OBO2OWL: Roundtrip between OBO and OWL

Syed Hamid Tirmizi, Daniel Miranker
Department of Computer Sciences
The University of Texas at Austin
{hamid, miranker} @ cs.utexas.edu

Abstract

Ontologies have traditionally been used in biomedicine for representing relationships among biological concepts, and as a result, large knowledge-bases like Gene Ontology (GO) have emerged. We believe use of Semantic Web technologies can allow better querying and collaboration of biomedical ontologies. As a migration path for biomedical ontologies, we have developed a mechanism for lossless roundtrip transformations between Open Biomedical Ontologies (OBO) format and OWL. We have methodically examined each of the constructs of OBO and mapped them to constructs in the Semantic Web stack. We have also enumerated constructs in each system that do not have simple syntactic equivalent in the other, important ones being GUIDs, various kinds of synonyms and subsets. We have implemented a tool that uses our transformation rules to translate OBO ontologies into OWL, and back, without loss of knowledge.

1 Introduction

Ontologies have traditionally been used in biomedicine for representing relationships among biological concepts. In recent years, the trend has been to harness the power of computers to build and query these structures. The results of this trend are large knowledge-bases like Gene Ontology (GO) [23, 24] and Zebrafish Anatomy [14], and the development of ontology representation formats (like OBO [19, 25]) and corresponding tools specifically for the biomedical domain.

Even though the idea of Semantic Web is in its early stages, it can prove to be a useful tool for the area of biomedicine, and can provide important features that will allow better querying and collaboration of biomedical ontologies. However, in order to do such a migration, it is important to understand and solve two major hurdles; (1) automatic translation of existing ontologies into Semantic Web, and (2) providing a way of using both pre-existing OBO tools and new Semantic Web tools for editing translated ontologies.

As a migration path for biomedical ontologies, we have developed a mechanism for lossless roundtrip transformations between OBO and OWL. We have methodically examined each of the constructs of OBO and mapped them to constructs in the Semantic Web stack. We find that most of OBO can be decomposed into layers with direct correspondence to the Semantic Web layer cake (see Figure 1). In the process we have enumerated constructs in each system that do not have a simple syntactic equivalent in the other, and have created new elements for them. Some major features that are incompatible are:

- The lack of globally unique identifiers (GUIDs) in OBO, which are one of the building blocks of Semantic Web technologies, and are essential for creating large scale and collaborative sources of knowledge.
- Presence of various kinds of synonym elements in OBO, perhaps emerging from biomedical domain, which are not clearly present in the Semantic Web languages like OWL.
- Availability of an element for defining multiple subsets of an ontology within its OBO representation. This feature is not available in OWL.

We have implemented a tool that uses our translation rules to transform an OBO ontology into OWL, and back, without loss of knowledge.

Our application context is the NSF's "Assembling the Tree of Life" (AToL) grand challenge. The grand challenge faced is in describing 5 to 10 million extant species, and computing and analyzing a unified phylogenetic tree. The effort spans organisms as far ranging as bacteria, plants and mammals. Numerous projects, organized around a particular group, e.g. fish, are organizing the terminology of their corpuses as ontologies and working to exploit Internet technology to tie this information together and make it highly available. A goal of our project, Morphster, includes image annotation. In our context, images are used to document the precise meaning or a biological concept. Thus, in our ontologies, an image, or parts of an image will become part of a concept definition. We anticipate both drawing concept labels from existing ontologies, [14], and adding new concepts to ontologies through image-based definitions.

The remainder of this document is organized as follows. Section 2 provides background information on key technologies under consideration. Section 3 explains the correspondence between OBO and Semantic Web feature using layer cakes, and goes into the details of similarities and differences between the two languages, OBO and OWL. Section 4 gives detailed listing and explanation of transformation rules, Section 5 provides a summary of implementation methodology, Section 6 gives some examples that explain transformations between OBO and OWL, Section 7 provides guidelines for editing the OWL produced by the tool, so that roundtrip transformation remains possible, Section 8 discusses related work, Section 9 talks about conclusions and possible directions for future work. Last two sections, Section 10 and 11, provide acknowledgements and references.

2 Background

Philosophically, ontology is the study of existence. For knowledge-based systems, it is a vocabulary of a set of objects and the describable relationships among them [11].

Ontologies are extensively used in areas like artificial intelligence [3, 6], Semantic Web [10, 12] and biology [20] 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 formats of representation makes the goal of having standardized large-scale and collaborative ontologies quite challenging [9, 15]. As a result, transformations between different languages of variable capabilities become important for merging pre-existing ontologies together and with newly created knowledge.

2.1 Ontologies in the Biomedical Domain

Building ontologies is a commonplace exercise in the biomedical domain. Biologists build ontologies to demonstrate relationships between different biological concepts like species, taxa, systems etc. Some examples are phylogenetic relationships among taxa (or species) and anatomical relationships among different parts of a skeleton (commonly known as a Nomina Anatomica).

2.2 Open Biomedical Ontologies (OBO) Flat File Format

The OBO flat file format is the most commonly used ontology representation language in the biomedical domain. OBO emerged from the Gene Ontology effort, and is now host to over 60 different ontologies [23, 24].

An ontology in OBO format consists of two parts; the first part contains the header tags and values, and the other part contains the domain knowledge described using term and typedef stanzas. 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 [19, 25].

The OBO flat file format is very human friendly. Therefore, it is easy for domain experts to understand it and express their knowledge in this format. Efforts are being made, especially in the biomedical community, for building useful GUI-based tool support for OBO format [24].

A drawback of OBO is the lack of globally unique identifiers, which makes it difficult to integrate different knowledge sources together. Also, OBO lacks adequate query and rule language support which is necessary to utilize the full potential of large knowledge-bases.

2.3 Semantic Web Technologies

The Semantic Web is an extension of the current World Wide Web that gives well-defined meaning to the content and enables computer and people to work in cooperation [4]. Some key technologies for developing the Semantic Web are described below.

- **Extensible Markup Language** (XML) is a language that provides arbitrary structure to documents by allowing user-defined markup tags. These tags, however, do not say anything about the meaning of the content.
- **Resource Description Framework** (RDF) is used to express meaning of data using triples. A triple is like a binary predicate and defines a relationship between two entities. RDF triples can be expressed using XML. The collection of XML tags for describing RDF is known as RDF/XML [1].
- Entities, either classes or relationship types, in the Semantic Web are identified using **Universal Resource Identifiers** (URIs). This means that each entity gets a globally unique identifier which can be accessed by everyone on the Web. Entities can be classified into groups using **XML namespaces**.
- Ontologies are an important component of the Semantic Web. Technologies like **RDF Schema** (RDF-S) and **Web Ontology Language** (OWL) are used for describing ontologies. RDF Schema allows description of valid classes and relationship types for an application, and some properties like subclasses, domains, ranges etc. OWL allows describing richer content and provides both ontology level and concept level annotations, set combinations, equivalences, cardinalities, deprecated content etc.

Semantic Web is currently an active area in terms of research and development of tool support. Languages like SPARQL are available for querying RDF-based knowledge sources [22]. 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. Therefore, RDF/XML is a common syntax for these as well. Since any XML is inherently hard to read for humans, choosing RDF/XML as a primary syntax (and hence Semantic Web as the primary technology) becomes hard for domain specific ontology builders. In addition, RDF/XML has high storage costs, is slower to parse than most XML, and is incompatible with currently available XML processing technologies like XSLT etc.

For the rest of our work, we assume RDF/XML to be the format for Semantic Web technologies (RDF, RDF Schema and OWL), and on occasion, we use OWL as an encompassing term for Semantic Web languages.

3 OBO and Semantic Web Layer Cake

The Semantic Web was envisioned in the form of a layer cake [5] and apparently most research so far has followed that roadmap.

In order to create a transformation mechanism between OBO flat files and Semantic Web technologies, we find it useful to create a layer cake for OBO, similar to that of the Semantic Web.

3.1 OBO Layer Cake

We have methodically examined each of the constructs of OBO and mapped them to constructs in the Semantic Web stack. We find that most of OBO can be decomposed into layers with direct correspondence to the Semantic Web layer cake. In the process we have enumerated constructs in each system that do not have a simple syntactic equivalent in the other. Elements of OBO "missing" in the semantic web are few, and can still be expressed in OWL. Thus, OBO ontologies may be translated to the Semantic Web. Further, we believe if certain ancillary information is retained during translation, the Semantic Web representation may be translated back to OBO, and the cycle repeated without any loss of knowledge.

OBO tags can be partitioned into three layers – OBO Core, OBO Vocabulary, and OBO Ontology Extensions (see Figure 1a, b).

- **OBO Core:** In OBO terminology, a concept can either be a term (class) or a typedef (relationship type). OBO Core deals with assigning IDs and namespaces to concepts, and representing some knowledge about those concepts using relationships; essentially triples.
- **OBO Vocabulary:** OBO Vocabulary allows annotating concepts with information like names, definitions and comments. In addition, it supports describing sub-class and sub-property relationships, as well as the domains and ranges of typedefs.
- **OBO Ontology Extensions:** In contrast to the previous two layers, which define tags with concept-level scope only, 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. Using OBO Ontology Extensions, we can also express specific properties of OBO terms (e.g. set combinations, disjoints etc.), and typedefs (e.g. transitivity, uniqueness, symmetry, cardinalities...).

3.2 Correspondence between OBO and Semantic Web Cakes

In our work, we define a set of transformation rules for converting OBO files to OWL. Since we have a mostly exact mapping of layers between the two formats (see Figure 1c), deciding which constructs to use for each kind of transformation becomes a lot easier. In other words, OBO Core tags can be transformed using RDF, OBO Vocabulary tags require using RDF Schema constructs, and OBO Ontology Extensions tags require us to use constructs defined in OWL.

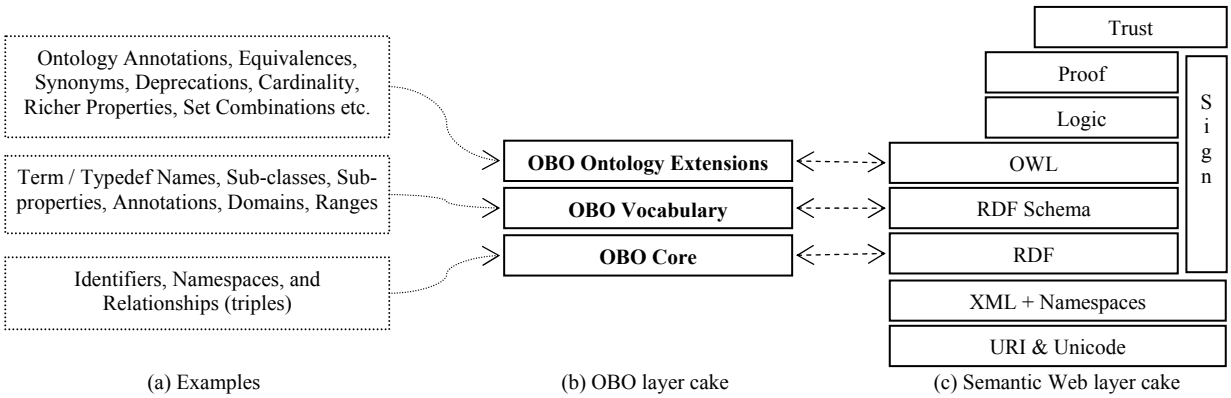

Figure 1: A layer cake for OBO, with some examples and a comparison with Semantic Web layers.

3.3 Incompatibilities between OBO and OWL

We classify incompatibilities between the two formats into one of the two categories. First, in certain cases, the semantic equivalent of a construct in one format is missing from the other format. Second, sometimes the semantics of constructs in OBO are not well-defined or documented, which forces us to define new equivalent constructs in OWL in order to allow the lossless transformation. Major examples of incompatibilities are the following:

- Entities in OWL are identified using URIs. However, OBO has very simple local identifier scheme. Transforming OBO into OWL requires transforming the OBO identifiers into globally unique IDs (GUIDs). Also, in order to make the roundtrip possible, it is necessary to extract the local identifier from the GUID.
- OBO format has the 'subset' construct, which does not have any 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.
- There are multiple kinds of synonym tags in OBO, as well as a way of expressing synonym relationship or equivalence using external database references (dbxrefs). The differences between these constructs and their usage are not very well documented. This requires defining new concepts in OWL which can later be mapped to new or already existing constructs in OWL.

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 in the document.

3.4 OBO and Different Flavors of OWL

Our investigation shows that a major portion of OBO Ontology Extensions maps to OWL Lite and provides similar level of expressiveness. Overall, OBO features match well with 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 migration of legacy OBO ontologies and their constituencies to the Semantic Web. Thus, that OBO is less expressive than OWL Full is the convenient direction of containment. It does mean that round trips can not be supported unless the editing of an OBO ontology while in OWL representation is restricted. We talk about editing in OWL format in the later sections.

4 Transformation Metadata and Rules

In this section, we list the rules for transformation of OBO ontology header and content into OWL, additional OWL tags defined for transformation and their usage, and for more complex transformations we describe the transformations and explain our approach.

4.1 Meta-OBO Tags for Semantic Web

In order to facilitate the transformation of OBO to OWL format, we define a set of OWL classes, object properties and annotation properties that correspond to OBO tags (see Table 1 and Table 2).

Tag	Type	Comments
`mo:name`	Annotation Property	Name of a term or typedef
`mo:comment`	Annotation Property	Comments about a term or typedef
`mo:formatVersion`	Annotation Property	Version of OBO format in which the ontology was originally written
`mo:dataVersion`	Annotation Property	Version of data in the ontology
`mo:savedDateTime`	Annotation Property	Date/time of last edit/save
`mo:savedBy`	Annotation Property	User who last edited/saved the ontology
`mo:autoGeneratedBy`	Annotation Property	The program used to generate the ontology
`mo:remark`	Annotation Property	General comments on the ontology
`mo:definedNamespace`	Object Property	A namespace defined for use in the ontology
`mo:defaultNamespace`	Object Property	Default namespace for the ontology
`mo:Subset`	Class	Subset of OBO terms from the ontology
`mo:definedSubset`	Object Property	Subset defined for the ontology

Table 1: Meta-OBO tags for header elements with types and definitions

4.2 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 defined as Meta-OBO tags. In this section, we list the elements with this kind of simple transformation.

Tag	Type	Comments
mo:inSubset	Object Property	Declares a term or typedef to be in a subset
mo:alternateID	Annotation Property	Alternate ID for a term or typedef
mo:definition	Annotation Property	Definition of the term or typedef
mo:xrefAnalogous	Annotation Property	Analogous external reference
mo:xrefUnknown	Annotation Property	External reference of unknown type
mo:synonym	Annotation Property	Synonym to a term or typedef
mo:relatedSynonym	Annotation Property	Related synonym to a term or typedef
mo:exactSynonym	Annotation Property	Exact synonym to a term or typedef
mo:broadSynonym	Annotation Property	Broad synonym to a term or typedef
mo:narrowSynonym	Annotation Property	Narrow synonym to a term or typedef
mo:isCyclic	Annotation Property	Cyclic property of a typedef
mo:isSymmetric	Annotation Property	Symmetric property of a typedef
mo:isTransitive	Annotation Property	Transitive property of a typedef

Table 2: Meta-OBO tags for terms and typedefs

4.2.1 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. The tags shown in Table 3 are common OBO header tags.

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

4.2.2 Terms

The concept of a term in OBO is similar to an OWL class. So, a term is translated into an `owl:Class` element (Table 5), and the tags associated with a term are contained within this element.

OBO terms and typedefs have a lot of metadata tags in common, shown in Table 4. Some other tags that are specific to OBO terms are shown in Table 5. Most of these tags have straightforward transformations to OWL elements.

Some important transformations among these are:

- The name and comments about a term are translated into `mo:name` and `mo:comment` elements. These elements are defined to be equivalent to RDF Schema elements `rdfs:label` and `rdfs:comment` respectively.
- The subclass relationship, defined using the 'is_a' tag in OBO, is translated into `rdfs:subClassOf` element defined in RDF Schema.

In some cases, however, like transformations of OBO's local IDs into globally unique OWL ID's, namespaces and deprecated content, more complex transformations are required. Each of these cases is explained in a dedicated section of this document.

OBO Tag	OWL Markup
format-version	`<mo:formatVersion> … </mo:formatVersion>`
typeref	`<owl:imports> … <owl:imports>`
version	`<mo:dataVersion> … </mo:dataVersion>`
date	`<mo:savedDateTime> … </mo:savedDateTime>`
saved-by	`<mo:savedBy> … </mo:savedBy>`
auto-generated-by	`<mo:autoGeneratedBy> … </mo:autoGeneratedBy>`
default-namespace	`<mo:defaultNamespace> … </mo:defaultNamespace>`
remark	`<mo:remark> … </mo:remark>`

Table 3: OBO tags and corresponding OWL elements for header section

OBO Tag	OWL Markup
id / namespace	`rdf:ID` (See more on identifiers and namespaces in section 4.3)
name	`<mo:name> … </mo:name>`
alt_id	`<mo:alternateID> … </mo:alternateID>`
def	`<mo:definition> … </mo:definition>`
comment	`<mo:comment> … </mo:comment>`
synonym	`<mo:synonym> … </mo:synonym>`
related_synonym	`<mo:relatedSynonym> … </mo:relatedSynonym>`
exact_synonym	`<mo:exactSynonym> … </mo:exactSynonym>`
broad_synonym	`<mo:broadSynonym> … </mo:broadSynonym>`
narrow_synonym	`<mo:narrowSynonym> … </mo:narrowSynonym>`
xref_analog	`<mo:xrefAnalogous> … </mo:xrefAnalogous>`
xref_unknown	`<mo:xrefUnknown> … </mo:xrefUnknown>`

Table 4: OBO tags common to terms and typedefs, and corresponding OWL elements

OBO Tag	OWL Markup
[TERM]	`<owl:Class> … </owl:Class>`
is_a	`<rdfs:subClassOf rdf:resource = "…"/>`
is_obsolete	`owl:DeprecatedClass` (See more on obsolete in section 4.6)

Table 5: OBO tags specific to terms, and corresponding OWL elements

4.2.3 Typedefs

Typedefs in OBO are similar to object properties in OWL. Therefore, 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 within this element. OBO tags and their corresponding OWL elements are shown in Table 6.

Some important cases in typedef transformations are:

- OBO typedefs can have associated domains and ranges. These are expressed by 'domain' and 'range' tags. These tags are translated into RDF Schema defined elements `rdfs:domain` and `rdfs:range` respectively.
- Typedefs may be cyclic ('is_cyclic' tag), transitive ('is_transitive' tag) or symmetric ('is_symmetric' tag). Corresponding OWL elements are `mo:isCyclic`, `mo:isTransitive` and `mo:isSymmetric` respectively. All these elements specify Boolean values. Although special properties tags like `owl:TransitiveProperty` etc. are available in OWL, we have not focused on them because of relatively very simple nature of existing OBO ontologies.
- 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 in a typedef stanza. This tag is translated into an `rdfs:subPropertyOf` element defined in RDF Schema.

Other important cases, like identifiers and namespaces etc are similar to that of terms and are explained in detail in later sections.

OBO Tag	OWL Markup
[TYPEDEF]	`<owl:ObjectProperty> … </owl:ObjectProperty>`
is_a	`<rdfs:subPropertyOf rdf:resource = "…"/>`
is_obsolete	`owl:DeprecatedProperty` (See more on obsolete in section 4.6)
domain	`<rdfs:domain rdf:resource = "…"/>`
range	`<rdfs:range rdf:resource = "…"/>`
is_cyclic	`<mo:isCyclic> … </mo:isCyclic>`
is_transitive	`<mo:isTransitive> … </mo:isTransitive>`
is_symmetric	`<mo:isSymmetric> … </mo:isSymmetric>`

Table 6: OBO tags specific to typedefs, and corresponding OWL elements

4.3 Transforming Identifiers and Namespaces

Each term or typedef in an OBO ontology has a locally unique identifier and belongs to a particular namespace. The identifier is expressed in the OBO file using the 'id' tag, and the namespace for a term or typedef is expressed using 'namespace' tag. An OBO ontology usually has a default namespace expressed using the 'default-namespace' tag. Every term or typedef for which a namespace is not explicitly specified is considered to be within the default namespace.

When translated into OWL, all the namespaces that exist in the ontology are translated into XML namespaces and are declared at the start of the file within the `rdf:RDF` element. Since OBO namespaces are locally unique strings, we transform them into globally unique names (URNs) by adding the prefix '`urn:obo-res:`' to every namespace identifier string. The default namespace is defined using the `xml:base` attribute.

In addition, all namespaces are declared within the `owl:Ontology` element using `mo:definedNamespace` elements. This is redundant information but is necessary for translating the ontology back into OBO. Similarly, the default namespace in OBO ontology is declared using `mo:defaultNamespace` element in the corresponding OWL version.

The identifier for a term or typedef (in OBO) is expressed in OWL using the `rdf:about` attribute of the `owl:Class` or `owl:ObjectProperty` element. The complete ID is the concatenation of namespace URN and local identifier separated by the '`#`' character. For example, a term in OBO with ID '`Alpha:123`' and namespace '`TestNS`' gets an ID of the format '`urn:obo-res:TestNS#Alpha:123`' in the corresponding OWL translation.

In case a term or typedef belongs to the default namespace, it is not necessary to start the OWL ID with the URN for the namespace. It is sufficient to start the ID with the '`#`' sign, e.g., '`#Alpha:123`'.

4.4 Transformation of Subsets

Terms in an OBO ontology can be organized into subsets of the ontology. 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 declared subset can then be used to assign terms.

A term can be assigned to a subset using the 'subset' tag. The value of this tag must be a subset ID as defined in the ontology header. Multiple 'subset' tags are used to assign the term to multiple subsets of the ontology.

Note that the subset definition is not assigned to a particular namespace in the OBO ontology. For simplicity of translation, we assume that the subsets belong to the default namespace.

When translated into OWL, the declaration of a subset becomes slightly more complex. 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 `mo:Subset` element, and is assigned to the default namespace in the OWL ontology. The quoted description of the subset is expressed using the `mo:comment` element within `mo:Subset`.

We have defined `mo:inSubset` annotation tags to assign terms to a subset. Within the definition of a term, i.e. `owl:Class` element, the property `mo:inSubset` is mentioned with the appropriate `mo:Subset` element ID.

4.5 Transformation of 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. An example is given in Table 1.

OBO Format	OWL Format
[TERM] id: SUB001 relationship: part_of OBJ001	<owl:Class rdf:about= "#SUB001"> <rdfs:subClassOf> <owl:Restriction> <owl:onProperty rdf:resource = "#part_of" /> <owl:someValuesFrom rdf:resource = "#OBJ001" /> </owl:Restriction> </rdfs:subClassOf> </owl:Class>

Table 7: Example of OBO relationship and transformation to OWL

There are multiple kinds of restrictions on relationships that can be expressed using OWL. OBO specifications [25] 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 an OBO relationship into OWL.

As shown in Table 7, a 'relationship' tag is transformed into a combination of `rdfs:subClassOf`, `owl:Restriction`, `owl:onProperty`, and `owl:someValuesFrom` elements. 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 [1, 8], and the transformed example can be read as, "there exists an instance of `OBJ001`, such that an instance of `SUB001` is a part of (`part_of`) it".

4.6 Transformation of Deprecated Content

OBO flat file format supports deprecated (or obsolete content). A term or typedef can be marked as obsolete using the 'is_obsolete' tag with a 'true' Boolean value. Obsolete terms and typedefs are not allowed to have any relationships with other terms or typedefs, including the subclass and sub-property relationships.

In order to specify a term or typedef that is to be considered equivalent (or a replacement) to the deprecated term or typedef, the 'use_term' tag is used. The value of this tag is the ID of the equivalent (or replacing) term or typedef. The 'use_term' tag is not allowed for terms and typedefs that do not have the value 'true' associated with the 'is_obsolete' tag.

When translated into OWL, an obsolete term is transformed into a deprecated class using `owl:DeprecatedClass` element. The 'use_term' references to the term in OBO are then translated into `owl:equivalentClass` elements in OWL.

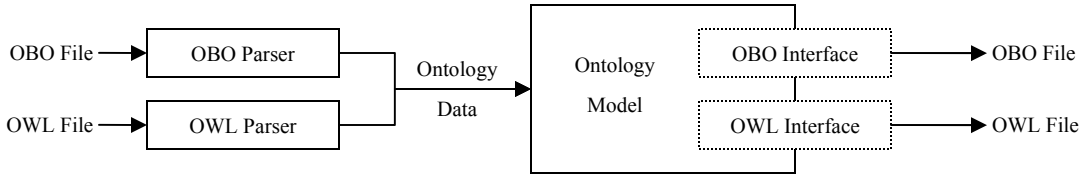

Figure 2: Components of the system with corresponding input and outputs.

Similarly, an obsolete OBO typedef is transformed into a deprecated property using the `owl:DeprecatedProperty` element. The 'use_term' references to the typedef in OBO are then translated into `owl:equivalentProperty` elements in OWL.

5 Implementation Details

There are three main components of the transformation system (see Figure 2), described as follows:

- **Ontology Model:** The Ontology Model component provides the data structures for internal storage and representation for data received as a part of an OBO or OWL ontology. The internal data structures are closer in nature to OBO file format. Some part of the source code for these data structures has been imported from OBO-Edit (specifically, from the older version known as DAG-Edit) [7, 18], which is a useful tool for viewing and editing OBO ontologies. In addition to having the data structures, the Ontology Model component provides appropriate interfaces for generating OBO or OWL representation of the ontology provided to the system.
- **OBO Parser:** The OBO Parser component parses files (ontologies) provided in OBO flat file format and forwards them to the Ontology Model. We have used a `javacc` grammar [20] to produce the parser for OBO file format.
- **OWL Parser:** The OWL Parser component reads OWL ontologies using Jena API for OWL [13], and provides the data to the Ontology Model. The system uses a subset of OWL to represent the ontologies transformed from OBO, and it is important that the parser allows only the same subset in order to make roundtrip transformations without any problems.

6 Examples of Roundtrip Transformations

In this section, we provide different example sets in order to clarify the transformation of different kinds of elements in an OBO ontology into OWL (see Table 8).

OBO Format	OWL Format
format-version: 1.0 date: 05:09:2006 17:28 saved-by: midori auto-generated-by: OBO-Edit 1.002 subsetdef: goslim_plant "Plant GO slim" subsetdef: goslim_yeast "Yeast GO slim" default-namespace: gene_ontology remark: cvs version: $Revision: 4.75 $	<owl:Ontology rdf:about="#OBO"> <mo:formatVersion>1.0</mo:formatVersion> <mo:savedDateTime>05:09:2006 17:28</mo:savedDateTime> <mo:savedBy>midori</mo:savedBy> <mo:autoGeneratedBy>OBO-Edit 1.002</mo:autoGeneratedBy> <mo:defaultNamespace>gene_ontology</mo:defaultNamespace> <mo:definedSubset rdf:resource="#goslim_yeast"/> <mo:definedSubset rdf:resource="#goslim_plant"/> <mo:remark>cvs version: $Revision: 4.75 $</mo:remark> </owl:Ontology> <mo:Subset rdf:about="#goslim_yeast"><mo:comment>"Yeast GO slim"</mo:comment></mo:Subset> <mo:Subset rdf:about="#goslim_plant"><mo:comment>"Plant GO slim"</mo:comment></mo:Subset>
(a) Header Information, containing subset definitions as well	
[Term] id: GO:0008029 name: pentraxin receptor activity def: "Combining with a pentraxin to initiate a change in cell activity." [GOC:add, ISBN:0781735149 "Fundamental Immunology"] comment: Note that pentraxins include such proteins as serum amyloid P component (SAP) and C-reactive protein (CRP). namespace: molecular_function exact_synonym: "pentaxin receptor" [] is_a: GO:0001864 is_a: GO:0001847	<owl:Class rdf:about="urn:obo-res:molecular_function#GO:0008029"> <mo:name>pentraxin receptor activity</mo:name> <mo:definition>"Combining with a pentraxin to initiate a change in cell activity." [GOC:add, ISBN:0781735149 "Fundamental Immunology"]</mo:definition> <mo:comment>Note that pentraxins include such proteins as serum amyloid P component (SAP) and C-reactive protein (CRP).</mo:comment> <mo:exactSynonym>"pentaxin receptor" [] </mo:exactSynonym> <rdfs:subClassOf rdf:resource="urn:obo-res:molecular_function#GO:0001847"/> <rdfs:subClassOf rdf:resource="urn:obo-res:molecular_function#GO:0001864"/> </owl:Class>
(b) Term definition, showing subclass relationships, namespace, and a kind of synonym	
[Typedef] id: part_of name: part of is_transitive: true	<owl:ObjectProperty rdf:about="#part_of"> <mo:name>part of</mo:name> <mo:isTransitive>true</mo:isTransitive> </owl:ObjectProperty>
(c) Typedef definition, showing transitivity property	
[Term] id: GO:0008041 name: storage protein of fat body (sensu Insecta) namespace: molecular_function is_obsolete: true	<owl:DeprecatedClass rdf:about="urn:obo-res:molecular_function#GO:0008041"> <mo:name>storage protein of fat body (sensu Insecta) </mo:name> </owl:DeprecatedClass>
(c) Transforming an obsolete (deprecated) term	
[Term] id: GO:0008047 name: enzyme activator activity namespace: molecular_function subset: gosubset_prok	<owl:Class rdf:about="urn:obo-res:molecular_function#GO:0008047"> <mo:name>enzyme activator activity</mo:name> <mo:inSubset rdf:resource="#gosubset_prok"/> </owl:Class>
(d) Transforming a subset entry	

Table 8: Examples of transformations between OBO and OWL formats

7 Rules for Editing of OWL Output

As discussed earlier, 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 an OBO ontology according to some guidelines while it is in OWL format.

Following are some of the guidelines for editing OWL produced by our system:

- Any new namespace that is added to the ontology using XML namespaces must be explicitly added to the document by adding the `mo:definedNamespace` tag to the ontology header. Naming of the new namespace should follow the methodology described in Section 4.3 regarding namespace transformations.
- New terms and typedefs should be defined using only the `owl:Class` and `owl:ObjectProperty` tags. Sometimes, OWL ontologies use RDF and RDF Schema tags like `rdf:Description` and `rdfs:Class`. Such tags should not be used. Also, special properties tags like `owl:TransitiveProperty` and `owl:SymmetricProperty` etc. may not be used. Such properties of typedefs should be expressed using tags like `mo:isTransitive`.
- Only the relationships with existential quantification should be used. In other words, when creating or editing existing relationships between classes, only the owl:someValuesFrom element should be used to restrict the relationship. Other kinds of restrictions are not supported.
- Our transformations deal only with OBO format version 1.0. Features like Boolean combinations etc. are not present in this version of OBO. Therefore, use of features and constructs should be restricted to the ones mentioned in this document.

8 Related Work

The only publicly available work concerning OBO and OWL mapping are Mungall's effort [17]. Their work describes a transformation from a small subset of OBO version 1.2 to W3C recommended OWL format. The technology used for making transformations in their work is XSLT. This mapping deals with very basic OBO elements like terms, typedefs, relationships, subclasses, etc. Transformations of a number of metadata tags and more complex elements like subsets, deprecated content and synonyms etc. have not been attempted in their work so far.

There are various kinds of differences between this work and our approach. First, we have concentrated on providing a migration path from existing OBO content to OWL in order to integrate the knowledge-bases of the two worlds. Since OBO 1.2 is still very new, and almost all OBO content is in OBO 1.0 format, we have focused on providing a much greater level of completeness in terms of metadata elements and other constructs provided by OBO 1.0. Second, we have approached the problem with the methodology of identifying correspondences between OBO and OWL. This way, we have been able to identify clearly matching elements between the two formats; we have also been able to list the features in OBO that require constructing new elements in OWL. The result of this methodology is that we have created bidirectional transformation rules which can provide lossless roundtrips between OBO 1.0 and OWL.

Mungall's transformation builds on the work of Aitken [1], which proposes a minimal OWL Full ontology for OBO and Gene Ontology (GO) terms. Aitken's work provides OWL constructs for some common OBO relationship types and also describes formal semantics for them.

9 Conclusion and Future Work

Building ontologies is not a new idea for the biology community. However, the utility of ontologies has not been fully realized due to wide acceptance of weaker but more readable languages like OBO. OBO does not provide global naming schemes, and there is lack of language support for querying OBO ontologies and performing rule-based inferences on them.

Semantic Web is the idea that once fully realized, can solve these problems. The use of URIs for assigning globally unique identifiers to concepts is one of the foundations of Semantic Web. Querying languages like SPARQL for ontologies expressed in RDF, and work is in progress on defining languages for rules and inference on the Semantic Web ontologies.

We believe our work is an important step towards building interoperable knowledge-bases, and that it draws an easy picture of the Semantic Web world to other communities that require them.

In future, we would like to extend this work to make it compatible with OBO 1.2 and provide a higher level of completeness. We would also like to allow more flexibility in editing the OWL version of the transformed ontologies, and make our tools richer enough to handle it.

Currently, we have simple checks in place to identify loss of knowledge in terms of size of the ontology. Successfully making the roundtrips with Gene Ontology (GO) which has more than 21 thousand terms gives us a reasonable level of confidence. However, having a formal way of proving the correctness of our roundtrip transformation rules according to the semantics of both OBO and OWL is another interesting direction for future work.

10 Acknowledgements

This research is supported by the National Science Foundation through grants IIS-0531767 and IIS-0325116.

11 References

1. Aitken, S. 2003. *A Minimal Ontology for OBO and GO*. http://www.aiai.ed.ac.uk/resources/go/
2. Antoniou, G., van-Harmelen, F. eds. 2004. *A Semantic Web Primer*. MIT Press.
3. Barker, K., Porter, B., Clark, P. 2001. *A Library of Generic Concepts for Composing Knowledge Bases*. First International Conference on Knowledge Capture.
4. Berners-Lee, T., Hendler, J., Lassila, O. 2001. *The Semantic Web*. Scientific American, 284(5):34-43, May 2001.
5. Berners-Lee, T. 2003. *Semantic Web Status and Direction*. International Semantic Web Conference (ISWC2003) Keynote.
6. Clark, P., Porter, B. 1997. *Building Concept Representations from Reusable Components*. Fourteenth National Conference on Artificial Intelligence (AAAI '97).
7. DAG-Edit User Guide. http://www.godatabase.org/dev/java/dagedit/docs/index.html
8. Dean, M., Schreiber, G., editors. 2004. *OWL Web Ontology Language Reference*. W3C Recommendation, 10 Feb 2004. http://www.w3.org/TR/2004/REC-owl-ref-20040210/

9. Farquhar, A., Fikes, R., Rice, J. 1997. *The Ontolingua Server: a Tool for Collaborative Ontology Construction*. Technical Report KSL-96-26, Knowledge Systems Laboratory, Computer Science Department, Stanford University, September 1996.
10. Fensel, D., van Harmelen, F., Horrocks, I., McGuinness, D.L., Patel-Schneider, P.F. 2001. *OIL: an ontology infrastructure for the Semantic Web*. IEEE Intelligent Systems, 16(2):38-45.
11. Gruber, T.R. 1993. *A Translation Approach to Portable Ontology Specification*. Knowledge Acquisition 5: 199-220.
12. Hendler, J. 2001. *Agents and the Semantic Web*. IEEE Intelligent Systems, 16(2):30-37.
13. Jena – A Semantic Web Framework for Java. http://jena.sourceforge.net/
14. Mabee, P.M., Haendel, M.A., Arratia G., Coburn M.M., Hilton E.J., Lundberg J.G., Mayden R.L. 2006. *ZFIN Anatomy Working Group: Skeletal System*. Manually curated data.
15. McGuinness, D.L., Fikes, R., Rice, J., Wilder, S. 2000. *An Environment for Merging and Testing Large Ontologies*. Principles of Knowledge Representation and Reasoning: Proceedings of the Seventh International Conference (KR2000).
16. McGuinness, D.L., van Harmelen, F., editors. 2004. *OWL Web Ontology Language*. W3C Recommendation, 10 Feb 2004. http://www.w3.org/TR/2004/REC-owl-features-20040210/
17. Mungall, C. 2005. *Mapping OBO to OWL*. Berkeley Drosophila Genome Project. http://www.godatabase.org/dev/doc/mapping-obo-to-owl.html
18. OBO-Edit, Gene Ontology Tools. http://www.geneontology.org/GO.tools.shtml
19. Open Biomedical Ontologies. http://obo.sourceforge.net/
20. Project Home for javacc. https://javacc.dev.java.net/
21. Smith, B., Ceusters, W., Klagges, B., Köhler, J., Kumar, A., Lomax, J., Mungall, C., Neuhaus, F., Rector, A.L., Rosse, C. 2005. *Relations in Biomedical Ontologies*. Genome Biology, 6(5):R46.
22. SPARQL Query Language for RDF. 2006. W3C Candidate Recommendation. http://www.w3.org/TR/rdf-sparql-query/
23. The Gene Ontology. http://www.geneontology.org/
24. The Gene Ontology Project. http://geneontology.sourceforge.net/
25. The OBO Flat File Format Specifications. http://www.geneontology.org/GO.format.shtml