OBO and OWL: Leveraging Semantic Web Technologies for the Life Sciences

Christine Golbreich[1], Matthew Horridge[2], Ian Horrocks[3], Boris Motik[3], and Rob Shearer[3]

[1]University of Versailles Saint-Quentin
55 avenue de Paris, 78035 Versailles, France
Christine.Golbreich@uvsq.fr

[2]School of Computer Science, University of Manchester
Oxford Road, Manchester, M13 9PL, UK
horridge@cs.man.ac.uk

[3]Oxford University Computing Laboratory
Wolfson Building, Parks Road, Oxford, OX1 3QD, UK
{Ian.Horrocks,Boris.Motik,Rob.Shearer}@comlab.ox.ac.uk

Abstract. OBO is an ontology language that has often been used for modeling ontologies in the life sciences. Its definition is relatively informal, so, in this paper, we provide a clear specification for OBO syntax and semantics via a mapping to OWL. This mapping also allows us to apply existing Semantic Web tools and techniques to OBO. We show that Semantic Web reasoners can be used to efficiently reason with OBO ontologies. Furthermore, we show that grounding the OBO language in formal semantics is useful for the ontology development process: using an OWL reasoner, we detected a likely modeling error in one OBO ontology.

1 Introduction

The Open Biomedical Ontologies (OBO) repository is a large library of ontologies from the biomedical domain hosted by the National Center for Biomedical Ontology (NCBO).[1] The majority of the ontologies in that repository are written in OBO Flat File Format[2]—an ontology language originally designed for the Gene Ontology (GO) [1]. This language (from now on called simply OBO) uses a simple textual syntax that was designed to be compact, readable by humans, and easy to parse. The OBO community has dedicated significant effort to developing tools such as OBO-Edit[3]—an integrated OBO editor and reasoner.

In parallel with the OBO effort, the Semantic Web community has developed the Web Ontology Language (OWL). Three dialects of the language have been defined, two of which are based on Description Logics—a well-understood family

[1] http://www.bioontology.org/repositories.html
[2] http://www.geneontology.org/GO.format.obo-1_2.shtml
[3] http://oboedit.org/

of knowledge representation formalisms with desirable computational properties. Formal semantics and the availability of efficient and provably correct reasoning tools have made the OWL DL dialect of OWL the language of choice for ontology development in fields as diverse as biology [22], medicine [5], geography [6], astronomy [2], geology,[4] agriculture [24], and defense [14]. Furthermore, OWL has been used to develop several large biomedical ontologies, such as the Biological Pathways Exchange (BioPAX) ontology [21], the GALEN ontology [20], the Foundational Model of Anatomy (FMA) [5], and the National Cancer Institute thesaurus [9]. Recently, the community of OWL users and developers proposed an extension of OWL called OWL 1.1 [18], which has been accepted as a member submission by the W3C. At the same time, a number of OWL-based tools have been developed, such as the Protégé [13] and SWOOP [11] editors, and the FaCT++ [25], RACER [8], and Pellet [23] reasoners.

In Section 2, we argue that there are many benefits in applying the tools and techniques of the Semantic Web to OBO. For example, Semantic Web reasoners could be used to provide guidance during ontology development; furthermore, modularization techniques for OWL [7] could simplify the reuse of existing OBO ontologies. This has been difficult up to now, however, since the OBO and Semantic Web communities have been largely disjoint.

To enable interoperability between OBO and Semantic Web tools and systems, we establish in Section 3 an exact relationship between OBO and OWL. This has not been straightforward, mainly because the OBO specification is quite informal. The syntax of the OBO language has not been formally specified, so our first step was to formalize the syntax of OBO itself; we discuss the results in Section 3.1. Likewise, there is no formal specification of OBO's semantics: the effects of different constructs have been described using natural language. We resolved ambiguities in these descriptions through extensive discussions with OBO developers. Hence, our mapping, presented in Section 3.2, formalizes the consensus interpretation in the OBO community. We also relate our mapping to several existing mappings from OBO to OWL.

In Section 4, we discuss how our mapping is used in practice. In Section 4.1 we discuss the technical aspects of our implementation. The complete replacement of OBO with OWL is not desirable for the OBO community, as many OBO users are familiar with both OBO-Edit and the OBO language, and find them convenient to use. Therefore, instead of simply implementing a translator from OBO to OWL, we have embedded support for OBO into existing Semantic Web ontology management infrastructure. In particular, we extended the well-known OWL API [10] with an OBO parser and serializer. All tools built on top of the OWL API can thus directly load, process, and save OBO ontologies. Moreover, tools such as OBO-Edit could use the new API to provide similar features, including direct access to OWL reasoners.

In Section 4.2, we show that reasoners implementing the formal semantics of OWL can derive subsumption inferences that are missed by OBO-Edit's rea-

[4] `http://sweet.jpl.nasa.gov/ontology/`

soner. In fact, on one of the OBO ontologies, our reasoner derived a new inference that highlights a probable modeling error.

Classifying large biomedical ontologies requires optimized reasoners. In Section 4.3, we show that OWL-based tools can be used to efficiently reason with OBO ontologies. To this end, we classified a number of OBO ontologies using the FaCT++ [25] and Pellet [23] systems, as well as the novel hypertableau-based reasoner HermiT [17].[5] The design of HermiT was motivated by an analysis of the structure of biomedical ontologies such as GALEN and NCI. Our results show that HermiT's improved handling of GALEN is applicable to OBO ontologies as well: on several ontologies containing complex cyclic definitions of terms, HermiT outperforms the other reasoners by orders of magnitude. Thus, our mapping allows the OBO community to benefit from current and future advances in reasoning technology while continuing to use their familiar ontology language and tools.

2 Why Map OBO to OWL 1.1?

2.1 OBO at a Glance

An OBO ontology is a collection of *stanzas*, each of which describes one element of the ontology. A stanza is introduced by a line containing a *stanza name* that identifies the type of element being described. The rest of the stanza consists of lines, each of which contains a *tag* followed by a colon, a *value*, and an optional comment introduced by "`!`".

The following is an example of an OBO stanza defining the *term* (the OBO equivalent of a class) `GO:0001555` with name `oocyte growth`. This term is a subclass of the term `GO:0016049`. (The comment tells us that `GO:0016049` is named `cell growth`.) Furthermore, `GO:0001555` has a `part_of` relationship to the term `GO:0048601` (which is named `oocyte morphogenesis`). Finally, `GO:0001555` is defined as an intersection of `GO:0040007` (`growth`) and of a relationship `has_central_participant` to `CL:0000023` (`oocyte`).

```
[Term]
id: GO:0001555
name: oocyte growth
is_a: GO:0016049 ! cell growth
relationship: part_of GO:0048601 ! oocyte morphogenesis
intersection_of: GO:0040007 ! growth
intersection_of: has_central_participant CL:0000023 ! oocyte
```

The following stanza defines the *relationship type* (the OBO equivalent of a property) `propreo:is_described_by`. The terms `propreo:chemical_entity` and `__Description177` are used as the domain and range, respectively, of the relationship type being defined.

[5] `http://www.cs.man.ac.uk/~bmotik/HermiT/`

```
[Typedef]
id: propreo:is_described_by
domain: propreo:chemical_entity
range: __Description177
```

Finally, the following stanza defines the *instance* (the OBO equivalent of an individual) `propreo:water_molecule`. The instance is a member of the term `propreo:inorganic_solvent_molecule` and has `propreo:CHEBI_15377` for the value of the relationship `propreo:is_described_by`.

```
[Instance]
id: propreo:water_molecule
instance_of: propreo:inorganic_solvent_molecule
property_value: propreo:is_described_by propreo:CHEBI_15377
```

2.2 Why Formalize OBO Syntax?

The line-oriented syntax of OBO makes parsing ontologies into stanzas and tag-value pairs straightforward. The tag values, however, usually have a structure that depends on the tag type. This structure is described in the OBO specification in natural language. For example, the structure of `intersection_of` tag values is described as follows:

> This tag indicates that this term represents the intersection of several other terms. The value is either a term id, or a relationship type id, a space, and a term id. [...]

This style of description is quite informal and it does not make the conceptual structure of the OBO language clear. For example, the above description does not provide any intuition behind the distinction between the two alternative structures of allowed values. Furthermore, the specification of the structure is conflated with low-level lexical issues, such as whitespace handling. As a result, neither aspect of the language is robustly addressed; for example, the treatment of escape characters is dependent on the structure of tag values. These issues make the implementation of an OBO parser quite difficult in practice.

2.3 Why Formalize OBO Semantics?

The semantics of OBO is also defined informally, by providing natural-language descriptions for different types of tag-value pairs. For example, the OBO specification defines the semantics of the `relationship` tag as follows:

> This tag describes a typed relationship between this term and another term. [...] The `necessary` modifier allows a relationship to be marked as "not necessarily true". [...]

Such a description is clearly ambiguous and informal. The notion of a relationship being "necessarily true" is completely undefined; in fact, the notion of a relationship has not been formalized either. Computational logic can be used to provide an unambiguous interpretation for such statements. For example, the `relation` tag from the stanza for the term `GO:0001555` from Section 2.1 can be interpreted in at least three different ways:

- *Existantial interpretation*: Each instance of the term `GO:0001555` must have at least one `part_of` relationship to an instance of the term `GO:0048601`. This reading corresponds to the DL axiom $\texttt{GO:0001555} \sqsubseteq \exists \texttt{part\_of}.\texttt{GO:0048601}$.
- *Universal interpretation*: Instances of `GO:0001555` can be connected through `part_of` relationships only to instances of `GO:0048601`. This reading corresponds to the DL axiom $\texttt{GO:0001555} \sqsubseteq \forall \texttt{part\_of}.\texttt{GO:0048601}$.
- *Constraint interpretation*: Instances of the term `GO:0001555` can be connected through `part_of` relationships; furthermore, the end-points of the relationship must be *known* to be instances of `GO:0048601`. Such a statement cannot be formalized in standard DLs; however, it can be expressed in various extensions of DLs [15, 3].

As another example, consider the natural-language explanation of the semantics for the `intersection_of` tag:

> [...] For example:
> `intersection_of: GO:00001`
> `intersection_of: part_of GO:00002`
> This means that the term is a subclass of any term that is both a subclass of `GO:00001` and has a `part_of` relationship to `GO:00002`. [...]

Here, it is not clear whether the defined term is equivalent to or a subclass of the intersection of the other terms. The textual description has a "procedural" flavor: it says that the defined term should be inferred to be a subclass of other terms, so one might conclude that the subclass relationship is the proper reading. Our discussions with the OBO developers, however, revealed that the intended interpretation is equivalence. The `union_of` tag suffers from analogous problems.

The description of OBO-Edit's reasoner provides an insight into the intended semantics of OBO. The OBO-Edit User's Guide[6] defines the following three reasoning rules:

1. For each transitive relationship R (such as `is_a` or `part_of`), whenever the ontology contains $a \rightarrow R \rightarrow b$ and $b \rightarrow R \rightarrow c$, an implied relationship $a \rightarrow R \rightarrow c$ is added.
2. For each term a defined as an intersection of terms b_1 and b_2, implied relationships $a \rightarrow \texttt{is\_a} \rightarrow b_1$ and $a \rightarrow \texttt{is\_a} \rightarrow b_2$ are added.
3. For each term a defined as an intersection of terms b_1 and b_2, whenever some term c has relationships $c \rightarrow \texttt{is\_a} \rightarrow b_1$ and $c \rightarrow \texttt{is\_a} \rightarrow b_2$, an implied relationship $c \rightarrow \texttt{is\_a} \rightarrow a$ is added.

[6] Available as part of the OBO-Edit distribution.

This definition is procedural, and it misses important inferences. Consider the following example:

```
[Term]                    [Term]
id: A                     id: B
relationship: R B         is_a: C
```

This simple OBO ontology says that `A` has an `R`-relationship to `B`, and that `B` is a subclass of `C`. Regardless of which of the three previously suggested interpretations for the `relationship` tag we choose, we should derive that `A` has an `R`-relationship to `C`; however, OBO-Edit's reasoning procedure does not derive that. Furthermore, the second and third inference rules clearly state that `intersection_of` is interpreted as equivalence, which may be in conflict with the natural-language description of the semantics.

To sum up, OBO suffers from problems very similar to those identified in semantic networks [19]. The DL family of ontology languages was developed precisely to address such problems—that is, to unambiguously specify the semantic properties of all ontology constructs. A mapping of OBO into OWL lends itself as an obvious way of providing formal semantics to OBO, and it allows for the application of sound and complete reasoning algorithms.

2.4 Why Use OWL 1.1?

In OBO, it is possible to make a property reflexive and/or (anti-)symmetric, as well as to say that one property is "transitive over" another: if P_1 is transitive over P_2, then for any individuals x, y, and z, the relationships $x \rightarrow P_1 \rightarrow y$ and $y \rightarrow P_2 \rightarrow z$ imply the relationship $x \rightarrow P_1 \rightarrow z$. Such axioms cannot be expressed in OWL DL; however, they can be expressed in the 1.1 extension of OWL. Thus, by using OWL 1.1 as the target language, we can capture a larger subset of OBO.[7] Since OWL 1.1 is fully backwards compatible with OWL, OBO ontologies that do not use any of the additional features of OWL 1.1 are mapped into OWL DL ontologies.

2.5 Reusing Existing Tools

An obvious practical benefit of a mapping from OBO to OWL is that it allows OBO users to exploit the multitude of existing OWL tools and services, instead of reimplementing the same functionality from scratch.

The foundation of many Semantic Web tools is provided by various APIs that provide means for the programmatic manipulation of ontologies. The OWL API [10] is a prominent example of such an API that is now very widely used. Recently, it has been completely reengineered and made compliant with the

[7] Our translation captures all of the OBO 1.2 specification except for cyclic properties (the semantics of which is not completely clear) and negative assertions about properties (e.g., the assertion that a property is *not* transitive).

OWL 1.1 version of the language. Jena[8] is a similar API that is comparable in its functionality with the OWL API and also has a large user base.

The OWL API has been used as the core data model for several ontology editors. For example, Protégé [13] is a well-known editor that can be used to edit OWL ontologies. Its newest incarnation, Protégé 4, supports all of OWL 1.1 and is based on the new OWL API. SWOOP [11] is another OWL editor that is based on the OWL API. These editors have been developed over years and are de facto standards for ontology editing. Furthermore, they are imbued with functionality that can provide guidance to the user during the ontology development and maintenance processes; for example, SWOOP supports ontology debugging [12] and modularization [7]. Finally, a number of plug-ins have been written for Protégé, implementing features such as UML-based ontology editing and graph-based ontology visualization.

Several highly optimized, sound, and complete reasoners for OWL exist. Pellet 1.4 [23] is built around the OWL API and is tightly integrated into SWOOP, and RacerPro [8], FaCT++ [25], and KAON2 [16] can be used with ontology editors through the Description Logics Implementors Group (DIG) protocol.[9] These reasoners can be used to classify an ontology and detect inconsistencies in class definitions, which is valuable during ontology development. Furthermore, reasoners can be used for query answering, which is the core inference underpinning many applications of OWL and OBO.

Apart from leveraging existing results, our integration allows the OBO community to reap the benefits of the future Semantic Web research. Conversely, OBO can provide to the OWL community significant user input as well as a growing library of OWL ontologies.

3 Providing a Formal Specification for OBO

In this section, we present a formal specification of the syntax and semantics of OBO. Due to space limitations, we highlight in this paper only the salient points; the full specification is available online.[10]

3.1 Formalization of OBO Syntax

We have formalized the OBO syntax by defining a BNF grammar, which maintains backward compatibility with the original OBO specification. Our grammar has been specifically designed to provide a conceptual description of the structure of OBO ontologies. To this end, it includes nonterminal symbols that identify and group certain OBO structures. Additionally, the grammar can be used to generate OBO parsers using automated tools.

For example, consider the definition of the `intersection_of` tag in BNF:

[8] `http://jena.sourceforge.net/`
[9] `http://dl.kr.org/dig/`
[10] `http://www.cs.man.ac.uk/~horrocks/obo/`

intersection := `intersection_of:` ***termOrRestr***
termOrRestr := ***term-id*** | ***restriction***
restriction := ***relationship-id term-id***

As explained in Section 2.2, the value of the `intersection_of` tag can be either a term, or a relationship type followed by a term. Our grammar introduces structure to such a "flat" definition as follows. We introduce a nonterminal ***term-id***, which denotes a "named" term (mimicking OWL's named classes), and a nonterminal ***restriction***, which denotes a "restricted term" (mimicking OWL's restriction classes). Then, we introduce the nonterminal ***termOrRestr*** (mimicking OWL's complex classes). Finally, we say that the value of the ***intersection*** tag is a ***termOrRestr*** (mimicking OWL's intersection classes that can contain arbitrary classes as conjuncts).

3.2 Mapping OBO to OWL

Our conceptualization of OBO's underlying model (described in Section 3.1) is quite similar to that of OWL, so the mapping between the two is relatively straightforward. We map OBO terms to OWL classes, OBO relationship types to OWL properties, and OBO instances to OWL individuals. The `id` assigned to each of these elements is used in OBO to uniquely identify each term, relationship type, and instance; hence, we use the value of `id` as the URI of the corresponding OWL element. The value of the `name` tag provides a human-readable description of OBO ontology elements, so it is translated into an OWL label.

Unlike OBO, OWL requires a strict separation between *object properties* (that relate individuals to each other) and *data properties* (that associate individuals with data values). To map OBO to OWL, we must infer which kind of property is appropriate for each OBO relationship type. If the range of a relationship type `R` is specified to be an XML datatype, or if `R` is asserted to be a subtype of another relationship type with such a range, then we translate `R` as an OWL datatype property; otherwise, we translate `R` as an object property.

OBO constructs such as `is_a`, `disjoint_from`, `domain`, and `range` have obvious equivalents in OWL and are translated in the straightforward manner. As discussed in Section 2.3, the official specification of the OBO language allows for several interpretations of the `intersection_of`, `union_of`, and `restriction` tags; hence, our mapping into OWL must pick the appropriate one.

Our discussions with OBO developers, as well as a survey of existing OBO ontologies, revealed that the existential interpretation (see Section 2.3) captures the intention behind the `relationship` tag. Hence, we translate the statement `relationship: R B` in a stanza defining the term `A` to the OWL axiom `SubClassOf(A ObjectSomeValuesFrom(R B))`.

Similarly, we concluded that the `intersection_of` tags should be interpreted as equivalences between classes (see Section 2.3). Furthermore, values for the `intersection_of` tag that consist of a relationship type and a term should be interpreted as existential constraints, just like `relationship` tags. Hence, we translate the statements `intersection_of: C` and `intersection_of: R D` in a stanza defining the term `A` to the following OWL axiom:

```
EntityAnnotation(OWLClass(GO_0001555) Label("oocyte_growth"))
SubClassOf(GO_0001555 GO_0016049) SubClassOf(GO_0001555
ObjectSomeValuesFrom(part_of GO_0048601))
EquivalentClasses(GO_0001555 ObjectIntersectionOf(
    GO_0040007 ObjectSomeValuesFrom(
        has_central_participant CL_0000023)))

ObjectPropertyDomain(propreo_is_described_by chemical_entity)
ObjectPropertyRange(propreo_is_described_by Description177)

ClassAssertion(propreo_water_molecule inorganic_solvent_molecule)
ObjectPropertyAssertion(
    is_described_by propreo_water_molecule CHEBI_15377)
```

Fig. 1. The OWL Interpretation of the Stanzas from Section 2.1

```
EquivalentClasses(A
    ObjectIntersectionOf(C ObjectSomeValuesFrom(R D)))
```

There was no consensus on the formal semantics of several OBO constructs, such as the `not_necessary` modifier to the `relationship` tag and the use of `relationship` tags in `Typedef` stanzas. In fact, these tags are likely to be deprecated in future releases of OBO, and are currently treated as annotations by our mapping.

Figure 1 shows the translation of the stanzas from Section 2.1.

Related Work. Other mappings between the OBO Flat File Format and OWL exist, and a summary can be found online.[11] None of these mappings are based on a formal analysis of OBO. The Common Mapping—a new version of a mapping originally developed by Chris Mungall—is defined via an XSLT style sheet,[12] and has been implemented as a Protégé plug-in. Common Mapping differs from our mapping in several important respects. For example, it reifies OBO annotation values and turns them into first-class elements of the interpretation domain subject to consistency checking and inferencing; in contrast, our translation simply encodes them as OWL 1.1 annotations. Common Mapping translates neither `reflexive` nor `transitive_over` tags, whereas our encoding preserves their semantics. Finally, for several OBO ontologies, Common Mapping produces OWL Full ontologies, which cannot be efficiently processed by existing Semantic Web tools. In contrast, our translation produces OWL 1.1 ontologies, which can be processed by existing tools without problems.

[11] `http://spreadsheets.google.com/ccc?key=pWN_4sBrd9l1Umn1LN8WuQQ`
[12] `http://www.godatabase.org/dev/xml/xsl/oboxml_to_owl.xsl`

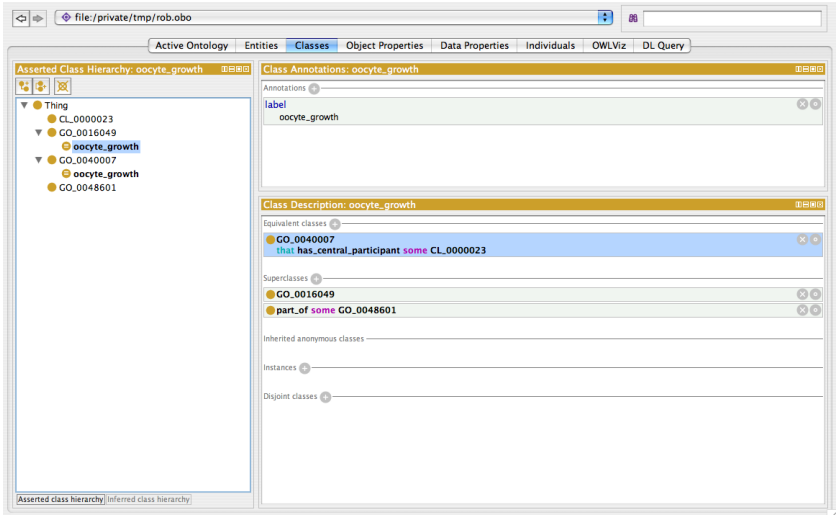

Fig. 2. A Screenshot of the Example From Section 2.1 Viewed in Protégé 4

4 Integrating OBO with the Semantic Web

4.1 Extending Semantic Web Tools to Support OBO

As mentioned in Section 1, simply replacing the OBO language with OWL would not be desirable for the OBO community—OBO users are familiar with the existing syntax of the language and the available tools, and want to continue to use them. Therefore, we adopted a less intrusive path of integration and have extended existing OWL tools with support for OBO.

As we discussed in Section 2.5, the OWL API lies at the core of many Semantic Web tools and supports fundamental tasks such as reading, saving, and manipulating ontologies. We extended the OWL API with an OBO parser and serializer, thus making OBO just another format supported by the API. This conveniently extends all applications based on the OWL API with support for OBO. For example, Protégé 4 can automatically read, edit, and save OBO ontologies; see Figure 2. Furthermore, the OWL API can be used to convert OBO files into OWL and vice versa by simply loading the file in one format and saving it in another. This functionality can be used to import OBO ontologies into tools that are not based on the OWL API and use custom OWL parsers.

The central new component in the API is an OBO parser, which consists of two distinct parts. The lower-level part is concerned with recognizing the syntax of OBO flat files, and it has been generated automatically from the BNF grammar described in Section 3.1. The upper-level part accepts different constructs from the OBO language and translates them into corresponding OWL 1.1 axioms according to the mapping described in Section 3.2.

4.2 Reasoning Support for OBO

An immediate benefit of our work is that it allows the application of Semantic Web reasoners to OBO ontologies. These reasoners are based on well-known algorithms with well-understood formal properties; furthermore, they provide formal guarantees about the completeness of reasoning, which makes the interpretation of derived results much easier. This can be quite useful in practice: on the OBO ontology `so.obo`, we used an OWL reasoner to detect a probable modeling error that is not detected by the OBO-Edit reasoner. This ontology contains the following stanzas that define the terms `SO:0000992` and `SO:0000914`:

```
[Term]
id: SO:0000992
name: BAC_cloned_genomic_insert
intersection_of: SO:0000914
```

```
[Term]
id: SO:0000914
name: cloned_genomic_insert
```

Note that the stanza for `SO:0000992` contains only one `intersection_of` tag. This seems to be a modeling error: presumably, the author simply forgot to add another `intersection_of` tag-value pair.

According to the mapping from Section 3.2, `intersection_of` defines a term to be equivalent to the intersection of other terms. Because the above intersection contains only one term, it effectively makes `SO:0000992` equivalent to `SO:0000914`. Indeed, OWL reasoners (correctly) derive that `SO:0000992` is a subclass of `SO:0000914` and vice versa. OBO-Edit's reasoner, however, only derives that `SO:0000992` is a subclass of `SO:0000914`, so the error remains undetected.

It is instructive to examine why OBO's reasoner does not derive the required inference. Namely, this inference could potentially be derived by applying the third inference rule from Section 2.3 on page 5 for $a = \texttt{SO:0000992}$ and $b_1 = b_2 = c = \texttt{SO:0000914}$; however, for the rule to be applicable, we would need an `is_a` relationship from `SO:0000914` to itself. Semantically, each class is a subclass of itself; this fact, however, is not represented explicitly in the OBO ontology model, so the mentioned inference rule is not applicable.

This error might have been detected by checking whether each stanza contains at least two `intersection_of` tags. Since the syntax of the OBO language, however, has not been formally specified, it is hard to implement a comprehensive set of such checks, so errors often fall thorough to the semantic level. Furthermore, if our example stanza contained two `intersection_of` tags with the same value, the ontology would be syntactically correct, but would imply the same consequence. The OBO-Edit reasoner does not derive all inferences even with respect to the informal semantics, so such an error would not be detected at the semantic level either. In contrast, the syntax and the semantics of OWL 1.1 have been formally specified, which makes the detection of errors easier.

4.3 Performance of Reasoning with OBO

Ontologies for the life sciences frequently contain many highly interconnected axioms with "cyclic definitions" of ontology terms. Such ontologies pose significant challenges to state-of-the-art tableau-based OWL reasoners [4, 17]. Hence,

Table 1. Performance of Reasoning with OBO Ontologies

Tools	No. of ontologies classified in						
	200 ms	1 s	5 s	25 s	53 s	163 s	3925 s
Pellet	2	13	36	51	59	64	79
FaCT++	25	58	72	77	78	80	80
HermiT	48	65	74	81	82	82	82

it is interesting to see whether the OBO ontologies can be effectively handled using modern Semantic Web reasoning tools. Our mapping would clearly be much less useful if OWL reasoners were unable to process OBO ontologies.

Therefore, we conducted several reasoning experiments using different OBO ontologies and tools. In particular, we measured the times needed to compute the subsumption hierarchy of a large set of OBO ontologies. We used the well-known reasoners Pellet and FaCT++, and a new reasoner HermiT. The latter reasoner employs novel reasoning algorithms based on hypertableau and hyperresolution [17]. The development of these algorithms has been motivated by an analysis of the structure of GALEN—the well-known biomedical terminology for which reasoning has proved to be quite hard. HermiT is currently the only reasoner that can classify the original version of GALEN [17].[13]

Although the reasoning algorithms from [17] support most of the OWL language, currently only the so-called Horn subset of OWL has been implemented in HermiT. Of the 88 ontologies available in the OBO repository, 83 fall into the supported fragment, so we used these ontologies in our performance tests. The ontologies are of varying sizes: the smallest one contains 166 axioms, whereas the largest one contains 37,943 axioms.

We summarize the times needed to classify these ontologies in Table 1. Because of the large number of ontologies, we do not present individual times for each ontology; instead, we just show how many ontologies each tool can classify within a certain time limit. The first four times were selected arbitrarily, whereas the last three times were chosen to show how long it takes for each tool to process the hardest ontology that it can classify.

Our results show that HermiT efficiently deals with all but one ontology—that is, it classifies them in at most 53 seconds. FaCT++ exhausts the available resources on two ontologies that HermiT can classify. Thus, HermitT's novel reasoning techniques seem to be critical in dealing with complex ontologies. In the future, similar advances are likely to follow. By defining OBO in terms of a mapping to OWL, the OBO community can reap the benefits of these advances while continuing to enjoy the existing OBO syntax and tool set.

[13] The original version of GALEN could not be processed by existing reasoners. Therefore, different versions of GALEN were derived from the original one by removing several cyclic definitions. Please refer to HermiT's web page for more information on this issue.

5 Conclusion

OBO is a language that has been extensively used for ontology modeling in the life sciences. Until now, the OBO language, as well as accompanying tools, have been developed independently from Semantic Web languages and tools. We argue that much can be gained from bringing the two languages closer together. On the one hand, this allows the OBO community to reuse OWL tools, while on the other hand, it provides new requirements and makes a large new corpus of ontologies available to the Semantic Web community.

The official specification of OBO is relatively informal. To obtain an unambiguous specification that can be easily implemented, we first formalized the syntax of the OBO language by capturing it in BNF. To capture the semantics, we developed a mapping between OBO and OWL. We have implemented this transformation in a new parser that we integrated into the OWL API, thus allowing numerous Semantic Web applications to use OBO as a native format. Finally, we showed that existing Semantic Web reasoners, such as HermiT, can be used to efficiently reason with OBO ontologies and can even identify likely modeling errors.

Acknowledgment

We thank Chris Mungall and John Day-Richter for their invaluable input on the definition and common usage of the OBO language.

References

1. M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, G. M. Rubin, and G. Sherlock. Gene Ontolgy: Tool for the Unification of Biology. *Nature Genetics*, 25(1):25–29, 2000.
2. S. Derriere, A. Richard, and A. Preite-Martinez. An Ontology of Astronomical Object Types for the Virtual Observatory. In *Proc. of the 26th meeting of the IAU: Virtual Observatory in Action: New Science, New Technology, and Next Generation Facilities*, pages 17–18, Prague, Czech Republic, August 21–22 2006.
3. F. M. Donini, D. Nardi, and R. Rosati. Description Logics of Minimal Knowledge and Negation as Failure. *ACM Transactions on Computational Logic*, 3(2):177–225, 2002.
4. T. Gardiner, I. Horrocks, and D. Tsarkov. Automated Benchmarking of Description Logic Reasoners. In *Proc. of DL 2006*, volume 189 of *CEUR*, 2006.
5. C. Golbreich, S. Zhang, and O. Bodenreider. The Foundational Model of Anatomy in OWL: Experience and Perspectives. *Journal of Web Semantics*, 4(3):181–195, 2006.
6. J. Goodwin. Experiences of using OWL at the Ordnance Survey. In *Proc. of OWL-ED 05*, volume 188 of *CEUR*, 2005.

7. B. Cuenca Grau, I. Horrocks, Y. Kazakov, and U. Sattler. Just the Right Amount: Extracting Modules from Ontologies. In *Proc. of WWW 2007*, pages 717–726, 2007.
8. V. Haarslev and R. Möller. RACER System Description. In *Proc. of IJCAR 2001*, pages 701–706, 2001.
9. F. W. Hartel, S. de Coronado, R. Dionne, G. Fragoso, and J. Golbeck. Modeling a Description Logic Vocabulary for Cancer Research. *Journal of Biomedical Informatics*, 38(2):114–129, 2005.
10. M. Horridge, S. Bechhofer, and O. Noppens. Igniting the OWL 1.1 Touch Paper: The OWL API. In *Proc. OWL-ED 2007*, volume 258 of *CEUR*, 2007.
11. A. Kalyanpur, B. Parsia, and J. Hendler. A Tool for Working with Web Ontologies. *International Journal on Semantic Web and Information Systems*, 1(1):36–49, 2005.
12. A. Kalyanpur, B. Parsia, E. Sirin, and J. Hendler. Debugging Unsatisfiable Classes in OWL Ontologies. *Journal of Web Semantics*, 3(4):243–366, 2005.
13. H. Knublauch, R. W. Fergerson, N. F. Noy, and M. A. Musen. The Protégé OWL Plugin: An Open Development Environment for Semantic Web Applications. In *Proc. of ISWC 2004*, pages 229–243, 2004.
14. L. Lacy, G. Aviles, K. Fraser, W. Gerber, A. Mulvehill, and R. Gaskill. Experiences Using OWL in Military Applications. In *Proc. of OWL-ED 05*, volume 188 of *CEUR*, 2005.
15. B. Motik, I. Horrocks, and U. Sattler. Bridging the Gap Between OWL and Relational Databases. In *Proc. of WWW 2007*, pages 807–816, 2007.
16. B. Motik and U. Sattler. A Comparison of Reasoning Techniques for Querying Large Description Logic ABoxes. In *Proc. of LPAR 2006*, pages 227–241, 2006.
17. B. Motik, R. Shearer, and I. Horrocks. Optimized Reasoning in Description Logics using Hypertableaux. In *Proc. of CADE-21*, volume 4603 of *LNAI*, pages 67–83. Springer, 2007.
18. P. F. Patel-Schneider and I. Horrocks. OWL 1.1 Web Ontology Language Overview. W3C Member Submission, December 19 2006.
Available at `http://www.w3.org/Submission/owl11-overview/`.
19. M. R. Quillian. Semantic Memory. In M. Minsky, editor, *Semantic Information Processing*, pages 216–270. MIT Press, Cambridge, MA, USA, 1968.
20. A. Rector and J. Rogers. 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*, pages 197–231, Lisbon, Portugal, September 4–8 2006.
21. A. Ruttenberg, J. Rees, and J. Luciano. Experience Using OWL DL for the Exchange of Biological Pathway Information. In *Proc. of OWL-ED 05*, volume 188 of *CEUR*, 2005.
22. A. Sidhu, T. Dillon, E. Chang, and B. Singh Sidhu. Protein Ontology Development using OWL. In *Proc. of OWL-ED 05*, volume 188 of *CEUR*, 2005.
23. E. Sirin and B. Parsia. Pellet: An OWL DL Reasoner. In *Proc. of DL 2004*, volume 104 of *CEUR*, 2004.
24. D. Soergel, B. Lauser, A. Liang, F. Fisseha, J. Keizer, and S. Katz. Reengineering Thesauri for New Applications: The AGROVOC Example. *Journal of Digital Information*, 4(4), 2004.
25. D. Tsarkov and I. Horrocks. FaCT++ Description Logic Reasoner: System Description. In *Proc. of IJCAR 2006*, pages 292–297, 2006.