

DOI:10.1145/1409360.1409377

**How ontologies provide the semantics, as explained here with the help of Harry Potter and his owl Hedwig.**

BY IAN HORROCKS

## Ontologies and the Semantic Web

While phenomenally successful in terms of amount of accessible content and number of users, today's Web is a relatively simple artifact. Web content consists mainly of distributed hypertext and hypermedia, accessible via keyword-based search and link navigation. Simplicity is one of the Web's great strengths and an important factor in its popularity and growth; even naive users quickly learn to use it and even create their own content.

However, the explosion in both the range and quantity of Web content also highlights serious shortcomings in the hypertext paradigm. The required content becomes increasingly difficult to locate via search and browse; for example, finding information about people with common names (or famous namesakes) can be frustrating. Answering more complex queries, along with more general information retrieval, integration, sharing, and processing, can be difficult or even impossible; for example, retrieving a list of the names of E.U. heads of state is apparently

beyond the capabilities of all existing Web query engines, in spite of the fact that the relevant information is readily available on the Web. Such a task typically requires the integration of information from multiple sources; for example, a list of E.U. member states can be found at [europa.eu](http://europa.eu), and a list of heads of state by country can be found at [rulers.org](http://rulers.org).

Specific integration problems are often solved through some kind of software "glue" that combines information and services from multiple sources. For example, in a so-called mashup, location information from one source might be combined with map information from another source to show the location of and provide directions to points of interest (such as hotels and restaurants). Another approach, seen increasingly in so-called Web 2.0 applications, is to harness the power of user communities in order to share and annotate information; examples include image- and video-sharing sites (such as Flickr and YouTube) and auction sites (such as eBay). In them, annotations usually take the form of simple tags (such as "beach," "birthday," "family," and "friends"). However, the meaning of tags is typically not well defined and may be impenetrable even to human users; examples (from Flickr) include "sasquatchmusicfestival," "celebrity-lookalikes," and "twab08."

Despite their usefulness, these approaches do not solve the general problem of how to locate and integrate information without human intervention. This is the aim of the semantic Web<sup>3</sup> according to the World Wide Web Consortium (W3C) Semantic Web FAQ; the goal is to "allow data to be shared effectively by wider communities, and to be processed automatically by tools as well as manually." The prototypical example of a semantic Web application is an automated travel agent that, given constraints and preferences, gives the user suitable travel or vacation suggestions. A key feature of such a "software agent" is that it would not simply exploit a predetermined set of informa-

tion sources but search the Web for relevant information in much the same way a human user might when planning a vacation.

A major difficulty in realizing this goal is that most Web content is primarily intended for presentation to and consumption by human users; HTML markup is primarily concerned with layout, size, color, and other presentation issues. Moreover, Web pages increasingly use images, often with active links, to present information; even when content is annotated, the annotations typically take the form of natural-language strings and tags. Human users are (usually) able to interpret the significance of such features and thus understand the information being presented, a task that may not be so easy for software agents.

This vision of a semantic Web is extremely ambitious and would require solving many long-standing research problems in knowledge representation and reasoning, databases, computational linguistics, computer vision, and agent systems. One such problem is the trade-off between conflicting requirements for expressive power in the language used for semantic annotations and the scalability of the systems used to process them<sup>7</sup>; another is that integrating different ontologies may prove to be at least as difficult as integrating the resources they describe.<sup>18</sup> Emerging problems include how to create suitable annotations and ontologies and how to deal with the variable quality of Web content.

Notwithstanding such problems, considerable progress is being made in the infrastructure needed to support the semantic Web, particularly in the development of languages and tools for content annotation and the design and deployment of ontologies. My aim here is to show here that even if a full realization of the semantic Web is still a long way off, semantic Web technologies already have an important influence on the development of information technology.

### Semantic Annotation

The difficulty of sharing and processing Web content, or resources, derives in part from the fact that much of it (such as text, images, and video) is unstructured; for example, a Web page

might include the following unstructured text:

Harry Potter has a pet named Hedwig.

As it stands, it would be difficult or impossible for a software agent (such as a search engine) to recognize the fact that this resource describes a young wizard and his pet owl. We might try to make it easier for agents to process Web content by adding annotation tags (such as `Wizard` and `Snowy Owl`). However, such tags are of only limited value. First, the problem of understanding the terms used in the text is simply transformed into the problem

providing definitive information about owls. RDF is a language that provides a flexible mechanism for describing Web resources and the relationships among them.<sup>14</sup> A key feature of RDF is its use of internationalized resource identifiers (IRIs)—a generalization of uniform resource locators (URLs)—to refer to resources. Using IRIs facilitates information integration by allowing RDF to directly reference non-local resources. IRIs are typically long strings (such as `hogwarts.net/HarryPotter`), though abbreviation mechanisms are available; here, I usually omit the prefix and just write `HarryPotter`.

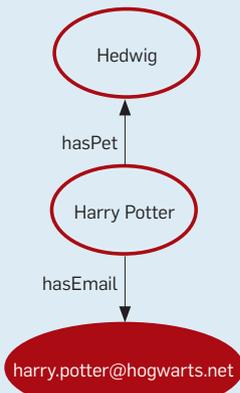


of understanding the terms in the tags; for example, a query for information about raptors may not retrieve the text, even though owls are raptors. Moreover, the relationship between Harry Potter and Hedwig is not captured in these annotations, so a query asking for wizards having pet owls might not retrieve Harry Potter.

We might also want to integrate information from multiple sources; for example, rather than coin our own term for Snowy Owl, we might want to point to the relevant term in a resource

RDF is a simple language; its underlying data structure is a labeled directed graph, and its only syntactic construct is the triple, which consists of three components, referred to as subject, predicate, and object. A triple represents a single edge (labeled with the predicate) connecting two nodes (labeled with the subject and object); it describes a binary relationship between the subject and object via the predicate. For example, we might describe the relationship between Harry and Hedwig using this triple:

Figure 1: Example RDF graph.



HarryPotter hasPet Hedwig . where HarryPotter is the subject, hasPet is the predicate, and Hedwig is the object. The subject of a triple is either an IRI or a blank node (an unlabeled node), while the object is an IRI, a blank node, or a literal value (such as a string or integer). For example, we could use the triple: HarryPotter hasemail "harry.potter@hogwarts.net". to capture information about Harry's email address. The predicate of a triple is always an IRI called a "property." IRIs are treated as names that identify particular resources. Blank nodes also denote resources, but the exact resource being identified is not specified, behaving instead like existentially quantified

variables in first-order logic.

A set of triples is called an RDF graph (see Figure 1). In order to facilitate the sharing and exchanging of graphs on the Web, the RDF specification includes an XML serialization. In RDF/XML the triples can be written as

```
<rdf:Description
rdf:about="#HarryPotter">
  <hasPet
rdf:resource="#Hedwig"/>
  <hasEmail>harry.potter@
hogwarts.net
</hasEmail>
</rdf:Description>
```

where #HarryPotter and #Hedwig are fragment identifiers.

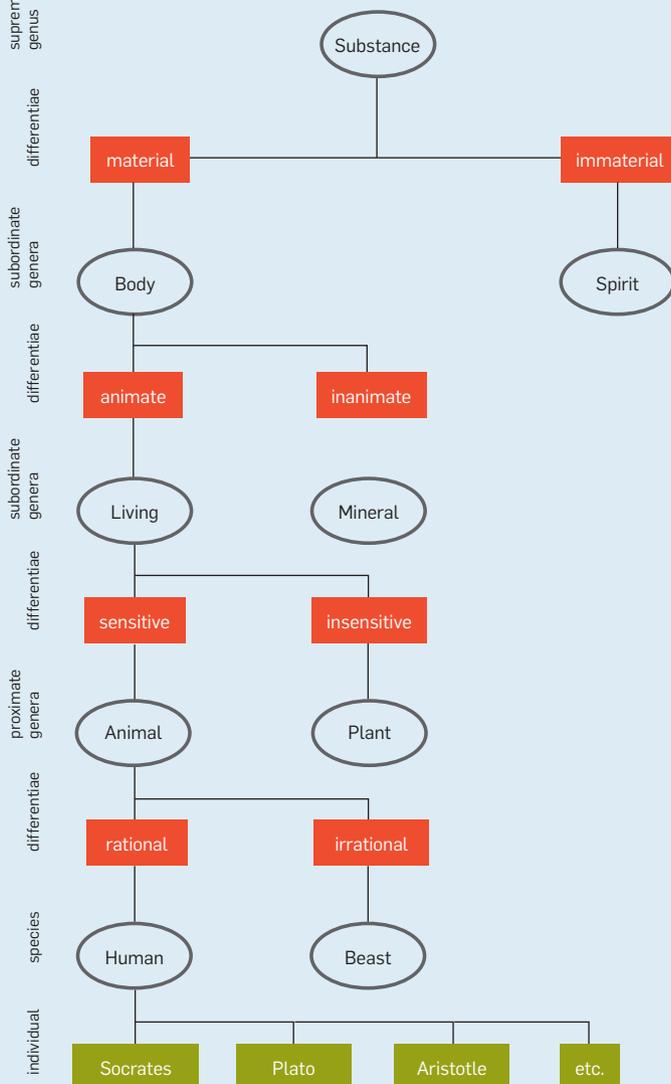
The RDF specification also extends the capabilities of the language by giving additional meaning to certain resources. One of the most important is `rdf:type`, a special property that captures the class-instance relationship; where `rdf` is an abbreviation (called a "namespace prefix") for the string `www.w3.org/1999/02/22-rdf-syntax-ns/`.

For example, we could use the triple: HarryPotter `rdf:type` Wizard . to represent the fact that Harry is an instance of Wizard.

RDF provides a flexible mechanism for adding structured annotations but does little to address the problem of understanding the meaning, or semantics, of the terms in annotations. One possible solution would be to fix a set of terms to be used in annotations and agree on their meaning. This works well in constrained settings like annotating documents; the Dublin Core Metadata Initiative (`dublincore.org/schemas/`) defines just such a set of terms, including, for example, the properties `dc:title`, `dc:creator`, `dc:subject`, and `dc:publisher`. However, this approach is limited with respect to flexibility and extensibility; only a fixed number of terms is defined, and extending the set typically requires a lengthy process in order to agree on which terms to introduce, as well as on their intended semantics. It may also be impractical to impose a single set of terms on all information providers.

An alternative approach is to agree on a language that can be used to define the meaning of new terms (such as by combining and/or restricting existing ones). Such a language should preferably be relatively simple and pre-

Figure 2: Tree of Porphyry.



cisely specified so as to be amenable to processing by software tools. This approach provides greatly increased flexibility, as new terms can be introduced as needed. This is the approach taken in the semantic Web, where ontologies are used to provide extensible vocabularies of terms, each with a well-defined meaning; for example, a suitable ontology might introduce the term `SnowyOwl` and include the information that a `SnowyOwl` is a kind of owl and that owl is a kind of raptor. Moreover, if this information is represented in a way that is accessible to our query engine, the engine would be able to recognize that Hedwig should be included in the answer to a query concerning raptors.

Ontology, in its original philosophical sense, is a branch of metaphysics focusing on the study of existence; its objective is to study the structure of the world by determining what entities and types of entities exist. The study of ontology can be traced back to the work of Plato and Aristotle, including their development of hierarchical categorizations of different kinds of entity and the features that distinguish them; for example, the “tree of Porphyry” identifies animals and plants as subcategories of living things distinguished from each other by animals having “sensitive” souls, with powers of sense, memory, and imagination (see Figure 2).

In computer science, an ontology is an engineering artifact, usually a model of (some aspect of) the world; it introduces vocabulary describing various aspects of the domain being modeled and provides an explicit specification of the intended meaning of that vocabulary. However, the specification often includes classification-based information, not unlike Porphyry’s tree; for example, Wizard may be described as a subcategory of human, with distinguishing features (such as the ability to perform magic).

The RDF vocabulary description language (RDF schema) extends RDF to include the basic features needed to define ontologies. This extension is achieved by giving additional meaning to more “special” resources, including `rdfs:Class`, `rdfs:subClassOf`, `rdfs:subPropertyOf`, `rdfs:domain`, and `rdfs:range`, where `rdfs` is an abbreviation for the string `www.w3.org/2000/01/rdf-schema#`. The

`rdfs:Class` resource is the class of all RDF classes; a resource (such as `Wizard`) that is the object of an `rdf:type` triple is itself an instance of the `rdfs:Class` resource. The `rdfs:subClassOf` and `rdfs:subPropertyOf` properties can be used in an ontology to describe a hierarchy of classes and properties, respectively. For example, the triples:

```
SnowyOwl rdfs:subClassOf Owl .
Owl rdfs:subClassOf Raptor .
```

can be used to represent the fact that a `SnowyOwl` is a kind of `Owl` and that an `Owl` is a kind of `Raptor`. Similarly, the triple:

```
hasBrother rdfs:subPropertyOf
hasSibling .
```

can be used to represent the fact that if  $x$  has a brother  $y$ , then  $x$  also has a sibling  $y$ . Additionally, a property’s domain and range can be specified using `rdfs:domain` and `rdfs:range`. For example, the triples:

```
hasPet rdfs:domain Human.
hasPet rdfs:range Animal.
```

can be used to represent the fact that only Humans can have pets and that all pets are Animals.

### The Web Ontology Language OWL

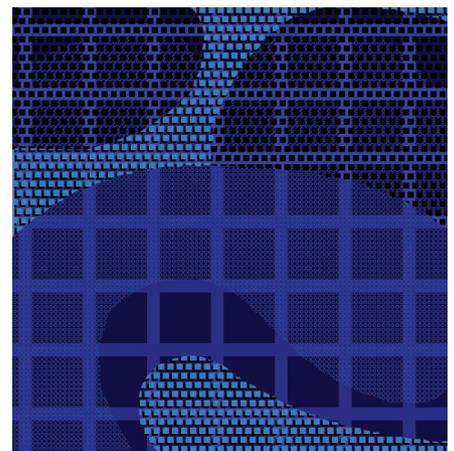
Though obviously an ontology language, RDF is rather limited; it is not able to, for example, describe cardinality constraints (such as Hogwarts students have at most one pet), a feature in most conceptual modeling languages, or describe even a simple conjunction of classes (such as `Student` and `Wizard`). In the late 1990s, the need for a more expressive ontology language was widely recognized within the nascent semantic Web research community and resulted in several proposals for new Web ontology languages, including Simple HTML Ontological Extensions (SHOE), the Ontology Inference Layer (OIL), and DAML+OIL.

In 2001, recognizing that an ontology-language standard is a prerequisite for the development of the semantic Web, the W3C set up a standardization working group to develop a standard for a Web ontology language. The result, in 2004, was the OWL ontology language standard ([www.w3.org/2004/OWL/](http://www.w3.org/2004/OWL/)), exploiting the earlier work on OIL and DAML+OIL while tightening the integration of these languages with RDF. Integrating OWL with RDF provided OWL with an RDF-based syntax,

with the advantage of making OWL ontologies directly accessible to Web-based applications, though the syntax is rather verbose and difficult to read; for example, in RDF/XML, the description of the class of Student Wizards would be written as:

```
<owl:Class>
  <owl:intersectionOf
    rdf:parseType="Collection">
    <owl:Class
      rdf:about="#Student"/>
    <owl:Class
      rdf:about="#Wizard"/>
  </owl:intersectionOf>
</owl:Class>
```

For this reason, here I use an informal “human-readable” syntax based on the one used in the Protégé 4 ontology de-



velopment tool ([protege.stanford.edu/](http://protege.stanford.edu/)) in which the description is written as:

```
Student and Wizard
```

A key feature of OWL is its basis in Description Logics (DLs), a family of logic-based knowledge-representation formalisms descended from Semantic Networks and KL-ONE but that have a formal semantics based on first-order logic.<sup>1</sup> These formalisms all adopt an object-oriented model like the one used by Plato and Aristotle in which the domain is described in terms of individuals, concepts (called “classes” in RDF), and roles (called “properties” in RDF). Individuals (such as Hedwig) are the basic elements of the domain; concepts (such as `Owl`) describe sets of individuals with similar characteristics; and roles (such as `hasPet`) describe relationships between pairs of individuals (such as “`HarryPotter hasPet Hedwig`”). To avoid confusion here I keep to the RDF terminology, referring to these basic language com-

ponents as individuals, classes, and properties.

Along with atomic-class names like Wizard and Owl, DLs also allow for class descriptions to be composed from atomic classes and properties. A given DL is characterized by the set of constructors provided for building class descriptions. OWL is based on a very expressive DL called *SHOIN(D)*, a sort of acronym derived from the features of the language.<sup>11</sup> The class constructors available in OWL include the Booleans and, or, and not, which in OWL are called, respectively, intersectionOf, unionOf, and complementOf, as well as restricted forms of existential ( $\exists$ ) and universal ( $\forall$ ) quantification, which in OWL are called, respectively, someValuesFrom and allValuesFrom restrictions. OWL also allows for properties to be declared transitive; if hasAncestor is a transitive property, then Enoch hasAncestor Cain and Cain hasAncestor Eve implies that Enoch hasAncestor Eve. The *S* in *SHOIN(D)* stands for this basic set of features.

In OWL, someValuesFrom restrictions are used to describe classes, the instances of which are related via a given property to instances of some other class. For example,

Wizard and hasPet some Owl describes Wizards having pet Owls. Note that such a description is itself a class, the instances of which are exactly those individuals that satisfy the description; in this case, they are instances of Wizard and are related via the hasPet property to an individual that is an instance of Owl. If an individual is asserted (stated) to be a member of this class, we know it must have a pet Owl, though we may be unable to identify the Owl in question; that is, someValuesFrom restrictions specify the existence of a relationship. In contrast, allValuesFrom restrictions constrain the possible objects of a given property and are typically used as a kind of localized range restriction. For example, we might want to state that Hogwarts students are allowed to have only owls, cats, or toads as pets without placing a global range restriction on the hasPet property (because other kinds of pet may be possible). We can do this in OWL like this:

```
Class: HogwartsStudent
```



**A key feature of OWL is its basis in Description Logics, a family of logic-based knowledge-representation formalisms that are descendants of Semantic Networks and KL-ONE but that have a formal semantics based on first-order logic.**



```
SubClassOf: hasPet only
(Owl or Cat or Toad)
```

OWL also allows for property hierarchies (the *H* in *SHOIN(D)*), extensionally defined classes using the *oneOf* constructor (*O*), inverse properties using the *inverseOf* property constructor (*I*), cardinality restrictions using the *minCardinality*, *maxCardinality*, and *cardinality* constructors (*N*) and XML Schema datatypes and values (**D**) ([www.w3.org/TR/xmlschema-2/](http://www.w3.org/TR/xmlschema-2/)). For example, we might also state that the instances of HogwartsHouse are Gryffindor, Slytherin, Ravenclaw, and Hufflepuff, that Hogwarts students have an email address (a string), and at most one pet, that isPetOf is the inverse of hasPet, and that a Phoenix can be the pet only of a Wizard:

```
Class: HogwartsHouse
EquivalentTo: {Gryffindor,
Slytherin, Ravenclaw,
Hufflepuff}
Class: HogwartsStudent
SubClassOf: hasEmail some
string
SubClassOf: hasPet max 1
ObjectProperty: hasPet
Inverses: isPetOf
Class: Phoenix
SubClassOf: isPetOf only
Wizard
```

An OWL ontology consists of a set of axioms. As in RDF, the axioms *subClassOf* and *subPropertyOf* can be used to define a hierarchy of classes and properties. In OWL, an *equivalentClass* axiom can also be used as an abbreviation for a symmetrical pair of subClassOf axioms. An *equivalentClass* axiom can be thought of as an “if and only if” condition; given the axiom C equivalentClass D, an individual is an instance of C if and only if it is an instance of D. Combining the axioms *subClassOf* and *equivalentClass* with class descriptions allows for easy extension of the vocabulary by introducing new names as abbreviations for descriptions. For example, the axiom

```
Class: HogwartsStudent
EquivalentTo: Student and
attendsSchool
```

value Hogwarts introduces the class name HogwartsStudent, asserting that its instances are exactly those Students who attend Hogwarts. Axioms can also be used to state that a set of classes is disjoint and describe additional char-

acteristics of properties. Besides being *Transitive*, a property can be *Symmetric*, *Functional*, or *InverseFunctional*; for example, the axioms

```
DisjointClasses: Owl Cat Toad
Property: isPetOf
```

Characteristics: Functional  
state that Owl, Cat, and Toad are disjoint (that is, they have no instances in common) and that `isPetOf` is functional (that is, pets can have only one owner).

These axioms describe constraints on the structure of the domain and play a role similar to the conceptual schema in a database setting; in DLs, such a set of axioms is called a terminology box (TBox). OWL also allows for axioms that assert facts about concrete situations, like data in a database setting; in DLs, such a set of axioms is called an assertion box (ABox). These axioms might, for example, include the facts

```
Individual: HarryPotter
Types: HogwartsStudent
Individual: Fawkes
Types: Phoenix
Facts: isPetOf Dumbledore
```

Basic facts, or those using only atomic classes, correspond directly to RDF triples; for example, the facts just discussed correspond to the following triples:

```
HarryPotter rdf:type, Hog-
wartsStudent .
Fawkes rdf:type Phoenix .
Fawkes isPetOf Dumbledore .
```

The term “ontology” is often used to refer to a conceptual schema or TBox, but in OWL an ontology can consist of a mixture of both TBox and ABox axioms; in DLs, this combination is called a knowledge base.

DLs are fully fledged logics and so have a formal semantics. They can, in fact, be understood as decidable subsets of first-order logic, with individuals being equivalent to constants, concepts to unary predicates, and roles to binary predicates. Besides giving a precise and unambiguous meaning to descriptions of the domain, the formal semantics also allows for the development of reasoning algorithms that can be used to correctly answer arbitrarily complex queries about the domain. An important aspect of DL research is the design of such algorithms and their implementation in (highly optimized) reasoning systems that can be used

by applications to help them “understand” the knowledge captured in a DL-based ontology.

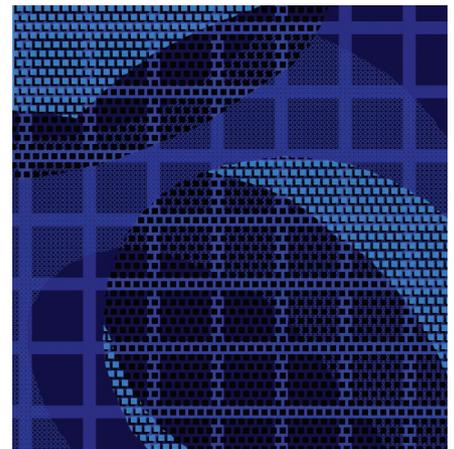
### Ontology Reasoning

Though there are clear analogies between databases and OWL ontologies, there are also important differences. Unlike databases, OWL has a so-called open-world semantics in which missing information is treated as unknown rather than as false and OWL axioms behave like inference rules rather than as database constraints. For example, we have asserted that Fawkes is a Phoenix and a pet of Dumbledore and that only a Wizard can have a pet Phoenix. In OWL, this leads to the implication that Dumbledore is a Wizard; if we were to query the ontology for instances of Wizard, then Dumbledore would be part of the answer. In a database setting the schema could include a similar statement about the Phoenix class, but it would (in this case) be interpreted as a constraint on the data. Adding the fact that Fawkes `isPetOf` Dumbledore without Dumbledore being known to be a Wizard would lead to an invalid database state; such an update would be rejected by a database management system as a constraint violation.

Unlike databases, OWL makes no unique name assumption; for example, given that `isPetOf` is a functional property, then additionally asserting that Fawkes `isPetOf` AlbusDumbledore would imply that Dumbledore and AlbusDumbledore are two names for the same Wizard. In a database setting this would again be treated as a constraint violation. Note that in OWL it is possible to assert (or infer) that two different names do not refer to the same individual; if such an assertion were made about Dumbledore and AlbusDumbledore, then asserting that Fawkes `isPetOf` AlbusDumbledore would make the ontology inconsistent. Unlike database management systems, ontology tools typically don’t reject updates that result in the ontology becoming wholly or partly inconsistent; they simply provide a suitable warning.

The treatment of schema and constraints in a database setting means they can be ignored when answering queries; in a valid database instance, all schema constraints must already be satisfied. This treatment makes query

answering highly efficient; for example, in order to determine if Dumbledore is in the answer to a query for Wizards, it is sufficient to check if this fact is explicitly present in the database. In OWL, the schema plays a much more important role and is actively considered at query time. Considering both the schema and the data can be very powerful, making it possible to answer conceptual, as well as extensional, queries; for example, we can ask not only if Dumbledore is a Wizard but if anybody having a Phoenix for a pet is necessarily a Wizard. This power does, however, make query-answering much more difficult (at least in the worst case); for example, in order to determine if Dumbledore is in the answer to a query



for Wizards, it is necessary to check if Dumbledore would be an instance of Wizard in every possible state of the world that is consistent with the axioms in the ontology. Query answering in OWL is thus analogous to theorem proving, and a query answer is often referred to as an “entailment.” OWL is therefore most suited to applications where the schema plays an important role, where it is not reasonable to assume that complete information about the domain is available, and where information has high value.

Ontologies may be very large and complex; for example, the Systematized Nomenclature of Medicine–Clinical Terms (SNOMED CT) ontology includes more than 400,000 class names. Building and maintaining such an ontology is costly and time-consuming, so providing tools and services to support the ontology-engineering process is critical to both the cost and the quality of the resulting ontology. Ontol-

ogy reasoning therefore plays a central role in both the development of high-quality ontologies and the deployment of ontologies in applications.

In spite of the complexity of reasoning with OWL ontologies, highly optimized DL reasoning systems (such as FaCT++, owl.man.ac.uk/factplusplus/, Racer, www.racer-systems.com/, and Pellet, pellet.owldl.com/) have proved effective in practice; the availability of such systems was one of the key motivations for the W3C to base OWL on a DL. State-of-the-art ontology-development tools (such as SWOOP, code.google.com/p/swoop/, Protégé 4, and TopBraid Composer, www.topbraidcomposer.com) use DL reasoners to give feedback to developers about the logical implications of their designs. This feedback typically includes warnings about inconsistencies and synonyms.

An inconsistent (sometimes called “unsatisfiable”) class is one for which its description is “overconstrained,” with the result that it can never have instances. This inconsistency is typically an unintended consequence of the design (why introduce a name for a class that can never have instances?) and may be due to subtle interactions among axioms. It is therefore useful to be able to detect such classes and bring them to the attention of the ontology engineer. For example, during the recent development of an OWL ontology at NASA’s Jet Propulsion Laboratory, the class “OceanCrustLayer” was found to be inconsistent. Engineers discovered (with the help of debugging tools) that this was the result of its being defined as both a region and a layer, one (a layer) a 2D object and the other (a region) a 3D object. The inconsistency thus highlighted a fundamental error in the ontology’s design.

It is also possible that the descriptions in an ontology mean that two classes necessarily have exactly the same set of instances; that is, they are alternative names for the same class. Having multiple names for the same class may be desirable in some situations (such as to capture the fact that “myocardial infarction” and “heart attack” are the same thing). However, multiple names could also be the inadvertent result of interactions among descriptions or of basic errors by the ontology designer; it is therefore use-



**Reliability and correctness are particularly important when ontology-based systems are used in safety-critical applications; in those involving medicine, for example, incorrect reasoning could adversely affect patient care.**



ful to be able to alert developers to the presence of such synonyms.

In addition to checking for inconsistencies and synonyms, ontology-development tools usually check for implicit subsumption relationships, updating the class hierarchy accordingly. This automated updating is also a useful design aid, allowing ontology developers to focus on class descriptions, leaving the computation of the class hierarchy to the reasoner; it can also be used by developers to check if the hierarchy induced by the class descriptions is consistent with their expert intuition. The two may not be consistent when, for example, errors in the ontology result in unexpected subsumption inferences or “underconstrained” class descriptions result in expected inferences not being found. Not finding expected inferences is common, as it is easy to inadvertently omit axioms that express “obvious” information. For example, an ontology engineer may expect the class of patients with a fracture of both the tibia and the fibula to be a subclass of “patient with multiple fractures”; however, this relationship may not hold if the ontology doesn’t include (explicitly or implicitly) the information that the tibia and fibula are different bones. Failure to find this subsumption relationship should prompt the engineer to add the missing DisjointClasses axiom.

Reasoning is also important when ontologies are deployed in applications, when it is needed to answer standard data-retrieval queries, and to answer conceptual queries about the structure of the domain. For example, biologists use ontologies (such as the Gene Ontology, or GO, and the Biological Pathways Exchange ontology, or BioPAX) to annotate (Web-accessible) data from gene-sequencing experiments, making it possible to answer complex queries (such as “What DNA-binding products interact with insulin receptors?”). Answering requires a reasoner to not only identify individuals that are (perhaps only implicitly) instances of DNA-binding products and of insulin receptors but to identify which pairs of individuals are related (perhaps only implicitly) via the interactsWith property.

Finally, in order to maximize the benefit of reasoning services, tools should be able to explain inferences; without explanations, developers may

find it difficult to repair errors in an ontology and may even start to doubt the correctness of inferences. Such an explanation typically involves computing a (hopefully small) subset of the ontology that still entails the inference in question and, if necessary, presenting the user with a chain of reasoning steps.<sup>12</sup> The explanation in Figure 3 (produced by the Protégé 4 ontology-development tool) describes the steps that lead to the inference mentioned earlier with respect to the inconsistency of OceanCrustLayer.

### Ontology Applications

The availability of tools and reasoning systems has contributed to the increasingly widespread use of OWL, which has become the de facto standard for ontology development in fields as diverse as biology,<sup>19</sup> medicine,<sup>18</sup> geography,<sup>8</sup> geology (the Semantic Web for Earth and Environmental Terminology project, [sweet.jpl.nasa.gov/](http://sweet.jpl.nasa.gov/)), agriculture,<sup>20</sup> and defense.<sup>15</sup> Applications of OWL are particularly prevalent in the life sciences where OWL is used by developers of several large biomedical ontologies, including SNOMED, GO, and BioPAX, mentioned earlier, as well as the Foundational Model of Anatomy ([sig.biostr.washington.edu/projects/fm/](http://sig.biostr.washington.edu/projects/fm/)) and the U.S. National Cancer Institute thesaurus ([www.cancer.gov/cancertopics/terminologyresourcehow](http://www.cancer.gov/cancertopics/terminologyresourcehow)).

The ontologies used in these applications might have been developed specifically for the purpose or without any particular application in mind. Many ontologies are the result of collaborative efforts within a given community

aimed at facilitating (Web-based) information sharing and exchange; some commercially developed ontologies are also subject to a license fee. Many OWL ontologies are available on the Web, identified by a URI and should, in principle, be available at that location. There are also several well-known ontology libraries and even ontology search engines (such as SWOOGLE, [swoogle.umbc.edu/](http://swoogle.umbc.edu/)) that are useful for locating ontologies. In practice, however, applications are invariably built around a predetermined ontology or set of ontologies that are well understood and known to provide suitable coverage of the relevant domains.

The importance of reasoning support in ontology applications was highlighted in a paper describing a project in which the Medical Entities Dictionary (MED), a large ontology (100,210 classes and 261 properties) used at the Columbia Presbyterian Medical Center in New York, was converted to OWL and checked using an OWL reasoner.<sup>13</sup> As reported in the paper, this check revealed “systematic modeling errors” and a significant number of missed subClassOf relationships that, if not corrected, “could have cost the hospital many missing results in various decision-support and infection-control systems that routinely use MED to screen patients.”

In another application, an extended version of the SNOMED ontology was checked using an OWL reasoner that found a number of missing subClassOf relationships. This ontology is being used by the U.K. National Health Service (NHS) to provide “a single and comprehensive system of terms, cen-

trally maintained and updated for use in all NHS organizations and in research” and as a key component of its \$6.2 billion “Connecting for Health” IT program ([www.connectingforhealth.nhs.ukhow](http://www.connectingforhealth.nhs.ukhow)). An important feature of the system is that it can be extended to provide more detailed coverage if needed by specialized applications; for example, a specialist allergy clinic may need to distinguish allergies caused by different kinds of nut so may need to add new terms to the ontology (such as AlmondAllergy):

```
Class: AlmondAllergy
    equivalentTo: Allergy and
    causedBy some Almond
```

Using a reasoner to insert this new term into the ontology ensures it is recognized as a subclassOf NutAllergy, something that is clearly of crucial importance for ensuring that patients with an AlmondAllergy are correctly identified in the national records system as patients with a NutAllergy.

Ontologies are also widely used to facilitate the sharing and integration of information. The Neurocommons project ([sciencecommons.org/projects/data/](http://sciencecommons.org/projects/data/)) aims to provide a platform for, for example, sharing and integrating knowledge in the neuroscience domain; a key component is an ontology of annotations to be used to integrate available knowledge on the Web, including major neuroscience databases. Similarly, the Open Biomedical Ontologies Foundry ([www.obofoundry.org](http://www.obofoundry.org)) is a library of ontologies designed to facilitate international information sharing and integration in the biomedical domain. In information-integration ap-

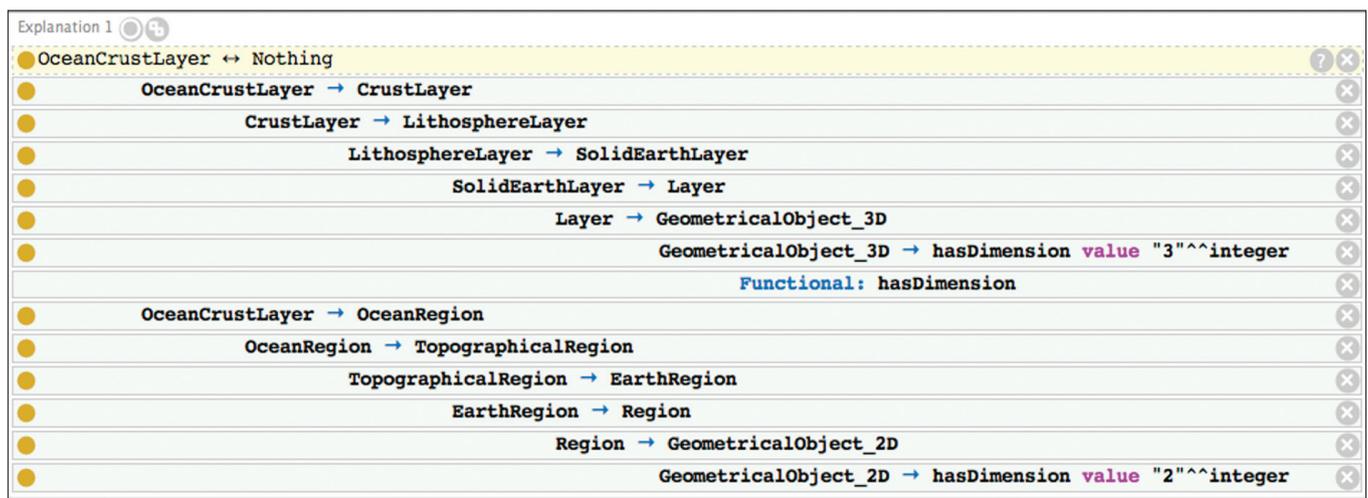


Figure 3: An explanation from Protégé 4.

plications the ontology could play several roles: provide a formally defined and extensible vocabulary for semantic annotations; describe the structure of existing sources and the information they store; and provide a detailed model of the domain against which queries are formulated. Such queries can be answered by using semantic annotations and structural knowledge to retrieve and combine information from multiple sources.<sup>22</sup> It should be noted that the use of ontologies in information integration is far from new and the subject of extensive research within the database community.<sup>2</sup>

With large ontologies, answering conceptual and data-retrieval queries may be a very complex task, and DL



reasoners allow OWL ontology applications to answer complex queries and provide guarantees about the correctness of the result. Reliability and correctness are particularly important when ontology-based systems are used in safety-critical applications; in those involving medicine, for example, incorrect reasoning could adversely affect patient care.

However, RDF and OWL are also used in a range of applications where reasoning plays only a relatively minor role in, for example, the Friend of a Friend, or FOAF, project ([www.foaf-project.org](http://www.foaf-project.org)) and the Dublin Core Metadata Initiative, ([dublincore.org](http://dublincore.org)) and when carrying annotations in Adobe's Extensible Metadata Platform ([www.adobe.com/products/xmp/](http://www.adobe.com/products/xmp/)). In them, RDF is typically used to provide a flexible and extensible data structure for annotations, with the added advantage that IRIs can be used to refer directly to Web resources.

In FOAF, for example, a simple RDF/OWL ontology provides a vocabulary of terms for describing and linking people and their interests and activities; terms include the `foaf:Person` class and properties, including `foaf:name`, `foaf:homepage`, and `foaf:knows`. OWL is used to declare that some properties (such as `foaf:homepage`) are InverseFunctional; that is, they can be used as a key to identify the subject of the property, often a person. However, the semantics of the vocabulary is mainly captured informally in textual descriptions of each term and procedurally interpreted by applications. This informality reduces the need for reasoning systems but limits the ability of applications to share and understand vocabulary extensions.

### Future Directions

The success of OWL also involves many challenges for the future development of both the OWL language and OWL tool support. Central to them is the familiar tension between requirements for advanced features, particularly increased expressive power, and raw performance, particularly the ability to deal with large ontologies and data sets.

Researchers have addressed them by investigating more expressive DLs, developing new and more highly optimized DL reasoning systems and identifying smaller logics that combine still-useful expressive power with better worst-case complexity or other desirable computational properties. Results from these efforts are being exploited by the W3C in order to refine and extend OWL, forming in October 2007 a new W3C Working Group for this purpose ([www.w3.org/2007/OWL/](http://www.w3.org/2007/OWL/)). The resulting language is called OWL 2 (initially called OWL 1.1) based on a more expressive DL called SROIQ.<sup>10</sup> OWL 2 extends OWL with the ability to “qualify” cardinality restrictions to, say, describe the hand as having four parts that are fingers and one part that is a thumb; assert that properties are reflexive, irreflexive, asymmetric, and disjoint (such as to describe `hasParent` as an irreflexive property); and compose properties into property chains (such as to capture the fact that a disease affecting a part of an organ affects the organ as a whole). OWL 2 also provides extended support for datatypes

and for annotations.

Besides increasing the expressive power of the complete language, OWL 2 also defines three so-called profiles, in effect language fragments with desirable computational properties ([www.w3.org/TR/owl2-profiles](http://www.w3.org/TR/owl2-profiles) [www.w3.org/TR/opw12-profiles/](http://www.w3.org/TR/opw12-profiles/)). One is based on DL Lite, a logic for which standard reasoning problems can be reduced to standard query language (SQL) query answering; another is based on EL++, a logic for which standard reasoning problems can be performed in polynomial time; and the third is based on DLP, a logic for which query answering can be implemented using rule-based techniques that have been shown to scale well in practice.

In some cases, even the increased expressive power of OWL 2 may not meet application requirements. One way to further increase the expressive power of the language would be to extend it with Horn-like rules; that is, implications like  $\text{parent}(x, y) \wedge \text{bother}(y, z) \Rightarrow \text{uncle}(x, z)$  stating that if  $y$  is a parent of  $x$  and  $z$  is a brother of  $y$  (the antecedent), then  $z$  is an uncle of  $x$  (the consequent). A notable proposal along these lines is the Semantic Web Rules Language ([www.w3.org/Submission/SWRL/](http://www.w3.org/Submission/SWRL/)). If the semantics of Horn-like rules is restricted so it applies only to named individuals, then its addition does not disturb the decidability of the underlying DL; this restricted form of rules is known as “DL-safe” rules.<sup>17</sup> A W3C working group was established in 2005 to produce a W3C language standard that will “allow rules to be translated between rule languages and thus transferred between rule systems” ([www.w3.org/2005/rules/](http://www.w3.org/2005/rules/)).

As I discussed earlier, reasoning-enabled tools provide vital support for ontology engineering. Recent work has shown how this support can be extended to modular design and module extraction, important techniques for working with large ontologies. When a team of ontology engineers is developing a large ontology, they should divide it into modules in order to make it easier to understand and facilitate parallel work. Similarly, it may be desirable to extract from a large ontology a module containing all the information relevant to some subset of the domain; the resulting small(er) ontology is easier for

humans to understand and applications to use. New reasoning services can be used to alert developers to unanticipated and/or undesirable interactions when modules are integrated and to identify a subset of the original ontology that is indistinguishable from it when used to reason about the relevant subset of the domain.<sup>4</sup>

The availability of an SQL has been an important factor in the success of relational databases, and there have been several proposals for a semantic Web query language. As in the case of RDF and OWL, the W3C in 2004 set up a standardization working group that in January 2008 completed its work on the SPARQL query language standard ([www.w3.org/TR/rdf-sparql-query](http://www.w3.org/TR/rdf-sparql-query)). Strictly speaking, this language is only for RDF, but it is easy to see how it could be extended for use with OWL ontologies, something already happening in practice.

As I mentioned earlier, major research efforts have been directed toward tackling some of the barriers to realizing the semantic Web; considerable progress has been made in such areas as ontology alignment (reconciling ontologies that describe overlapping domains),<sup>18</sup> ontology extraction (extracting ontologies from text),<sup>16</sup> and the automated annotation of both text<sup>6</sup> and images.<sup>5</sup> Of particular interest is the growth of Web 2.0 applications, showing how it might be possible for user communities to collaboratively annotate Web content, as well as create simple forms of ontology via the development of hierarchically organized sets of tags, or folksonomies.<sup>21</sup> Progress has also been made in developing the infrastructure needed to add structured annotations to existing Web resources. For example, in October 2008 the W3C produced a Recommendation for RDFa, a mechanism for embedding RDF in existing XHTML documents ([www.w3.org/TR/rdfa-syntax/](http://www.w3.org/TR/rdfa-syntax/)).

### Conclusion

Semantic Web research aims to help Web-accessible information and services be more effectively exploited, particularly by software agents and applications. As a first step, the W3C developed new languages, including RDF and OWL, that allow for the description of Web resources and the representation

of knowledge to enable applications to use resources more intelligently.

Although a wide range of semantic Web applications is available today, fully realizing the semantic Web still seems a long way off and would first require the solution of many challenging research problems, including those in knowledge representation and reasoning, databases, computational linguistics, computer vision, and agent systems. Moreover, most of the Web is yet to be semantically annotated, and relatively few ontologies are available (even fewer high-quality ones).

However, semantic Web research already has a major influence on the development and deployment of ontology languages and tools (often called semantic Web technologies). They have become a de facto standard for ontology development and are seeing increased use in research labs, as well as in large-scale IT projects, particularly those where the schema plays an important role, where information has high value, and where information may be incomplete. This emerging role is reflected in extended support for semantic Web technologies, including commercial tools, implementations, and applications, from commercial vendors, including Hewlett-Packard, IBM, Oracle, and Siemens.

Related challenges involve both expressive power and scalability. However, the success of the technologies also motivates research and development efforts in academic institutions and industry to address these challenges; it seems certain these efforts will have a major influence on the future development of information technology.

### Acknowledgment

I want to thank Uli Sattler of the University of Manchester and Franz Baader of Dresden Technical University for letting me borrow the idea of using Harry Potter in the ontology examples. 

### References

1. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., and Patel-Schneider, P.F., Eds. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, Cambridge, U.K., 2003.
2. Batini, C., Lenzerini, M., and Navathe, S.B. A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys* 18, 4 (Dec. 1986), 323–364.
3. Berners-Lee, T., Hendler, J., and Lassila, O. The semantic Web. *Scientific American* 284, 5 (May 2001), 34–43.

4. Cuenca Grau, B., Horrocks, I., Kazakov, Y., and Sattler, U. Modular reuse of ontologies: Theory and practice. *Journal of Artificial Intelligence Research* 31 (Apr. 2008), 273–318.
5. Datta, R., Joshi, D., Li, J., and Wang, J.Z. Image retrieval: Ideas, influences, and trends of the new age. *ACM Computing Surveys* 40, 2 (Apr. 2008).
6. Dill, S., Eiron, N., Gibson, D., Gruhl, D., Guha, R., Jhingran, A., Kanungo, T., Rajagopalan, S., Tomkins, A., Tomlin, J.A., and Zien, J.Y. Semtag and seeker: Bootstrapping the semantic Web via automated semantic annotation. In *Proceedings of the 12th International World Wide Web Conference* (Budapest, Hungary, May). ACM Press, New York, 2003, 178–186.
7. Doyle, J. and Patil, R.S. Two theses of knowledge representation: Language restrictions, taxonomic classification, and the utility of representation services. *Artificial Intelligence* 48, 3 (Apr. 1991), 261–297.
8. Golbreich, C., Zhang, S., and Bodenreider, O. The foundational model of anatomy in OWL: Experience and perspectives. *Journal of Web Semantics* 4, 3 (Sept. 2006), 181–195.
9. Goodwin, J. Experiences of using OWL at the ordnance survey. In *Proceedings of the First OWL Experiences and Directions Workshop* (Galway, Ireland, Nov.). CEUR-WS, 2005; CEUR-WS.org/Vol-188/.
10. Horrocks, I., Kutz, O., and Sattler, U. The even more irresistible SROIQ. In *Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning* (Lake District, U.K., June). AAAI Press, Menlo Park, CA, 2006, 57–67.
11. Horrocks, I. and Sattler, U. A tableau decision procedure for SHOIQ. *Journal of Automated Reasoning* 39, 3 (Oct. 2007), 249–276.
12. Kalyanpur, A., Parsia, B., Sirin, E., and Hendler, J. Debugging unsatisfiable classes in OWL ontologies. *Journal of Web Semantics* 3, 4 (Dec. 2005), 243–366.
13. Kershenbaum, A., Fokoue, A., Patel, C., Welty, C., Schonberg, E., Cimino, J., Ma, L., Srinivas, K., Schloss, R., and Murdock, J.W. A view of OWL from the field: Use cases and experiences. In *Proceedings of the Second OWL Experiences and Directions Workshop* (Athens, GA, Nov.). CEUR-WS, 2006; CEUR-WS.org/Vol-216/.
14. Klyne, G. and Carroll, J.J. *Resource Description Framework (RDF): Concepts and Abstract syntax*. W3C Recommendation, Feb. 10, 2004; [www.w3.org/TR/rdf-concepts/](http://www.w3.org/TR/rdf-concepts/).
15. Lacy, L., Aviles, G., Fraser, K., Gerber, W., Mulvehill, A., and Gaskill, R. Experiences using OWL in military applications. In *Proceedings of the First OWL Experiences and Directions Workshop* (Galway, Ireland, Nov. 2005); CEUR-WS.org/Vol-188/.
16. Maedche, A. and Staab, S. Ontology learning for the semantic Web. *IEEE Intelligent Systems* 16, 2 (Mar./Apr. 2001), 72–79.
17. Motik, B., Sattler, U., and Studer, R. Query answering for OWL-DL with rules. *Journal of Web Semantics* 3, 1 (July 2005), 41–60.
18. Shvaiko, P. and Euzenat, J. A survey of schema-based matching approaches. *Journal on Data Semantics IV, Lecture Notes in Computer Science* 3730 (Nov. 2005), 146–171.
19. Sidhu, A., Dillon, T., Chang, E., and Sidhu, B.S. Protein ontology development using OWL. In *Proceedings of the First OWL Experiences and Directions Workshop* (Galway, Ireland, Nov. 2005); CEUR-WS.org/Vol-188/.
20. Soergel, D., Lauser, B., Liang, A., Fisseha, F., Keizer, J., and Katz, S. Reengineering thesauri for new applications: The AGROVOC example. *Journal of Digital Information* 4, 4 (2004).
21. Spyns, P., de Moor, A., Vandenbussche, J., and Meersman, R. From folksonomies to ontologies: How the twin meet. In *Proceedings of On the Move to Meaningful Internet Systems 2006: CoopIS, DOA, GADA, and ODBASE, Lecture Notes in Computer Science* 4275 (Montpellier, France, Oct. 29–Nov. 3). Springer, 2006, 738–755.
22. Stevens, R., Baker, P., Bechhofer, S., Ng, G., Jacoby, A., Paton, N.W., Goble, C.A., and Brass, A. Tambis: Transparent access to multiple bioinformatics information sources. *Bioinformatics* 16, 2 (Feb. 2000), 184–186.

**Ian Horrocks** (Ian.Horrocks@comlab.ox.ac.uk) is a professor of computer science in the Oxford University Computing Laboratory and a fellow of Oriel College, Oxford, U.K.

© 2008 ACM 0001-0782/08/1200 \$5.00