möller, R. 2001. High performance reasoning with very large knowledge bases: A practical case study. In *Proc. of IJCAI-01*.

Hollunder, B., and Baader, F. 1991. Qualifying number restrictions in concept languages. In *Proc. of KR-91*, 335–346.

Horrocks, I., and Sattler, U. 2001. Ontology reasoning in the $\mathcal{SHOQ}$(D) description logic. In *Proc. of IJCAI-01*. Morgan Kaufmann.

Horrocks, I., and Tessaris, S. 2000. A conjunctive query language for description logic Aboxes. In *Proc. of AAAI 2000*, 399–404.

Horrocks, I.; Sattler, U.; and Tobies, S. 1999. Practical reasoning for expressive description logics. In Ganzinger, H.; McAllester, D.; and Voronkov, A., eds., *Proc. of LPAR'99*, 161–180. Springer-Verlag.

Horrocks, I.; Sattler, U.; and Tobies, S. 2000. Reasoning with individuals for the description logic $\mathcal{SHIQ}$. In *Proc. of CADE-17*, LNAI, 482–496.

Horrocks, I. 1998a. The FaCT system. In de Swart, H., ed., *Proc. of TABLEAUX-98*, 307–312. Springer-Verlag.

Horrocks, I. 1998b. Using an expressive description logic: FaCT or fiction? In *Proc. of KR-98*, 636–647.

McGuinness, D. L. 1998. Ontological issues for knowledge-enhanced search. In *Proc. of FOIS*, Frontiers in Artificial Intelligence and Applications. IOS-press.

McIlraith, S.; Son, T.; and Zeng, H. 2001. Semantic web services. *IEEE Intelligent Systems* 16(2):46–53.

Patel-Schneider, P. F. 1998. DLP system description. In *Proc. of DL'98*, 87–89. CEUR Electronic Workshop Proceedings, http://ceur-ws.org/Vol-11/.