What Are Ontologies Good For?

Ian Horrocks

Oxford University, Oxford, UK,
Ian.Horrocks@comlab.ox.ac.uk,
http://web.comlab.ox.ac.uk/people/Ian.Horrocks/

Abstract. 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. In contrast, in computer science an ontology is an engineering artifact, usually a "conceptual model" of (some aspect of) the world, typically formalised as a logical theory. Formalising an ontology using a suitable logic opens up the possibility of using automated reasoning to support both ontology design and deployment. The value of such support has already been demonstrated in medical applications, where it has been used to help repair and enrich ontologies that play an important role in patient care.
Even with the aid of reasoning enabled tools, developing and maintaining good quality ontologies is a difficult and costly task, and problems related to the availability of good quality ontologies threaten to limit the deployment of ontology based information systems. This has resulted in ontology engineers increasingly looking to the philosophy community for possible solutions, and in particular as a source of relevant expertise in the organisation and formalisation of knowledge.

1 Introduction

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 1).

In contrast, in computer science an ontology is an engineering artifact, usually a so-called "conceptual 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 by describing the relationships between different vocabulary terms. These relationships do, however, invariably include classification based relationships not unlike those used in Porphyry's tree.

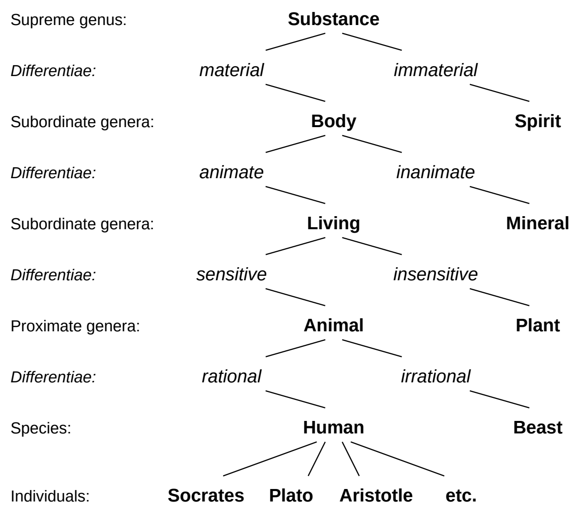


Fig. 1. Tree of Porphyry.

In a logic-based (formal) ontology, the vocabulary can be thought of as predicates and constants. For example, Animal and Plant can be thought of as (unary) predicates, and Socrates, Plato and Aristotle as constants. Relationships between vocabulary terms can be specified using *axioms*, logical sentences such as

$$\forall x[\mathsf{Human}(x) \rightarrow \mathsf{Animal}(x) \wedge \mathsf{Sensitive}(x)]$$
$$\mathsf{Human}(\mathsf{Socrates})$$

An ontology can be thought of simply as a set of such axioms—i.e., a logical theory.

Viewing an ontology as a logical theory opens up the possibility of using automated reasoning to check, e.g., internal consistency (the theory does not entail "false"), and other (non-) entailments. This is not just a theoretical possibility, but is the raison d'être of many logic based ontology languages, in particular those based on description logics—for these languages, sophisticated reasoning tools have been developed and are in widespread use. Such tools can be used, for example, to check for *concept subsumption*, i.e., an entailment of the form $\mathcal{O} \models \forall x[C(x) \rightarrow D(x)]$ for an ontology $\mathcal{O}$, and concepts C and D.

The (possible) existence of such tools was an important factor in the design of the OWL ontology language [12] and its basis in description logics. OWL was initially developed for use in the so-called Semantic Web; the availability of description logic based reasoning tools has, however, contributed to the increasingly widespread use of OWL, not only in the Semantic Web per se, but as a popular language for ontology development in fields as diverse as biology [22], medicine [7], geography [8], geology [27], astronomy [5], agriculture [24] and defence [18].

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 Biological Pathways Exchange (BioPAX) ontology [21], the GALEN ontology [20], the Foundational Model of Anatomy (FMA) [7], and the National Cancer Institute thesaurus [10].

The importance of reasoning support was highlighted by [16], who described a project in which the Medical Entities Dictionary (MED), a large ontology used at the Columbia Presbyterian Medical Center, was checked using a reasoner. This check revealed “systematic modelling errors”, and many missed concept subsumptions, the combination of which “could have cost the hospital many missing results in various decision support and infection control systems that routinely use MED to screen patients”.

2 The Web Ontology Language OWL

The Web Ontology Language (OWL) [19] is currently by far the most widely used ontology language. OWL was developed by a World Wide Web Consortium (W3C) working group in order to extend the capabilities of the Resource Description Framework (RDF), a language for representing basic information about entities and relationships between them [12]. OWL exploited existing work on langauges such as OIL [6] and DAML+OIL [11] and, like them, was based on a description logic (DL).

Description logics (DLs) are a family of logic-based knowledge representation formalisms; they are descendants of Semantic Networks [29] and KL-ONE [2]. 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* (usually called *classes* in ontology languages), and *roles* (usually called *relationships* or *properties* in ontology languages). Individuals, e.g., “Socrates”, are the basic elements of the domain; concepts, e.g., “Human”, describe sets of individuals having similar characteristics; and roles, e.g., “hasPupil” describe relationships between pairs of individuals, such as “Socrates hasPupil Plato”.

As well as *atomic* concept names such as Human, DLs also allow for concept descriptions to be composed from atomic concepts and roles. Moreover, it is possible to assert that one concept (or concept description) is subsumed by (is a sub-concept of), or is exactly equivalent to, another. This allows for easy extension of the vocabulary by introducing new names as abreviations for descriptions. For example, using standard DL notation, we might write:

$$\mathsf{HappyParent} \equiv \mathsf{Parent} \sqcap \forall \mathsf{hasChild}.(\mathsf{Intelligent} \sqcup \mathsf{Athletic})$$

This introduces the concept name $\mathsf{HappyParent}$, and asserts that its instances are just those individuals that are instances of Parent, and all of whose children are instances of either $\mathsf{Intelligent}$ or $\mathsf{Athletic}$.

Another distinguishing feature of DLs is that they are logics, and so have a formal semantics. DLs can, in fact, be seen as decidable subsets of first-order predicate logic, with individuals being equivalent to constants, (atomic) concepts to unary predicates and (atomic) roles to binary predicates. Similarly, complex concepts are equivalent to formulae with one free variable, and standard axioms are equivalent to (bi-) implications with the free variable universally quantified at the outer level. For example, the concept used above to describe a happy parent is equivalent to the following formula

$$\mathsf{Parent}(x) \land \forall y[\mathsf{hasChild}(x, y) \rightarrow (\mathsf{Intelligent}(y) \lor \mathsf{Athletic}(y))]$$

and the DL axiom introducing the concept name HappyParent is equivalent to the following bi-implication:

$$\forall x[\mathsf{HappyParent}(x) \iff \mathsf{Parent}(x) \land \forall y[\mathsf{hasChild}(x, y) \rightarrow (\mathsf{Intelligent}(y) \lor \mathsf{Athletic}(y))]]$$

As well as giving a precise and unambiguous meaning to descriptions of the domain, the use of a logic, and in particular of a decidable logic, also allows for the development of reasoning algorithms that can be used to answer complex questions 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.

A given DL is characterised by the set of constructors provided for building concept descriptions. These typically include at least intersection ($\sqcap$), union ($\sqcup$) and complement ($\neg$), as well as restricted forms of existential ($\exists$) and universal ($\forall$) quantification, which in OWL are called, respectively, *someValues-From* and *allValuesFrom* restrictions. OWL is based on a very expressive DL called $\mathcal{SHOIN}$ that also provides cardinality restrictions ($\geqslant$, $\leqslant$) and enumerated classes (called *oneOf* in OWL) [12, 13]. Cardinality restrictions allow, e.g., for the description of a concept such as people who have at least two children, while enumerated classes allow for classes to be described by simply enumerating their instance, e.g.,:

$$\mathsf{EUcountries} \equiv \{\mathsf{Austria}, \ldots, \mathsf{UK}\}$$

$\mathcal{SHOIN}$ also provides for transitive roles, allowing us to state, e.g., that if y is an ancestor of x and z is an ancestor of y, then z is also an ancestor of x, and for inverse roles, allowing us to state, e.g., that if z is an ancestor of x, then x is also an descendent of z. The constructors provided by OWL, and the equivalent DL syntax, are summarised in Figure 2.

In DLs it is usual to separtate the set of statements that establish the vocabulary to be used in describing the domain (what we might think of as the schema) from the set of statements that describe some particular situation that instantiates the schema (what we might think of as data); the former is called the TBox (Terminology Box), and the latter the ABox (Assertion Box). An OWL ontology is simply equivalent to a set of $\mathcal{SHOIN}$ TBox and ABox statements. This mixing of schema and data is quite unusual (in fact ontologies are usually thought of as consisting only of the schema part), but does not affect the meaning—from a logical perspective, $\mathcal{SHOIN}$ KBs and OWL ontologies are just sets of axioms.

Constructor	DL Syntax	Example
`intersectionOf`	$C_1 \sqcap \ldots \sqcap C_n$	$\mathsf{Human} \sqcap \mathsf{Male}$
`unionOf`	$C_1 \sqcup \ldots \sqcup C_n$	$\mathsf{Doctor} \sqcup \mathsf{Lawyer}$
`complementOf`	$\neg C$	$\neg\mathsf{Male}$
`oneOf`	$\{x_1 \ldots x_n\}$	$\{\mathsf{john}, \mathsf{mary}\}$
`allValuesFrom`	$\forall P.C$	$\forall\mathsf{hasChild.Doctor}$
`someValuesFrom`	$\exists r.C$	$\exists\mathsf{hasChild.Lawyer}$
`hasValue`	$\exists r.\{x\}$	$\exists\mathsf{citizenOf}.\{\mathsf{USA}\}$
`minCardinality`	$(\geqslant n\ r)$	$(\geqslant 2\ \mathsf{hasChild})$
`maxCardinality`	$(\leqslant n\ r)$	$(\leqslant 1\ \mathsf{hasChild})$
`inverseOf`	r^-	$\mathsf{hasChild}^-$

Fig. 2. OWL constructors

```
<owl:Class>
  <owl:intersectionOf rdf:parseType=" collection">
    <owl:Class rdf:about="#Parent"/>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasChild"/>
      <owl:allValuesFrom>
        <owl:unionOf rdf:parseType=" collection">
          <owl:Class rdf:about="#Intelligent"/>
          <owl:Class rdf:about="#Athletic"/>
        </owl:unionOf>
      </owl:allValuesFrom>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```

Fig. 3. An examle of OWL's RDF syntax.

The main difference between OWL and $\mathcal{SHOIN}$ is that OWL ontologies use an RDF based syntax intended to facilitate their use in the context of the Semantic Web. This syntax is rather verbose, and not well suited for presentation to human beings. Figure 3, for example, illustrates how the description of happy parent given above would be written in OWL's RDF syntax.

3 Ontologies and Databases

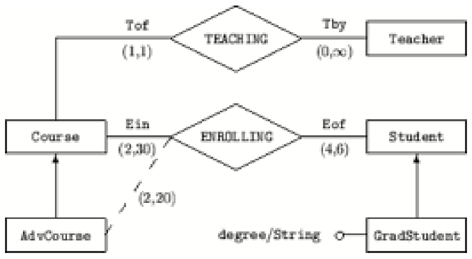


Fig. 4. An entity relationship schema.

Ontologies and ontology based information systems are closely related in both form and function to databases and database systems: both support the development of domain models that can be queried and updated. Figure 4, for example, illustrates a database entity relationship (ER) schema that models (a fragment of) a university domain in much the same way as the conceptual (TBox) part of an ontology, and (some of) the information it captures could be expressed using the following DL axioms:

$$\mathsf{Course} \sqsubseteq (\leqslant 1\,\mathsf{Tby}) \sqcap (\geqslant 1\,\mathsf{Tby}) \sqcap (\forall \mathsf{Tby}.\mathsf{Teacher}) \quad (\mathrm{A1})$$
$$\exists \mathsf{Tby}.\top \sqsubseteq \mathsf{Course} \quad (\mathrm{A2})$$
$$\mathsf{GradStudent} \sqsubseteq \mathsf{Student} \quad (\mathrm{A3})$$

where (A1) states that every course is taught by exactly one teacher, (A2) states that only courses can be taught (i.e., being taught by something implies being a course), and (A3) states that graduate students are a subclass of students.

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, it is stated that only courses can be taught; in OWL, if we know that IntroductionToAI is taught by Dave, this leads to the implication that IntroductionToAI is a Course—if we were to query the ontology for instances of Course, then IntroductionToAI would be part of the answer. In a database setting the schema is interpreted as a set of constraints on the data: adding the fact that IntroductionToAI is taught by Dave without IntroductionToAI being already *known* to be a Course 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, from axiom (A1) we know that a course can be taught by only one teacher, so additionally asserting that IntroductionToAI is taught by David would lead to the implication that Dave and David 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 Dave and David, then asserting that IntroductionToAI is taught by both Dave and David 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 IntroductionToAI is in the answer to a query for courses, 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 Dave is a Teacher, but if it is the case that anybody teaching a Course must be a Teacher. It does, however, make query answering *much* more difficult (at least in the worst case): in order to determine if Dave is in the answer to a query for teachers, it is necessary to check if Dave would be an instance of Teacher 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.

4 Ontology Reasoning

The design and implementation of reasoning systems is an important aspect of DL research and, as mentioned above, the availability of such reasoning systems was one of the motivations for basing OWL on a DL. This is because reasoning can be used in tools that support both the design of high quality ontologies, and the deployment of ontologies in applications.

4.1 Reasoning at design time

Ontologies are often large and complex: the well known SNOMED clinical terms ontology includes, for example, more than 400,000 class names [25]. 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. State of the art ontology development tools, such as SWOOP [14], Protégé 4 [17], and TopBraid Composer (see `http://www.topbraidcomposer.com/`), use a DL reasoner, such as FaCT++ [28], Racer [9] or Pellet [23], to provide feedback to the user about the logical implications of their design. This typically includes (at least) warnings about inconsistencies and redundancies.

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, where the axioms describing 2-dimensional and 3-dimensional objects ensured that these two classes were disjoint (had no instances in common). The inconsistency thus highlighted a fundamental error in the design of the ontology, discovering and repairing which obviously improved the quality of the ontology.

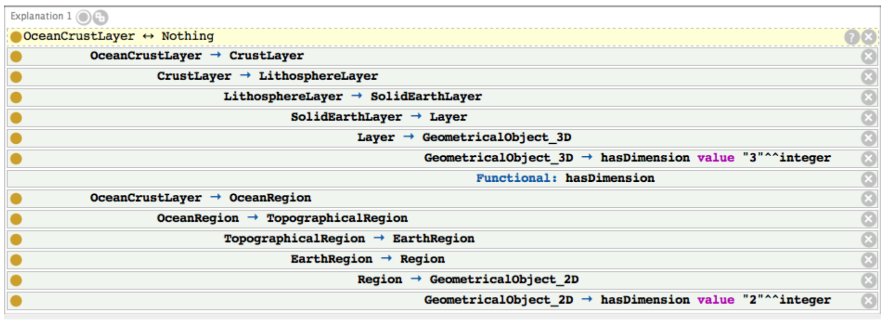
It is also possible that the descriptions in the ontology mean that two classes necessarily have exactly the same set of instances, i.e., that they are alternative names for the same class. This may be desirable in some situations, e.g., to capture the fact that "Myocardial infarction" and "Heart attack" mean the same thing. It could, however, also be the inadvertent result of interactions between descriptions, and so it is also useful to be able to alert users to the presence of such "synonyms". For example, when developing a medical terminology ontology a domain expert added the following two axioms:

$$\begin{aligned}\mathsf{AspirinTablet} &\equiv \exists \mathsf{hasForm}.\mathsf{Tablet}\\ \mathsf{AspirinTablet} &\sqsubseteq \mathsf{AspirinDrug}\end{aligned}$$

intending to capture the information that aspirin tablets are just those aspirin drugs that have the form of a tablet. Instead, these axioms had the effect of making *every* kind of tablet be an aspirin tablet. This was immediately corrected when the reasoner alerted the domain expert to the unexpected equivalence between Tablet and $\mathsf{AsprinTablet}$.

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 the expected subsumption relationship will alert the engineer to the missing DisjointClasses axiom.

Recent work has also shown how reasoning can be used to support modular design [4] and module extraction [3], important techniques for working with large ontologies. When developing a large ontology such as SNOMED, it is useful if not essential to divide the ontology into modules, e.g., to facilitate parallel work by a team of ontology developers. Reasoning techniques can be used to alert the developers to unanticipated and/or undesirable interactions between the various modules. Similarly, it may be desirable to extract from a large ontology a smaller module containing all the information relevant to some subset of the domain, e.g., heart disease—the resulting small(er) ontology will be easier for humans to understand and easier for applications to use. Reasoning can be used to compute a module that is as small as possible while still containing all the necessary information.



```
Explanation 1
OceanCrustLayer ↔ Nothing
        OceanCrustLayer → CrustLayer
                CrustLayer → LithosphereLayer
                        LithosphereLayer → SolidEarthLayer
                                SolidEarthLayer → Layer
                                        Layer → GeometricalObject_3D
                                                GeometricalObject_3D → hasDimension value "3"^^integer
                                                        Functional: hasDimension
        OceanCrustLayer → OceanRegion
                OceanRegion → TopographicalRegion
                        TopographicalRegion → EarthRegion
                                EarthRegion → Region
                                        Region → GeometricalObject_2D
                                                GeometricalObject_2D → hasDimension value "2"^^integer
```

Fig. 5. An explanation from Protégé 4

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 [15]. Figure 5, 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.

4.2 Reasoning in deployment

Reasoning is also important when ontologies are deployed in applications—it is needed, e.g., in order to answer structural queries about the domain and to retrieve data. For example, biologists use ontologies such as the Gene Ontology (GO) and the Biological Pathways Exchange ontology (BioPAX) to annotate 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.

It is easy to imagine that, with large ontologies, query answering 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. This is obviously of crucial importance when ontologies are used in safety critical applications such as medicine; it is, however, also important if ontology based systems are to be used as components in larger applications, such as the Semantic Web, where the correct functioning of automated processes may depend on their being able to (correctly) answer such queries.

5 Ontology Applications

The availability of tools and reasoning systems such as those mentioned in Section 3 has contributed to the increasingly widespread use of OWL. 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 above, the Foundational Model of Anatomy (FMA) [7] and the National Cancer Institute thesaurus [10].

Many ontologies are the result of collaborative efforts within a given community, and are developed with the aim of facilitating information sharing and exchange. Some ontologies are even commercially developed and subject to a licence fee. In most cases, an ontology focuses on a particular domain, although there are some well known "foundational" or "upper" ontologies, such as DOLCE[1] and SUMO,[2] whose coverage is more general; their aim is, however, mainly to provide a carefully formalised basis for the development of domain specific ontologies.

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,[3] that can be used to locate ontologies. In practice, however, applications are invariably built around a predetermined

[1] `http://www.loa-cnr.it/DOLCE.html`
[2] `http://www.ontologyportal.org/`
[3] `http://swoogle.umbc.edu/`

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 by recent work on the MED ontology in which potentially critical errors were discovered (see Section 1). 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 multi-billion pound "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:

$$\mathsf{AlmondAllergy} \equiv \mathsf{Allergy} \sqcap \exists \mathsf{causedBy}.\mathsf{Almond}$$

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,[4] 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[5] 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 [26]. 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 [1].

6 Discussion

Ontology, in its original philosophical sense, is a fundamental branch of metaphysics focusing on the study of existence; its objective is to determine what

[4] `http://sciencecommons.org/projects/data/`
[5] `http://www.obofoundry.org/`

entities and types of entities actually exist, and thus to study the structure of the world. In contrast, in computer science an ontology is an engineering artifact, usually a "conceptual model" of (some aspect of) the world, typically formalised as a logical theory.

Formalising ontologies using logic opens up the possibility of using automated reasoning to aid both their design and deployment. The availability of reasoning tools was an important factor in the design of the OWL ontology language and its basis in description logics and has contributed to the increasingly widespread use of OWL, not only in the Semantic Web per se, but as a popular language for ontology development in diverse application areas. The value of reasoning support has already been demonstrated in medical applications, where it has been used to help repair and enrich ontologies that play an important role in patient care.

The benefits of ontologies and ontology reasoning come at a cost, however. Even with the aid of reasoning enabled tools, developing and maintaining good quality ontologies is a difficult and costly task. Moreover, compared to more established database systems, query answering in an ontology based information system is likely to be much more difficult, and in the worst case could be highly intractable. The use of ontologies is, therefore, perhaps best 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.

Although reasoning enabled tools are a boon to ontology engineers, the design of good quality ontologies is still a difficult and time consuming task, and problems related to the availability of good quality ontologies threaten to limit the deployment of ontology based information systems. This problem can be tackled to some extent by extending the range and capability of ontology engineering tools, and this is a very active research area. However, tools alone cannot compensate for a lack of understanding of and expertise in "knowledge engineering". This has resulted in ontology engineers increasingly looking to the philosophy community for possible solutions, and in particular as a source of relevant expertise in the organisation and formalisation of knowledge. The results of such collaborations are already becoming evident in, for example, the design of foundational ontologies such as DOLCE and SUMO, and in ambitious ontology development projects such as the OBO foundry.

References

1. Batini, C., Lenzerini, M., Navathe, S.B.: A comparative analysis of methodologies for database schema integration. ACM Computing Surveys 18(4), 323–364 (1986)
2. Brachman, R.J., Schmolze, J.G.: An overview of the KL-ONE knowledge representation system. Cognitive Science 9(2), 171–216 (1985)
3. Cuenca Grau, B., Horrocks, I., Kazakov, Y., Sattler, U.: Just the right amount: Extracting modules from ontologies. In: Proc. of the Sixteenth International World Wide Web Conference (WWW 2007) (2007), `download/2007/CHKS07a.pdf`

4. Cuenca Grau, B., Kazakov, Y., Horrocks, I., Sattler, U.: A logical framework for modular integration of ontologies. In: Proc. of the 20th Int. Joint Conf. on Artificial Intelligence (IJCAI 2007). pp. 298–303 (2007), `download/2007/CKHS07a.pdf`
5. Derriere, S., Richard, A., Preite-Martinez, A.: An ontology of astronomical object types for the virtual observatory. Proc. of Special Session 3 of the 26th meeting of the IAU: Virtual Observatory in Action: New Science, New Technology, and Next Generation Facilities (2006)
6. Fensel, D., van Harmelen, F., Horrocks, I., McGuinness, D., Patel-Schneider, P.F.: OIL: An ontology infrastructure for the semantic web. IEEE Intelligent Systems 16(2), 38–45 (2001), `download/2001/IEEE-IS01.pdf`
7. Golbreich, C., Zhang, S., Bodenreider, O.: The foundational model of anatomy in OWL: Experience and perspectives. J. of Web Semantics 4(3), 181–195 (2006)
8. Goodwin, J.: Experiences of using OWL at the ordnance survey. In: Proc. of the First OWL Experiences and Directions Workshop. CEUR Workshop Proceedings, vol. 188. CEUR (`http://ceur-ws.org/`) (2005)
9. Haarslev, V., Möller, R.: RACER system description. In: Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2001). Lecture Notes in Artificial Intelligence, vol. 2083, pp. 701–705. Springer (2001)
10. Hartel, F.W., de Coronado, S., Dionne, R., Fragoso, G., Golbeck, J.: Modeling a description logic vocabulary for cancer research. Journal of Biomedical Informatics 38(2), 114–129 (2005)
11. Horrocks, I., Patel-Schneider, P.F., van Harmelen, F.: Reviewing the design of DAML+OIL: An ontology language for the semantic web. In: Proc. of the 18th Nat. Conf. on Artificial Intelligence (AAAI 2002). pp. 792–797. AAAI Press (2002), `download/2002/AAAI02IHorrocks.pdf`
12. Horrocks, I., Patel-Schneider, P.F., van Harmelen, F.: From $\mathcal{SHIQ}$ and RDF to OWL: The making of a web ontology language. J. of Web Semantics 1(1), 7–26 (2003), `download/2003/HoPH03a.pdf`
13. Horrocks, I., Sattler, U.: A tableaux decision procedure for $\mathcal{SHOIQ}$. In: Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005). pp. 448–453 (2005), `download/2005/HoSa05a.pdf`
14. Kalyanpur, A., Parsia, B., Sirin, E., Cuenca-Grau, B., Hendler, J.: SWOOP: a web ontology editing browser. J. of Web Semantics 4(2) (2005)
15. Kalyanpur, A., Parsia, B., Sirin, E., Hendler, J.: Debugging unsatisfiable classes in OWL ontologies. J. of Web Semantics 3(4), 243–366 (2005), `http://www.mindswap.org/papers/debugging-jws.pdf`
16. Kershenbaum, A., Fokoue, A., Patel, C., Welty, C., Schonberg, E., Cimino, J., Ma, L., Srinivas, K., Schloss, R., Murdock, J.W.: A view of OWL from the field: Use cases and experiences. In: Proc. of the Second OWL Experiences and Directions Workshop. CEUR Workshop Proceedings, vol. 216. CEUR (`http://ceur-ws.org/`) (2006)
17. Knublauch, H., Fergerson, R., Noy, N., Musen, M.: The Protégé OWL Plugin: An open development environment for semantic web applications. In: McIlraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) Proc. of the 3rd International Semantic Web Conference (ISWC 2004). Lecture Notes in Computer Science, vol. 3298, pp. 229–243. Springer (2004)
18. Lacy, L., Aviles, G., Fraser, K., Gerber, W., Mulvehill, A., Gaskill, R.: Experiences using OWL in military applications. In: Proc. of the First OWL Experiences and Directions Workshop. CEUR Workshop Proceedings, vol. 188. CEUR (`http://ceur-ws.org/`) (2005)

19. Patel-Schneider, P.F., Hayes, P., Horrocks, I.: OWL Web Ontology Language semantics and abstract syntax. W3C Recommendation (10 February 2004), `http://www.w3.org/TR/owl-semantics/`, available at `http://www.w3.org/TR/owl-semantics/`
20. Rector, A., Rogers, J.: Ontological and practical issues in using a description logic to represent medical concept systems: Experience from GALEN. In: Reasoning Web, Second International Summer School, Tutorial Lectures. LNCS, vol. 4126, pp. 197–231. SV (2006)
21. Ruttenberg, A., Rees, J., Luciano, J.: Experience using OWL DL for the exchange of biological pathway information. In: Proc. of the First OWL Experiences and Directions Workshop. CEUR Workshop Proceedings, vol. 188. CEUR (`http://ceur-ws.org/`) (2005)
22. Sidhu, A., Dillon, T., Chang, E., Sidhu, B.S.: Protein ontology development using OWL. In: Proc. of the First OWL Experiences and Directions Workshop. CEUR Workshop Proceedings, vol. 188. CEUR (`http://ceur-ws.org/`) (2005)
23. Sirin, E., Parsia, B., Cuenca Grau, B., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. J. of Web Semantics 5(2), 51–53 (2007), `http://www.mindswap.org/papers/PelletJWS.pdf`
24. Soergel, D., Lauser, B., Liang, A., Fisseha, F., Keizer, J., Katz, S.: Reengineering thesauri for new applications: The AGROVOC example. J. of Digital Information 4(4) (2004)
25. Spackman, K.: Managing clinical terminology hierarchies using algorithmic calculation of subsumption: Experience with SNOMED-RT. J. of the Amer. Med. Informatics Ass. (2000), fall Symposium Special Issue
26. Stevens, R., Baker, P., Bechhofer, S., Ng, G., Jacoby, A., Paton, N.W., Goble, C.A., Brass, A.: Tambis: Transparent access to multiple bioinformatics information sources. Bioinformatics 16(2), 184–186 (2000)
27. Semantic web for earth and environmental terminology (SWEET). Jet Propulsion Laboratory, California Institute of Technology (2006), `http://sweet.jpl.nasa.gov/`
28. Tsarkov, D., Horrocks, I.: FaCT++ description logic reasoner: System description. In: Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2006). Lecture Notes in Artificial Intelligence, vol. 4130, pp. 292–297. Springer (2006), `download/2006/TsHo06a.pdf`
29. Woods, W.A.: What's in a link: Foundations for semantic networks. In: Brachman, R.J., Levesque, H.J. (eds.) Readings in Knowledge Representation, pp. 217–241. Morgan Kaufmann Publishers, San Francisco, California (1985), previously published in D. G. Bobrow and A. M. Collins, editors, *Representation and Understanding: Studies in Cognitive Science*, pages 35-82. New York Academic Press, 1975.