OBO & OWL: Roundtrip Ontology Transformations

Syed Hamid Tirmizi[1], Stuart Aitken[2,3], Dilvan A. Moreira[4], Chris Mungall[5], Juan Sequeda[1], Nigam H. Shah[6], and Daniel P. Miranker[1,7]

[1]Department of Computer Science, the University of Texas at Austin
[2]Artificial Intelligence Applications Institute, the University of Edinburgh
[3]Informatics Life-Sciences Institute, the University of Edinburgh
[4]Department of Computer Science, Mathematics and Computing Institute, University of São Paulo
[5]Lawrence Berkeley National Laboratory
[6]Center for Biomedical Informatics Research, School of Medicine, Stanford University
[7]Institute for Cell and Molecular Biology, the University of Texas at Austin
hamid@cs.utexas.edu, stuart@inf.ed.ac.uk, dilvan@gmail.com, cjm@berkeleybop.org, jsequeda@cs.utexas.edu, nigam@stanford.edu, miranker@cs.utexas.edu

Abstract. The Open Biomedical Ontology (OBO) format emerged from the Gene Ontology, and now supports many other important ontologies. If we compare OBO to OWL, the ontology language of the Semantic Web, the latter anticipates integral query languages, rule languages and distributed infrastructure for information interchange. A convenient method for leveraging these other features for OBO ontologies is by transforming OBO ontologies to OWL. We have developed a methodology for translating OBO ontologies to OWL using the organization of the Semantic Web layer cake to guide the work. The approach reveals that the constructs of OBO can be grouped together to form a similar layer cake. Thus we were able to identify the constructs of OBO that have easy semantic equivalence to a construct in the OWL stack, and as well as those constructs that entail the challenges to a transformation system. As a result, we have developed a standard *common mapping* between OBO and OWL for the OBO community. Our mapping produces OWL-DL – a Description Logics based dialect of OWL with desirable computational properties for efficiency and correctness. Our Java implementation of the mapping is part of the official Gene Ontology project source. Our transformation system provides a lossless roundtrip mapping for OBO ontologies, i.e. an OBO ontology may be translated to OWL and back without loss of knowledge.

Keywords: Knowledge engineering methodologies, Knowledge Representation Formalisms and Methods, Ontology languages, Semantic Web.

1 Introduction

Two ontology based systems, the Open Biomedical Ontologies (OBO) and the Semantic Web, each associated with a large community are being developed indepen-

dently. Ontologies in biomedicine are used for cataloging biological concepts and representing relationships among them. Major results include the Gene Ontology (GO) [9] and the Zebrafish Anatomy Ontology (ZFA) [28]. OBO format, which originated with GO, continues to evolve in support of the needs of the biomedical community. Over 100 OBO ontologies are available on the NCBO Bioportal [17]. Thus OBO is the backbone for ontology tools in this domain.

The Semantic Web is an evolving extension of the World Wide Web based on ontologies, intended to facilitate search and information integration. Built on the foundations of artificial intelligence, the Semantic Web envisions the Web becoming a global knowledgebase through distributed development of ontologies using formally defined semantics, global identifiers and expressive languages for defining rules and queries on ontologies. The Semantic Web has been organized in the form of a layer cake where each layer provides a representation language of increasing expressive power (see Fig. 1). The Web Ontology Language (OWL), a component of the Semantic Web, provides the capability of expressing ontologies in multiple dialects. OWL-DL, a Description Logics based dialect, has become the language of choice due to the availability of reasoning tools. In the biomedical domain, some important ontologies such as NCI Thesaurus [19] and BioPAX [11] have been modeled in OWL.

Given the volume and growth of OBO content, integrating the features promised by Semantic Web technologies with OBO content would provide significant benefit to the biomedical community. One way to provide those features is to create a system that allows back and forth translation of OBO ontologies between the two systems.

This paper describes precisely such a round-trip and the methodology that was followed in the course of its creation. The results in this paper represent a community effort to create a standard transformation mapping, initiated by the OBO foundry. A goal was to reconcile a number of independent efforts. In addition to this paper, an early product of this collaboration is a Google spreadsheet [20], mediated by Nigam Shah that lists the transformation choices of the respective contributors and a mediated set of transforms, named the *common mapping*. Supplemental material on the mapping is also available [7]. The final results produce OWL-DL, as validated by WonderWeb OWL Ontology Validator [39]. A full implementation was done in Java, and is a part of the Gene Ontology project source [10], hosted at sourceforge.net. It provides a lossless roundtrip mapping for OBO ontologies, i.e. ontologies that are originally in OBO can be translated into OWL and back into OBO.

A basis for reconciling the efforts was an observation that the Semantic Web layer cake itself could serve as a guideline for studying the representation of ontologies in OBO and creating the transformation system. Compared to an approach that deals with each construct individually, we found that this method gave a better organization to our work and enabled us to identify matches and mismatches between the two languages more efficiently. We found that most of OBO can be decomposed into layers with direct correspondence to the Semantic Web layer cake. Discussions became a two step process where it was first determined if an OBO construct had a clear correspondance to a Semantic Web layer, with respect to its intended expressive power, and if so, to which level it belonged. It followed that constructs that fell into the same equivalence class should be handled similarly. Deep discussion could be limited to those OBO constructs that could not be easily situated in this structure. These include, (1) local identifiers in OBO compared to global identifiers in OWL, (2) various kinds

of synonym elements in OBO, and (3) defining subsets of OBO ontology. Even these constructs can be expressed in OWL-DL, albeit not by obvious construct substitution. We conclude that OWL-DL is strictly more expressive than OBO.

An additional consequence of this work is that, in effect, it defines a subset of OWL-DL that captures the expressive power of OBO and can be seen as a way of introducing formal semantics to OBO. We include a discussion of how OWL tools can be restricted to this subset so as to assure that ontologies developed with OWL tools may be translated to OBO. Similarly and perhaps more importantly, how to assure that OWL tools do not break OBO ontologies that have been translated to OWL such that, after using OWL tools, an updated ontology may be returned to OBO form. The exception handling in the Java based OWL to OBO translator was developed such that the translator itself serves double duty as a validator for this subset of OWL. At least two biomedical ontology tools, OBO-Edit [21] and Morphster [15] already exploit this translator.

Related Work: Each of the authors of this paper, as well as Mikel Egana, Erick Antezana, and LexBio group at Mayo Clinic, contributed some earlier independent effort at creating a transformation system. The resuls of these efforts are documented in our spreadsheet. No single effort survived in its entirety in the common mapping [20].

Another independent and important effort was that of Golbreich et al [2], [3] (hereafter Golbreich). Note that this group did not participate in the community effort to standardize the mapping. Golbreich developed a BNF grammar for OBO syntax, as well as a mapping between OBO and OWL 1.1 (now known as OWL 2). The differences between the Golbreich work and the common mapping effort presented in this paper comprise a difference of methodology and practical focus. Golbreich's work laid out valuable syntactic groundwork to formalize the semantics of a large subset of OBO. Much like most of the other first efforts, a complete transformation system was not specified. This particular effort deferred resolving OBO annotations, synonyms, subsets, and deprecation tags. Golbreich's work also did not address the mapping of local identifiers in OBO into global identifiers. The transformations that are specified by Golbreich are largely consistent with the common mappings. Given that OWL 2 is not yet ratified by the W3C, and therefore not yet in common use, we see Golbreich effort as corroborative rather than competitive.

2 Background

In knowledge-based systems, an ontology is a vocabulary of a set of concepts and the describable relationships among them [37]. Ontologies are extensively used in areas like artificial intelligence [13], [25], the Semantic Web [6] and biology [1], [9], [28] as a form of knowledge representation. They generally describe individual objects (or instances), classes of objects, attributes, relationship types, and relationships among classes and objects within a domain. Such ontologies are also called domain ontologies (or domain-specific ontologies).

A number of formal languages for writing ontologies exist, each having a different level of expressive power, inference capability, human readability, machine readability, and acceptance within their target domains.

The presence of domain ontologies and different languages and formats of representation makes the goal of having standardized large-scale and collaborative ontologies quite challenging. As a result, transformations between different languages of variable capabilities become important for merging pre-existing ontologies together and with newly created ones.

2.1 Open Biomedical Ontologies (OBO)

An ontology in OBO consists of two parts; the first part is the header that contains tag-value pairs describing the ontology, and the other part contains the domain knowledge described using *term* and *typedef* (more commonly known as a relationship type) stanzas [23]. A stanza generally defines a concept (term or typedef) and contains a set of tag-value pairs to describe it. The terms and typedefs defined in OBO ontology are assigned local IDs and namespaces. Relationships between different terms are expressed using the 'relationship' tag.

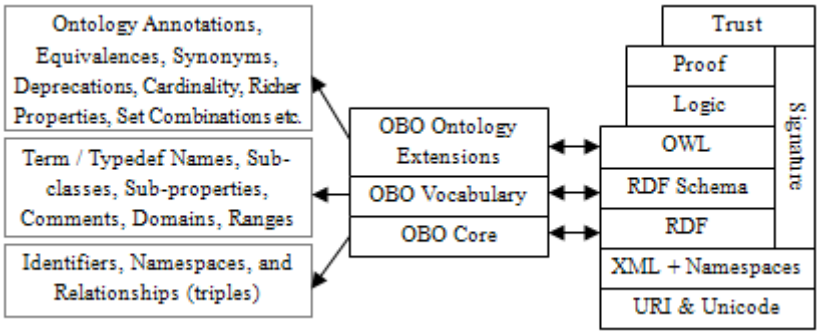
The OBO flat file format is human friendly. Therefore, it is easy for domain experts to understand it and express their knowledge in this language. Useful GUI-based tools like OBO-Edit are available for building ontologies in OBO [21].

As OBO continues to evolve as a language and hosted content, there is emphasis on formalizing the syntax and semantics of OBO format. Also, given the ongoing adoption of ontologies by the biomedical community and emerging new ontology building projects, OBO foundry has developed standard ontologies such as the Relations Ontology [1], which provide consistent and unambiguous formal definitions of the relationship types (or typedefs) used in such ontologies. While this effort is designed to assist developers and users in avoiding errors in ontology building, it also promises to simplify the process of ontology alignment in the future for the OBO community.

2.2 Semantic Web Technologies

The Semantic Web gives well-defined meaning to the content on the World Wide Web and enables computer and people to work in cooperation. Some key technologies that form the Semantic Web are:

1. Extensible Markup Language (XML) is a language that provides structure to documents by allowing user-defined markup elements. XML does not say anything about the meaning of the content.
2. Resource Description Framework (RDF) can express meaning of data using triples. A triple is a binary predicate that defines a relationship between two entities. RDF triples can be expressed using XML. The collection of XML elements for describing RDF is known as RDF/XML.



a) Examples b) OBO layers c) Semantic Web layers

Fig. 1. A layer cake for OBO, with some examples and a comparison with the Semantic Web layers; the mapping between the two layer cakes is generally quite straightforward, which makes it easy to understand the constructs in OBO and their mappings in OWL.

3. The Semantic Web uses Universal Resource Identifiers (URIs). This means that each entity gets a globally unique identifier that can be accessed by everyone on the Web.
4. RDF Schema (RDF-S) and Web Ontology Language (OWL) are ontology languages. RDF Schema allows description of valid classes and relationship types for an application, and some properties like subclasses, domains, ranges etc. OWL further allows describing constraints on instances and provides both ontology level and concept level annotations, set combinations, equivalences, cardinalities, deprecated content etc.

The Semantic Web is currently an active area in terms of research and development of tool support. Languages like SPARQL [31] are available for querying RDF-based knowledge sources. Other important technologies that are a part of Semantic Web vision are rule languages, inference and proofs etc.

RDF Schema and OWL are built on top of RDF. RDF/XML is a common syntax for these as well. In order to present mapping examples in this paper, we have chosen RDF/XML. On occasion, we use OWL as an encompassing term for these languages.

3 System Description

3.1 OBO and Semantic Web Layers

The Semantic Web was envisioned as an expressive hierarchy that is often illustrated as a layer cake [36] (see Fig. 1c). At the beginning of this research it was our conjecture that the precise organization of the hierarchy transcends the Semantic Web and could be used, retroactively, to formalize the structure of other data and concept modeling systems. Thus, as a first step towards the creation of a transformation mechan-

Table 1: Layer cake assignments for OBO constructs

OBO Core: id, idspace, relationship
OBO Vocabulary: name, definition, comment, is_a, domain, range
OBO Ontology Extensions: format-version, version, date, saved-by, auto-generated-by, namespace, default-namespace, subsetdef, alt_id, relationship, subset, synonym, is_obsolete, is_cyclic, is_transitive, is_symmetric, import, synonymtypedef, intersection_of, union_of, disjoint_from, replaced_by, consider, inverse_of, transitive_over

ism between OBO and OWL, we created a layer cake for OBO whose structure mirrored that of the Semantic Web layer cake. This allowed us to identify straightforward mappings as well as the cases that do not match as well. We term this the '*two layer cakes*' methodology. This methodology has also been successfully applied towards the transformation of SQL databases into OWL ontologies [34].

3.2 OBO Layer Cake

We methodically examined each of the constructs of OBO. We find that most of OBO can be decomposed into layers with direct correspondence to the Semantic Web: OBO Core, OBO Vocabulary, and OBO Ontology Extensions (see Fig. 1a, b).

1. **OBO Core:** In OBO, a concept can either be a term (class) or a typedef (relationship type). OBO Core deals with assigning IDs and ID spaces to concepts, and representing relationships as triples.
2. **OBO Vocabulary:** OBO Vocabulary allows annotating concepts with metadata like names and comments. It also supports describing sub-class and sub-property relationship types, as well as the domains and ranges of typedefs.
3. **OBO Ontology Extensions:** In addition to concept-level tags, OBO Ontology Extensions (OBO-OE) layer defines tags for expressing metadata on the entire ontology as well. It also allows defining synonyms, equivalences and deprecation of OBO concepts. OBO-OE layer can also express specific properties of OBO terms (e.g. set combinations, disjoints etc.), and typedefs (e.g. transitivity, uniqueness, symmetry, cardinalities).

Table 1 provides assignments of OBO constructs to appropriate layers in the OBO layer cake.

Since we mostly have an exact mapping of layers between the two languages (see Fig. 1), deciding which constructs to use for each kind of transformation is simplified. OBO Core tags can be transformed using RDF. OBO Vocabulary tags require using RDF Schema constructs. OBO Ontology Extensions tags require constructs defined in OWL.

3.3 Incompatibilities between OBO and OWL

We classify incompatibilities between the two languages into one of the two categories. First, in certain cases, the semantic equivalent of a construct in one language is missing from the other language. Second, sometimes the semantics of constructs in

OBO are not sufficiently well-defined to map to a formally defined OWL construct, which forces us to define new vocabulary in OWL in order to allow the lossless transformation.

1. Entities in OWL have globally unique identifiers (URIs). On the other hand, OBO allows local identifiers. Transforming OBO into OWL requires transforming the local identifiers in an OBO ontology into URIs. Also, in order to make the roundtrip possible, it is necessary to extract the local identifier back from the URI.
2. OBO language has the 'subset' construct, which does not have an equivalent construct in OWL. An OBO subset is a collection of terms, and is defined as a part of an ontology. An ontology can contain multiple subsets and each term can be a part of multiple subsets. In order to make the transformation possible, we need to define an OWL construct equivalent to OBO subset, and some relationship concepts to represent terms being in a subset, and a subset being a part of an ontology.
3. There are multiple kinds of synonym tags in OBO, e.g. related, narrow, broad, exact etc. The differences between these constructs are not formally documented. This requires defining new concepts in OWL, which can perhaps be mapped to new or already existing constructs in OWL.

Elements of OBO "missing" in Semantic Web are few, and can still be constructed in OWL. Thus, OBO ontologies may be translated to Semantic Web. However, in order to make the roundtrip possible, we find it important to store some ancillary information about the OBO ontology in the OWL file, e.g. a base URI etc., so it can be translated back without any loss of knowledge. It is important to note that even changing a local identifier within the whole knowledgebase is counted as loss of knowledge from the original source, even if the overall structure of the ontology remains intact.

The presence of such incompatibilities requires us to make some complex choices regarding the transformation process. Our solutions to these problems are explained in detail later.

3.4 OBO and Sublanguages of OWL

OWL has three increasingly expressive sublanguages; OWL Lite, OWL DL and OWL Full. Each of these sublanguages extends its simpler predecessor with richer constructs that affect the computational completeness and decidability of the ontology.
Our investigation shows that a major portion of OBO Ontology Extensions maps to OWL Lite and provides similar level of expressiveness. Overall, OBO features are a strict subset of OWL DL.

In OBO, the definition of a term, or a typedef, is rigid and not as expressive as OWL Full. OWL Full allows restrictions to be applied on the language elements themselves [8], [16]. In other words, an OWL Full Class can also be an OWL Full Property and an Instance and vice versa. Such features are not supported in OBO.

Recall, the primary concern is the use of the Semantic Web technology and tools for OBO ontologies. Thus, that OBO is less expressive than OWL is the convenient direction of containment. It does mean that round trips cannot be supported unless the

Table 2: Some OBO elements and their mappings in OWL. OBO examples in this table have been taken from ZFA.

OBO	OWL
[Typedef] id: part_of name: part of is_transitive: true	<owl:TransitiveProperty rdf:about="…#part_of"> <rdfs:label>part of</rdfs:label> </owl:TransitiveProperty>
Example A Simple transformations: name, transitivity	
[Term] id: ZFA:0000434 name: skeletal system is_a: ZFA:0001439	<owl:Class rdf:about="...#ZFA_0000434"> <rdfs:label>skeletal system</rdfs:label> <rdfs:subClassOf rdf:resource="...#ZFA_0001439"/> </owl:Class>
Example B Transformation of 'is-a'	
[Term] id: ZFA:0001439 name: anatomical system relationship: part_of ZFA:0001094	<owl:Class rdf:about= "…#ZFA_0001439"> <rdfs:label>anatomical system</rdfs:label> <rdfs:subClassOf><owl:Restriction> <owl:onProperty rdf:resource = "…#part_of" /> <owl:someValuesFrom rdf:resource = "…#ZFA_0001094" /> </owl:Restriction></rdfs:subClassOf> </owl:Class>
Example C Transformation of a relationship	
[Term] id: ZFA:0000437 name: stomach is_obsolete: true	<owl:Class rdf:about="&oboInOwl;ObsoleteClass"/> <owl:Class rdf:about="...#ZFA_0000437"> <rdfs:label>stomach</rdfs:label> <rdfs:subClassOf rdf:resource="&oboInOwl;ObsoleteClass"/> </owl:Class>
Example D Transformation of obsolete term	

editing of any OBO ontology while in OWL representation is restricted. We talk about editing of transformed ontologies while in OWL language in a later section.

While transforming OBO ontologies into OWL, we must ensure producing a representation that can be used by description logic based inference engines. One of the intended goals of our transformation is to produce OWL DL, and not OWL Full.

4 Transformation Metadata and Rules

In this section, we present some of the rules for the transformation of OBO ontologies into OWL. For more complex transformations we describe the transformations and explain our approach.

In order to facilitate the transformation, we have defined a set of OWL meta-classes that correspond to the vocabulary of OBO tags. Complete listing of mappings between OBO and OWL are available in a Google Spreadsheet [20].

4.1 Simple Transformation Rules

Most of the transformations follow simple rules. For most header and term/typedef tags, there is a one-to-one correspondence between OBO tags and OWL elements, either pre-existing or newly defined. In this section, we list the elements with this kind of simple transformation. Table 2 Example A provides some examples.

Header: The set of tag-value pairs at the start of an OBO file, before the definition of the first term or typedef, is the header of the ontology.

When translated into OWL language, each of the OBO header tags gets translated into the corresponding OWL markup element. The whole ontology header is contained in the *owl:Ontology* element in the new OWL file, and can appear anywhere within the file, as opposed to the start of file in OBO language.

Terms: A term in OBO is a class in OWL. So, a term declaration is translated into an *owl:Class* element and the tags associated with a term are contained within this element. Some tags that have straightforward transformations to OWL elements are:

1. The elements for 'name' and 'comment' about a term fall into the OBO Vocabulary layer, and are translated into *rdfs:label* and *rdfs:comment* respectively. A 'definition' tag is translated into *hasDefinition* annotation property, and is therefore placed in the OBO Ontology Extensions layer.
4. The 'is_a' tag in OBO specifies a subclass relationship, and is placed in the OBO Vocabulary layer. It is translated into an *rdfs:subClassOf* element (Table 2 Example B).

Typedefs: A typedef in OBO is an object property in OWL. A typedef stanza in an OBO file is translated into an *owl:ObjectProperty* element in OWL. The other information associated with the typedef is expressed as elements nested within this element. Some simple transformations are:

1. OBO typedefs can have associated domains and ranges. These are expressed by 'domain' and 'range' tags, and are in the OBO Vocabulary layer. These tags are translated into RDF Schema defined elements *rdfs:domain* and *rdfs:range* respectively.
2. Just like subclasses for terms, a property can be a sub-property to another property. A sub-property relationship is expressed using the 'is_a' tag, from OBO Vocabulary layer, in a typedef stanza. This tag is translated into an *rdfs:subPropertyOf* element defined in RDF Schema.
3. Typedefs may be cyclic ('is_cyclic' tag), transitive ('is_transitive' tag) or symmetric ('is_symmetric' tag). These tags fall into the OBO Ontology Extensions layer. The corresponding elements in OWL are annotation property *isCyclic*, and property types *owl:TransitiveProperty* and *owl:SymmetricProperty* respectively. The *isCyclic* property specifies a Boolean value.

4.2 Identifiers and ID Spaces

OBO has a local identifier scheme. As OBO evolves, ID spaces have been introduced to allow specifying global identifiers. OBO identifiers have no defined syntax, but they are recommended to be of the form:
"<IDSPACE>:<LOCALID>"

However, OBO ontologies may contain flat identifiers, ones that do not mention the ID space. OBO identifiers must be converted to URIs for use in OWL. The rules for converting OBO identifiers to URIs in the current mapping are as follows:

If the OBO header declares an ID space of the form: "*idspace: GO http://www.go.org/owl#*", all OBO identifiers with the prefix *GO:* will be mapped to the provided URI, e.g. "*http://www.go.org/owl#GO_0000001*".

If an OBO ID space prefix does not have a declaration in the header, all identifiers that mention that prefix will be transformed using a default base URI, for example an identifier of the form "*SO:0000001*" will become "*<default-base-uri>SO_0000001*". In case the OBO identifier is flat, e.g. *foo*, the transformation again uses the default base URI to create "*<default-base-uri>UNDEFINED_foo*". Notice that the URI contains "*UNDEFINED_*", which clarifies that the URI should be translated into a flat identifier when translating the OWL version back to OBO.

Recall that OBO Relations Ontology standardizes certain typedefs for use across OBO ontologies. Such typedefs have OBO identifiers prefixed with ID space *OBO_REL*. OBO ontology assumes the presence of this ID space with URI "*http://www.obofoundry.org/ro/ro.owl*" even if it is not explicitly stated. When translated into OWL, an XML namespace *xmlns:oboRel* with the same URI is added to the ontology, and the newly created object property is assigned that namespace. As a result, we ensure that all Relations Ontology constructs are mapped to the same URIs across ontologies.

4.3 Relationships

Relationships between OBO terms can be defined using the 'relationship' tag. A defined relationship is like a binary predicate and consists of a subject (the term being described in the stanza), a relationship type and an object.

There are multiple kinds of restrictions on relationships that can be expressed using OWL. OBO specifications do not specify any formal semantics of the 'relationship' tag that match a specific relationship type restriction defined in OWL. Therefore, we have selected the most general restriction to transform OBO relationships into OWL.

An example of relationship transformation is shown in Table 2 Example C. The *owl:someValuesFrom* element specifies the type of restriction that is applied to the OWL relationship. This restriction is similar to the existential quantifier of predicate logic [8], [16].

4.4 Subsets

Terms in an OBO ontology can be organized into subsets. A term can belong to multiple subsets.

In order to declare a subset, a value for the tag 'subsetdef' is specified in the OBO ontology header. This value consists of a subset ID (or subset name) and a quoted description about the subset. A term can be assigned to a defined subset using the 'subset' tag. Multiple 'subset' tags are used to assign the term to multiple subsets of the ontology.

Table 3: Results from the evaluation of our roundtrip transformation on some OBO ontologies.

Ontology*	Original OBO	OWL Translation**	Roundtrip OBO**
ZFA	Terms: 2219 Typedefs: 4	Classes: 2219 Object Properties: 4	Terms: 2219 Typedefs: 4
MA	Terms: 2882 Typedefs: 1	Classes: 2882 Object Properties: 1	Terms: 2882 Typedefs: 1
SPD	Terms: 494 Typedefs: 1	Classes: 494 Object Properties: 1	Terms: 494 Typedefs: 1
GO	Terms: 28667 Typedefs: 5	Classes: 28667 Object Properties: 5	Classes: 28667 Typedefs: 5

* ZFA = Zebrafish Anatomical Ontology, MA = Adult Mouse Gross Anatomy, SPD = Spider Ontology, GO: Gene Ontology

** Class counts do not include obsolete classes, or ancillary information required for roundtrips

When the ontology is translated into OWL, the mapping of subsets is one of the more complex processes. This is due to the fact that subsets do not have a semantic equivalent in OWL. Therefore, we use some OWL features to construct elements that serve as subsets. Subsets fall in the OBO Ontology Extensions in the OBO layer cake. The local ID (or name) assigned to the subset, which is locally unique, becomes the OWL ID of a subset resource. A subset resource is declared using an *oboInOwl:Subset* element. The *inSubset* annotation is used to assign terms to a subset, and it is expressed within the *owl:Class* element.

4.5 Obsolete Content

OBO format supports obsolete content. A term or typedef can be marked as obsolete using the 'is_obsolete' tag with a 'true' Boolean value. The 'is_obsolete' tag is in the OBO Ontology Extensions.

Obsolete terms and typedefs are not allowed to have any relationships with other terms or typedefs, including the subclass and sub-property relationships.

When translated into OWL, an obsolete term becomes a subclass of *oboInOwl:ObsoleteClass* (Table 2 Example D). Similarly, an obsolete typedef becomes a subproperty of *oboInOwl:ObsoleteProperty*.

Notice that while OWL provides elements to handle deprecation, obsoletion in OBO has different semantics, hence requires a different mapping.

5 Implementation and Evaluation

Based on the mapping rules, we have implemented a Java implementation of the transformation. Our implementation is part of the official Gene Ontology project source [10]. Gene Ontology project is an open source project on Sourceforge.net, and is home to the OBO ontology editor OBO-Edit. Our implementation is part of the OBO API that provides data structures for storing OBO ontologies, as well as read and write capabilities for OBO and OWL, among other operations. The source code

Table 4: Identifying the semantics for OBO constructs using OWL mappings.

Description	OBO	OWL	Semantics*
x is a subclass of y	is_a	rdfs:subClassOf	CEXT(x) ⊆ CEXT(y)
x is a subproperty of y	is_a	rdfs:subPropertyOf	EXT(x) ⊆ EXT(y)
x is the domain of property y	domain	rdfs:domain	<z,w> ∈ EXT(y) implies z ∈ CEXT(x)
x is the range of property y	range	rdfs:range	<w,z> ∈ EXT(y) implies z ∈ CEXT(x)
x is disjoint from y	disjoint_from	owl:disjointWith	CEXT(x) ∩ CEXT(y) = {}
p is a transitive property	is_transitive	owl:TransitiveProperty	<x,y>,<y,z> ∈ EXT(p) implies <x,z> ∈ EXT(p)

* CEXT(c): the set of instances of class c; EXT(p): the set of pairs <x,y> related by property p

for our transformation tool is available at [22]. Our tool is also used in Morphster [15], a knowledge acquisition tool for systematic biology that demonstrates the use of the Semantic Web technologies on OBO ontologies. We elaborate on this further in a later discussion on interconnecting OBO with the Semantic Web.

Finally, we have deployed our transformation as a web service for general use: *http://www.cs.utexas.edu/~hamid/oboowl.html*

In the OBO API, we have created *NCBOOboInOWLMetadataMapping* class in the package *org.obo.owl.datamodel.impl*. This class implements the roundtrip mapping between OBO and OWL. In order to provide console-based use of the transformation tool, we have created *Obo2Owl* and *Owl2Obo* classes in *org.obo.owl.test* package.

In order to evaluate the OWL output of our implementation, we have tested our tool on Gene Ontology, Zebrafish Anatomical Ontology, Spider Ontology and Adult Mouse Gross Anatomy, obtained from NCBO BioPortal. After transformation of these ontologies into OWL, we have successfully loaded the OWL files into Protégé [24], an ontology development tool for the Semantic Web. Using the 'summary' feature of Protégé, we have compared the overall class and object property count with the term and typedef count obtained for the original OBO file, using OBO-Edit's 'extended information' feature The results of the comparison (Table 3) show equal values for both versions of the ontologies. Similarly, for testing the roundtrip, we compared the original OBO file with the roundtrip version, again using OBO-Edit's feature. Our evaluation showed that the two OBO ontologies had the same term and typedef counts (Table 3).

6 Discussion: Implications of Transformation

6.1 OBO Semantics by Transformation

The transformation system has the additional effect of formalizing the semantics of the OBO language. The semantics of OBO are operationally defined by means of GO

and the software systems that support GO. The semantics of OWL have been formally defined using model theory [26], [27]. Though we have not written it out, a formal document specifying OBO semantics can be created, mechanically, from the contents of this paper and the OWL semantics documents. The contents of that document would comprise an enumeration of the pairwaise mapping of constructs between the two languages, restating, in each mapping, the semantics stated for the involved OWL construct.

In Table 4, we present a few examples where our transformation mapping could provide formal semantics for OBO constructs, taken directly from OWL semantics specifications. So,

1. *x is_a y*: all instances of *x* are also instances of *y*.
2. *x is domain of y*: the subject entity for all relationships of type *y* is an instance of *x*.
3. *x is disjoint from y*: *x* and *y* do not have any common instances.

While the identification is straightforward in these cases, in certain other situations, it is not very clear. Finding the semantics of relationships in OBO is one such case. OBO specifications do not provide the semantics of the construct used to specify relationships between two terms using a typedef. Therefore, it is hard to decide which of the available relationship constraints in OWL (*owl:allValuesFrom*, *owl:someValuesFrom*) to use, the former being similar to a universal quantifier, and the latter to an existential quantifier. In our transformations, we use *owl:someValuesFrom*, since already built ontologies show examples of use of OBO relationship construct in a way compatible to that of *owl:someValuesFrom*. We recommend that the semantics of relationships should always be defined to match *owl:someValuesFrom* restriction.

Other OBO tags that do not clearly match with OWL elements, such as synonyms and subsets, as well as the semantics for the 'obsolete' tag also present a more significant challenge in the identification of semantics.

6.2 Updating OBO Ontologies in OWL

The set of constructs for ontology representation provided by OWL is considerably larger than the set of constructs provided by OBO. Therefore, in order to allow roundtrip transformations on OBO ontologies, it is important to restrict the editing of such ontologies per some guidelines while they are being represented in OWL.

Our transformation mappings essentially provide a subset of OWL elements that may be used for adding or updating contents of the ontology. We refer to this subset of OWL as *OWL-Bio*, for biomedical ontologies hosted by OBO. Since our mapping produces OWL DL, OWL-Bio is a subset of OWL DL by definition.

Compared to the general use of OWL, there are two key points to keep in mind:

1. To create relationships, use *owl:someValuesFrom* relations. Since OBO does not have a corresponding relationship mechanism for *owl:alValuesFrom*, it is not a part of OWL-Bio.
2. Obsolescence of terms in the ontology should be done using the obsolete elements *oboInOwl:ObsoleteClass* and *oboInOwl:ObsoleteProperty* instead of built in deprecation elements in OWL.

6.3 Interconnecting OBO and Semantic Web

The implications of our work in providing semantics to OBO as well as in defining a "biomedical flavor" for OWL strongly suggest the use of this mapping as a potential bridge between the OBO and the Semantic Web worlds. Compared to the existing work by Golbreich et al. [2], our ability to make roundtrips between OBO and OWL-Bio could enable fluid interconnections between the two worlds. While OWL-Bio could serve as a common ground for the two languages, our roundtrip tool could be used as a validator for ontologies updated in OWL.

It is common for biologists to develop and refine their OBO ontologies as their work progresses. Our work provides a path for accessing and querying the Semantic Web as well as OBO content in an integrated fashion, and to assimilate linked data available on the Semantic Web.

The Morphster tool [15] exercises our rountrip transformation mechanism to jumpstart the integration of OBO ontologies with the Semantic Web. Morphster has successfully accomplished the use of a Semantic Web based triple store Jena SDB [30] for storage of large OBO ontologies and querying by the SPARQL query language for RDF. It also enables the use of XML Web Services with OBO ontologies to obtain and link diverse data such as images from Morphbank [14], and authoritative taxonomic names from uBio [38] etc.

7 Conclusion

Building ontologies is not a new idea for the biology community, and precedes the development of the Semantic Web. While ontologies are a central part of the architecture of the Semantic Web, the Semantic Web vision inclues a broad range of technologies from the Artificial Intelligence field, such as inference and querying mechanisms, as well as anticipating additional elements of distributed computation, such as global identifiers and the use of XML and HTTP as middleware. OBO, on the other hand, has appropriate tool support for building ontologies and hosts a number of important biomedical ontologies. Hence the OBO community has the biggest and most immediate need for the features being developed by the Semantic Web community.

We have standardized the mapping between the two systems to allow the OBO community to utilize the tool base developed for the Semantic Web world, and will also standardize the transformation across OBO tools. We have indirectly formalized the semantics of OBO by creating a roundtrip transformation between OBO and OWL. We have also implemented our transformation tool in Java and it is available as a part of open source Gene Ontology project, and also as a web service. We believe our work is an important step towards building interoperable knowledge bases between OBO and the Semantic Web communities.

A key difference between the OBO community and the Semantic Web is the methodology for content development across ontologies. The Semantic Web has adapted a completely distributed development mechanism for ontologies that may be integrated using URIs. On the other hand, the OBO community uses a hybrid of centralized and distributed development. While the users of OBO develop ontologies inde-

pendently, the OBO foundry has the goal of creating a suite of orthogonal interoperable reference ontologies, such as the Relations Ontology, in the biomedical domain. Our transformation system enriches the Semantic Web by providing this this additional structured ontology content and the access to the wealth of data annotated using it.

Acknowledgment

For developing part of the Java implementation of the transformation, we have used the PERL implementation by Erick Antezana [18] as a guide. Also, we thank Smriti Ramakrishnan for her help in the development and deployment of the OBO OWL transformation web service.

This research was supported by the National Science Foundation grant IIS-0531767, National Institutes of Health grant U54 HG004028-01, Biotechnology and Biological Sciences Research Council grant BB/F015976/1, and CAPES-Brazil.

References

[1] B. Smith, W. Ceusters, B. Klagges, J. Köhler, A. Kumar, J. Lomax, C. Mungall, F. Neuhaus, A.L. Rector and C. Rosse. Relations in Biomedical Ontologies. Genome Biology, 6(5):R46, 2005.

[2] C. Golbreich, M. Horridge, I. Horrocks, B. Motik and R. Shearer. OBO and OWL: Leveraging Semantic Web Technologies for the Life Sciences. ISWC/ASWC 2007: 169-182.

[3] C. Golbreich and I. Horrocks. The OBO to OWL mapping, GO to OWL 1.1! Workshop on OWL: Experiences and Directions, Innsbruck, Austria, Jun 6-7, 2007.

[4] C. Mungall. Mapping OBO to OWL. Berkeley Drosophila Genome Project. 2005. http://www.godatabase.org/dev/doc/mapping-obo-to-owl.html

[5] DAG-Edit User Guide. http://www.godatabase.org/dev/java/dagedit/docs/index.html

[6] D. Fensel, F. van Harmelen, I. Horrocks, D.L. McGuinness and P.F. Patel-Schneider. OIL: an ontology infrastructure for the Semantic Web. IEEE Intelligent Systems, 16(2):38-45, 2001.

[7] D. Moreira, C. Mungall, N. Shah, S. Aitken, J. Richter, T. Redmond and M. Musen. The NCBO OBOF to OWL Mapping. Available from Nature Precedings, 2009, http://hdl.handle.net/10101/npre.2009.3938.1

[8] D.L. McGuinness and F. van Harmelen, editors. OWL Web Ontology Language. W3C Recommendation, 10 Feb 2004. http://www.w3.org/TR/2004/REC-owl-features-20040210/

[9] Gene Ontology. http://www.geneontology.org/

[10] Gene Ontology at Sourceforge. https://sourceforge.net/projects/geneontology/

[11] G.D. Bader, M. Cary and C. Sander. BioPAX – Biological Pathway Data Exchange Format. Encyclopedia of Genomics, Proteomics and Bioinformatics, 2006, New York: John Wiley & Sons, Ltd.

[12] Jena – A Semantic Web Framework for Java. http://jena.sourceforge.net/

[13] K. Barker, B. Porter and P. Clark. A Library of Generic Concepts for Composing Knowledge Bases. First International Conference on Knowledge Capture, 2001.

[14] Morphbank – Biological Imaging. http://www.morphbank.net/

[15] Morphster AToL Project. http://www.morphster.org/

[16] M. Dean and G. Schreiber, editors. OWL Web Ontology Language Reference. W3C Recommendation, 10 Feb 2004. http://www.w3.org/TR/2004/REC-owl-ref-20040210/
[17] NCBO BioPortal. http://www.bioontology.org/bioportal.html
[18] NCBO Wiki OboInOwl. http://bioontology.org/wiki/index.php/OboInOwl:Main_Page
[19] NCI Thesaurus. http://ncit.nci.nih.gov/
[20] OBO2OWL Mappings. http://spreadsheets.google.com/ccc?key=0AqAZUdKw1IVHcFdOXzRzQnJkOWwxVW1uMUxOOFd1UVE&hl=en
[21] OBO-Edit, Gene Ontology Tools. http://www.geneontology.org/GO.tools.shtml
[22] OBO-Edit Source Wiki. http://wiki.geneontology.org/index.php/OBO-Edit:_Getting_the_Source_Code
[23] OBO Flat File Format Specifications. http://www.geneontology.org/GO.format.shtml
[24] Protégé Ontology Editor. http://protege.stanford.edu
[25] P. Clark and B. Porter. Building Concept Representations from Reusable Components. Fourteenth National Conference on Artificial Intelligence (AAAI), 1997.
[26] P. Hayes, editor. RDF Semantics. W3C Recommendation, 10 Feb 2004. http://www.w3.org/TR/2004/REC-rdf-mt-20040210/
[27] P.F. Patel-Schneider, P. Hayes and I. Horrocks, editors. OWL Web Ontology Language: Semantics and Abstract Syntax. W3C Recommendation, 20 Feb 2004. http://www.w3.org/TR/2004/REC-owl-semantics-20040210/
[28] P.M. Mabee, M.A. Haendel, G. Arratia, M.M. Coburn, E.J. Hilton, J.G. Lundberg and R.L. Mayden. ZFIN Anatomy Working Group: Skeletal System. Manually curated data, 2006.
[29] R. Page. Taxonomic Names, Metadata, and the Semantic Web. Biodiversity Informatics, 3, pp 1-15, 2006.
[30] SDB – A SPARQL Database for Jena. http://jena.sourceforge.net/SDB/
[31] SPARQL Query Language for RDF. 2006. W3C Candidate Recommendation. http://www.w3.org/TR/rdf-sparql-query/
[32] S. Aitken. A Minimal Ontology for OBO and GO. 2003. http://www.aiai.ed.ac.uk/resources/go/
[33] S.H. Tirmizi and D.P. Miranker. OBO2OWL: Roundtrip between OBO and OWL. The University of Texas at Austin, Department of Computer Sciences. Technical Report TR-06-47. October 2, 2006.
[34] S.H. Tirmizi, J. Sequeda and D.P. Miranker. Translating SQL Applications to the Semantic Web. DEXA 2008: 450-464.
[35] T. Berners-Lee, J. Hendler and O. Lassila. The Semantic Web. Scientific American, 284(5):34-43, May 2001.
[36] T. Berners-Lee. Semantic Web Status and Direction. International Semantic Web Conference, 2003 Keynote.
[37] T.R. Gruber. A Translation Approach to Portable Ontology Specification. Knowledge Acquisition 5: 199-220, 1993.
[38] Universal Biological Indexer and Organizer (uBio). http://www.ubio.org/
[39] WonderWeb OWL Validator http://www.mygrid.org.uk/OWL/Validator