Ontologies and the Semantic Web

Ian Horrocks
Oxford University Computing Laboratory
Oxford, UK
Ian.Horrocks@comlab.ox.ac.uk

ABSTRACT

The goal of semantic web research is to allow the vast range of web-accessible information and services to be more effectively exploited by both humans and automated tools. To facilitate this process, RDF and OWL have been developed as standard formats for the sharing and integration of data and knowledge—the latter in the form of rich conceptual schemas called ontologies. These languages, and the tools developed to support them, have rapidly become de facto standards for ontology development and deployment; they are increasingly used, not only in research labs, but in large scale IT projects. Although many research and development challenges still remain, these "semantic web technologies" are already starting to exert a major influence on the development of information technology.

1. INTRODUCTION

While phenomenally successful in terms of size and number of users, today's World Wide Web is fundamentally a relatively simple artefact. Web content consists mainly of distributed hypertext and hypermedia, and is accessed via a combination of keyword based search and link navigation. This simplicity has been one of the great strengths of the web, and has been an important factor in its popularity and growth: naive users are able to use it, and can even create their own content.

The explosion in both the range and quantity of web content has, however, highlighted some serious shortcomings in the hypertext paradigm. In the first place, the required content becomes increasingly difficult to locate using search and browse. For example, finding information about people with very common names (or with famous namesakes) can be a frustrating experience. 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 all the heads of state of EU countries seems to be beyond the capabilities of any web query engine, in spite of the fact that the relevant information is readily available on the web. This kind of task typically requires the integration of information from multiple sources: a list of EU member states can, for example, be found at `europa.eu`, while `rulers.org` lists heads of state by country.

Specific integration problems can often be solved by using some kind of software "glue" to combine 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 in order to show the location of and provide directions to points of interest such as hotels and restaurants. Another approach, increasingly seen 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[1] and YouTube,[2] and auction sites such as eBay.[3] In these applications, annotations usually take the form simple tags, such as "beach", "birthday", "family" and "friends". The meaning of tags is, however, typically not well defined, and may be impenetrable even to human users: typical examples (from flickr) include "sasquatchmusicfestival", "celebritylookalikes", and "twab08".

While these approaches are very useful, they do not solve the general problem of how to locate and integrate information on demand and without human intervention. This is the aim of the semantic web [3], the ultimate goal being to "allow data to be shared effectively by wider communities, and to be processed automatically by tools as well as manually".[4] The classic example of a semantic web application is an automated travel agent that, given various constraints and preferences, would offer 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 information sources, but would search the web for relevant information in much the same way that a human user might do when planning a vacation.

A major difficulty in realising this goal is that most web content is primarily intended for presentation to and consumption by human users: HTML markup is mainly concerned with layout, size, colour and other presentational issues. Moreover, web pages increasingly use images, often including active links, to present information, and even when content is annotated, the annotations typically take the form

[1] `http://www.flickr.com/`
[2] `http://youtube.com/`
[3] `http://www.ebay.com/`
[4] `http://www.w3.org/2001/sw/SW-FAQ#swactivity` (W3C Semantic Web Frequently Asked Questions).

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, but this may not be so easy for a software agent. A key idea behind the semantic web is to address this problem by giving machine accessible semantics to annotations. This is to be achieved by using ontologies—rich conceptual schemas—to give formally defined meanings to the terms used in annotations, transforming them into *semantic annotations*.

This vision of a semantic web is clearly extremely ambitious, and its full realisation would require the solution of many very hard and long-standing research problems in areas such as knowledge representation and reasoning, databases, computational linguistics, computer vision and agent systems. One such problem is the tradeoff between conflicting requirements for expressive power in the language used for semantic annotations and scalability of the systems used to process them [7]; another is that integrating different ontologies may prove to be at least as hard as integrating the resources that they describe [18]. New problems include the need to create suitable annotations and ontologies, and the variable quality of web information sources.

Notwithstanding the above mentioned problems, considerable progress has been made in the development of the infrastructure needed to support the semantic web. In particular, there has been impressive progress in the development of languages and tools for content annotation and for the design and deployment of ontologies. In the following sections I will describe some of this work in a little more detail. I hope to show that, even if a full realisation of the semantic web is still some way off, these "semantic web technologies" have already been successfully deployed in a range of applications, and are starting to exert a major influence on the development of information technology.

2. SEMANTIC ANNOTATION

The difficulty in sharing and processing web content, or *resources*, derives at least in part from the fact that many web resources are unstructured, and consist of text, images, video etc. For example, we might find a web page including the following piece of unstructured text:

> Harry Potter has a pet called Hedwig.

As it stands, it would be difficult for a software agent, such as a search engine, to recognise the fact that this resource describes a wizard and an owl. We might try to alleviate the problem by adding annotation tags such as ⟨`Wizard`⟩ and ⟨`SnowyOwl`⟩. Taken in isolation, however, such annotations are of only limited value. In the first place the problem of understanding the terms used in the text has simply been transformed into the problem of understanding the terms used in the tags. A query for information about raptors, for example, may not retrieve this text, even though owls are raptors. Moreover, the relationship between Harry Potter and Hedwig has not been captured in these annotations, so a query for wizards having pet owls might not retrieve Harry Potter. In the web setting we might also want to integrate information from multiple sources; for example, instead of coining our own term for SnowyOwl, we might want to point to the relevant term in some resource providing definitive information about owls.

The Resource Description Framework (RDF) is a language that has been developed in order to provide a flexible mechanism for describing web resources and relationships between them [14]. A key feature of RDF is the use of Internationalized Resource Identifiers (IRIs)—a generalisation of Uniform Resource Locators (URLs)—to refer to resources. This facilitates information integration by allowing RDF to directly reference non-local resources. IRIs are typically long strings such `http://hogwarts.net/HarryPotter`, although various abbreviation mechanisms are available; I will, however, usually omit the prefix and just write `HarryPotter`.

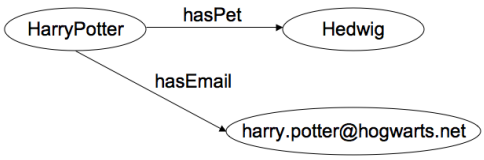

Figure 1: An example of an RDF graph

RDF is a very simple language: its underlying data structure is a labelled directed graph, and its only syntactic construct is the *triple*. As its name suggests, a triple consists of three components, referred to as the *subject*, *predicate* and *object*. A triple represents a single edge (labelled with the predicate) connecting two nodes (labelled 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 the triple:

```
HarryPotter hasPet Hedwig .
```

where `HarryPotter` is the subject, `hasPet` is the predicate and `Hedwig` is the object. The subject of a triple can be either an IRI or a blank node (an unlabelled node), while the object can be 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, and an IRI that is used in the predicate position of a triple is called a *property*. An IRI is treated as a name that identifies a particular resource. Blank nodes also denote resources, but the exact resource being identified is not specified—they behave like existentially quantified variables in First Order Logic.

A set of triples is called an RDF graph. The above triples correspond to the RDF graph shown in Figure 1. In order to facilitate the sharing and exchanging of graphs on the web, an XML serialisation has also been defined. In RDF/XML the above triples could 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 capabilities of RDF have been extended by giving additional meaning to certain resources. One of the most important of these is `rdf:type`,[5] a special property that cap-

[5]Where rdf: is an abbreviation (known as a names-

tures the class-instance relationship. For example, we could use the triple:

```
HarryPotter rdf:type Wizard .
```

to represent the fact that Harry is an instance of Wizard.

As described so far, RDF would provide a flexible mechanism for adding structured annotations, but would do little to address the problem of understanding the meaning, or *semantics*, of the terms used in annotations. One possible solution to this problem would be to fix a set of terms to be used in annotations and to agree on their meaning. This can work well in constrained settings such as annotating documents: the Dublin Core Metadata Initiative[6] defines just such a set of terms, including, for example, the properties `dc:title`, `dc:creator`, `dc:subject` and `dc:publisher`. This approach is, however, quite limited with respect to flexibility and extensibility: only a fixed number of terms are defined, and extending this set typically requires a lengthy process in order to agree on which terms to introduce and 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 which can be used to define the meaning of new terms, e.g., by combining and/or restricting existing ones. Such a language should preferably be relatively simple and precisely specified so as to be amenable to processing by software tools. This approach provides greatly increased flexibility, as new terms can be introduced whenever needed, and it is this approach that is taken in the semantic web, where ontologies are used to provide extensible vocabularies of terms, each with a well defined meaning. A suitable ontology might, for example, introduce the term SnowyOwl, and include the information that SnowyOwls are kinds of Owl, and that Owls are kinds of Raptor. Moreover, if this information were represented in a way that is accessible to our query engine, then it would be able to recognise that Hedwig should be included in the answer to a query for raptors.

Ontology, in its original philosophical sense, is a fundamental branch of metaphysics focusing on the study of existence; its objective is to determine what entities and types of entities actually exist, and thus to study the structure of the world. The study of ontology can be traced back to the work of Plato and Aristotle, and includes the development of hierarchical categorisations of different kinds of entity and the features that distinguish them: the well known "tree of Porphyry", for example, identifies animals and plants as sub-categories of living things distinguished by animals being *sensitive*, and plants being *insensitive* (see Figure 2).

In contrast, in computer science an ontology is an engineering artefact, usually a model of (some aspect of) the world; it introduces vocabulary describing various aspects of the domain being modelled, and provides an explicit specification of the intended meaning of the vocabulary. However, this specification often includes classification based information not unlike that in Porphyry's tree. For example, Wizard may be described as a sub-category of Human, with distinguishing features including the ability to perform magic.

pace prefix) for the string "`http://www.w3.org/1999/02/22-rdf-syntax-ns#`"; i.e., the fully expanded form of `rdf:type` is `http://www.w3.org/1999/02/22-rdf-syntax-ns#Type`.

[6] `http://dublincore.org/schemas/`

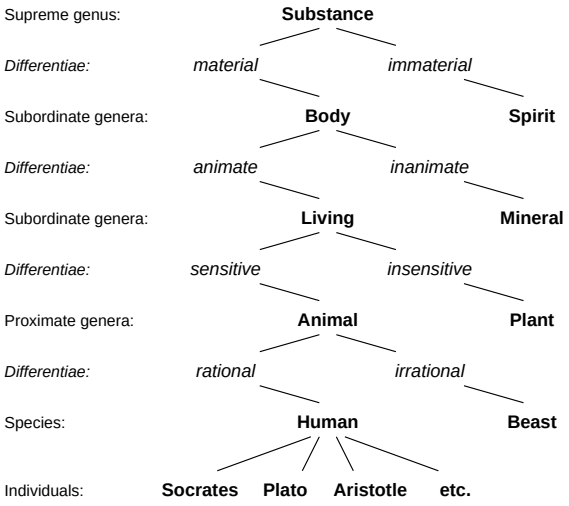

Figure 2: Tree of Porphyry.

The RDF Vocabulary Description Language (RDF Schema) extends RDF to include the basic features needed to define ontologies. This is achieved by giving additional meaning to more "special" resources, including `rdfs:Class`, `rdfs:subClassOf`, `rdfs:subPropertyOf`, `rdfs:domain` and `rdfs:range`.[7] 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 to describe a hierarchy of classes and properties respectively. For example, the triples:

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

could 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 .
```

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

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

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

3. THE WEB ONTOLOGY LANGUAGE OWL

Although already recognisable as an ontology language, the capabilities of RDF are rather limited: they do not, for example, include the ability to describe cardinality constraints (such as Hogwarts Students having *at most one* pet), a feature found in most conceptual modelling languages, or to describe even a simple conjunction of classes

[7] Where rdfs: is an abbreviation for the string "`http://www.w3.org/2000/01/rdf-schema#`".

(such as Student *and* Wizard). The need for a more expressive ontology language was widely recognised within the nascent semantic web research community, and resulted in several proposals for "web ontology languages", including SHOE, OIL and DAML+OIL.

The architecture of the web depends on agreed standards and, recognising that an ontology language standard would be a prerequisite for the development of the semantic web, the World Wide Web Consortium (W3C) set up a standardisation working group to develop a standard for a web ontology language. The result of this activity was the OWL ontology language standard.[8] OWL exploited the earlier work on OIL and DAML+OIL, and also tightened the integration of these languages with RDF.

The integration of OWL with RDF includes the provision of an RDF based syntax. This has the advantage of making OWL ontologies directly accessible to web based applications, but the syntax is rather verbose and not easy to read. For example, the description of the above mentioned class of Student Wizards would be written in RDF/XML as:

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

In the remainder of this article I will instead use an informal "human readable" syntax based on the one used in the Protégé 4 ontology development tool.[9] In this syntax, the above 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 that are descendants of Semantic Networks and KL-ONE, but that have a formal semantics based on first-order logic [1]. These formalisms all adopt an object-oriented model, similar to 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, e.g., "Hedwig", are the basic elements of the domain; concepts, e.g., "Owl", describe sets of individuals having similar characteristics; and roles, e.g., "hasPet", describe relationships between pairs of individuals, such as "HarryPotter hasPet Hedwig". In order to avoid confusion I will keep to the already introduced RDF terminology and from now on refer to these basic language components as individuals, classes and properties.

As well as *atomic* class names such as Wizard and Owl, DLs also allow for class descriptions to be composed from atomic classes and properties. A given DL is characterised by the set of constructors provided for building class descriptions. OWL is based on a very expressive DL called $\mathcal{SHOIN}(\mathbf{D})$—a sort of acronym derived from the various features of the language [11]. The class constructors available in OWL include the Booleans and, or and not, which in OWL are called *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 to be transitive—if `hasAncestor` is a transitive property, then `Enoch hasAncestor Cain` and `Cain hasAncestor Eve` implies that `Enoch hasAncestor Eve`. The $\mathcal{S}$ in $\mathcal{SHOIN}(\mathbf{D})$ stands for this basic set of features.

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

```
Wizard and hasPet some Owl
```

describes those Wizards having pet Owls. Note that such a description is itself a class, the instances of which are just those individuals that satisfy the description; in this case, those individuals that are instances of Wizard and that are related via the hasPet property to an individual that is an instance of Owl. If an individual is asserted to be a member of this class, then we know that they must have a pet Owl, although we may not be able to identify the Owl in question, i.e., 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 localised range restriction. For example, we might want to state that Hogwarts students can 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 in general). We can do this in OWL as follows:

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

In addition to the above mentioned features, OWL also allows for property hierarchies (the $\mathcal{H}$ in $\mathcal{SHOIN}(\mathbf{D})$), extensionally defined classes using the *oneOf* constructor ($\mathcal{O}$), inverse properties using the *inverseOf* property constructor ($\mathcal{I}$), cardinality restrictions using the *minCardinality*, *maxCardinality* and *cardinality* constructors ($\mathcal{N}$), and the use of XML Schema datatypes and values ($\mathbf{D}$).[10] For example, we could additionally state that the instances of HogwartsHouse are exactly Gryffindor, Slytherin, Ravenclaw and Hufflepuff, that Hogwarts students have an email address (which is a string) and at most one pet, that `isPetOf` is the inverse of `hasPet` and that a Phoenix can only be the pet 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, subClassOf and subPropertyOf axioms 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

[8] `http://www.w3.org/2004/OWL/`

[9] `http://protege.stanford.edu/`

[10] `http://www.w3.org/TR/xmlschema-2/`

axiom can be thought of as an "if and only if" condition: given the axiom C equivalentClass D, then an individual is an instance of C if and only if it is an instance of D. Combining subClassOf and equivalentClass axioms with class descriptions allows for easy extension of the vocabulary by introducing new names as abbreviations for descriptions. For example, the following axiom

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

introduces the class name `HogwartsStudent`, and asserts that its instances are just those Students that attend Hogwarts. Axioms can also be used to state that a set of classes is disjoint, and to describe additional characteristics of properties: as well as 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 (i.e., that they have no instances in common), and that isPetOf is Functional (i.e., pets can have at most one owner).

The above mentioned axioms describe constraints on the structure of the domain, and play a similar role to the conceptual schema in a database setting; in DLs such a set of axioms is called a TBox (Terminology Box). OWL also allows for axioms asserting facts about some concrete situation, similar to data in a database setting; in DLs such a set of axioms is called an ABox (Assertion Box). These might, for example, include the facts:

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

Basic facts (i.e., those using only atomic classes) correspond directly to RDF triples—the above facts, for example, correspond to the following triples:

```
HarryPotter rdf:type, HogwartsStudent .
Fawkes rdf:type Phoenix .
Fawkes isPetOf Dumbledore .
```

The term ontology is often used to refer just 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 known as a Knowledge Base.

Description Logics are fully fledged logics and so have a formal semantics. DLs can, in fact, be seen as decidable subsets of first-order logic, with individuals being equivalent to constants, concepts to unary predicates and roles to binary predicates. As well as giving a precise and unambiguous meaning to descriptions of the domain, this also allows for the development of reasoning algorithms that can provide correct answers to arbitrarily complex queries about the domain. An important aspect of DL research has been the design of such algorithms, and their implementation in (highly optimised) reasoning systems that can be used by applications to help them "understand" the knowledge captured in a DL based ontology.

4. ONTOLOGY REASONING

Although 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 false, and OWL axioms behave like inference rules rather than database constraints. In the above axioms, for example, Fawkes is said to be a Phoenix and to be the pet of Dumbledore, and it is also stated 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 in this case it would be interpreted as a constraint on the data: adding the fact that Fawkes isPetOf Dumbledore without Dumbledore being already *known* to be a Wizard would lead to an invalid database state, and such an update would therefore be rejected by a database management system as a constraint violation.

In contrast to databases, OWL also makes no unique name assumption (UNA). For example, given that isPetOf is a Functional property, then additionally asserting that Fawkes isPetOf AlbusDumbledore would lead to the implication that Dumbledore and AlbusDumbledore are two names for the same individual. 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 that they can be ignored at query time—in a valid database instance all the schema constraints must already be satisfied. This makes query answering very efficient: 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. This can be very powerful, and makes it possible to answer conceptual as well as extensional queries—for example, we can ask not only if Dumbledore is a Wizard, but if it is the case that anybody having a Phoenix for a pet must be a Wizard. It does, however, make query answering *much* more difficult (at least in the worst case): 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: the well known SNOMED Clinical Terms ontology includes, for example, more than 400,000 class names. Building and maintaining

such ontologies is very costly and time consuming, and providing tools and services to support this "ontology engineering" process is of crucial importance to both the cost and the quality of the resulting ontology. Moreover, as we have seen above, query answering in OWL is not simply a matter of checking the data, but may require complex reasoning to be performed. Ontology 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 optimised DL reasoning systems such as FaCT++,[11] Racer,[12] and Pellet[13] have proved to be very effective in practice—in fact the availability of such reasoning systems was one of the key motivations for basing OWL on a DL. State of the art ontology development tools, such as SWOOP,[14] Protégé 4, and TopBraid Composer,[15] use DL reasoners to provide feedback to the user about the logical implications of their design. This typically includes (at least) warnings about inconsistencies and synonyms.

An inconsistent (sometimes called unsatisfiable) class is one whose description is "over-constrained", with the result that it can never have any instances. This is typically an unintended feature of the design—why introduce a name for a class that can never have any instances—and may be due to subtle interactions between axioms. It is, therefore, very useful to be able to detect such classes and bring them to the attention of the ontology engineer. For example, during the development of an OWL ontology at the NASA Jet Propulsion Laboratory, the class "OceanCrustLayer" was found to be inconsistent. This was discovered (with the help of debugging tools) to be the result of its being defined to be both a region and a layer, one of which (layer) was a 2-dimensional object and the other a 3-dimensional object. The inconsistency thus highlighted a fundamental error in the design of the ontology.

It is also possible that the descriptions in an ontology mean that two classes necessarily have exactly the same set of instances, i.e., that they are alternative names for the same class. This may be desirable in some situations, e.g., to capture the fact that "Myocardial infarction" and "Heart attack" mean the same thing. It could, however, also be the inadvertent result of interactions between descriptions or of basic errors on the part of the ontology designer; it is, therefore, also useful to be able to alert users to the presence of such synonyms.

In addition to checking for inconsistencies and synonyms, ontology development tools usually check for implicit subsumption relationships, and update the class hierarchy accordingly. This is also a very useful design aid: it allows ontology developers to focus on class descriptions, leaving the computation of the class hierarchy to the reasoner, and it can also be used by developers to check if the hierarchy induced by the class descriptions is consistent with their intuition. This may not be the case when, for example, errors in the ontology result in unexpected subsumption inferences, or "under-constrained" class descriptions result in expected inferences not being found. The latter case is extremely common, as it is easy to inadvertently omit axioms that express "obvious" information. For example, an ontology engineer may expect the class of patients who have a fracture of both the tibia and the fibula to be a subClassOf "patient with multiple fractures"; however, this may not be the case 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 will alert the engineer to the missing DisjointClasses axiom.

Reasoning is also important when ontologies are deployed in applications—as we have already seen, it is needed in order to answer standard data retrieval queries as well as to answer conceptual queries about the structure of the domain. For example, biologists use ontologies such as the Gene Ontology (GO) and the Biological Pathways Exchange ontology (BioPAX) to annotate (web-accessible) data from gene sequencing experiments so as to be able to answer complex queries such as "what DNA binding products interact with insulin receptors". Answering this query requires a reasoner not only to identify individuals that are (perhaps only implicitly) instances of DNA binding products and of insulin receptors, but also to identify which pairs of individuals are (perhaps only implicitly) related via the interactsWith property.

Finally, in order to maximise the benefit of reasoning services, tools should be able to explain inferences: without this facility, users may find it difficult to repair errors in the ontology and may even start to doubt the correctness of inferences. 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 [12]. Figure 3, for example, shows an explanation, produced by the Protégé 4 ontology development tool, of the above mentioned inference with respect to the inconsistency of OceanCrustLayer.

5. ONTOLOGY APPLICATIONS

The availability of tools and reasoning systems such as those mentioned in Section 4 has contributed to the increasingly widespread use of OWL, and it has become the de facto standard for ontology development in fields as diverse as biology [19], medicine [8], geography [9], geology [23], agriculture [20] and defence [15]. Applications of OWL are particularly prevalent in the life sciences where it has been used by the developers of several large biomedical ontologies, including the SNOMED, GO and BioPAX ontologies mentioned in Section 4, the Foundational Model of Anatomy (FMA) and the National Cancer Institute thesaurus.

The ontologies used in such applications may have been specifically developed for the purpose, or may have been developed 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 ontologies are even commercially developed and subject to a licence fee.

Many OWL ontologies are now available on the web—an OWL ontology is identified by a URI, and the ontology should, in principle, be available at that location. There are also several well known ontology libraries, and even ontology search engines such as SWOOGLE,[16] that can be used

[11] http://owl.man.ac.uk/factplusplus/
[12] http://www.racer-systems.com/
[13] http://pellet.owldl.com/
[14] http://code.google.com/p/swoop/
[15] http://www.topbraidcomposer.com/
[16] http://swoogle.umbc.edu/

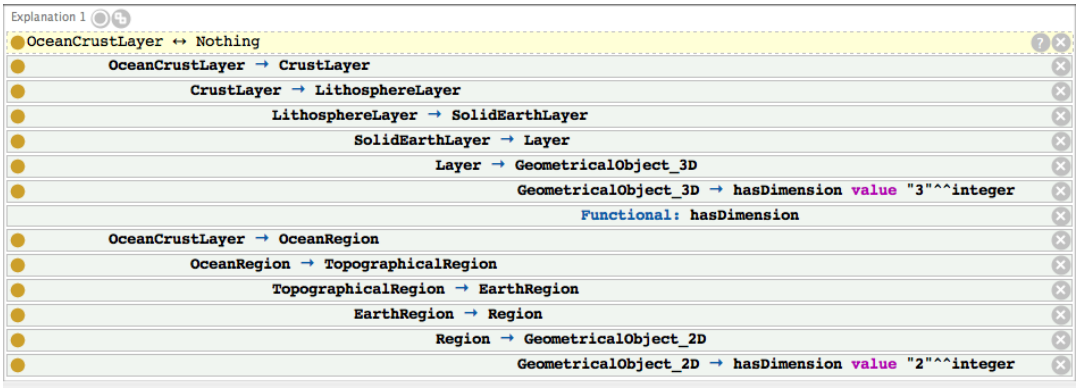

Figure 3: An explanation from Protégé 4

to locate 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 recent paper describing a project in which the Medical Entities Dictionary (MED), a large ontology (100,210 classes and 261 properties) that is used at the Columbia Presbyterian Medical Center, was converted into OWL and checked using an OWL reasoner [13]. This check revealed "systematic modelling errors", and a significant number of missed subClassOf relationships which, 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".

Similarly, an extended version of the SNOMED ontology was checked using an OWL reasoner, and a number of missing subClassOf relationships found. This ontology is being used by the UK National Health Service (NHS) to provide "A single and comprehensive system of terms, centrally maintained and updated for use in all NHS organisations and in research", and as a key component of their new £6.2 billion "Connecting for Health" IT programme. An important feature of this system is that it can be extended to provide more detailed coverage if needed by specialised applications. For example, a specialist allergy clinic may need to distinguish allergies caused by different kinds of nut, and so may 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 will ensure that it is recognised as a subClassOf NutAllergy. This is clearly of crucial importance in order to ensure that patients with an AlmondAllergy are correctly identified in the national records system as patients having a NutAllergy.

Ontologies are also widely used to facilitate the sharing and integration of information. The Neurocommons project,[17] for example, aims to provide a platform for sharing and integrating knowledge in the neuroscience domain. A key component is an ontology of annotations that will be used to integrate available knowledge on the web, including major neuroscience databases. Similarly, the OBO Foundry[18] is a library of ontologies designed to facilitate information sharing and integration in the biomedical domain.

In information integration applications the ontology can play several roles: it can provide a formally defined and extensible vocabulary for use in semantic annotations, it can be used to describe the structure of existing sources and the information that they store, and it can provide a detailed model of the domain against which queries can be formulated. Such queries can be answered by using semantic annotations and structural knowledge to retrieve and combine information from multiple sources [22]. It should be noted that the use of ontologies in information integration is far from new, and has already been the subject of extensive research within the database community [2].

It is easy to imagine that, with large ontologies, answering conceptual and data retrieval queries may be a very complex task. The use of DL reasoners allows OWL ontology applications to answer complex queries and to provide guarantees about the correctness of the result. Reliability and correctness are clearly important features of any information system; they are particularly important if ontology based systems are to be used in safety-critical applications such as medicine, where incorrect reasoning could adversely impact patient care.

RDF and OWL have, however, also been used in a range of applications where reasoning plays only a relatively minor role. Examples include the Friend of a Friend (FOAF)[19] project, the Dublin Core Metadata Initiative, and the use of RDF to carry annotations in Adobe's Extensible Metadata Platform (XMP).[20] In these applications, RDF is typically used to provide a flexible and extensible data structure for annotations, with the added advantage that IRIs can be used to directly refer to web resources.

In FOAF, for example, a simple RDF/OWL ontology is used to provide a vocabulary of terms for describing and linking people and their interests and activities; these terms include the `foaf:Person` class and properties such as `foaf:name`, `foaf:homepage` and `foaf:knows`. OWL is used to declare that properties such as `foaf:homepage` are In-

[17] `http://sciencecommons.org/projects/data/`

[18] `http://www.obofoundry.org/`

[19] `http://www.foaf-project.org/`

[20] `http://www.adobe.com/products/xmp/`

verseFunctional, i.e., that they can be used as a key to uniquely identify the subject of the property (often a person). The semantics of the vocabulary is, however, mainly captured informally in textual descriptions of each term, and is procedurally interpreted by applications. This reduces the need for reasoning systems, but limits the ability of applications to share and understand vocabulary extensions.

6. FUTURE DIRECTIONS

As we have seen in Section 5, OWL is already being successfully used in many applications. This success brings with it, however, many challenges for the future development of both the OWL language and OWL tool support. Central to these is the familiar tension between requirements for advanced features, in particular increased expressive power, and raw performance, in particular the ability to deal with very large ontologies and data sets.

Researchers have addressed these problems by investigating more expressive DLs, developing new and more highly optimised DL reasoning systems, and identifying smaller logics that combine useful expressive power with lower worst case complexity or other desirable computational properties. Results from these research efforts are now being exploited in order to refine and extend OWL, a new W3C Working Group having been formed for this purpose.[21] The resulting language is called OWL 2,[22] and is based on a more expressive DL called $\mathcal{SROIQ}$ [10]. OWL 2 extends OWL with the ability to "qualify" cardinality restrictions, e.g., to describe the hand as having four parts *that are fingers* and one part *that is a thumb*; the ability to assert that properties are reflexive, irreflexive, asymmetric and disjoint, e.g., to describe hasParent as an irreflexive property; and the ability to compose properties into property chains, e.g., 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.

As well as increasing the expressive power of the complete language, OWL 2 also defines several *profiles*: language fragments that have desirable computational properties.[23] These include a profile based on DL Lite, a logic for which standard reasoning problems can be reduced to SQL query answering; a profile based on $\mathcal{EL}^{++}$, a logic for which standard reasoning problems can be performed in polynomial time; and a profile 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 provided by 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, i.e., implications such as $\text{parent}(x, y) \wedge \text{brother}(y, z) \Rightarrow \text{uncle}(x, z)$, which states 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). There have been several proposals along these lines, most notably the Semantic Web Rules Language (SWRL).[24] If the semantics of such rules is restricted so that they only apply to named individuals, then their addition does not disturb the decidability of the underlying DL; this restricted form of rules is known as DL-safe rules [17]. Efforts are now underway to produce a W3C language standard that will "allow rules to be translated between rule languages and thus transferred between rule systems".[25]

As we saw in Section 4, 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 developing a large ontology, it is useful if not essential to divide the ontology into modules in order to make it easier to understand and to facilitate parallel work by a team of ontology engineers. 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 will be easier for humans to understand and easier for applications to use. New reasoning services can be used both 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 [4].

The availability of a standard query language (SQL) has been an important factor in the success of relational databases. It has long been recognised that the semantic web, and semantic web knowledge representation languages such as RDF and OWL, would also benefit from the availability of a standardised query language, and there have been several proposals for such a language. As in the case of RDF and OWL, a W3C standardisation working group was set up, and has recently completed its work on the SPARQL query language standard.[26] Strictly speaking, this is only a query language for RDF, but it is easy to see how it could be extended for use with OWL ontologies, and this is already happening in practice.

Major research efforts have also been directed towards tackling some of the barriers to realising the semantic web mentioned in Section 1, and considerable progress has been made in areas such as ontology alignment (reconciling ontologies that describe overlapping domains) [18], ontology extraction (extracting ontologies from, e.g., text) [16], and the automated annotation of both text [6] and images [5]. A particularly interesting development has been the growth of Web 2.0 applications—this has shown how it might be possible for user communities to collaboratively annotate web content, as well as to create simple forms of ontology via the development of hierarchically organised sets of tags, often called folksonomies [21]. Progress has also been made in developing the infrastructure needed to add structured annotations to existing web resources. The recent RDFa proposal, for example, provides a mechanism for embedding RDF in existing XHTML documents.[27]

7. CONCLUSIONS

As we have seen, the goal of semantic web research is to enable the vast range of web-accessible information and services to be more effectively exploited, in particular by software agents and applications. As a first step, languages such

[21] `http://www.w3.org/2007/OWL/`
[22] It was initially called OWL 1.1.
[23] `http://www.w3.org/TR/owl2-profiles/`
[24] http://www.w3.org/Submission/SWRL/
[25] `http://www.w3.org/2005/rules/`
[26] `http://www.w3.org/TR/rdf-sparql-query/`
[27] `http://www.w3.org/TR/rdfa-syntax/`

as RDF and OWL have been developed; these languages allow for the description of web resources, and for the representation of knowledge that will enable applications to use resources more intelligently.

Although a wide range of semantic web applications have now appeared, fully realising the semantic web still seems some way off, and would require the solution of many very hard and long-standing research problems. Moreover, the vast majority of the web has yet to be semantically annotated, and there are still relatively few ontologies available (and even fewer high quality ones).

Semantic web research has, however, already had a major impact on the development and deployment of ontology languages and tools—now often called semantic web technologies. These technologies have rapidly become a de facto standard for ontology development, and are seeing increasing use not only in research labs but 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 is reflected in extended support for semantic web technologies, including commercial tools, implementations and applications, from major players such as HP, IBM, Oracle and Siemens.

The success of semantic web technologies, and their increasing use in large scale applications, has brought with it new challenges, both with respect to expressive power and scalability. However, this success also means that major research and development efforts in both academia and industry are now focused on addressing these challenges, and it seems certain that these efforts will have a major influence on the future development of information technology.

8. ACKNOWLEDGEMENTS

Thanks to all my friends and collaborators in the semantic web and description logic communities for inspiration, help, and a lot of fun. Particular thanks to Uli Sattler and Franz Baader from whom I borrowed the idea of using Harry Potter in ontology examples.

9. REFERENCES

[1] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2003.

[2] C. Batini, M. Lenzerini, and S. B. Navathe. A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys*, 18(4):323–364, 1986.

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

[4] B. Cuenca Grau, I. Horrocks, Y. Kazakov, and U. Sattler. Modular reuse of ontologies: Theory and practice. *J. of Artificial Intelligence Research*, 31:273–318, 2008.

[5] R. Datta, D. Joshi, J. Li, and J. Z. Wang. Image retrieval: Ideas, influences, and trends of the new age. *ACM Computing Surveys*, 40(2), 2008.

[6] S. Dill, N. Eiron, D. Gibson, D. Gruhl, R. Guha, A. Jhingran, T. Kanungo, S. Rajagopalan, A. Tomkins, J. A. Tomlin, and J. Y. Zien. Semtag and seeker: Bootstrapping the semantic web via automated semantic annotation. In *Proc. of the Twelfth International World Wide Web Conference (WWW 2003)*, 2003.

[7] J. Doyle and R. S. Patil. Two theses of knowledge representation: Language restrictions, taxonomic classification, and the utility of representation services. *Artificial Intelligence*, 48:261–297, 1991.

[8] C. Golbreich, S. Zhang, and O. Bodenreider. The foundational model of anatomy in OWL: Experience and perspectives. *J. of Web Semantics*, 4(3), 2006.

[9] J. Goodwin. Experiences of using OWL at the ordnance survey. In *Proc. of the First OWL Experiences and Directions Workshop*, volume 188 of *CEUR Workshop Proceedings*. CEUR (`http://ceur-ws.org/`), 2005.

[10] I. Horrocks, O. Kutz, and U. Sattler. The even more irresistible $\mathcal{SROIQ}$. In *Proc. of the 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2006)*, pages 57–67. AAAI Press, 2006.

[11] I. Horrocks and U. Sattler. A tableau decision procedure for $\mathcal{SHOIQ}$. *J. of Automated Reasoning*, 39(3):249–276, 2007.

[12] A. Kalyanpur, B. Parsia, E. Sirin, and J. Hendler. Debugging unsatisfiable classes in owl ontologies. *J. of Web Semantics*, 3(4):243–366, 2005.

[13] A. Kershenbaum, A. Fokoue, C. Patel, C. Welty, E. Schonberg, J. Cimino, L. Ma, K. Srinivas, R. Schloss, and J. W. Murdock. A view of OWL from the field: Use cases and experiences. In *Proc. of the Second OWL Experiences and Directions Workshop*, volume 216 of *CEUR Workshop Proceedings*. CEUR (`http://ceur-ws.org/`), 2006.

[14] G. Klyne and J. J. Carroll. Resource description framework (RDF): Concepts and abstract syntax. W3C Recommendation, 10 February 2004. Available at `http://www.w3.org/TR/rdf-concepts/`.

[15] L. Lacy, G. Aviles, K. Fraser, W. Gerber, A. Mulvehill, and R. Gaskill. Experiences using OWL in military applications. In *Proc. of the First OWL Experiences and Directions Workshop*, volume 188 of *CEUR Workshop Proceedings*. CEUR (`http://ceur-ws.org/`), 2005.

[16] A. Maedche and S. Staab. Ontology learning for the semantic web. *IEEE Intelligent Systems*, 16(2):72–79, 2001.

[17] B. Motik, U. Sattler, and R. Studer. Query answering for OWL-DL with rules. *J. of Web Semantics*, 3(1):41–60, 2005.

[18] P. Shvaiko and J. Euzenat. A survey of schema-based matching approaches. *J. on Data Semantics*, IV:146–171, 2005.

[19] A. Sidhu, T. Dillon, E. Chang, and B. S. Sidhu. Protein ontology development using OWL. In *Proc. of the First OWL Experiences and Directions Workshop*, volume 188 of *CEUR Workshop Proceedings*. CEUR (`http://ceur-ws.org/`), 2005.

[20] D. Soergel, B. Lauser, A. Liang, F. Fisseha, J. Keizer, and S. Katz. Reengineering thesauri for new applications: The AGROVOC example. *J. of Digital Information*, 4(4), 2004.

[21] P. Spyns, A. de Moor, J. Vandenbussche, and R. Meersman. From folksologies to ontologies: How the twain meet. In *Proc. of OTM Conferences (1)*,

volume 4275 of *Lecture Notes in Computer Science*, pages 738–755, 2006.

[22] R. Stevens, P. Baker, S. Bechhofer, G. Ng, A. Jacoby, N. W. Paton, C. A. Goble, and A. Brass. Tambis: Transparent access to multiple bioinformatics information sources. *Bioinformatics*, 16(2):184–186, 2000.

[23] Semantic web for earth and environmental terminology (SWEET). Jet Propulsion Laboratory, California Institute of Technology, 2006. `http://sweet.jpl.nasa.gov/`.