DAML+OIL: a Reason-able Web Ontology Language

Ian Horrocks

Department of Computer Science, University of Manchester
Oxford Road, Manchester M13 9PL, UK
horrocks@cs.man.ac.uk

Abstract. Ontologies are set to play a key role in the "Semantic Web", extending syntactic interoperability to semantic interoperability by providing a source of shared and precisely defined terms. DAML+OIL is an ontology language specifically designed for use on 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 logical basis of the language means that reasoning services can be provided, both to support ontology design and to make DAML+OIL described Web resources more accessible to automated processes.

1 Introduction

The World Wide Web has been made possible through a set of widely established standards which guarantee interoperability at various levels: the TCP/IP protocol has ensured that nobody has to worry about transporting bits over the wire anymore; similarly, HTTP and HTML have provided a standard way of retrieving and presenting hyperlinked text documents. Applications were able to use this common infrastructure and this has led to the WWW as we know it now.

The current Web can be characterised as the second generation Web: the first generation Web was characterised by handwritten HTML pages; the second generation made the step to machine generated and often active HTML pages. These generations of the Web were meant for direct human processing (reading, browsing, form-filling, etc.). The third generation Web aims to make Web resources more readily accessible to automated processes by adding meta-data annotations that describe their content—this coincides with the vision that Tim Berners-Lee calls the Semantic Web in his recent book "Weaving the Web" [5].

1.1 Ontologies

If meta-data annotations are to make resources more accessible to automated agents, it is essential that their 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.

Examples of the use of ontologies could include:

- in e-commerce sites [25], where ontologies can facilitate machine-based communication between buyer and seller, enable vertical integration of markets (see, e.g., http://www.verticalnet.com/), and allow descriptions to be reused in different marketplaces;
- in search engines [26], where ontologies 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 (see, e.g., http://www.hotbot.com/);
- in Web services [28], where ontologies can provide semantically richer service descriptions that can be more flexibly interpreted by intelligent agents.

2 Ontology Languages

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 [9]. RDF Schema (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 primitive language (the above is an almost complete description of its functionality), and more expressive power would clearly be necessary/desirable in order to describe resources in sufficient detail. 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.

2.1 DAML and OIL

In 1999 the DARPA Agent Markup Language (DAML) program[3] was initiated with the aim of providing the foundations of a next generation "semantic" Web [17]. As a first step, it was decided that the adoption of a common ontology language would facilitate semantic interoperability across the various projects

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

making up the program. RDFS was seen as a good starting point, and was already a proposed World Wide Web Consortium (W3C) standard, but it was not expressive enough to meet DAML's requirements. A new language called DAML-ONT was therefore developed that extended RDF with language constructors from object-oriented and frame-based knowledge representation languages. Like RDFS, DAML-ONT suffered from a rather weak semantic specification, and it was soon realised that this could lead to disagreements, both amongst humans and machines, as to the precise meaning of terms in a DAML-ONT ontology.

At around the same time, a group of (largely European) researchers with aims similar to those of the DAML researchers (i.e., to provide a foundation for the next generation Web) had designed another Web oriented ontology language called OIL (the Ontology Inference Layer) [11, 12]. Like DAML-ONT, OIL had an RDFS based syntax (as well as an alternative XML syntax) and a set of language constructors based on frame-based languages. The developers of OIL, however, placed a stronger emphasis on formal rigor, and the language was explicitly designed so that its semantics could be specified via a mapping to a very expressive description logic, $\mathcal{SHIQ}$ [22].

It became obvious to both groups that their objectives could best be served by combining their efforts, the result being the merging of DAML-ONT and OIL to produce DAML+OIL. The merged language has a formal (model theoretic) semantics that provides machine and human understandability (as well as an axiomatization [13]), and a reconciliation of the language constructors from the two languages.

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). More recently, DAML+OIL has been submitted to W3C[4] and is to form the basis for the W3C's Web ontology language which the Web-Ontology Working Group has been mandated to deliver.[5]

2.2 DAML+OIL

DAML+OIL is an ontology language, and as such 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, e.g., subsumption relationships between classes or properties. Asserting that resources[6] (pairs of resources) are instances of DAML+OIL classes (properties) is left to RDF, a task for which it

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

[5] http://www.w3c.org/2001/sw/WebOnt/

[6] Everything describable by RDF is called a resource. A resource could be Web accessible, e.g., a Web page or part of a Web page, but it could also be an object that is not directly accessible via the Web, e.g., a person. Resources are named by URIs plus optional anchor ids. See http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/ for more details.

is well suited. When a resource r is an instance of a class C we say that r has type C.

From a formal point of view, DAML+OIL can be seen to be equivalent to a very expressive description logic, with a DAML+OIL ontology corresponding to a DL terminology (Tbox). As in a DL, DAML+OIL classes can be names (URIs) 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 supported, and by the kinds of axiom supported.

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$2hasChild.Lawyer
maxCardinalityQ	$\leqslant n P.C$	$\leqslant$1hasChild.Male
cardinalityQ	$=n\, P.C$	$=$1 hasParent.Female

Fig. 1. DAML+OIL class constructors

Figure 1 summarises the constructors supported by DAML+OIL. The standard DL syntax is used 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>
```

while $\geqslant$2hasChild.Lawyer would be written as

```
<daml:Restriction daml:minCardinalityQ="2">
  <daml:onProperty rdf:resource="#hasChild"/>
  <daml:hasClassQ rdf:resource="#Lawyer"/>
</daml:Restriction>
```

The meaning of the first three constructors (intersectionOf, unionOf and complementOf) is relatively self-explanatory: they are just the standard boolean operators that allow classes to be formed from the intersection, union and negation of other classes. The oneOf constructor allows classes to be defined existentially, i.e., 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 hasValue 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 $\geqslant nP.C$ ($\leqslant nP.C$, $=n\,P.C$) 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. Moreover, XML Schema *datatypes* (e.g., so called primitive datatypes such as strings, decimal or float, as well as more complex derived datatypes such as integer subranges) can be used anywhere that a class name might appear.

The formal semantics of the class constructors is given by DAML+OIL's model-theoretic semantics[7] or can be derived from the specification of a suitably expressive DL (e.g., see [21]).

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	$\top \sqsubseteq \leqslant 1P$	$\top \sqsubseteq \leqslant$1hasMother
unambiguousProperty	$\top \sqsubseteq \leqslant 1P^-$	$\top \sqsubseteq \leqslant$1isMotherOf$^-$

Fig. 2. DAML+OIL axioms

As already mentioned, besides the set of constructors supported, the other aspect of a language that determines its expressive power is the kinds of axiom supported. Figure 2 summarises the axioms supported by 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.

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

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 an atomic 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 and unambiguousProperty axioms, can be reduced to subClassOf and sameClassAs axioms (as can be seen from the DL syntax). In fact sameClassAs could also be reduced to subClassOf as a sameClassAs axiom $C \equiv D$ is equivalent to a pair of subClassOf axioms, $C \sqsubseteq D$ and $D \sqsubseteq C$.

As we have seen, DAML+OIL 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.

3 Reasoning Services

As we have seen, DAML+OIL is equivalent to a very expressive description logic. More precisely, DAML+OIL is equivalent to the $\mathcal{SHIQ}$ DL [22] with the addition of existentially defined classes (i.e., the oneOf constructor) and *datatypes* (often called concrete domains in DLs [1]). This equivalence allows DAML+OIL to exploit the considerable existing body of description logic research, e.g.:

- to define the semantics of the language and to understand its formal properties, in particular the decidability and complexity of key inference problems [10];
- as a source of sound and complete algorithms and optimised implementation techniques for deciding key inference problems [22, 21];
- to use implemented DL systems in order to provide (partial) reasoning support [19, 29, 16].

A important consideration in the design of DAML+OIL was that key inference problems in the language, in particular class consistency/subsumption,[8] 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 [20, 15].

Maintaining the decidability of the language requires certain constraints on its expressive power that may not be acceptable to all applications. However, the

[8] In propositionally closed languages like DAML+OIL, class consistency and subsumption are mutually reducible. Moreover, in DAML+OIL the consistency of an entire "knowledge base" (an ontology plus a set of class and property membership assertions) can be reduced to class consistency.

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 [4]. 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" [3] and/or computing least common subsumers [2] 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 [8], 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 (e.g., name reconciliation [27]), 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.

4 Datatypes

DAML+OIL supports the full range of XML Schema datatypes. 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, it is assumed that that 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 interpretation 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 [18].
– 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 may 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 [21].

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 data types, 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.

5 Research Challenges for DAML+OIL

Class consistency/subsumption reasoning in DAML+OIL is know to be decidable (as it is contained in the C2 fragment of first order logic [14]), 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.

5.1 Individuals

The OIL language was designed so that it could be mapped to the $\mathcal{SHIQ}$ DL, thereby providing a implementation path for reasoning services. This mapping is made possible by a very weak treatment of individuals occurring in existentially

defined classes, which are treated not as single elements but as the extensions of corresponding primitive classes. This is a well known technique for avoiding the reasoning problems that arise with existentially defined classes (such as classes defined using DAML+OIL's oneOf constructor) and is also used, e.g., in the CLASSIC knowledge representation system [6].

In contrast, DAML+OIL gives a standard semantics to such individuals, i.e., they are interpreted as single elements in the domain of discourse. 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 (oneOf) Italy are kinds of person that have at most one country of residence:

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

Unfortunately, the combination of such 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 [21], and in the absence of individuals (in existentially 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,[9] but in this case reasoning with individuals would not be sound and complete with respect to the semantics of the language.

5.2 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 [20, 15], but this is mostly w.r.t. logics that do not include inverse properties (e.g., $\mathcal{SHF}$ [10]). 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 [23].

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

[9] This approach is taken by some existing applications, e.g., OilEd [4].

[10] $\mathcal{SHF}$ is equivalent to $\mathcal{SHIQ}$ without inverse properties and with only functional properties instead of qualified number restrictions [22].

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.

5.3 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 description logics has already been done [30, 7, 24], 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 in Section 3, reasoning problems such as matching and computing least common subsumers could also be important in ontology design.

6 Summary

DAML+OIL is an ontology language specifically designed for use on the Web; it exploits existing Web standards (XML and RDF), adding the formal rigor of a description logic and the ontological primitives of object oriented and frame based systems.

This combination of features has proved very attractive and DAML+OIL has already been widely adopted, with some major efforts already committed to encoding their ontologies in DAML+OIL. This has been particularly evident in the bio-ontology domain, where the Bio-Ontology Consortium has specified DAML+OIL as their ontology exchange language, and the Gene Ontology [31] is being migrated to DAML+OIL in a project partially funded by GlaxoSmithKline Pharmaceuticals in cooperation with the Gene Ontology Consortium.[11]

What of the future? The development of the semantic Web, and of Web ontology languages, presents many opportunities and challenges. A "practical" (satisfiability/subsumption) algorithm for the full DAML+OIL language has yet to be developed, and it is not yet clear that sound and complete reasoners can provide adequate performance for typical Web applications.

Acknowledgements

I would like to acknowledge the contribution of all those involved in the development of DAML-ONT, OIL and DAML+OIL, amongst whom Dieter Fensel, Frank van Harmelen, Deborah McGuinness and Peter F. Patel-Schneider deserve particular mention.

[11] http://www.geneontology.org/.

References

1. F. Baader and P. Hanschke. A schema for integrating concrete domains into concept languages. In *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI'91)*, pages 452–457, 1991.
2. F. Baader and R. Küsters. Computing the least common subsumer and the most specific concept in the presence of cyclic $\mathcal{ALN}$-concept descriptions. In *Proc. of the 22nd German Annual Conf. on Artificial Intelligence (KI'98)*, volume 1504 of *Lecture Notes in Computer Science*, pages 129–140. Springer-Verlag, 1998.
3. F. Baader, R. Küsters, A. Borgida, and D. L. McGuinness. Matching in description logics. *J. of Logic and Computation*, 9(3):411–447, 1999.
4. S. Bechhofer, I. Horrocks, C. Goble, and R. Stevens. OilEd: a reason-able ontology editor for the semantic web. In *Proc. of the Joint German/Austrian Conf. on Artificial Intelligence (KI 2001)*, number 2174 in Lecture Notes in Artificial Intelligence, pages 396–408. Springer-Verlag, 2001.
5. T. Berners-Lee. *Weaving the Web*. Harpur, San Francisco, 1999.
6. A. Borgida and P. F. Patel-Schneider. A semantics and complete algorithm for subsumption in the CLASSIC description logic. *J. of Artificial Intelligence Research*, 1:277–308, 1994.
7. D. Calvanese, G. De Giacomo, and M. Lenzerini. Answering queries using views in description logics. In *Proc. of the 1999 Description Logic Workshop (DL'99)*, pages 9–13. CEUR Electronic Workshop Proceedings, http://ceur-ws.org/Vol-22/, 1999.
8. D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Information integration: Conceptual modeling and reasoning support. In *Proc. of the 6th Int. Conf. on Cooperative Information Systems (CoopIS'98)*, pages 280–291, 1998.
9. S. Decker, F. van Harmelen, J. Broekstra, M. Erdmann, D. Fensel, I. Horrocks, M. Klein, and S. Melnik. The semantic web: The roles of XML and RDF. *IEEE Internet Computing*, 4(5), 2000.
10. F. M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. *Information and Computation*, 134:1–58, 1997.
11. 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'00)*, number 1937 in Lecture Notes in Artificial Intelligence, pages 1–16. Springer-Verlag, 2000.
12. 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.
13. R. Fikes and D. L. McGuinness. An axiomatic semantics for rdf, rdf schema, and daml+oil. In *Stanford University KSL Technical Report KSL-01-01*. http://www.ksl.stanford.edu/people/dlm/daml-semantics/abstract-axiomatic-semantics.html, 2001.
14. E. Grädel, M. Otto, and E. Rosen. Two-variable logic with counting is decidable. In *Proc. of the 12th IEEE Symp. on Logic in Computer Science (LICS'97)*, pages 306–317. IEEE Computer Society Press, 1997.
15. V. Haarslev and R. Möller. High performance reasoning with very large knowledge bases: A practical case study. In *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001)*, 2001.
16. V. Haarslev and R. Möller. RACER system description. In *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2001)*, 2001.

17. J. Hendler and D. L. McGuinness. The darpa agent markup language". *IEEE Intelligent Systems*, 15(6):67–73, 2000.
18. B. Hollunder and F. Baader. Qualifying number restrictions in concept languages. In *Proc. of the 2nd Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'91)*, pages 335–346, 1991.
19. I. Horrocks. The FaCT system. In H. de Swart, editor, *Proc. of the 2nd Int. Conf. on Analytic Tableaux and Related Methods (TABLEAUX'98)*, volume 1397 of *Lecture Notes in Artificial Intelligence*, pages 307–312. Springer-Verlag, 1998.
20. 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.
21. I. Horrocks and U. Sattler. Ontology reasoning in the $\mathcal{SHOQ}$(D) description logic. In *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001)*. Morgan Kaufmann, Los Altos, 2001.
22. I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In H. Ganzinger, D. McAllester, and A. Voronkov, editors, *Proc. of the 6th Int. Conf. on Logic for Programming and Automated Reasoning (LPAR'99)*, number 1705 in Lecture Notes in Artificial Intelligence, pages 161–180. Springer-Verlag, 1999.
23. I. Horrocks, U. Sattler, and S. Tobies. Reasoning with individuals for the description logic $\mathcal{SHIQ}$. In *Proc. of the 17th Int. Conf. on Automated Deduction (CADE 2000)*, number 1831 in Lecture Notes in Artificial Intelligence, pages 482–496. Springer-Verlag, 2000.
24. I. Horrocks and S. Tessaris. A conjunctive query language for description logic aboxes. In *Proc. of the 17th Nat. Conf. on Artificial Intelligence (AAAI 2000)*, pages 399–404, 2000.
25. D. L. McGuinness. Ontological issues for knowledge-enhanced search. In *Proc. of FOIS*, Frontiers in Artificial Intelligence and Applications. IOS-press, 1998.
26. D. L. McGuinness. Ontologies for electronic commerce. In *Proc. of the AAAI '99 Artificial Intelligence for Electronic Commerce Workshop*, 1999.
27. D. L. McGuinness, R. Fikes, J. Rice, and S. Wilder. The Chimaera ontology environment. In *Proc. of the 17th Nat. Conf. on Artificial Intelligence (AAAI 2000)*, 2000.
28. S. McIlraith, T. Son, and H. Zeng. Semantic web services. *IEEE Intelligent Systems*, 16(2):46–53, March/April 2001.
29. P. F. Patel-Schneider. DLP system description. In *Proc. of the 1998 Description Logic Workshop (DL'98)*, pages 87–89. CEUR Electronic Workshop Proceedings, http://ceur-ws.org/Vol-11/, 1998.
30. M.-C. Rousset. Backward reasoning in ABoxes for query answering. In *Proc. of the 1999 Description Logic Workshop (DL'99)*, pages 18–22. CEUR Electronic Workshop Proceedings, http://ceur-ws.org/Vol-22/, 1999.
31. The Gene Ontology Consortium. Gene ontolgy: tool for the unification of biology. *Nature Genetics*, 25(1):25–29, 2000.