Handbook of Knowledge Representation
Edited by F. van Harmelen, V. Lifschitz and B. Porter
© 2008 Elsevier B.V. All rights reserved
DOI: 10.1016/S1574-6526(07)03021-0

Chapter 21

The Semantic Web: Webizing Knowledge Representation

Jim Hendler and Frank van Harmelen

Abstract
The World Wide Web opens up new opportunities for the use of knowledge representation: a formal description of the semantic content of Web pages can allow better processing by computational agents. Further, the naming scheme of the Web, using Universal Resource Indicators, allows KR systems to avoid the ambiguities of natural language and to allow linking between semantic documents. These capabilities open up a raft of new possibilities for KR, but also present challenges to some traditional KR assumptions.

21.1 Introduction

The web-page http://www.cs.rpi.edu/~hendler is not much different than most other pages in many ways. Besides content, it contains many links to other pages: links to pages of students, links to downloadable files, links to various digital libraries, links to the Web resources used in classes and to University pages that describe when the classes were given, what the prerequisites were, etc. In short, a great deal of the information "on" this page is not actually on the page at all, it is provided by the linking mechanisms of the Web. It is, in fact, exactly this network effect of gaining advantage by linking to information created by other people, rather than recreating it locally, that makes the Web so powerful.

Now consider knowledge representation. When trying to create a machine-readable KR page that would contain similar information, we could not get this kind of network effect using the KR techniques described in most of the chapters in this book. First, even if we were to use a particular representation technique, and even if it is a well-defined technique like FOL, there is still the issue of using information defined by someone else. One author might write:

ForAll(x)(Advisor(x, Hendler) $\rightarrow$ StudentOf(Hendler,x).

while another might put

```
Advisor(_x,_ y) :- PhDAdvisor(_y,_x).
PhDAdvisor(Hendler,Smith).
```

Try to unify these KBs. Even though there is no logical mismatch, the mere syntactic differences between the representations makes it impossible to simply re-use the knowledge between KBs. The problem would be even worse when the two KBs were using different forms of KR, say some particular subset of FOL, some particular temporal logic, or some kind of modal operators.

Even when using the same exact logical language, say the Conceptual Graphs that John Sowa describes in Chapter 5, and even when using the same implementation (so syntax matters go away), we still do not have the kind of linking we have on the Web. Most KR systems do not have a mechanism by which to specify that a KB living somewhere else should be included at query time so as to make use of the knowledge defined by someone else. In short, we do not have a way to get the network effect in KR that we get in the Web world.

In fact, in many KR systems the notion of knowledge not directly under the control of a single mechanism, and not incorporated at what would be the equivalent of compile time, is anathema to the design. It can lead to inconsistency in all sorts of nasty ways. For example, one KB might be using knowledge in a way that is incompatible or inconsistent with another via unexpected interactions. If one KB said "man" implies "male" where the other was using the term in the non-gendered "all men are mortals" sense, then, when our KBs are linked a mother from one system becomes a male in the other system, but mothers are known to be female, and thus we have a contradiction from which all manner of improper things could be inferred in many systems, requiring belief revision at the least. Or consider even if the terms are used correctly, but at query time the other KB's server is down, and thus the list of students varies, depending on the uptime of the server, again leading to potential problems.

Traditionally, the field of knowledge representation has faced these potential problems by either ignoring them (by assuming people are using the same KR system, or doing all merging at "compile" time), by addressing them as special cases (such as in the design of temporal reasoners (cf. Chapter 12) or belief revision systems (cf. Chapter 8)) or by defining the problem away. This latter is generally done by using inexpressive languages that do not allow inconsistency, or defining inconsistency as an "error" that will be handled offline.

Additionally, there is another issue that KR systems in AI have tended to ignore: the issue of scaling. KR often talks of algorithmic complexity, or even performance issues, but compared to the size of a good database system, or an incredible information space like the World Wide Web, KR systems have lagged far behind. The engineering challenges proposed by KBs that could be linked together to take advantage of the network effect that could be achieved thereby, are beyond the scaling issues explored in most AI work today.

In short, there is a set of KR challenges that have not been widely explored until recently. First, solving syntactic interoperability problems demands standards: not just at some kind of KR logic level, but all the way down to the nitty-gritty syntactic details. Second, linking KR systems requires "extra-logical" infrastructure that can be exploited to achieve the network effect. Third, the languages designed need to be scalable, at least in some sense thereof, to much larger sizes than traditional in AI work. Fourth, and finally, achieving such linkage presents challenges to current KR formulations demanding new kinds of flexibility and addressing issues that have largely been previously ignored.

From a KR perspective, designing systems to overcome these challenges, using the Web itself for much of the extra-logical infrastructure, is the very definition of what has come to be known as the "Semantic Web". It was this thinking that the authors of a widely cited vision paper on the Semantic Web [2] to conclude that

> Knowledge representation is currently in a state comparable to that of hypertext before the advent of the web: it is clearly a good idea, and some very nice demonstrations exist, but it has not yet changed the world. It contains the seeds of important applications, but to unleash its full power it must be linked into a single global system.

Other articles have been written that explain how Semantic Web systems are like traditional KR systems (cf. Chapter 3 which describes the correspondence of the Semantic Web language OWL to description logics), and thus this article will concentrate on the other side of this: the things that make Semantic Web KR different from traditional systems.

However, before we go on, it is important to note one way in which this chapter differs from many of the others in this Handbook. The KR languages that we will discuss here are not academic efforts aimed at extending the philosophical reach of computational reasoning. They are languages that were designed as standards with an eye towards widespread use. The languages RDF, RDFS and OWL, which we will discuss in the remainder of this chapter, are without question the most widely used KR languages in history. A web search performed around the beginning of 2007 finds millions of RDF and RDFS documents, and tens of thousands of OWL ontologies. The user community goes way beyond the traditional AI users, and these languages form the basis of a new phase of commercial development going forward under the name "Web 3.0". This article discusses these languages from a KR perspective, but a realization of the scale of the deployment, the wide range of users, and the power that has been achieved through the standardization of these KR languages is crucial to an understanding of their design.

21.2 The Semantic Web Today

The Semantic Web is an extension of the current World Wide Web in which information is tied to machine-readable metadata, making it easy to exchange, integrate and process data in a systematic, machine-automated manner. Using standardized languages, published as World Wide Web Consortium (W3C) recommendations, Semantic Web data cannot only explicitly describe the knowledge content underlying HTML pages, but also specify the implicit information contained in media like images and videos, or be a publicly accessible and usable representation of an otherwise inaccessible database or other resource.

The standardized languages which are the basis of the Semantic Web form a layered stack, at the bottom of which lies the Resource Description Framework (RDF)

[16]. RDF is a simple assertional language that is designed to represent information in the form of triples, i.e., statements of the form: subject, predicate, object. RDF predicates may be thought of as attributes of resources and in this sense correspond to traditional attribute-value pairs. $p(s, o)$: resource s has resource o as value for attribute p. The arguments to RDF predicates must always be ground values except for the possibility of local existential variables to represent anonymous objects: $\mathit{colleague}(\mathit{Jim}, \_1)$, $\mathit{hometown}(\_1, \mathit{Amsterdam})$ states that Jim has some (otherwise unknown) colleague whose hometown is Amsterdam.

RDF however, contains no mechanisms for describing these predicates, nor does it support description of relationships between predicates and other resources. This is provided by the RDF vocabulary description language, RDF Schema (RDFS [6]). RDFS allows the specification of classes (generalized categories or unary relations) and properties (predicates or binary relations), which can be arranged in a generalization hierarchy: a hierarchy for classes, and a hierarchy for properties. In addition, it allows simple typing of such properties, by stating the classes to which subject and object of a particular property must belong. This allows simple inferencing of the following forms: inferring class membership and subclass relations through transitive inference in the subclass hierarchy, inferring class membership through occurrence in typed property-positions, and inferring property values and subproperty relations through transitive inference in the subproperty hierarchy.

From an AI perspective, RDFS is similar to some of our early frame systems in its representational capabilities. Notably, RDF and RDFS lack any notion of negation or disjunction and (as mentioned above) have only a very limited notion of existential quantification. Together this makes for a language with very limited expressive power. One illustration of this limited expressivity is the fact that (barring the use of XML datatypes), it is not possible to express inconsistencies in RDF. Also, it has turned out to be practical to perform exhaustive forward inferencing, i.e., to compute the entire deductive closure of an RDF graph. In fact, some of the most widely used RDF storage and query engines (e.g., Sesame [4]) work in this way. This is clearly only possible with a sufficiently weak language which does not in practice cause the exponential blow-up that deductive closures of richer languages suffer from.

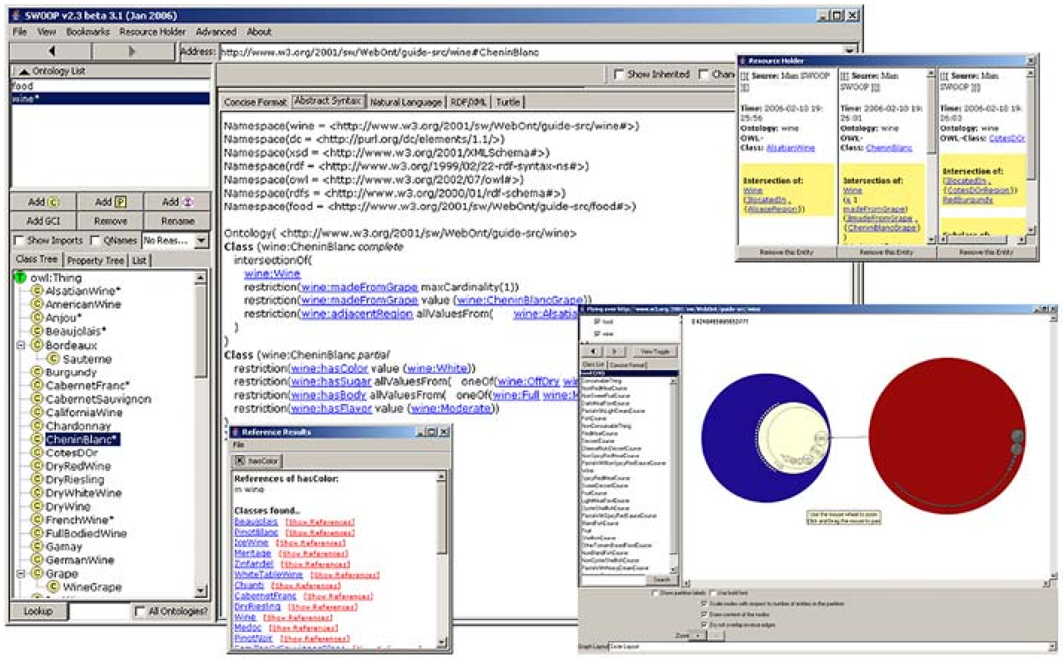
The Web Ontology Language (OWL) [7], released in February 2004 as a W3C recommendation, is a more expressive ontology language that is layered on top of RDF and RDFS. OWL can be used to define classes and properties as in RDFS, but in addition, it provides a rich set of constructs to create new class descriptions as logical combinations (intersections, unions, or complements) of other classes; define value and cardinality restrictions on properties (e.g., a restriction on a class to have only one value for a particular property) and so on. OWL's expressivity is sufficient to cover most of the well-known Description Logic formalisms, and some of its representational characteristics largely resemble those of DL. However, OWL is unique in two ways. First, it is the first reasonably expressive ontology language to become a standard recognized by a major standards body. This is very important for tool interoperability and ontology reuse, which we discuss below. In addition, OWL is the first widely-used ontology language whose design is based on the Web architecture, i.e., it is open (non-proprietary); it uses Universal Resource Identifiers (URIs) to unambiguously identify resources on the Web (similar to RDF and RDFS); it supports the linking of terms across ontologies making it possible to cross-reference and reuse information; and it has an XML syntax (RDF/XML) for easy data exchange.

OWL provides three increasingly expressive sub-languages: OWL Lite, OWL DL, and OWL Full, each with a different intended audience based on scope and complexity of the application domain. For example, the goal of OWL Lite is to provide a language that is viewed by tool builders to be easy enough and useful enough to support, thereby acting as an entry ontology language for semantic web application developers, whereas OWL Full provides more freedom in domain modeling at the cost of a higher learning curve. At the time of this writing, an effort is underway to define another sublanguage, sometimes referred to as RDFS+ and other times as OWL Very Lite, which is intended to be a much simpler version that provides only simple reasoning extensions to RDFS to allow for very efficient scalability. A second effort [5] has identified a number of subsets of OWL that have polynomial reasoning performance.

Within the KR community, the most used form of OWL is OWL DL, due to its support for automated reasoning. OWL DL has a formal model-theoretic semantics [18] providing a rigorous and provably decidable semantics for the language. As discussed in Chapter 3, DLs are a decidable subset of First Order Logic (FOL), being restricted to the 2-variable fragment of FOL. The decidability of the logic ensures that sound and complete DL reasoners can be built to check the consistency of an OWL DL ontology, i.e., verify whether there are any logical contradictions in the ontology axioms. Furthermore, reasoners can be used to derive inferences from the asserted information, e.g., infer whether a particular concept in an ontology is a subconcept of another, or whether a particular individual in an ontology belongs to a specific class. Popular existing DL reasoners in the OWL community include Pellet [21] and FaCT [13] which are available for free download and use, as well as several commercial products.

In addition to reasoners, numerous OWL ontology browsers/editors such as Protégé [17], SWOOP [14] and KAON [3] have been built to aid in the design and construction of OWL ontology models. Most of these OWL tools have expanded their functionality beyond basic editing to include features such as change management and query handling, and in a lot of cases included a reasoner for consistency checking of the ontology. For example, Protégé allows integration of any DIG-compliant reasoner and has plug-ins for collaborative ontology development, ontology change-management, ontology visualization, import and export to and from various representation formats. SWOOP provides the ability to automatically partition, collaboratively annotate and version control OWL ontologies. For example, Fig. 21.1 shows some of the features of the Swoop editor being used to browse an OWL ontology.

While tools such as these are familiar to many in the AI community, the need for wider deployment and ontology development by non-AI-experts (for example, subject matter experts in some domain), requires that these tools explore making the AI concepts available to others. Current efforts include using the "cultural metaphors" of the online culture, such as hypertext links, expandable menus, Web-browser-like look and feel, etc. to make these tools more comfortable to users who are familiar with the Web but not with AI.



```
Namespace(wine = <http://www.w3.org/2001/sw/WebOnt/guide-src/wine#>)
Namespace(dc = <http://purl.org/dc/elements/1.1/>)
Namespace(xsd = <http://www.w3.org/2001/XMLSchema#>)
Namespace(rdf = <http://www.w3.org/1999/02/22-rdf-syntax-ns#>)
Namespace(owl = <http://www.w3.org/2002/07/owl#>)
Namespace(rdfs = <http://www.w3.org/2000/01/rdf-schema#>)
Namespace(food = <http://www.w3.org/2001/sw/WebOnt/guide-src/food#>)

Ontology( <http://www.w3.org/2001/sw/WebOnt/guide-src/wine>
Class (wine:CheninBlanc complete
  intersectionOf(
    wine:Wine
    restriction(wine:madeFromGrape maxCardinality(1))
    restriction(wine:madeFromGrape value (wine:CheninBlancGrape))
    restriction(wine:adjacentRegion allValuesFrom(    wine:Alsati
  )
)
Class (wine:CheninBlanc partial
  restriction(wine:hasColor value (wine:White))
  restriction(wine:hasSugar allValuesFrom(  oneOf(wine:OffDry wi
  restriction(wine:hasBody allValuesFrom(  oneOf(wine:Full wine:
  restriction(wine:hasFlavor value (wine:Moderate))
)
```

```
References of hasColor:
in wine

Classes found..
Beaujolais  [Show References]
PinotBlanc  [Show References]
IceWine  [Show References]
Meritage  [Show References]
Zinfandel  [Show References]
WhiteTableWine  [Show References]
Chianti  [Show References]
CabernetFranc  [Show References]
DryRiesling  [Show References]
Wine  [Show References]
Medoc  [Show References]
PinotNoir  [Show References]
```

Figure 21.1: The standard syntax, and Web features, of OWL have led to the development of a number of new, Web-based, tools. SWOOP, shown in this figure, is an example of an ontology browser/editor developed for the Semantic Web.

21.3 Semantic Web KR Language Design

Despite these primary similarities to traditional AI work, there are some key differences in the design of OWL, and of current efforts to build new languages adding features missing from OWL, from traditional AI work. These differences are in many ways similar to the ways in which the World Wide Web was different from traditional Hypertext systems, and thus the term Semantic Web is most correctly applied to systems which focus on these features. In the remainder of this article we describe some of these differences, focusing on

- The importance of standards based on the Web infrastructure.
- The "Webization" of ontology language.
- The emphasis on scalability.

We will then discuss some of the emerging trends on the Semantic Web including work on bringing rule languages and FOL to the Semantic Web. We conclude by discussing some of the challenges to traditional AI reasoning that we will need to overcome if we are going to "unleash KR's full power".

21.3.1 Web Infrastructure

There are two reasons why the decision to build OWL and other Semantic Web languages based on Web standards, as opposed to other attempts to standardize knowledge exchange [9, 15], are so critical to the uptake of this technology. One is the importance of being able to exploit the Web infrastructure for wider deployment, which we discuss in this section, and one is more important to the technical underpinnings of KR, the grounding of assertions and definitions dereferencably.

The building of the Semantic Web on top of the Web infrastructure is largely motivated by a lesson learned from the efforts to more widely disseminate expert systems technology in the mid-1980's. Often seen as a failure, despite wide use today of rule-based technologies, one of the reasons expert systems had trouble with uptake is that, especially in the early days, they did not "play nice" with the rest of a user or organization's computer infrastructure. The need for special languages and machines, and the difficulty in embedding rule-bases into existing code was a major impediment. For a web of KR to succeed, it must be deployable via existing infrastructure, and the Web infrastructure is the mostly widely deployed and used in the world today.

Underlying the web is the Hypertext Transfer Protocol [8], the ubiquitous HTTP typed into web pages. For the facts and axioms of the Semantic Web to be sharable on the Web infrastructure, it is clearly crucial that they must be encoded in an HTTP friendly way, mandating use of HTML, XML, RDF or some other widely used web format (MIME type) for exchange. The Semantic Web is built largely on RDF, for reasons discussed in the next section. However, Web embedding is more than using these languages: simply HTTP-GETting a document to display in the browser is not akin to putting KR on the Web.

Most Web applications today use a three-tiered architecture in their client–server communication. The client sends the HTTP-GET request to the server which is to return a document in HTML or other specified MIME type (XML is becoming much more prevalent, and many applications are switching to that today). The server, rather than just serving up the document as would be done for static HTML, generates the document by using a database or other backend which keeps the base information in whatever proprietary form the provider uses. The "middle tier" issues queries (or similar) to find the relevant information and transform it into the requested document format, and this is in turn returned to the client as the response to the GET request.

For a KR infrastructure to live on the Web, it is important that it can be integrated into such applications. The Semantic Web infrastructure was designed with this in mind. While proprietary knowledge bases and knowledge base languages could underlie the applications, without standards for the exchange formats, what is requested by one cannot be generated by another. So a major aspect of the Semantic Web languages is simply this: that it can coexist with other web applications, be made to work through server modifications, and to integrate well into current and future Web based architectures. This is also an important economic incentive to wider adoption of Semantic Web KR by industry, the deployed infrastructure for information exchange, web servers and clients, does not require replacement to get any reasoning benefits the Semantic Web can offer, and many end-users will see the benefit of the Semantic Web solely as new functionality delivered to them through their Web browser.

21.3.2 Webizing KR

With the advent of the Web, the neologism "webize" has come into being to refer to bringing new resources to the Web in a way that allows them to be integrated into the existing infrastructure, as described above, but also to be linked to one another to achieve the network effect that makes the Web succeed. In the words of Tim Berners-Lee, the inventor of the Web

> The essential process in webizing is to take a system which is designed as a closed world, and then ask what happens when it is considered as part of an open world. Practically, this effect on a computer language is to replace the names/tokens/identifiers for URIs. Thus, where before reference could only be made to something in the same document/program/module one can with equal ease make reference to something in a different one somewhere in that abstract space which is the Web. (Berners-Lee, 2001)

In essence, we make something a first-class citizen of the web by assigning it its own Universal Resource Identifier (URI). This is an identifier obeying a set of rules of web access, but essentially is equivalent to providing a pointer into a near infinite name space. Thus, the URI http://www.w3.org/2003/08/owlfaq.html is the identifier for the W3C OWL FAQ, as a pointer into Web space.

There are a number of important aspects of URIs with respect to making the Web work: the definition of URI scheme, the convention that the first element is the server at which the resource named can be found, etc. From the KR point of view, however, there is another feature that is very crucial: any resource on the Web can be given an identifier, and any Web server pointed at that name will retrieve the same underlying representation of the resource. Using RDF as the basis of Semantic Web KR ensures that any term defined in a Web-based ontology is given a globally recognized identifier.

In a traditional KR system we can generally assert a new class or predicate or formula, and within the KB it resides the name is unique. However, if we want to refer to it from outside that KB, there is generally not a way to do so. So if we want to say "the concept **Student** which is used by **Jim Hendler's Web Page**" or "the concept **person** as defined in **CYC**" there needs to be an identifier for the concept, and traditional KR has not provided an externally addressable referent.

On the Web, URIs provide this function, and RDF was designed precisely to take advantage of this. The URI

http://www.cs.umd.edu/users/hendler/onts/Research.owl#student

is an identifier that cannot be used for other definition but the student concept. This URI thus provides a label that could be used anywhere in "web space" and remains unambiguous—two different KBs that each refer to this term must be referring to the same thing by definition.

The ability to have global names is a very powerful concept in and of its own right, and there are a number of philosophical issues in KR to be discussed in this respect (a couple of which we touch on later in this chapter). The most important, however, is the notion of *dereferencability*. On the Web, a URI can be used not only to name a document, but as a reference to a document—in your browser when you click on an HTTP URI, a document is fetched and typically a presentation is displayed by your browser.

RDFS and OWL are defined so that the concepts created in the ontology definition documents are assigned URIs that dereference to a representation of the document that defined them. So, for example, the student URI defined above not only gives a precise name to the student concept, but also if an HTTP-GET is performed on the URI, an OWL document containing the definition will be returned to the client performing the GET. Thus, while simply by examining the name there is no way to tell whether the definition of "student" is a class name, a predicate, or an individual, by retrieving the document and parsing it into RDF, an assertion will be found that answers this question. Thus, we would find a piece of OWL that entails that

```
http://www.cs.umd.edu/users/hendler/onts/Research.owl#student
rdf:type owl:class
```

(RDF statements are comprised of triples read as object predicate object, thus this says that the binary relation "rdf:type" holds between the subject URI and the URI "Owl:class"—for details on the RDF representation as both triples and as XML documents, see [7].)

Typically, when dealing with OWL as a KR language we use an XML rendering, or other presentation syntax, to remove the details of the RDF triples and provide a level of abstraction. For example, the triple above could be rendered in the N3 presentation syntax as

```
@prefix: "http://www.cs.umd.edu/users/hendler/onts/Research.owl".
: student a owl:class.
```
[1]

Other, more complex relations can also be similarly shortened, for example, the OWL specification [7] also contains an "abstract syntax" so that a statement such as

```
Ontology(<http://www.cs.umd.edu/users/hendler/onts/Research.owl>
Class (Research:CS_Course partial
restriction(Research:offeredIn someValuesFrom(Research:CS_Department))
Research:Course))
```

which states that the concept defined at the URI
http://www.cs.umd.edu/users/hendler/onts/Research.owl#CS_Course
is a subclass of those things which are in the intersection of Courses and those things which are existentially quantified as being offered in a CS_Department (and further that CS_Department, CS_Course, Course and offeredIn are also defined). This statement would be rendered in XML as the much less readable (but nicely Web compatible)

```
<owl:Class rdf:about="#CS_Course">
   <rdfs:subClassOf>
      <owl:Class rdf:about="#Course"/>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
      <owl:Restriction>
         <owl:onProperty>
            <owl:ObjectProperty rdf:about="#offeredIn"/>
         </owl:onProperty>
         <owl:someValuesFrom>
            <owl:Class rdf:about="#CS_Department"/>
         </owl:someValuesFrom>
      </owl:Restriction>
```

[1] rdf: and owl: are common abbreviations for actual deferencable URIs that link to the standards documents that define RDF and OWL, respectively.

```
    </rdfs:subClassOf>
  </owl:Class>
```

which in turn would “compile” into a large number of RDF triples, forming a labeled, directed graph of URIs, that could be stored in an RDF datastore and used by RDF, RDFS and or OWL tools.

Regardless, however, of the representation syntax used, the semantics of the expression are defined in the OWL Model Theory [18] and the URIs for the classes, individuals, and predicates defined in such expressions are uniquely and globally assigned to allow Semantic Web systems to use each others' data and domain descriptions in a clean, Web-accessible, and distributed (open) manner.

There are other interoperability advantages which we will not go into here. For example, the is an emerging standard for an RDF-based query language (SPARQL [19]) which can be used for querying data (or Abox) assertions that have been defined against RDFS and OWL ontologies. For more about the Semantic Web from the perspective of interoperability, see the W3C Semantic Web Activity Web page (http://www.w3.org/2001/sw).

21.3.3 Scalability and the Semantic Web

There are two aspects of scalability that apply to the design of Semantic Web KR systems, one is the scalability of the underlying reasoning itself, as is a concern in most KR work, and the other is a scalability in the sense that the Web is scalable, the creation of an open and distributed KR world. This latter puts some constraints on the requirements for OWL, and any successor languages, and is a point where Web and AI research come very much into contact (and sometimes conflict).

The first kind of scalability on the Web is the one that is usually discussed, the fact that the Web itself is massive, with hundreds of billions of documents of many different kinds, some open and accessible, some of limited access (sometimes called the deep web). In addition, with one of the goals of the Semantic Web being to bring significantly more data resources to the Web, and to make these more accessible via linking to ontological knowledge, the scale of the emerging Semantic Web Knowledge Bases dwarves just about anything tried in AI before now.

For one example, a number of people in the “Health Care and Life Sciences Interest Group”[2] are working to develop ontologies in biological areas and to link these to sets of data coming out of various datasources to better integrate these data sources. As one example, the Uniprot (Universal Protein Resource) Web site offers access to protein sequencing data being produced at a number of sites. Knowledge Bases, containing hundreds of millions of triples have been developed, and these are being tied to OWL ontologies that range in size from tens to thousands of classes. Scaling AI reasoning techniques, even when using the decidable fragment of OWL, to these sorts of scales is a major engineering challenge for Semantic Web researchers.[3] At the time of writing, the most scalable RDFS system can handle up to a few tens of billions of RDF triples

[2]http://www.w3.org/2001/sw/hcls/.

[3]It is worth noting that a number of large OWL ontologies also exist without instances, for example, the National Cancer Institute maintains a metathesaurus that is released in OWL. At the time of this writing it has over 50,000 class definitions [10].

while still complying with the standard semantics. For the various OWL variants, these numbers are significantly smaller. In particular reasoning with very large numbers of instances has traditionally imposed significant problems for DL reasoners.

The other scaling challenge to Semantic Web KR, and one of the key challenges in the design of OWL, was designing a language that would fit into the overall Web architecture and its constraints. Consider again the example of the introduction, where we consider linked knowledge documents as analogous to linked Web pages. It may seem like it is simple to talk about the contents of a Web page, but in reality it is extremely difficult (and still not well defined). Is the content just what is returned by a single HTTP-Get (i.e., the "page" you see in your browser), is it all possible renderings of that page in different MIME types, is it the page plus all the documents it is linked to directly, or all the pages on the same site? In fact, if we consider the "transitive closure" of the link space, then the content associated with any particular page is, in the worst case, the entirety of World Wide Web!

In creating a knowledge representation for the Web, it was important to keep in mind that it too would include documents (cf. ontologies), linked to datasets (cf. RDF triple stores), linked to possibly other documents, other datasets, and to regular web resources (for example, it is useful to say that the person described in the Web page http://www.cs.umd.edu/~hendler is named "Jim Hendler"—combining an HTML referent and a KB referent in a smooth way). If one assumed that such knowledge sources were being created dynamically, for example, via a web crawl or dynamic mapping from multiple databases to a triple store, then it appeared that full knowledge of all the assertions associated with a fact on the Web, would essentially map to the problem of finding all the content linked to a particular web page—in the worst case, the entire Semantic Web.

Given this notion of Web KR being amenable to applications like crawlers, which might at any point in time have an "incomplete" view of the word, the design of Semantic Web languages has favored the open-world semantics of FOL to the closed-world semantics of databases. In addition, assuming an incremental addition of information (i.e., a crawler accreting knowledge over time) a monotonic logic ended up being favored. For example, in the design of the OWL language, an original objective was to have the language include default reasoning, motivated by many AI applications, but no non-monotonic solution amenable to Web architecture concerns was found. This debate continues at the time of this writing in the design of a rules language for the Semantic Web (cf. [12]) where the need for negation as failure is recognized as important to many applications, but no mechanism for closing the world of discourse, compatible with open and distributed Web principles, has been developed.

21.4 OWL—Defining a Semantic Web KR Language

The best example of a current KR language for the Web, which meets the requirements above but still meets the needs of many KR projects is the Web Ontology Language OWL, which became a World Wide Web Consortium recommendation in February of 2004. OWL was designed to be expressive enough for many practical problems, simple enough for "real users" to get a start without taking an AI course, and designed to meet the needs of companies interested in deploying AI-related applications on the

Web. As we will discuss, OWL comes in three "dialects" known as OWL Lite, OWL DL, and OWL Full. As we shall see, the rationale behind having these three dialects helps explains how OWL supports the different needs of different user communities.

To start with, OWL is designed as an extension to the Resource Description Framework (RDF) and RDF Schema (RDFS) language developed by earlier standards efforts. RDF and RDFS provide several useful KR concepts, roughly corresponding to the ISA-hierarchy and simple slot definitions of early frame-based KR languages. One difference, however, is that RDF is designed such that several important aspects of the Web are built in. RDF provides a mechanism for assigning URIs to class names (thus giving them the global addressing property described previously), it provides a mechanism for the internationalization of terms (based on Unicodes), and it provides a mechanism for accessing the "XML Schema Datatypes" that are used on the Web for providing standard definitions for common datatypes such as strings, integers, dates, etc. In short, by building OWL on RDF the needs for web embedding are largely met.

However, embedding on RDF was something of a struggle for designing a Web KR language. Traditional KR languages have not provided mechanisms for external references, externally defined datatypes ad the like. In addition, some common features of KR languages (cf. variables in arguments, closed lists, and mechanisms for asserting equivalence of terms) were not provided in the original RDF. The working group thus had to provide a design that either avoided the problems, created solutions at the OWL level, or required working with those updating the RDF standard to provide common solutions.

In addition, the designers of OWL inherited some constraints from the Web domain. Some of the designers felt that a Web ontology language should be monotonic and without defaults, given that new information is often discovered on the Web and reinferencing in the presence of new information. Some designers felt that it was crucial the language be decidable, others argued that there should be a well-designed decidable fragment of the language. In addition, although RDF allowed properties on classes, these were always universal, and the designers of OWL felt it was crucial on the Web to be able to have class descriptions that could be restricted to only some subset of a class as this would mean that if definitions from multiple documents were merged, there would be means to separate aspects of the class definitions. Note that from these definitions it becomes clear why OWL resembles a Description Logic language—DLs largely meet these KR design goals.

However, there were also KR design goals of OWL that could not be met within standard approached to DL, but which use cases mandated for OWL necessitated. A good example of this is inverse functional datatypes. One of the features of OWL that is very important in many Web applications is the ability to designate two individuals or classes to be equivalent. OWL provides mechanisms for directly asserting this, but very important in many cases was the use of an "Inverse Functional Property" definition, which allows an OWL ontology to designate some property as being unique to individuals. That is,

$$\text{P1 an inverseFunctionalProperty} \Leftrightarrow \text{P1(x,y)} \wedge \text{P1(z,y)} \Rightarrow \text{equivalent(x,z)}.$$

(An example of the use of this feature is in FOAF, where we can designate that individuals with the same "foaf:mbox" property (i.e., same email address) should be merged into the same node in the FOAF networks.)

However, there was a problem. If, using standard DL semantics, one designated a datatypeProperty to be inverse functional, then the system becomes undecidable. On the one hand, decidability is a desirable feature, on the other hand, many use cases of OWL require that datatypes be inverse functional (in particular, database keys mapped to RDF require inverse functional datatype properties). No single design could satisfy both goals. This was one of the reasons for defining both a general form of OWL (OWL Full) which does not restrict datatypes in this way, and OWL DL, a decidable profile of OWL which does. Several other features of the use of OWL also have similar dichotomies, including using classes as instances (i.e., metamodeling) and having classes that have not been typed (Datatype, objectType, annotationType, or ontologyProperty) in the defining document. In all these cases, the semantics of OWL DL could be kept clean and decidable and the semantics of OWL Full included some features that allow undecidability, necessary for some of the webized applications.

In addition, in designing OWL one option was to choose the language to include the largest decidable subset of FOL known (i.e., the most expressive DLs known) while another option was to include only options whose value was proven important and useful. OWL Lite is a subset of OWL DL that removes some features that were felt to be outside this class, or that might be confusing to novice users.

Other challenges in the design of OWL required providing the semantics for some of the Web features of the language that were not included in many standard KR languages. For example, since OWL defines the URI mechanism, it is easy for an OWL document to refer to terms in other ontology documents. Thus, the vocabulary of OWL that can be used within a single document can be used to express relations between classes in an ontology and those defined in other ontologies. This could include simple assertions, perhaps stating that what some European document refers to as :footballTeam is different from what a US document means by the same term, or that it is equivalent to what some US document calls a :soccerTeam.

The links between documents can, however, also be more complex than this. For example, supposing we would like to say that our pet cat is a short-tailed Abyssinian cat and that we would like to use the properties of Cat that are already in CYC, but perhaps Cyc does not have all the features we need (for example, tail-length or the AbyssinianCat Class). In OWL we can simply extend the classes from CYC by creating a document that defines the CYC: namespace as pointing to CYC's URIs, and asserting, for example:

```
:AbyssinianCat a cyc:petCat.
:tailLength a owl:datatypeProperty;
   range cyc:Cat; {note that in CYC a petCat would be asserted to be a Cat}
   domain xsd:string.
:myCat a :AbyssinianCat;
   :tailLength "short".[4]
```

However, if a reasoner sees this document, what semantics should it adopt? Should the terms from CYC be expected to include all the semantics of the CYC ontology, should there be no "official" semantics for this, letting external links be defined by

[4]We make length a string for simplicity of this example, we leave defining and enumerated class of appropriate lengths as an exercise to the reader.

some sort of extra-logical mechanism, or could some other solution be devised and standardized at some later time. OWL provides mechanisms for declaring that one ontology "imports" another, and therefore all the semantics should be observed, and a mechanism by which this can be defined as an annotation property (or, in OWL Full, left unspecified) essentially asserting that no semantics should be assigned to the class *a priori* but rather that systems might use externally defined mechanisms to meet user expectations.

A recurring issue in the design of web-based KR languages is the choice between open world semantics and closed world semantics. A closed world semantics typically allows the derivation of conclusions from the absence of conclusions to the contrary. In programming languages such as Prolog, this is known as Negation by Failure, and is closely related to default reasoning. Although in general the Web would seem more suited to open-world reasoning (and indeed both RDFS and OWL adopt an open-world semantics) there are many use-cases where a closed-world semantics is appropriate: students in a class, customers of a company, cities in a country are all examples of closed sets: if a student is not listed as enrolled, we can safely assume she is not enrolled. Although useful in many cases, there is currently no practical mechanism in RDFS or OWL to state that a given set of individuals (or facts) is "closed".

A related, although different, issue is the unique name assumption. Typically, database systems assume a single, unique name for each individual. If we encounter two individuals with different names, we can safely assume they are indeed different individuals. Again, on the web this assumption would be too strong. In a world as large as the web, many individuals are known under multiple names ("Jim Hendler", "James Hendler", "Prof. J. Hendler", "the author of Chapter 21", etc.). When encountering two such different names, we should safely assume that they may or may not designate the same individual, until further reasoning decides the issue one way or the other. OWL contains a simple device to state that all individuals in an enumerated set are known to be different (i.e., that they are not just different names for some of the same individuals), but this language construct (`owl :allDifferent`) requires the explicit enumeration of these names, which can be either impractical, or even impossible in principle.

Traditionally, systems such as databases and logic programming systems have tended to support closed-worlds and unique names, while knowledge representation systems and theorem provers support open-worlds and non-unique names. Ontologies are sometimes in need of one, and sometimes in need of the other. This conundrum was nicely resolved in [11], which identified a fragment of OWL baptized DLP, for Description Logic Programming: this fragment is the largest fragment on which the choice for CWA and UNA does not matter as depicted in Fig. 21.2. That is to say, OWL DLP is weak enough so that the differences between the choices do not show up. The advantage of this is that people or applications that wish to make different choices on these assumptions can still exchange ontologies in OWL DLP without harm. Of course, as soon as they go outside OWL DLP, they will notice that they draw different conclusions from the same statements. In other words, they will notice that they disagree on the semantics.

Fortunately, DLP is still large enough that it can be used for useful representation and reasoning tasks. It allows the use of such OWL constructors as class and property equivalence, equality and inequality between individuals, inverse, transitive, symmet-

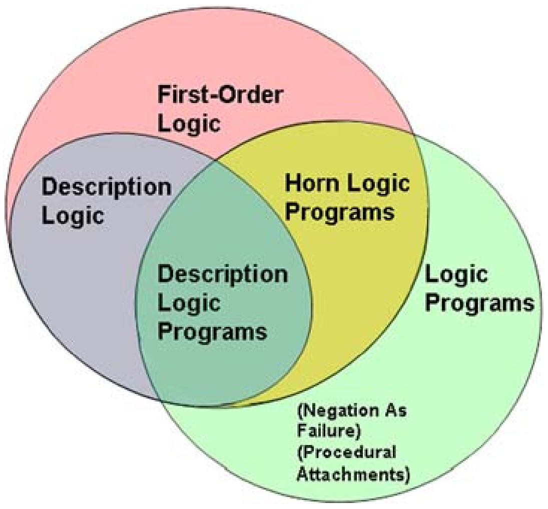


Figure 21.2: Relation of OWL-DLP to other KR languages.

ric and functional properties, and the intersection of classes. It excludes however constructors such as intersection and arbitrary cardinality-constraints. These constructors do not only allow useful expressivity for many practical cases, while guaranteeing correct interchange between OWL reasoners independent of their CWA and UNA, they also allow for a translation into efficiently implementable reasoning techniques based on databases and logic programs.

As is already clear from the above two points, RDFS and OWL do not allow any form of default reasoning, even though many years of KR applications have shown this to be a very useful device for dealing with incomplete knowledge. This would be particularly important in a world as large as the Web, where not all properties of all objects will be explicitly known, but must often be inferred by default until shown otherwise. However, a lack of concensus in the KR community on how to best formalize defaults has prevented such features from being included in the Semantic Web standardized representation languages.

Finally, a point often raised is that the large and open world of the Web will almost certainly need some forms of uncertainty and fuzziness. Again, lack of concensus has prevented such language features from being included, although it would seem clear that they will ultimately be needed in some form or other, either in the representation or in the inference mechanisms.

As time progresses, new work also continues which pushes OWL in different directions. In practical use on the Web, OWL has needed to be scaled to problems that have been much larger than those previously attempted in KR research. Very large Tboxes (thousands of class definitions) coupled with extremely larges Aboxes (millions of individuals) turned out to be relatively easy to construct, and necessary for Web uses. To this end, as we mentioned earlier, several groups are exploring tractable subsets of OWL, some of which are very close to RDFS others of which attempt to provide more functionality while remaining polynomial (cf. [20] which describes a number of these). On the other hand, some usages are exploring more expressive features that were not included in OWL including qualified restrictions, limited forms of non-monotonicity, integration with rules, mereological constructs, and others.

21.5 Semantic Web KR Challenges

Perhaps the most interesting thing about OWL, and future Semantic Web KR languages, is not what was solved in the OWL design, but what was left unsolved. For example, above we described what happens when one document imports another. However, on the Web the typical mechanism for implementing this inclusion is an HTTP-Get of the imported document. What happens if the server where that document lives is down, or if the owner of the document makes changes that remove the class I am referring to (or even worse, makes a change that subtly changes the semantics)? Or what happens if the document we link to links, in turn, to some other document which has a different semantic interpretation of some shared terms we use? For example, we may have said cyc:cat is a cyc:mammal, someone else that it is a cyc:insect, as they prefer the cat class for referring to caterpillars, and cyc: contains an assertion that mammals and insects are disjoint classes. At this point, all my instances of cats become inconsistent, a real problem especially when trying to use some typical logical reasoner that uses some form of reasoning by negation, in which case it would follow that any unasserted fact was true. Definitely not the desired behavior!

Most KR systems have been designed in the past to assume that inconsistency is a problem, and to define mechanisms to rule it out (either by limiting expressivity or defining inconsistency as an error condition) or which provide some mechanism (like a belief revision mechanism) that triggers from knowing the sources of the inconsistency. Semantic Web KR appears to mandate either some form of local consistency or the development of paraconsistent or other, some argue higher order, logics that disallow the general proof of all concepts from an inconsistency.[5]

In addition to attempts to explore semantics that handle some of the Web problems in KR, there are also attempts to explore the provision of capabilities that OWL disallows, such as providing mechanisms for scoping RDF graphs to allow default reasoning and negation as failure or to provide unifying logics in which other Web KR languages can be expressed, providing semantic interoperability without an insistence (as in the case of OWL) on syntactic uniformity.

21.6 Beyond OWL

The continued use of Semantic Web KR, beyond the OWL language, requires the design of other reasoning frameworks in ways that provide the same opportunities for interoperability standards and linking that OWL provides for basic KR vocabularies. A number of efforts have looked, for example, at bringing the power of rules to the Web for providing the linking of properties that OWL does not provide. A number of these efforts came together in the RuleML effort [12] as well as the development of Web specific rule languages like N3 [1], aimed specifically at providing support for RDF-based ontologies. At the time this chapter is being written, the World Wide Web Consortium has created the Rules Interchange Format (RIF) Working Group to explore the standardization of rules for the Web and to formalize a mapping between OWL and this emerging rules language.

[5] The interested reader is also directed to SCL [20] an attempt to provide a unifying logic for the Web which allows some higher-order-like reasoning within the constricts of FOL.

Other efforts are exploring other kinds of KR on the Web. Probabilistic extensions to OWL are also a current area of interest and several efforts are underway to extend OWL to provide richer expressivity ranging from extensions that aim to maintain the OWL DL guarantees to new, post OWL languages that extend the logic in many of the ways found in other chapters in this book.

A key point to note about OWL is that it was *not* intended to be the be-all and end-all knowledge representation for the Web. Like any good standard, it was designed to be a consensus language that could be used by a wide variety of users, sacrificing some of the advanced expressivity features that were not yet ready for standardization or for which there did not yet seem to be use cases compelling to the non-researchers involved in the standardization process. It is a truism in the standards community that good standards evolve, and the many activities looking to extend OWL in various directions are a healthy sign that OWL adoption is taking place.

21.7 Conclusion

At the time of this writing, subsets of the OWL language are being supported by major database vendors and RDF and RDFS are seeing wide use in both corporate and wider Web applications under the name "Web 3.0". In the academic arena, significant investment from the US and EU governments have helped to create a large community exploring many aspects of the use of Semantic Web technologies. New efforts in the standardization community are exploring adding rules to the Semantic Web, the use of semantic web ontologies in health care and life sciences, approaches to embedding Semantic annotations in traditional Web pages, adding probability to OWL, and others (See in particular the World Wide Web Consortium's Semantic Web Activity.[6]) OWL has become the most used KR language in the history of the field, not because of its particular representational power, but rather because it was designed to be a common syntax usable by many KR systems, to be webized for easier sharing of ontologies and concepts, and to be expressive enough for many problems without totally sacrificing scalability.

Acknowledgements

Funding for Dr. Hendler's research at the University of Maryland and RPI has been provided by: Fujitsu Laboratories of America—College Park, Lockheed Martin Advanced Technology Laboratory, NTT Corp., Kevric Corp., SAIC, National Science Foundation, DARPA, National Geospatial Intelligence Agency, Northrop Grumman Electronic Systems, US Army Research Laboratory, NIST, Other DoD sources and the Rensselaet Polytechnic Institute endowment. Special thanks to Guus Schreiber for comments on this article, to Vladimir Kolovski and Chris Halaschek-Weiner for their help in its preparation, and to the MINDSWAP project at Maryland where much of the foundations Semantic Web work took place.

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

Bibliography

[1] T. Berners-Lee, D. Connolly, L. Kagal, Y. Scharf, and J. Hendler. N3Logic: A logic for the web. *Theory and Practice of Logic Programming*, 2007 (Special Issue on Logic Programming and the Web).

[2] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic Web. *Scientific American*, 284(5):34–43, May 2001.

[3] E. Bozsak, M. Ehrig, and S. Handschub. Kaon—towards a large scale semantic web, 2002.

[4] J. Broekstra, A. Kampman, and F. van Harmelen. Sesame: A generic architecture for storing and querying rdf and rdf schema. In I. Horrocks and J. Hendler, editors. *Proceedings of the First International Semantic Web Conference*, *Lecture Notes in Computer Science*, vol. 2342, pages 54–68. Springer-Verlag, July 2002.

[5] B. Cuenca-Grau. Owl 1.1 web ontology language tractable fragments (editor's draft of 6 April 2007). Available at: http://webont.com/owl/1.1/tractable.html, 2007.

[6] D. Brickley and R. Guha. RDF vocabulary description language 1.0: RDF schema. http://www.w3.org/tr/rdf-schema/, February 2004.

[7] M. Dean and G.E. Schreiber. Owl web ontology language reference, w3c recommendation. Available at http://www.w3.org/TR/owl-ref/, 2004.

[8] J. Fielding, J. Gettys, H. Mogul, H. Frystyk, and T. Berners-Lee. Hypertext transfer protocol—http/1.1. IETF Network Working Group Request for Comments: 2068, 1997.

[9] M.R. Genesereth and R.E. Fikes. Knowledge interchange format, version 3.0 reference manual. Available at: http://www-ksl.stanford.edu/knowledge-sharing/kif/.

[10] J. Golbeck, G. Fragoso, F. Hartel, J. Hendler, B. Parsia, and J. Oberthaler. The national cancer institute's thesaurus and ontology, 2003.

[11] B.N. Grosof, I. Horrocks, R. Volz, and S. Decker. Description logic programs: combining logic programs with description logic. In *Proceedings of the World Wide Web Conference*, pages 48–57, 2003.

[12] D. Hirtle, H. Boley, B. Grosof, M. Kifer, M. Sintek, S. Tabet, and G. Wagner. Schema specification of RuleML 0.91. Available at: http://www.ruleml.org/0.91/, 2006.

[13] I. Horrocks. The fact system. In *TABLEAUX '98: Proceedings of the International Conference on Automated Reasoning with Analytic Tableaux and Related Methods*, pages 307–312. Springer-Verlag, London, UK, 1998.

[14] A. Kalyanpur, B. Parsia, E. Sirin, B.C. Grau, and J. Hendler. Swoop: A web ontology editing browser. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, 4(2), June 2006.

[15] P.D. Karp, K.L. Myers, and T.R. Gruber. The generic frame protocol. In *IJCAI (1)*, pages 768–774, 1995.

[16] O. Lassila and R. Swick. Resource description framework (rdf) model and syntax specification.

[17] M. Musen, J.H. Gennari, H. Eriksson, and A.R. Puerta. Computer support for development of intelligent systems from libraries of components. *Medinfo*, 8(1):766, 1995.

[18] P. Patel-Schneider, P. Hayes, and I. Horrocks. OWL web ontology language semantics and abstract syntax w3c recommendation. http://www.w3.org/tr/2004/rec-owl-semantics-20040210/, February 2004.

[19] E. Prud'hommeaux and A. Seaborne. SPARQL query language for rdf, w3c working draft. Available at: http://www.w3.org/TR/rdf-sparql-query/, 2007.

[20] E. Prud'hommeaux and A. Seaborne. SPARQL query language for rdf, w3c working draft. Available at: http://www.w3.org/TR/rdf-sparql-query/, March 2007.

[21] E. Sirin, B. Parsia, B.C. Grau, A. Kalyanpur, and Y. Katz. Pellet: A practical owl-dl reasoner. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, 2006.