14

OWL: a Description Logic Based Ontology Language for the Semantic Web

Ian Horrocks
Peter F. Patel-Schneider
Deborah L. McGuinness
Christopher A. Welty

Abstract

It has long been realized that the web could benefit from having its content understandable and available in a machine processable form. The Semantic Web aims to achieve this via annotations that use terms defined in ontologies to give well defined meaning to Web accessible information and services. OWL, the ontology language recommended by the W3C for this purpose, was heavily influenced by Description Logic research. In this chapter we review briefly some early efforts that combine Description Logics and the Web, including predecessors of OWL such as OIL and DAML+OIL. We then go on to describe OWL in some detail, including the various influences on its design, its relationship with RDFS, its syntax and semantics, and a range of tools and applications.

14.1 Background and history

The World Wide Web, while wildly successful in growth, may be viewed as being limited by its reliance on languages like HTML that are focused on *presentation* (i.e., text formatting) rather than *content*. Languages such as XML do add some support for capturing the meaning of Web content (instead of simply how to render it in a browser), but more is needed in order to support intelligent applications that can better exploit the ever increasing range of information and services accessible via the Web. Such applications are urgently needed in order to avoid overwhelming users with the sheer volume of information becoming available.

The Semantic Web has been envisaged as an evolution of the existing Web from a linked document repository into an application platform where "information is given well-defined meaning, better enabling computers and people

to work in cooperation" [Berners-Lee *et al.*, 2001]. This is to be achieved by augmenting the existing layout information with semantic annotations that add descriptive terms to Web content, with the meaning of such terms being defined in *ontologies*.

In order for the meaning of semantic annotations to be accessible to applications (as well as humans), the ontology language being used should have a precisely defined semantics and should be amenable to automated processing. Description Logics appear to be ideally suited to this role: they have a formal logic-based semantics, and are often equipped with decision procedures that have been designed with a view to implementation in automated reasoning systems. This view of the potential place of Description Logics in the Semantic Web led to the development of a number of languages that brought Description Logic concepts to the Semantic Web, culminating in the development of the Web Ontology Language OWL . OWL is the World Wide Web Consortium (W3C) recommended ontology language for the Semantic Web, and exploits many of the strengths of Description Logics, including well defined semantics and practical reasoning techniques.

In this chapter we first review briefly the history of Description Logic efforts related to the Semantic Web, in particular OIL and DAML+OIL. We then go on to describe OWL in some detail, and to show how it brings Description Logic concepts fully into the Semantic Web.

14.1.1 Early Uses of Description Logics in the Semantic Web

Before the development of Description Logic-related languages designed for the Semantic Web, there were several systems that used Description Logics in the context of the web. We will describe some salient features of two systems, UNTANGLE and FINDUR, that illustrate early Description Logic usage on the web.

The relationship between hypertext and semantic networks has long been realized, but one of the earliest Description Logic systems to realize this relationship was the UNTANGLE system [Welty and Jenkins, 2000], a Description Logic system for representing bibliographic (card-catalog) information. The UNTANGLE project began as a bit of exploratory research in using Description Logics for digital libraries [Welty, 1994], but out of sheer temporal coincidence with the rise of the web, a web interface was added and the first web-based Description Logic system was born.

The original UNTANGLE web interface was developed in 1994 [Welty, 1996a], and combined LISP-CLASSIC and the COMMONLISP Hypermedia

Server (CL-HTTP) [Mallery, 1994] to implement a hypertext view of the ABox and TBox semantic networks, and used nested bullet lists to view the concept taxonomy, with in-page cross references for concepts having multiple parents. The interface was interesting in some respects as a tool to visualize Description Logic and semantic network information, though this aspect was never fully developed.

As the World Wide Web (WWW) became the primary means of dissemination of computer science research, the goals of the UNTANGLE project shifted in 1995 to cataloging and classifying pages on the web [Welty, 1996b], which was viewed as a massive and unstructured digital library [Welty, 1998].

Another early project using Description Logics for the web was the FINDUR system at AT&T [McGuinness, 1998; McGuinness *et al.*, 1997]. The basic notion of FINDUR was *query expansion*,[1] that is, taking synonyms or hyponyms (more specific terms) and including them in the input terms, thereby expanding the query.

The FINDUR system represented a simple background knowledge base containing mostly thesaurus information built in a Description Logic (CLASSIC) using the most basic notions of Wordnet (synsets and hyper/hyponyms) [Miller, 1995]. Concepts corresponding to sets of synonyms (synsets) were arranged in a taxonomy. These synsets also contained an informal list of related terms. Site specific search engines (built on Verity—a commercial search engine) were hooked up to the knowledge base. Any search term would first be checked in the knowledge base, and if it was contained in any synset, a new query would be constructed consisting of the disjunction of all the synonymous terms, as well as all the more specific terms (hyponyms).

The background knowledge was represented in CLASSIC, but the Description Logic was not itself part of the on-line system. Instead, the information used by the search engine was statically generated on a regular basis and used to populate the search engine. The true power of using a Description Logic as the substrate for the knowledge base was realized mainly in the maintenance task. The Description Logic allowed the maintainer of the knowledge base to maintain some amount of consistency, such as discovering cycles in the taxonomy and disjoint synsets. These simple constraints proved effective tools for maintaining the knowledge since the knowledge itself was very simple.

Additional use of the Description Logic approach in FindUR was realised

[1] Work on a subsequent Description Logic based approach to query expansion addressed some formal issues in evaluating the soundness and completeness of alternative approaches [Rousset, 1999]. Other work on Description Logic (or Description Logic inspired) approaches to retrieval also exists, e.g., [Meghini *et al.*, 1997].

in applications that exposed more structured searches, exploiting subclass hierarchies and property relationships, such as the medical applications of FindUR in the P-CHIP Primary Care Search Application [Brachman *et al.*, 1999]. This type of structured search, exploiting background ontologies and relationships between terms, can also be seen in later work, e.g., in the SHOE project [Heflin *et al.*, 2003].

14.2 Steps Towards Integration with the Semantic Web: OIL and DAML+OIL

The first major effort to build a language that combined Description Logics and the Semantic Web was OIL (the Ontology Inference Layer) [Horrocks *et al.*, 2000a], a part of the On-To-Knowledge research project funded by the European Union. The OIL language was explicitly designed as "a web-based representation and inference language for ontologies [combining] the widely used modeling primitives from frame-based languages with the formal semantics and reasoning services provided by description logics" (`http://www-ontoknowledge.org/oil/oilhome.shtml`).

Description Logics provide the semantics for OIL, so much so that the semantics of OIL is specified via a mapping to the Description Logic $\mathcal{SHIQ}$ [Fensel *et al.*, 2001; Horrocks *et al.*, 1999]. OIL has a syntax based on the Resource Description Framework (RDF), as well as an XML syntax, that provided the connection to the Semantic Web of the time.[2] OIL allows the grouping of Description Logic constructs in a way similar to frame systems, providing a more intuitive feel to the language as opposed to the logically inspired syntax usually used for Description Logics. These three influences—Description Logics, frames, and the Semantic Web—are present not only in OIL, but also in all of its successors.

14.2.1 OIL

The OIL language is designed to combine frame-like modeling primitives with the increased (in some respects) expressive power, formal rigor and automated reasoning services of an expressive Description Logic [Fensel *et al.*, 2000]. OIL also comes "web enabled" by having both XML and RDFS based serializations (as well as a formally specified "human readable" form, which we will use here). In frame languages, classes (concepts) are described by *frames*, whose main components consist of a list of superclasses and a list

[2] At the time that OIL was developed, RDF—the base language of the Semantic Web—was without a fully specified semantic foundation.

of *slot-filler* pairs. A slot corresponds to a role in a Description Logic, and a slot-filler pair corresponds to either a value restriction (a concept of the form $\forall R.C$) or an existential quantification (a concept of the form $\exists R.C$)—one of the criticisms leveled at frame languages is that they are often unclear as to exactly which of these is intended by a slot-filler pair.

OIL extends this basic frame syntax so that it can capture the full power of an expressive Description Logic.

In order to allow users to choose the expressive power appropriate to their application, and to allow for future extensions, a layered family of OIL languages was described. The base layer, called "Core OIL" [Bechhofer *et al.*, 2000], is a cut-down version of the language that closely corresponds with RDFS (i.e., it includes only class and slot inclusion axioms, and slot range and domain constraints). The standard language is called "Standard OIL", and when extended with ABox axioms (i.e., the ability to assert that individuals and tuples are, respectively, instances of classes and slots), is called "Instance OIL". Finally, "Heavy OIL" was the name given to a further layer that was to include still unspecified language extensions.

Figure 14.1 illustrates an OIL ontology (using the human readable serialization) corresponding to an example terminology from Chapter **??**. A full specification of OIL, including DTDs for the XML and RDFS serializations, can be found in [Horrocks *et al.*, 2000a].

Standard OIL can be seen as a syntactic variant of the Description Logic $\mathcal{SHIQ}$ [Horrocks *et al.*, 1999] extended with simple concrete datatypes [Baader and Hanschke, 1991; Horrocks and Sattler, 2001]; we will call this Description Logic $\mathcal{SHIQ}(\mathcal{D})$. Rather than providing the usual model-theoretic semantics, OIL defines a translation that maps an OIL ontology into an equivalent $\mathcal{SHIQ}(\mathcal{D})$ terminology. From this mapping, OIL derives both a clear semantics and a means to exploit the reasoning services of Description Logic systems such as FaCT [Horrocks, 1998b], Racer [Haarslev and Möller, 2001] and Pellet [Pellet, 2003] that implement (most of) $\mathcal{SHIQ}(\mathcal{D})$.

14.2.2 The DAML project and **DAML+OIL**

At about the same time as OIL was being developed, the DARPA Agent Markup Language (DAML) program was started in the United States. DAML was initiated in order to provide the foundation for the next generation of the web which, it was anticipated, would increasingly utilize agents and programs rather than relying so heavily on human interpretation of web information [Hendler and McGuinness, 2000].

```
name "Family"
documentation "Example ontology describing family relationships"
definitions
  slot-def hasChild
    inverse isChildOf
  class-def defined Woman
    subclass-of Person Female
  class-def defined Man
    subclass-of Person not Woman
  class-def defined Mother
    subclass-of Woman
    slot-constraint hasChild
      has-value Person
  class-def defined Father
    subclass-of Man
    slot-constraint hasChild
      has-value Person
  class-def defined Parent
    subclass-of or Father Mother
  class-def defined Grandmother
    subclass-of Mother
    slot-constraint hasChild
      has-value Parent
  class-def defined MotherWithManyChildren
    subclass-of Mother
    slot-constraint hasChild
      min-cardinality 3
  class-def defined MotherWithoutDaughter
    subclass-of Mother
    slot-constraint hasChild
      value-type not Woman
```

Fig. 14.1. OIL "family" ontology.

One of the early widely-distributed contributions of the DAML program was DAML-ONT—a proposal for an ontology language for the web [Hendler and McGuinness, 2000; McGuinness *et al.*, 2002]. This language began with the requirement to build on the best practice in web languages of the time, and in particular to extend W3C's Resource Description Framework, with the aim of adding expressive power suited to agent and service interoperation.

It became obvious that the goals of DAML-ONT and OIL were so similar that these objectives could best be served by combining the two efforts. The resulting language, DAML+OIL, has a formal (model-theoretic) semantics that provides machine and human understandability [van Harmelen *et al.*, 2001], an axiomatization [Fikes and McGuinness, 2001], and a reconciliation of the language constructors from the two precursor languages.

DAML+OIL is similar to OIL in many respects, but is more tightly integrated with RDFS, which provides the only specification of the language and its only serialization. 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.

The solution to this problem adopted by DAML+OIL is to define the semantics of the language in such a way that it gives a meaning to any (parts of) ontologies that conform to the RDFS specification, including "strange" constructs such as slot constraints with multiple slots and classes. This is made easier by the fact that, unlike OIL, the semantics of DAML+OIL is directly defined.

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 Description Logic 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.*, 2001b], because the axioms may be susceptible to many different frame-like groupings [Bechhofer *et al.*, 2001a].

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.

The initial release of DAML+OIL did not include any specification of datatypes. The language was, however, subsequently extended with arbitrary datatypes from the XML Schema type system, which can be used in restrictions (slot constraints) and range constraints. As in $\mathcal{SHOQ}(\mathcal{D})$ [Horrocks and Sattler, 2001], a clean separation is maintained 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 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).

14.3 Full Integration into the Semantic Web: OWL

With its tighter integration with RDF and RDFS, DAML+OIL was the first Description Logic-inspired language to be completely integrated into the fabric of the Semantic Web (as it was then defined). There was, however, no formal semantic integration of DAML+OIL with RDF and RDFS, as RDF and RDFS did not have a formal semantics at the time when DAML+OIL was being developed. DAML+OIL was also only the product of an *ad-hoc* group of researchers, and was thus not an official part of the Semantic Web.

The fix these problems, the W3C chartered two working groups in 2001. The RDF Core Working Group was to be responsible for updating the RDF recommendation and providing a formal semantics for RDF and thus for the Semantic Web. The Web Ontology Working Group was to be responsible for designing an ontology language for the web, compatible with the new version of RDF. The Web Ontology Working Group developed the Web Ontology Language OWL [Bechhofer *et al.*, 2004; Patel-Schneider *et al.*, 2004], which became a W3C recommendation on 10 February 2004. In parallel, the RDF Core Working Group developed a formal semantics for RDF and RDFS [Hayes, 2004].

14.3.1 Influences on the Design of OWL

The design of OWL was subject to a variety of influences. These included influences from established formalisms and knowledge representation paradigms, influences from existing ontology languages, and influences from existing Semantic Web languages.

Some of the most important influences on the design of OWL came, via its predecessor DAML+OIL, from Description Logics, from the frames paradigm, and from RDF. In particular, the formal specification of the language was influenced by Description Logics, the surface structure of the language (as seen in the abstract syntax) was influenced by the frames paradigm, and the RDF/XML exchange syntax was influenced by a requirement for upwards compatibility with RDF.

Description Logics, and insights from Description Logic research, had a strong influence on the design of OWL, particularly on the formalisation of the semantics, the choice of language constructors, and the integration of datatypes and data values. In fact OWL DL and OWL Lite (two of the three species of OWL) can be viewed as expressive Description Logics, with an ontology being equivalent to a Description Logic knowledge base. More precisely, OWL DL and OWL Lite are equivalent to $\mathcal{SHIF}(\mathbf{D})$ and $\mathcal{SHOIN}(\mathbf{D})$ respectively (see Sections 14.3.3 and 14.3.5 for more details).

This design was motivated by practical considerations. The designers of OWL wanted to have some idea as to how difficult it would be for tools and applications to support the language. It was therefore important to understand its formal properties, e.g., with respect to the decidability and complexity of key inference problems. These properties followed directly from the correspondences with Description Logics. These correspondences would allow tools and applications to exploit known reasoning algorithms and even (highly optimised) implementations.

In the Semantic Web context, where users with a wide range of expertise might be expected to create or modify ontologies, readability and general ease of use are important considerations for an ontology language. In the design of OIL [Fensel *et al.*, 2001], one of the languages on which DAML+OIL was based, these requirements were addressed by providing a surface syntax based on the frames paradigm. Frames group together information about each class, making ontologies easier to read and understand, particularly for users not familiar with (Description) Logics. The frames paradigm has been used in a number of well known knowledge representation and ontology environment systems including the PROTÉGÉ ontology design tool [Grosso *et al.*, 1999], the Ontolingua ontology environment tool [Farquhar *et al.*, 1996], the OKBC knowledge model [Chaudhri *et al.*, 1998], and the Chimaera Ontology Evolution Environment [McGuinness *et al.*, 2000]. The design of OIL was influenced by XOL [Karp *et al.*, 1999]—a proposal for an XML syntax for OKBC Lite (a cut down version of the OKBC knowledge model).

In frame based languages, each class is described by a frame. The frame includes the name of the class, identifies the more general class (or classes) that it specialises, and lists a set of "slots". A slot may consist of a property-value pair, or a constraint on the values that can act as slot "fillers" (in this

context, value means either an individual or a data value). This structure was used in the OIL language, with some enrichment of the syntax for specifying classes and slot constraints so as to enable the full power of a Description Logic style language to be captured. In addition, property frames were used to describe properties, e.g., specifying more general properties, range and domain constraints, transitivity and inverse property relationships.

A class frame is semantically equivalent to a Description Logic axiom asserting that the class being described by the frame is a subclass of each of the classes that it specialises and of each of the property restrictions corresponding to the slots. As well as a richer slot syntax, OIL also offered the possibility of asserting that the class being described by the frame was exactly equivalent to the relevant intersection class, (i.e., that they were mutually subsuming). A property frame is equivalent to a set of axioms asserting the relevant subproperty relationships, range and domain constraints etc. OIL was designed so that OIL frames could easily be mapped to equivalent axioms in the $\mathcal{SHOQ}(\mathcal{D})$ Description Logic [Decker *et al.*, 2000].

The formal specification and semantics of OWL are given by an abstract syntax [Patel-Schneider *et al.*, 2004] that has been heavily influenced by frames in general and by the design of OIL in particular. In the abstract syntax, axioms are compound constructions that are very like an OIL-style frame. For classes, they consist of the name of the class being described, a *modality* of "partial" or "complete" (indicating that the axiom is asserting a subclass or equivalence relationship respectively), and a sequence of property restrictions and names of more general classes. Similarly, a property axiom specifies the name of the property and its various features.

The frame style of the abstract syntax, which borrows heavily in spirit from the human readable syntax for OIL, makes it *much* easier to read (compared to the RDF/XML syntax), and also easier (for non-logicians) to understand and to use. Moreover, abstract syntax axioms have a direct correspondence with Description Logic axioms, and they can also be mapped to a set of RDF triples.

The third major influence on the design of OWL was the requirement to maintain the maximum upwards compatibility with existing web languages, and in particular with RDF [Manola and Miller, 2004]. On the face of it this requirement made good sense as RDF (and in particular RDF Schema) already included several of the basic features of a class and property based ontology language, e.g., it allows subclass and subproperty relationships to be asserted. Moreover, the development of RDF preceded that of OWL, and it seemed reasonable to try to appeal to any user community already established by RDF.

It may seem easy to meet this requirement simply by giving OWL an RDF-based syntax, but, in order to provide maximum upwards compatibility, it was also thought necessary to ensure that the semantics of OWL ontologies was consistent with the semantics of RDF. This proved to be difficult, however, given the greatly increased expressive power provided by OWL. This will be discussed in more detail in Section 14.3.2.

14.3.2 Layering (on **RDFS***) and three "species" of OWL*

As mentioned above, the the design of OWL included a requirement to maintain the maximum upwards compatibility with RDF [Manola and Miller, 2004]. On the face of it this requirement made good sense, but it led to a number of problems in the design of OWL.

In the first place, RDF/XML is extremely verbose. Compare, for example, information about a class as it would be given in a Description Logic syntax

$$\texttt{Student}\ \mathcal{D}\ \texttt{Person} \sqcap \geqslant 1\, \texttt{enrolledIn}$$

(a `Student` is a `Person` who is `enrolledIn` at least 1 thing), with how it would most naturally be written using the OWL RDF/XML syntax[3]

```
<owl:Class rdf:ID="Student">
   <owl:intersectionOf rdf:parsetype="Collection">
     <owl:Class rdfs:about="Person" />
     <owl:Restriction>
        <owl:onProperty rdf:resource="enrolledIn" />
        <owl:minCardinality rdfs:datatype="&xsd;Integer">
                1
        </owl:minCardinality>
     </owl:Restriction>
   <owl:intersectionOf>
</owl:Class>
```

This verbosity of OWL's RDF/XML syntax may not, in itself, be a serious problem given the capabilities and bandwidths of modern computers and communication systems, and that this syntax is mainly intended for exchanging data between OWL applications. The RDF/XML syntax does, however, lead to some more serious problems. RDF is itself a graph based formalism, with graphs expressed as set of subject-predicate-object triples,

[3] Full details on the OWL RDF/XML syntax can be found in the OWL Reference document [Bechhofer *et al.*, 2004].

where each triple represents a labelled edge (the predicate) connecting two vertices (the subject and object). This means that many OWL constructs, such as property restrictions, have to be encoded as several triples. There is no requirement that these triples occur together, so parsing becomes difficult as it may be necessary to scan the entire input in order to locate all of the components of a given construction. Moreover, circular and other unusual structures with ill defined meanings cannot be ruled out.

Even more problematical is the relationship between the semantics of an OWL ontology and the semantics of the RDF triples used to encode it. This was not as much of an issue when OIL and DAML+OIL were designed, as at that time the meaning of RDF was not precisely specified. OIL in particular did not bother to relate the RDF meaning of its RDF/XML syntax to the OIL meaning of this syntax—the RDF/XML syntax for some OIL constructs does more-or-less line up with the RDF meaning of these constructs, but this is by no means the case for all such constructs.

DAML+OIL did a better job of abiding by the RDF meaning of its syntax. The DAML+OIL model theory [van Harmelen *et al.*, 2001] included a semantic condition for triples that was close to the RDF meaning (as defined at that time) for triples. Moreover, DAML+OIL used the built-in RDF and RDFS vocabulary to a greater extent than did OIL, and used it in a way generally compatible with the RDF or RDFS meaning. For example DAML+OIL uses `rdfs:subClassOf` to relate classes to superclasses.

By the time OWL was being designed, RDF was being given a formal meaning via the RDF model theory [Hayes, 2004]. It proved to be extremely difficult to design an RDF syntax for OWL such that the Description Logic semantics assigned by OWL to its various constructs was compatible with the semantics of the RDF triples used to encode them. Moreover, there was a fundamental incompatibility between the requirement to be fully backwards compatible with OWL (i.e., to allow arbitrary RDF graphs to be interpreted as OWL ontologies), and the requirement for OWL to be equivalent to an expressive description logic. (See [Horrocks *et al.*, 2003] for a detailed discussion of these issues.)

After much argument within the OWL working group, these problems were eventually "resolved" by defining two "species" of OWL: OWL Full and OWL DL. Only a subset of RDF graphs correspond to OWL DL ontologies. In particular, the graph cannot include cyclical and other problematical constructions (including "malformed" OWL syntax), and the sets of class, property, and individual names must be disjoint. For RDF graphs that satisfy these syntactic restrictions, a fairly standard Description Logic style model theory is used to define the semantics of the resulting OWL constructs. In contrast, OWL Full uses an RDF style model theory to give a semantic account of the use of OWL constructors in arbitrary RDF graphs. (See Sections 14.3.4 and 14.3.5 for more details.)

As well as arguments about the importance of compatibility with RDF, there were also tensions within the working group regarding the expressive power that was appropriate for OWL, with some members arguing that OWL DL was too complex to be understood by new users or implemented by application developers. Weight was lent to these arguments by the fact that, at the time, no tableaux decision procedure was known for $\mathcal{SHOIN}(\mathbf{D})$ (the description logic language underlying OWL DL), and no implemented system supported it. In response to these concerns, a third OWL species was defined, namely OWL Lite.

OWL Lite is a syntactic subset of OWL DL that prohibits and/or restricts the use of certain constructors and axioms with the aim of making the language easier to understand and/or implement. It has subsequently been shown, however, that by combining other constructors and axioms, most of the expressive power of OWL DL can be regained, and that OWL Lite is expressively equivalent to $\mathcal{SHIF}(\mathbf{D})$ [Horrocks and Patel-Schneider, 2003].

Given that OWL Full does not correspond either syntactically or semantically to a Description Logic, in the remainder of this chapter we will focus our attention on OWL DL and OWL Lite.

14.3.3 OWL DL abstract syntax and semantics

OWL DL has some differences from standard Description Logics. These differences provide a bridge between the formal Description Logic world and the Semantic Web world.

- OWL uses URI references as names, and constructs these URI references in the same manner as that used by RDF. It is thus common in OWL to use qualified names as shorthands for URI references, using, for example, the qualified name owl:Thing for the URI reference `http://www.w3.org/2002/07/owl#Thing`.
- OWL gathers information into ontologies, which are generally stored as Web documents written in RDF/XML. Ontologies can import other ontologies, adding the information from the imported ontology to the current ontology.
- OWL allows RDF annotation properties to be used to attach information to classes, properties, and ontologies, such as `owl:DeprecatedClass`. These annotations are RDF triples, and are therefore required to carry a full semantic weight in RDF. In OWL DL, however, such annotations carry a separate, limited meaning.

Abstract Syntax	DL Syntax
Descriptions (C)	
A	A
`owl:Thing`	$\top$
`owl:Nothing`	$\bot$
`intersectionOf(`$C_1 \ldots C_n$`)`	$C_1 \sqcap \ldots \sqcap C_n$
`unionOf(`$C_1 \ldots C_n$`)`	$C_1 \sqcup \ldots \sqcup C_n$
`complementOf(`C`)`	$\neg C$
`oneOf(`$o_1 \ldots o_n$`)`	$\{o_1\} \sqcup \ldots \sqcup \{o_n\}$
`restriction(`R `someValuesFrom(`C`))`	$\exists R.C$
`restriction(`R `allValuesFrom(`C`))`	$\forall R.C$
`restriction(`R `hasValue(`o`))`	$R : o$
`restriction(`R `minCardinality(`n`))`	$\geqslant n\, R$
`restriction(`R `minCardinality(`n`))`	$\leqslant n\, R$
`restriction(`U `someValuesFrom(`D`))`	$\exists U.D$
`restriction(`U `allValuesFrom(`D`))`	$\forall U.D$
`restriction(`U `hasValue(`v`))`	$U : v$
`restriction(`U `minCardinality(`n`))`	$\geqslant n\, U$
`restriction(`U `maxCardinality(`n`))`	$\leqslant n\, U$
Data Ranges (D)	
D	D
`oneOf(`$v_1 \ldots v_n$`)`	$\{v_1\} \sqcup \ldots \sqcup \{v_n\}$
Object Properties (R)	
R	R
`inv(`R`)`	R^-
Datatype Properties (U)	
U	U
Individuals (o)	
o	o
Data Values (v)	
v	v

Fig. 14.2. OWL DL Descriptions, Data Ranges, Properties, Individuals, and Data Values

- OWL uses the facilities of RDF datatypes and XML Schema datatypes to provide datatypes and data values (a very restricted form of concrete domains **??**).
- OWL DL and OWL Lite have a frame-like abstract syntax, whereas RDF/XML is the official exchange syntax for all three species of OWL.

As mentioned above, OWL DL is very closely related to $\mathcal{SHOIN}(\mathbf{D})$, which extends $\mathcal{SHOIQ}$ [Horrocks and Sattler, 2005] with datatypes like those in $\mathcal{SHOQ}(\mathcal{D})$ [Horrocks and Sattler, 2001], but allows only unqualified number restrictions (see Chapter **??**). OWL DL can form descriptions of classes, datatypes, individuals and data values using the constructs shown in Figure 14.2. In this table the first column gives the OWL abstract syntax for the construction, while the second column gives the equivalent Description Logic syntax. The letters A, D, R, U, o and v represent, respectively, names for classes (concepts), data ranges, object properties (abstract roles), datatype properties (concrete roles), individuals (nominals) and data values; C, possibly subscripted, represents an arbitrary class description. In OWL, data values are RDF literals (i.e., instances of datatypes such as string or integer), and all other names are URI references. As mentioned above, `owl:Thing` and `owl:Nothing` are shorthand for the URI references `http://www.w3.org/2002/07/owl#Thing` and `http://www.w3.org/2002/07/owl#Nothing` respectively.

The treatment of concrete values in OWL is somewhat different from the usual treatment of concrete values in Description Logics. OWL uses datatypes from XML Schema datatypes as its concrete types, so `xsd:integer` is a concrete type in OWL, namely the type of integers. OWL uses the RDF syntax for these concrete values, so `"2"^^xsd:integer` is the way to write the integer 2. OWL also allows plain RDF literals, which are a combination of a string and an optional language tag. These plain literals belong to the datatype rdfs:Literal. OWL also allows sets of data values to be used in concept expressions, as in `oneOf("1"^^xsd:integer "2"^^xsd:integer "3"^^xsd:integer)`.

OWL uses these description-forming constructs in axioms that provide information about classes, properties, and individuals, as shown in Figure 14.3. Again, the frame-like abstract syntax is given in the first column, and the standard Description Logic syntax is given in the second column. The letters A, D, R, U, o and v (in each case possibly subscripted) represent, respectively, names for classes (concepts), data ranges, object properties (abstract roles), datatype properties (concrete roles), individuals (nominals) and data values; C (possibly subscripted) represents an arbitrary class description.

Either partial or complete information can be stated about a class, as in

```
Class(ex:Country partial owl:Thing)
Class(ex:Person partial owl:Thing)
Class(ex:Student partial ex:Person)
```

Abstract Syntax	DL Syntax
Class(A partial $C_1 \ldots C_n$)	$A \sqsubseteq C_1 \sqcap \ldots \sqcap C_n$
Class(A complete $C_1 \ldots C_n$)	$A \equiv C_1 \sqcap \ldots \sqcap C_n$
EnumeratedClass(A $o_1 \ldots o_n$)	$A \equiv \{o_1\} \sqcup \ldots \sqcup \{o_n\}$
SubClassOf(C_1 C_2)	$C_1 \sqsubseteq C_2$
EquivalentClasses($C_1 \ldots C_n$)	$C_1 \equiv \ldots \equiv C_n$
DisjointClasses($C_1 \ldots C_n$)	$C_i \sqcap C_j \subseteq \bot, i \not{\mathcal{D}} j$
Datatype(D)	
ObjectProperty(R super(R_1)...super(R_n)	$R \sqsubseteq R_i$
domain(C_1)...domain(C_m)	$\geqslant 1\, R \sqsubseteq C_i$
range(C_1)...range(C_ℓ)	$\top \sqsubseteq \forall R.C_i$
[inverseOf(R_0)]	$R \equiv R_0^-$
[Symmetric]	$R \equiv R^-$
[Functional]	$\top \sqsubseteq\, \leqslant 1\, R$
[InverseFunctional]	$\top \sqsubseteq\, \leqslant 1\, R^-$
[Transitive])	$Tr(R)$
SubPropertyOf(R_1 R_2)	$R_1 \sqsubseteq R_2$
EquivalentProperties($R_1 \ldots R_n$)	$R_1 \equiv \ldots \equiv R_n$
DatatypeProperty(U super(U_1)...super(U_n)	$U \sqsubseteq U_i$
domain(C_1)...domain(C_m)	$\geqslant 1\, U \sqsubseteq C_i$
range(D_1)...range(D_ℓ)	$\top \sqsubseteq \forall U.D_i$
[Functional])	$\top \sqsubseteq\, \leqslant 1\, U$
SubPropertyOf(U_1 U_2)	$U_1 \sqsubseteq U_2$
EquivalentProperties($U_1 \ldots U_n$)	$U_1 \equiv \ldots \equiv U_n$
AnnotationProperty(S)	
OntologyProperty(S)	
Individual(o type(C_1)...type(C_n)	$o \in C_i$
value(R_1 o_1)...value(R_n o_n)	$\langle o, o_i \rangle \in R_i$
value(U_1 v_1)...value(U_n v_n))	$\langle o, v_i \rangle \in U_i$
SameIndividual($o_1 \ldots o_n$)	$\{o_1\} \equiv \ldots \equiv \{o_n\}$
DifferentIndividuals($o_1 \ldots o_n$)	$\{o_i\} \sqsubseteq \neg\{o_j\}, i \not{\mathcal{D}} j$

Fig. 14.3. OWL DL Axioms and Facts

```
Class(ex:Canadian complete ex:Person
                  hasValue(ex:nationality ex:Canada))
```

which makes `ex:Country` and `ex:Person` classes, `ex:Student` a subclass of `ex:Person`, and `ex:Canadian` precisely those people who have `ex:nationality` `ex:Canada`.[4]

In OWL DL properties are divided into object properties, like `ex:nationality`, which relate individuals to other individuals, datatype properties, like `ex:age`, which relate individuals to data values, and annotation properties, which can be used to add uninterpreted information (such as versioning information) to individuals, classes, and properties. Constraints, such as domains and ranges, can be given for object properties and datatype properties, but not for annotation properties.

```
DatatypeProperty(ex:age domain(ex:Person)
                        range(xsd:integer))
ObjectProperty(ex:nationality domain(ex:Person)
                              range(ex:Country))
```

Object properties can also be specified to be transitive, symmetric, functional, and inverse functional. Not all of these can be specified for a particular object property, as to retain decidability of OWL DL properties that are specified as being transitive, and their super-properties and their inverses, cannot have their cardinality restricted, either via the functional part of property axioms or in cardinality restrictions (see [Horrocks *et al.*, 1999]).

Annotation properties are a way of associating uninterpreted information with classes, properties, and individuals. The syntax for annotations is not given in Figure 14.3. Many axioms (`Class`, `EnumeratedClass`, `Datatype`, `ObjectProperty`, `DatatypeProperty`, `AnnotationProperty`, `OntologyProperty`, and `Individual` axioms) can have an annotation that provides uninterpreted information about that class, property, or individual. To prevent annotations influencing the semantics of OWL DL, little can be said about annotation properties: they cannot participate in restrictions, nor can they be given domains, ranges, or any other aspect of other kinds of properties. These limitations result in no non-trivial inference related to annotation properties in OWL DL.

Any axiom that starts with a URI reference can have an annotation that provides uninterpreted information about that URI reference. To prevent undue influences on the semantics of OWL DL, little can be said about annotation properties. Annotation properties cannot participate in restrictions, nor can they be given domains, ranges, or any other aspect of other kinds of properties.

[4] For a more extensive example of how to use OWL, see the OWL Guide [Smith *et al.*, 2004].

One use of annotation properties is to provide comment information for classes and properties, as in

```
Class(ex:Country partial
      annotation(rdfs:comment "Countries of the world")
      owl:Thing)
```

This is, perhaps, the most useful purpose for annotations in OWL DL.

Information about individuals can also be provided in OWL, for either named individuals or anonymous individuals, as in:

```
Individual(ex:Canada type(ex:Country))
Individual(ex:England type(ex:Country))
Individual(ex:Peter type(ex:Canadian)
                    value(ex:age "48"^^xsd:integer))
Individual(value(ex:nationality ex:England)
           value(ex:age "44"^^xsd:integer))
```

In OWL, axioms and facts are grouped into ontologies, with the result that an OWL DL ontology is equivalent to a Description Logic Knowledge Base, i.e., a Tbox plus an Abox (see Section **??**). This is not completely standard, as ontologies are more typically thought of as describing only the structure of a domain (in terms of classes and properties), and not as describing a particular situation (in terms of instances of classes and properties); in this more common usage, an ontology is therefore equivalent to a Description Logic Tbox, and not to the combination of a Tbox and an Abox.

Ontologies can be given annotations, just like classes, properties, and individuals. Ontologies can also have ontology properties as annotations. Ontology properties are just like annotations except that they relate ontologies to other ontologies. Most ontology properties act just like annotation properties (i.e., they provide uniterpreted information), but one special property, `owl:imports`, can be used to "import" other ontologies. Importing an ontology effectively treats the content of a web page as part of the current ontology, as in:

```
Ontology(Simple
   Annotation(rdfs:comment "A simple ontology for nationality")
   Annotation(owl:imports http://www.foo.ex/simpler.owl)
   ....
)
```

The meaning of the above imports statement is that the set of axioms in the `Simple` ontology is taken to include the set of axioms in the `simpler.owl` ontology.

14.3.4 Semantics for OWL DL

A formal semantics, very similar to the semantics provided for Description Logics (see Section **??**), is provided for this style of using OWL. Full details on this model theory can be found in the OWL Semantics and Abstract Syntax [Patel-Schneider *et al.*, 2004].

Because OWL includes datatypes, the semantics for OWL is very similar to that of Description Logics that also incorporate datatypes, in particular $\mathcal{SHOQ}(\mathcal{D})$. However, the particular datatypes used in OWL are taken from RDF and XML Schema Datatypes [Biron and Malhotra, 2001]. Data values such as `"44"^^xsd:integer` thus mean what they would mean as XML Schema data values.

The semantics for OWL DL does include some aspects that may be viewed as unusual from a description logic perspective. Annotations are given a simple separate meaning, not shown here, that can be used to associate information with classes, properties, and individuals in a manner compatible with the RDF semantics. Ontologies also live within the semantics and can be given annotation information. Finally, `owl:imports` is given a meaning that involves finding the referenced ontology (if possible) and adding its meaning to the meaning of the current ontology. In other respects, however, the meaning of an OWL ontology should be exactly equivalent to the meaning of the Description Logic knowledge base derived via the correspondences given in Figures 14.2 and 14.3.

What makes OWL DL (and OWL Lite) a Semantic Web language, therefore, is not its semantics, which are quite standard for a Description Logic, but instead the use of URI references for names, the use of XML Schema datatypes for data values, and the ability to connect to documents in the World Wide Web.

14.3.5 OWL Lite and OWL Full

As we have seen, OWL DL is related to $\mathcal{SHOIN}(\mathbf{D})$, a very expressive Description Logic. This Description Logic is somewhat difficult to present to naive users, as it is possible to build complex boolean descriptions using,

for example, union and complement. $\mathcal{SHOIN}(\mathbf{D})$ is also difficult to reason with, as key inference problems have NExpTime complexity, and somewhat difficult to build even non-reasoning tools for, because of the complex descriptions.

For these reasons, a subset of OWL DL has been identified that should be easier on all the above metrics; this subset is called OWL Lite. OWL Lite prohibits unions and complements, restricts intersections to the implicit intersections in the frame-like class axioms, limits all embedded descriptions to class names, does not allow individuals to occur in descriptions or class axioms, and limits cardinalities to 0 or 1.

These restrictions make OWL Lite expressively equivalent to $\mathcal{SHIF}(\mathbf{D})$. As in $\mathcal{SHIF}(\mathbf{D})$, key inferences in OWL Lite can be computed in worst case exponential time (ExpTime), and there are already several optimized reasoners for logics equivalent to OWL Lite (see Section 14.3.7). This improvement in tractability comes with relatively little loss in expressive power—although OWL Lite syntax is more restricted than that of OWL DL, it is still possible to express complex descriptions by introducing new class names and exploiting the implicit negations introduced by disjointness axioms [Horrocks and Patel-Schneider, 2003]. Using these techniques, all OWL DL descriptions can be captured in OWL Lite except those containing either individual names or cardinalities greater than 1.

OWL DL and OWL Lite are extensions of a restricted use of RDF and RDFS, because, unlike RDF and RDFS, they do not allow classes to be used as individuals, and the language constructors cannot be applied to the language itself. For users who want these capabilities, a version of OWL that is upward compatible with RDF and RDFS has been provided; this version is called OWL Full. In OWL Full, all RDF and RDFS combinations are allowed. For example, in OWL Full, it is possible to impose a cardinality constraint on `rdfs:subClassOf`, if so desired.

OWL Full contains OWL DL, but goes well outside the standard Description Logic framework. The penalty to be paid here is two-fold. First, reasoning in OWL Full is undecidable. (Showing the undecidability is trivial, because restrictions required in order to maintain the decidability of OWL DL do not apply to OWL Full [Horrocks *et al.*, 1999], but also as a result of the ability to apply OWL's expressive power to RDF syntax, as exemplified above [Motik, 2005].) Second, the abstract syntax for OWL DL is inadequate for OWL Full, and the official OWL exchange syntax, RDF/XML, must be used.

OWL Full has been given a model-theoretic semantics that is a vocabulary extension of the RDF model theory [Patel-Schneider *et al.*, 2004; Hayes, 2004]. A correspondence between this semantics and the semantics of OWL DL has also been established: it has been shown that the model theory for OWL DL has very similar consequences to this RDF-style model theory for those OWL ontologies that can be written in the OWL DL abstract syntax [Patel-Schneider *et al.*, 2004].

More formally, given two OWL DL ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$, written in the abstract syntax, if $\mathcal{O}_1$ entails $\mathcal{O}_2$ according to the OWL DL model theory then the mapping of $\mathcal{O}_1$ into RDF triples will entail the mapping of $\mathcal{O}_2$ into RDF triples according to the OWL Full model theory.

The converse, however, is not true. Although the correspondence is usually exact, it has been shown that there are at least some pathological cases where the correspondence breaks down. For example, in OWL Full `owl:Thing` contains individuals corresponding to the RDFS and OWL vocabularies (such as rdf:Property, rdfs:subClassOf, and owl:hasValue). (There are no such individuals in the model theory for OWL DL.) Limiting the extent of `owl:Thing`, for example as in

```
SubClassOf(owl:Thing oneOf(ex:foo ex:bar))
```

forces equalities in this vocabulary, which have truely unusual effects, in the OWL Full model theory.

In order to avoid any possible confusion as to the meaning of OWL DL, the OWL Full model theory has been given “non-normative” status (i.e., it is only informative) for OWL ontologies that can be written in the abstract syntax—for such ontologies, the definitive semantics is given by the OWL DL model theory.

14.3.6 OWL Datatypes

As well as dealing with “abstract” classes such as `Person` and `Animal`, many practical applications need to represent and reason about datatypes and values such as integers and strings. The integration of datatypes in the OWL language is again heavily influenced by Description Logic research, which has demonstrated that care is required in order to avoid complexity blow-ups or even undecidability being caused by datatypes [Lutz, 2002]. In the $\mathcal{SHOQ}(\mathcal{D})$ Description Logic it was shown that this could be achieved by strictly separating the interpretation of datatypes and values from that of classes and individuals: $\mathcal{SHOQ}(\mathcal{D})$ interpretations include an additional interpretation domain for data values $\imath^{\mathcal{I}}_{\mathbf{D}}$ which is disjoint from the domain of individuals $\imath^{\mathcal{I}}$. Datatypes, such as integer, are interpreted as a subset

of $\imath^{\mathcal{I}}_{\mathbf{D}}$, and values such as the integer “35” are interpreted as elements of $\imath^{\mathcal{I}}_{\mathbf{D}}$. The separation is further strengthened by dividing properties into two disjoint sets of abstract and datatype properties. Abstract properties such as `brother` are interpreted as binary relations on $\imath^{\mathcal{I}}$ (i.e., subsets of $\imath^{\mathcal{I}} \times \imath^{\mathcal{I}}$), while datatype properties such as `age` are interpreted as binary relations between $\imath^{\mathcal{I}}$ and $\imath^{\mathcal{I}}_{\mathbf{D}}$ (i.e., subsets of $\imath^{\mathcal{I}} \times \imath^{\mathcal{I}}_{\mathbf{D}}$).

This design has the advantage that reasoning with datatypes and values can be almost entirely separated from reasoning with classes and individuals—a class based reasoner simply needs access to a datatype “oracle” that can answer simple questions with respect to datatypes and values (e.g., “is -5 a nonNegative Integer?”). Moreover, the language remains decidable if datatype and value reasoning is decidable, i.e., if the oracle can guarantee to answer all questions of the relevant kind for supported datatypes. This can easily be achieved for a range of common datatypes such as integers, decimals and strings [Lutz, 2002].

As well as these practical considerations, it can also be argued that the separation of classes and datatypes makes sense from a philosophical standpoint as datatypes are already structured by built in predicates such as greater-than and less-than. From this point of view, it does not make sense to use ontology axioms to add further structure to datatypes or to form “hybrid” classes such as the class of red integers.

14.3.7 Reasoners, tools and applications

As discussed in Section 14.3.1, an important motivation for the design of OWL DL and OWL Lite was the ability for applications to exploit known reasoning algorithms and existing (highly optimised) reasoner implementations. This meant that, even before the OWL specification was finalised, prototype tools and applications could make use of Description Logic reasoning systems such as FaCT [Horrocks, 1998a], Pellet [Pellet, 2003] and Racer [Haarslev and Möller, 2001]. The use of FaCT and Racer was also facilitated by the fact that both systems provide a standard application interface designed by the Description Logic Implementation Group (DIG) [Bechhofer *et al.*, 1999].[5]

At the time when OWL achieved recommendation status (the final stage in the W3C standardisation process), the above systems were only able to support OWL Lite reasoning. This was because, at that time, no suitable algorithm was known for $\mathcal{SHOIN}$, and all of the implementations were

[5] The Pellet system now also provides a DIG compliant interface.

based on the $\mathcal{SHIQ}$ algorithms described in [Horrocks *et al.*, 1999] and [Horrocks *et al.*, 2000b]. In order to process OWL DL ontologies, reasoning systems typically applied some "work-around" with respect to nominals, e.g., by treating them as primitive classes. This work-around is sound but incomplete for subsumption. I.e., given ontologies $\mathcal{O}$and $\mathcal{O}'$, where $\mathcal{O}'$ has been derived from $\mathcal{O}$by replacing each occurrence of a nominal o with a primitive class C_o, a class C is subsumed by a class D with respect to $\mathcal{O}$if C is subsumed by a D with respect to $\mathcal{O}'$, but if C is *not* subsumed by a D with respect to $\mathcal{O}'$, then we cannot be sure that C is *not* subsumed by a D with respect to $\mathcal{O}$. Clearly, the work-around is unsound for satisfiability, i.e., there may be concepts that are satisfiable with respect to $\mathcal{O}'$, but unsatisfiable with respect to $\mathcal{O}$.

This situation was clearly very unsatisfactory given the motivation for OWL's Description Logic based design, and it was always anticipated that existing tableaux decision procedures for $\mathcal{SHIQ}$ and $\mathcal{SHOQ}$ would soon be extended to $\mathcal{SHOIQ}$. Although this took a little longer than anticipated, a tableaux decision procedure for $\mathcal{SHOIQ}$ was eventually developed [Horrocks and Sattler, 2005], and the Pellet system now uses this algorithm to fully support OWL DL. It is expected that other reasoners, including FaCT++ [Tsarkov and Horrocks, 2005] (the successor of the FaCT system), will soon follow suit.

It is interesting to note that OWL is now supported by commercial Description Logic systems. These include the Cerebra system from Cerebra Inc (formerly Network Inference), and RacerPro, a commercial version of the Racer system. There are also several OWL reasoners that are not based on tableaux decision procedures. These include KAON2, a system that uses a reduction of $\mathcal{SHIQ}(\mathcal{D})$ to disjunctive datalog [Hustadt *et al.*, 2004], and Hoolet, a system that uses the Vampire First Order Theorem prover via a translation of $\mathcal{SHOIN}(\mathbf{D})$ into FOL [Tsarkov *et al.*, 2004].

The growing importance of ontologies, and the emergence of the OWL standard, has also given impetus to the development of ontology engineering tools, including tools for editing, validating, visualising, merging and debugging OWL ontologies. Several Application Programming Interfaces (APIs) for OWL are also available.

Of the available OWL Editing tools, probably the best known and most widely used is Protégé [Gennari *et al.*, 2003]. Protégé is a frame based editor that supports OWL via an OWL Plugin. The Plugin uses a range of techniques (some of which were first developed in the OilEd editor [Bechhofer *et al.*, 2001b]) to extend the language that can be dealt with, e.g.,

by explicitly specifying quantification with slots and allowing for multiple necessary and sufficient conditions in class definitions. Even so, Protégé still has some restrictions with respect to the OWL DL language; it does not, for example, support arbitrary classes in restrictions (but only named classes). In addition to the frame editor, there are additional Protégé plugins supporting, e.g., ontology visualisation, ontology documentation and "wizards" that can automate some basic steps in the ontology development process. Protégé can connect to any Description Logic reasoner with a DIG compliant interface, and uses the reasoner to check class consistency, to compute the class hierarchy, and to compute the most specific class(es) that each individual is an instance of.

Several other OWL editing tools are also available. These include OilEd [Bechhofer *et al.*, 2001b] (from Manchester University), SWOOP [Kalyanpur *et al.*, 2005a] (from the Pellet team) and Construct (from Cerebra). The design of OilEd is based on that of Protégé, but it provides more complete support for OWL DL. Like Protégé, OilEd can use any DIG compliant reasoner to reason over the ontology. SWOOP is browser based, and is much more tightly linked to OWL's syntactic structure; it provides both abstract syntax and RDF/XML syntax editing modes, and fully supports OWL DL. SWOOP uses the Pellet system for reasoning support, and also has an integrated debugger [Kalyanpur *et al.*, 2005b]. Construct is a graphical tool that uses a UML like notation; it uses the Cerebra reasoner to provide reasoning support.

OWL editing tools are also expanding into the software engineering realm, as tools such as Sandpiper Software's Medius Visual Ontology Modeler supports ontology development using UML modelling tools with output in OWL (see `http://www.sandsoft.com/`). This enables broader communities to model, edit, and integrate with OWL ontologies.

Although OWL was initially developed as an ontology language for the Semantic Web, OWL has also been widely adopted as a de facto standard for ontology based applications. These include "traditional" applications in e-Science and Medicine, as well as applications in industry and government. The advantages of using OWL in such applications include the relative stability and interoperability conferred by a W3C standard, and the availability of an expanding range of reasoners and tools.

Examples of OWL ontology applications include:

- Ontologies developed by members of the Open Biomedical Ontologies Consortium (see `http://obo.sourceforge.net/`), which recommends OWL as the exchange language for all Life Science ontologies. These include the widely used Gene Ontology (GO) and Microarray Gene Expression Data (MGED) ontology.

- The US National Cancer Institute (NCI) "thesaurus", an ontology containing the working vocabulary used in NCI data system (see `http://ncicb.nci.nih.gov/NCICB/core/EVS/`).
- United Nations Food and Agriculture Organization (FAO) is using OWL to develop a range of ontologies covering areas such as agriculture and fisheries (see `http://www.fao.org/agris/aos/Applications/intro.htm`).
- The Semantic Web for Earth and Environmental Terminology (SWEET) ontologies developed at the US National Aeronautics and Space Administration (NASA) Jet Propulsion Laboratory (see `http://sweet.jpl.nasa.gov/ontology/`). These include ontologies describing space, the biosphere and the sun.
- An ontology used at General Motors in a project to help quality improvement activities for assembly line processes in different production sites [Morgan *et al.*, 2005].

14.4 Summary

The Semantic Web is envisaged as an evolution of the existing Web where terms defined in ontologies will be used to give well defined and machine-processable meaning to Web accessible information and services. OWL, a Description Logic based ontology language, has been designed for this purpose.

Because of the ambitious design goals for OWL, because of the multiple influences on OWL, and also because of the structural requirements constraining OWL, the development of OWL has not been without problems. Through hard work and compromise, these problems have largely been overcome, resulting in a ontology language that is truly part of the Semantic Web.

It was not possible to simultaneously satisfy all of the constraints on OWL, so two styles of using OWL have been developed, each appropriate under different circumstances.

One style of OWL usage is motivated by the need unambiguously represent information in an expressive language, but one that can still be reasoned with predictably. When this is a primary goal, OWL DL will be the target language, and it has been used in a number of existing applications. When using OWL DL, some compatibility with RDF is lost, mostly having to do with using classes and properties as individuals. On the other hand, users of OWL DL benefit from decidable inference, and the availability of an increasingly wide range of tools and infrastructure, including efficient reasoning systems and sophisticated ontology development environments.

OWL DL also has a frame-like alternative syntax that can be used to make working with OWL easier.

Even though OWL DL is, essentially, a description logic, it also includes features that place it firmly in the Semantic Web. OWL DL uses the datatyping mechanisms from RDF and many of the built-in XML Schema datatypes. OWL DL uses RDF URI references as names, including the names from RDF, RDFS, and XML Schema datatypes that are relevant. Entailment in OWL DL is compatible with entailment in RDF and RDFS.

For users who still need unambiguous representation and predictable reasoning, but for whom simplicity is more important than expressive power, the OWL Lite subset of OWL DL may be a good choice. This sublanguage rules out some of the things that can be said in OWL DL, but still retains considerable expressive power. Moreover, OWL Lite is supported by a wider range of reasoning tools, and as key reasoning tasks are of lower worst case complexity, these reasoners might be expected to be more efficient, in general, than OWL DL reasoners.

The other style of OWL usage is one where compatibility with RDF is the overarching concern. In this case, OWL Full would be an appropriate choice. OWL Full extends RDF and RDFS to a full ontology language, with a well-specified entailment relationship that extends entailment in RDF and RDFS, while avoiding any paradoxes that might arise. However, entailment in OWL Full is undecidable, which is a significant issue in most circumstances, and no effective tools for reasoning are available for OWL Full, nor are they expected to appear. Also, the user-friendly alternative syntax is not adequate for OWL Full, so RDF/XML must be used.

In practice, relatively few OWL Full applications have emerged to date, and where OWL Full ontologies are found, they often turn out to be outside the OWL DL subset only as the result of minor syntactic errors. Fragments of OWL are, however, sometimes used as ad hoc extensions to RDFS. A common example is the use of OWL functional properties, and explicit equivalences and (in)equalities, in what would otherwise be an RDFS ontology.

There remain, of course, significant issues that are deliberately not handled by OWL, but which are definitely relevant to many Semantic Web use cases:

- OWL avoids anything related to nonmonotonicity (such as default reasoning and localised closed world assumptions);
- OWL's limited expressiveness excludes operations such as property-chaining, or, more generally, axioms with variables, such as rules (although there are already proposals for extensions in this direction [Horrocks *et al.*, 2005; Eiter *et al.*, 2004;

Motik *et al.*, 2004]);
- OWL's import mechanism is limited, and does not support fine-grained operations (such as the importation of parts of ontologies);
- OWL integrates datatypes in a very clean way, but there is no notion of operations on these datatypes (such integer arithmetic or string operations).

Extending the current Semantic Web with some or all of these features will require not only a standardisation effort, but sets a significant research challenge to the community.

Bibliography

[Baader and Hanschke, 1991] Franz Baader and Philipp 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.

[Bechhofer *et al.*, 1999] Sean Bechhofer, Ian Horrocks, Peter F. Patel-Schneider, and Sergio Tessaris. A proposal for a description logic interface. In *Proc. of the 1999 Description Logic Workshop (DL'99)*, pages 33–36. CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/Vol-22/`, 1999.

[Bechhofer *et al.*, 2000] Sean Bechhofer, Jeen Broekstra, Stefan Decker, Michael Erdmann, Dieter Fensel, Carole Goble, Frank van Harmelen, Ian Horrocks, Michel Klein, Deborah L. McGuinness, Enrico Motta, Peter F. Patel-Schneider, Steffen Staab, and Rudi Studer. An informal description of OIL-Core and Standard OIL: a layered proposal for DAML-O. Technical Report KSL-00-19, Stanford University KSL, November 2000. Available at `http://www.ontoknowledge.org/oil/downl/dialects.pdf`.

[Bechhofer *et al.*, 2001a] Sean Bechhofer, Carole Goble, and Ian Horrocks. DAML+OIL is not enough. In *Proc. of the 2001 Int. Semantic Web Working Symposium (SWWS 2001)*, pages 151–159, 2001. Available at `http://www.semanticweb.org/SWWS/program/full/SWWSProceedings.pdf`.

[Bechhofer *et al.*, 2001b] Sean Bechhofer, Ian Horrocks, Carole Goble, and Robert 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, 2001. Appeared also in Proc. of the 2001 Description Logic Workshop (DL 2001).

[Bechhofer *et al.*, 2004] Sean Bechhofer, Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. OWL web ontology language reference. W3C Recommendation, 10 February 2004.

[Berners-Lee *et al.*, 2001] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific American*, 284(5):34–43, May 2001.

[Biron and Malhotra, 2001] Paul V. Biron and Ashok Malhotra. XML schema part 2: Datatypes. W3C Recommendation, May 2001. Available at `http://www.w3.org/TR/xmlschema-2/`.

[Brachman *et al.*, 1999] Ronald J. Brachman, Alexander Borgida, Deborah L. McGuinness, and Peter Patel-Schneider. Reducing CLASSIC to practice:

Knowledge representation theory meets reality. *Artificial Intelligence*, 114(1–2):203–237, 1999.

[Chaudhri *et al.*, 1998] Vinay K. Chaudhri, Adam Farquhar, Richard Fikes, Peter D. Karp, and James Rice. OKBC: A programmatic foundation for knowledge base interoperability. In *Proc. of the 15th Nat. Conf. on Artificial Intelligence (AAAI'98)*, pages 600–607, 1998.

[Decker *et al.*, 2000] Stefan Decker, Dieter Fensel, Frank van Harmelen, Ian Horrocks, Sergey Melnik, Michel Klein, and Jeen Broekstra. Knowledge representation on the web. In *Proc. of the 2000 Description Logic Workshop (DL 2000)*, pages 89–97. CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/Vol-33/`, 2000.

[Eiter *et al.*, 2004] Thomas Eiter, Thomas Lukasiewicz, Roman Schindlauer, and Hans Tompits. Combining answer set programming with description logics for the semantic web. In *Proc. of the 9th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2004)*, pages 141–151. Morgan Kaufmann, Los Altos, 2004.

[Farquhar *et al.*, 1996] Adam Farquhar, Richard Fikes, and James Rice. The ontolingua server: A tool for collaborative ontology construction. In *Proceedings of the Tenth Knowledge Acquisition for Knowledge-Based Systems Workshop*, November 1996. Also available as Stanford University Knowledge Systems Lab Tech Report: KSL-TR-96-26.

[Fensel *et al.*, 2000] Dieter Fensel, Ian Horrocks, Frank van Harmelen, Stefan Decker, Michael Erdmann, and Michel 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.

[Fensel *et al.*, 2001] Dieter Fensel, Frank van Harmelen, Ian Horrocks, Deborah L. McGuinness, and Peter F. Patel-Schneider. OIL: An ontology infrastructure for the semantic web. *IEEE Intelligent Systems*, 16(2):38–45, 2001.

[Fikes and McGuinness, 2001] Richard E. Fikes and Deborah L. McGuinness. An axiomatic semantics for RDF, RDF Schema, and DAML+OIL. Technical Report KSL-01-01, Stanford University KSL, 2001. Available at `http://www.ksl.stanford.edu/people/dlm/daml-semantics/abstract-axiomatic-semantics.html`.

[Gennari *et al.*, 2003] John H. Gennari, Mark A. Musen, William E. Grosso, Monica Crubézy, Henrik Eriksson, Natalya F. Noy, and Samson W. Tu. The evolution of protégé. *International Journal of Human Computer Studies*, 58(1):89–123, 2003.

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

[Haarslev and Möller, 2001] Volker Haarslev and Ralf 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.

[Hayes, 2004] Patrick Hayes. RDF model theory. W3C Recommendation, 10 February 2004.

[Heflin *et al.*, 2003] Jeff Heflin, James Hendler, and Sean Luke. SHOE: A blueprint for the semantic web. In D. Fensel, J. Hendler, H. Lieberman, and

W. Wahlster, editors, *Spinning the Semantic Web*. MIT Press, Cambridge, MA, 2003.

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

[Horrocks and Patel-Schneider, 2003] Ian Horrocks and Peter F. Patel-Schneider. Reducing OWL entailment to description logic satisfiability. In Dieter Fensel, Katia Sycara, and John Mylopoulos, editors, *Proc. of the 2003 International Semantic Web Conference (ISWC 2003)*, number 2870 in Lecture Notes in Computer Science, pages 17–29. Springer, 2003.

[Horrocks and Sattler, 2001] Ian Horrocks and Ulrike Sattler. Ontology reasoning in the $\mathcal{SHOQ}$(D) description logic. In *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001)*, pages 199–204, 2001.

[Horrocks and Sattler, 2005] Ian Horrocks and Ulrike Sattler. A tableaux decision procedure for $\mathcal{SHOIQ}$. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*, 2005.

[Horrocks *et al.*, 1999] Ian Horrocks, Ulrike Sattler, and Stephan Tobies. Practical reasoning for expressive description logics. In Harald Ganzinger, David McAllester, and Andrei 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, 1999.

[Horrocks *et al.*, 2000a] I. Horrocks, D. Fensel, J. Broekstra, S. Decker, M. Erdmann, C. Goble, F. van Harmelen, M. Klein, S. Staab, R. Studer, and E. Motta. OIL: The Ontology Inference Layer. Technical Report IR-479, Vrije Universiteit Amsterdam, Faculty of Sciences, September 2000.

[Horrocks *et al.*, 2000b] Ian Horrocks, Ulrike Sattler, and Stephan Tobies. Reasoning with individuals for the description logic $\mathcal{SHIQ}$. In David McAllester, editor, *Proc. of the 17th Int. Conf. on Automated Deduction (CADE 2000)*, volume 1831 of *Lecture Notes in Computer Science*, pages 482–496. Springer, 2000.

[Horrocks *et al.*, 2003] Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. From $\mathcal{SHIQ}$ and RDF to OWL: The making of a web ontology language. *J. of Web Semantics*, 1(1):7–26, 2003.

[Horrocks *et al.*, 2005] Ian Horrocks, Peter F. Patel-Schneider, Sean Bechhofer, and Dmitry Tsarkov. OWL rules: A proposal and prototype implementation. *J. of Web Semantics*, 3(1):23–40, 2005.

[Horrocks, 1998a] Ian Horrocks. The FaCT system. In Harrie 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, 1998.

[Horrocks, 1998b] Ian 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.

[Hustadt *et al.*, 2004] Ullrich Hustadt, Boris Motik, and Ulrike Sattler. Reducing SHIQ-description logic to disjunctive datalog programs. In *Proc. of the 9th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2004)*, pages 152–162, 2004.

[Kalyanpur *et al.*, 2005a] Aditya Kalyanpur, Bijan Parsia, and James Hendler. A tool for working with web ontologies. *International Journal on Semantic Web and Information Systems*, 1(1):36–49, 2005.

[Kalyanpur *et al.*, 2005b] Aditya Kalyanpur, Bijan Parsia, Evren Sirin, and James

Hendler. Debugging unsatisfiable classes in owl ontologies. In *Journal of Web Semantics — Special Issue of the Semantic Web Track of WWW2005*, 2005. (To Appear).

[Karp *et al.*, 1999] Peter D. Karp, Vinay K. Chaudhri, and Jerome Thomere. XOL: An XML-based ontology exchange language. Technical Report SRI AI Technical Note 559, SRI International, Menlo Park (CA, USA), 1999.

[Lutz, 2002] Carsten Lutz. *The Complexity of Reasoning with Concrete Domains*. PhD thesis, Teaching and Research Area for Theoretical Computer Science, RWTH Aachen, 2002.

[Mallery, 1994] John Mallery. A Common LISP hypermedia server. In *Proc. of the 1st Int. Conf. on The World-Wide Web*. CERN, 1994. Available at `http://www.ai.mit.edu/projects/iiip/doc/cl-http/server-abstract.html`.

[Manola and Miller, 2004] Frank Manola and Eric Miller. RDF primer. W3C Recommendation, 10 February 2004.

[McGuinness *et al.*, 1997] Deborah L. McGuinness, Harley Manning, and Tom Beattie. Knowledge augmented intranet search. In *Proc. of the 6th World Wide Web Conference CDROM version*, 1997.

[McGuinness *et al.*, 2000] Deborah L. McGuinness, Richard Fikes, James Rice, and Steve Wilder. An environment for merging and testing large ontologies. In *Proc. of the 7th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2000)*, pages 483–493, 2000.

[McGuinness *et al.*, 2002] Deborah L. McGuinness, Richard Fikes, Lynn A. Stein, and James Hendler. DAML-ONT: An ontology language for the semantic web. In Dieter Fensel, Hendler, Henry Lieberman, and Wolfgang Wahlster, editors, *The Semantic Web: Why, What, and How*. The MIT Press, 2002.

[McGuinness, 1998] Deborah L. McGuinness. Ontological issues for knowledge-enhanced search. In *Proceedings of Formal Ontology in Information Systems.*, 1998. Also published in *Frontiers in Artificial Intelligence and Applications*, IOS-Press, 1998.

[Meghini *et al.*, 1997] Carlo Meghini, Fabrizio Sebastiani, and Umberto Straccia. Modelling the retrieval of structured documents containing texts and images. In Costantino Thanos, editor, *Proc. of the 1st European Conf. on Research and Advanced Technology for Digital Libraries (ECDL'97)*, volume 1324 of *Lecture Notes in Computer Science*. Springer, 1997.

[Miller, 1995] George A. Miller. WordNet: A lexical database for English. *Communications of the ACM*, 38(11):39–41, 1995.

[Morgan *et al.*, 2005] Alexander P. Morgan, John A. Cafeo, Kurt Godden, Ronald M. Lesperance, Andrea M. Simon, Deborah L. McGuinness, and James L. Benedict. The general motors variation-reduction adviser. *AI Magazine*, 26(2), 2005.

[Motik *et al.*, 2004] Boris Motik, Ulrike Sattler, and Rudi Studer. Query answering for OWL-DL with rules. In *Proc. of the 2004 International Semantic Web Conference (ISWC 2004)*, pages 549–563, 2004.

[Motik, 2005] Boris Motik. On the properties of metamodeling in OWL. In *Proc. of the 2005 International Semantic Web Conference (ISWC 2005)*, Lecture Notes in Computer Science, page To appear. Springer, 2005.

[Patel-Schneider *et al.*, 2004] Peter F. Patel-Schneider, Patrick Hayes, and Ian Horrocks. OWL web ontology language semantics and abstract syntax. W3C Recommendation, 10 February 2004.

[Pellet, 2003] Pellet OWL reasoner. Maryland Information and Network Dynamics

Lab, 2003.

[Rousset, 1999] Marie-Christine Rousset. Query expansion in description logics and CARIN. In *Working Notes of the AAAI Fall Symposium on "Question Answering Systems"*, 1999.

[Smith *et al.*, 2004] Michael K. Smith, Chris Welty, and Deborah L. McGuinness. OWL web ontology language guide. W3C Recommendation, 10 February 2004.

[Tsarkov and Horrocks, 2005] Dmitry Tsarkov and Ian Horrocks. Ordering heuristics for description logic reasoning. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*, 2005.

[Tsarkov *et al.*, 2004] Dmitry Tsarkov, Alexandre Riazanov, Sean Bechhofer, and Ian Horrocks. Using Vampire to reason with OWL. In Sheila A. McIlraith, Dimitris Plexousakis, and Frank van Harmelen, editors, *Proc. of the 2004 International Semantic Web Conference (ISWC 2004)*, number 3298 in Lecture Notes in Computer Science, pages 471–485. Springer, 2004.

[van Harmelen *et al.*, 2001] Frank van Harmelen, Peter F. Patel-Schneider, and Ian Horrocks. A model-theoretic semantics for DAML+OIL (March 2001), March 2001.

[Welty and Jenkins, 2000] Christopher Welty and Jessica Jenkins. Untangle: a new ontology for card catalog systems. In *Proc. of the 17th Nat. Conf. on Artificial Intelligence (AAAI 2000)*, pages 1137–1138. AAAI Press/The MIT Press, 2000.

[Welty, 1994] Christopher Welty. A knowledge-based email distribution system. In *Proc. of the 1994 Florida AI Research Symposium*. AAAI Press/The MIT Press, May 1994.

[Welty, 1996a] Christopher Welty. An HTML interface for Classic. In *Proc. of the 1996 Description Logic Workshop (DL'96)*, number WS-96-05 in AAAI Technical Report. AAAI Press/The MIT Press, 1996.

[Welty, 1996b] Christopher Welty. Intelligent assistance for navigating the web. In *Proc. of the 1996 Florida AI Research Symposium*. AAAI Press/The MIT Press, May 1996.

[Welty, 1998] Christopher Welty. The ontological nature of subject taxonomies. In *Proc. of the Int. Conf. on Formal Ontology in Information Systems (FOIS'98)*, Frontiers in Artificial Intelligence. IOS Press, 1998.