DAML+OIL: a Description Logic for the Semantic Web

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. For example, the TCP/IP protocol has ensured interoperability at the transport level, while HTTP and HTML have provided a standard way of retrieving and presenting hyperlinked text documents. Applications have been able to use this common infrastructure and this has made possible the World Wide Web as we know it now.

The "first generation" Web consisted largely of handwritten HTML pages. The current Web, which can be described as the second generation, has made the transition to machine generated and often active HTML pages. Both the first and second generation 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 idea was first delineated, and named the *Semantic Web*, in Tim Berners-Lee's recent book "Weaving the Web" [5].

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. This is where *ontologies* will play a crucial role, providing a source of shared and precisely defined terms that can be used in meta-data. An ontology typically consists of a hierarchical description of important concepts in a domain, along with

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

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 obviously facilitates machine understanding. Examples of the use of ontologies could include e-commerce sites [16], search engines [17] and Web services [19].

2 Web 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 [7]. 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.

The recognition of these requirements has led to the development of DAML+OIL, an expressive Web ontology language. DAML+OIL is the result of a merger between DAML-ONT, a language developed as part of the US DARPA Agent Markup Language (DAML) programme[3]) and OIL (the Ontology Inference Layer) [9], developed by a group of (mostly) European researchers.[4]

3 DAML+OIL and Description Logics

DAML+OIL is designed to describe the *structure* of a domain; it takes an object oriented approach, describing the structure 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[5] (pairs of resources) are instances of DAML+OIL classes (properties) is left to RDF, a task for which it 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 (DL), 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.

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, $\geq$2hasChild.Lawyer would be written as

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

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

[2]`http://www.w3c.org/RDF/`

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

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

[5]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.

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`	$\geq nP.C$	$\geq$2hasChild.Lawyer
`maxCardinalityQ`	$\leq nP.C$	$\leq$1hasChild.Male
`cardinalityQ`	$=n\,P.C$	=1 hasParent.Female

Figure 1: DAML+OIL class constructors

```
    <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 `hasClass` and `oneOf`.

The `minCardinalityQ`, `maxCardinalityQ` and `cardinalityQ` constructors (known in DLs as qualified number restrictions) are generalisations of the `toClass` and `hasClass` constructors. The class $\geq nP.C$ ($\leq nP.C$, $=n\,P.C$) 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 sub-ranges) 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[6].

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.

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

[6]`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`	$\top \sqsubseteq \leq 1 P$	$\top \sqsubseteq \leq 1$hasMother
`unambiguousProperty`	$\top \sqsubseteq \leq 1 P^-$	$\top \sqsubseteq \leq 1$isMotherOf$^-$

Figure 2: DAML+OIL axioms

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), unambiguous (i.e., its inverse is functional) or transitive, as well as to use inverse properties.

4 Reasoning Services

As we have shown, DAML+OIL is equivalent to a very expressive description logic. More precisely, DAML+OIL is equivalent to the $\mathcal{SHIQ}$ DL [15] 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 [8];
- as a source of sound and complete algorithms and optimised implementation techniques for deciding key inference problems [15, 14];
- to use implemented DL systems in order to provide (partial) reasoning support [12, 20, 11].

A important consideration in the design of DAML+OIL was that key inference problems in the language, in particular class consistency/subsumption,[7] 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

[7]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.

applications in spite of their high worst case complexity [13, 10]. In particular, DAML+OIL is able to exploit highly optimised reasoning services provided by DL systems such as FaCT [12], DLP [20], and Racer [11], although these systems do not, as yet, support the whole DAML+OIL language (none is able to reason with existentially defined classes, i.e., the `oneOf` construct, or to provide support for all XML Schema datatypes).

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 (as demonstrated in the OilEd[8] ontology editor [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 [6], 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 [18]), 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.

5 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. As well as providing the formal underpinnings of the language, the connection to DLs can be exploited as a source of algorithms and implementation techniques, and to provide (partial) reasoning support for DAML+OIL applications by using implemented DL systems.

DAML+OIL has already been widely adopted, with some major efforts having already committed to encoding their ontologies in the language. 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 [21] is being migrated to DAML+OIL in a project partially funded by GlaxoSmithKline Pharmaceuticals in cooperation with the Gene Ontology Consortium.[9]

[8]`http://img.cs.man.ac.uk/oil`
[9]`http://www.geneontology.org/`.

What of the future? The development of the semantic Web, and of Web ontology languages, presents many challenges. As we have seen, no DL system yet provides reasoning support for the full DAML+OIL language. Developing a "practical" satisfiability/subsumption algorithm (i.e., one that is amenable to highly optimised implementation) for the whole language would present a major step forward in DL (and semantic web) research. Moreover, even if such an algorithm can be developed, it is not clear if even highly optimised implementations of sound and complete algorithms will be able to 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, Jim Hendler, Deborah McGuinness and Peter F. Patel-Schneider deserve particular mention.

References

[1] F. Baader and P. Hanschke. A schema for integrating concrete domains into concept languages. In *Proc. of 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 KI'98*, volume 1504 of *LNCS*, 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 LNAI, pages 396–408. Springer-Verlag, 2001.

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

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

[7] 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.

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

[9] 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.

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

[11] V. Haarslev and R. Möller. RACER system description. In *Proc. of IJCAR-01*, 2001.

[12] I. Horrocks. The FaCT system. In H. de Swart, editor, *Proc. of TABLEAUX-98*, volume 1397 of *LNAI*, pages 307–312. Springer-Verlag, 1998.

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

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

[15] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In H. Ganzinger, D. McAllester, and A. Voronkov, editors, *Proc. of LPAR'99*, number 1705 in LNAI, pages 161–180. Springer-Verlag, 1999.

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

[17] D. L. McGuinness. Ontologies for electronic commerce. In *Proc. of the AAAI '99 Artificial Intelligence for Electronic Commerce Workshop*, 1999.

[18] D. L. McGuinness, R. Fikes, J. Rice, and S. Wilder. The Chimaera ontology environment. In *Proc. of AAAI 2000*, 2000.

[19] S. McIlraith, T. Son, and H. Zeng. Semantic web services. *IEEE Intelligent Systems*, 16(2):46–53, March/April 2001.

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

[21] The Gene Ontology Consortium. Gene ontolgy: tool for the unification of biology. *Nature Genetics*, 25(1):25–29, 2000.