Relational patterns in OWL and their application to OBO

Robert Hoehndorf[1], Anika Oellrich[1], Michel Dumontier[2], Janet Kelso[4], Heinrich Herre[3], and Dietrich Rebholz-Schuhmann[1]

[1] European Bioinformatics Institute, {hoehndor,anika,rebholz}@ebi.ac.uk
[2] Department of Biology, Institute of Biochemistry and School of Computer Science, Carleton University, Michel_Dumontier@carleton.ca
[3] Institute for Medical Informatics, Statistics and Epidemiology, University of Leipzig, heinrich.herre@imise.uni-leipzig.de
[4] Department of Evolutionary Genetics, Max Planck Institute for Evolutionary Anthropology, kelso@eva.mpg.de

Abstract. Directed acyclic graphs are commonly used to represent ontologies in the biomedical domain. They provide an intuitive means to formalize relations that hold between ontological categories. However, their semantics is usually not explicit. We provide a semantics for a part of the OBO Flatfile Format by extending OWL with a method to express relational patterns. These patterns are OWL axioms with variables for classes. The variables can only be filled with named classes. Additionally, we provide a semantics for open patterns in OWL. Our method is applicable to the OBO Flatfile Format, and provides a means to design OWL ontologies using complex ontology design patterns. Therefore, it leads not only to an integration of the OBO Flatfile Format and OWL, but extends OWL with an intuitive interface for designing ontologies using complex definition patterns. A prototypic implementation and test results are available at http://bioonto.de/obo2owl.

1 Introduction

Directed acyclic graphs (DAG) have been a popular representation format for biomedical ontologies. The original representation of the Gene Ontology (GO) has been in the form of a DAG [1]. From the DAG representation of the GO, the *OBO Flatfile Format* [9], a graph-based knowledge representation language, was derived. Currently, many ontologies in the biomedical domain are being developed in the OBO Flatfile Format.

In the OBO Flatfile Format, nodes represent ontological categories and edges represent relations between these categories. The OBO Relationship Ontology (RO) provides formal definitions for commonly used relations between ontological categories [15]. However, at the moment, there is no explicit semantics for the OBO Flatfile Format that can accommodate the relation definitions from the RO.

We developed an extension to the OBO Flatfile Format and to the Manchester OWL Syntax [8] based on the assumption that any statement in OWL in

which two variables for classes occur, determines a relation between these two classes. For example, we define the **CC-has-part** relation as `?X SubclassOf: has-part some ?Y`, where **CC-has-part** is a relation between classes and **has-part** a relation between individuals. Based on this assumption, we provide a novel implementation of the OBO Relationship Ontology in OWL and a software application to convert OBO ontologies to OWL. Furthermore, we provide another software application which uses OWL reasoning to infer new binary relations between classes. Our method and software applications lead to an integration of the OBO Flatfile Format with OWL while maintaining the semantics for relations provided by the OBO Relationship Ontology.

2 OBO Flatfile Format

The OBO Flatfile Format is a knowledge representation format that has been developed for biomedical ontologies. The basic elements of the OBO Flatfile Format are `typedef` and `term` statements. Term statements define *nodes* in the DAG. A node has an identifier and a name. Additionally, several restrictions can be asserted for a node. In particular, the edges originating from the node are specified in the node description. For this purpose, either the `is_a` or `relationship` statements are used. Edges represent relations between nodes. An example of a term definition taken from the Celltype Ontology (CL) [3] is:

```
[Term]
id: CL:0000028
name: CNS neuron (sensu Nematoda and Protostomia)
is_a: CL:0000540 ! neuron
relationship: develops_from CL:0000338
```

Typedef statements specify the *kinds* of edges in the DAG. They represent the relation that is intended to be established between two nodes when an edge of a certain kind is used. An edge has an identifier and a name. Additionally, properties for the edges can be asserted, such as transitivity. A definition of the develops_from edge kind taken from the CL is:

```
[Typedef]
id: develops_from
is_transitive: true
```

The intended meaning of the graph representations is that nodes represent ontological categories and edges represent relations between these categories. The OBO Relationship Ontology (RO) [15] provides formal definitions in first order logic for a set of commonly used relations in biomedical ontologies.

However, the semantics of the OBO Flatfile Format is not explicit, and several competing solutions have been proposed. One type of semantics is provided by the RO, which uses a first order semantics tailored to each relation type. Another type of semantics uses a fixed semantics for relations and relational statements [9]. An edge of type R between nodes A and B is usually given the semantics of the OWL statement

```
A SubclassOf: R some B
```

The latter kind provides a semantics for intersection, union and disjointness statements as well, using the corresponding operations for classes in description logics.

Although the rigid semantics for relational statements in the OBO Flatfile Format is adequate for many relation types, it fails in several cases. For example, the relation **lacks-part** must not be translated using an existential assertion. Similarily, the relation **realized-by** must use a universal quantification instead of an existential one [14]: an existential quantifier would entail the false assertion that every function or disposition is actually realized by some process, while dispositions and functions serve to express potentials for realizations.

Both kinds of semantics are incompatible, and each has drawbacks. The first does not provide a semantics for the OBO Flatfile Format *as a knowledge representation language*, because it only provides a semantics for parts of the OBO Flatfile Format, and this semantics depends on the used relations. New relations cannot easily be introduced or defined. On the other hand, using a rigid semantics for relations does not correspond to the intuitions of ontology designers and often leads to assertions which are false within a domain.

This observation motivated us to develop and implement an extension to OWL that can be applied to solve the problems with the OBO Flatfile Format, and is general enough to be applicable in other domains of knowledge representation using OWL.

3 Pattern definitions in OWL

To provide compatibility with the OBO Flatfile Format, we focus on binary relations between classes first and extend our method later. We introduce a new type in OWL which represents relational pattern specifications.

To specify the intensions of binary relations between classes, we have extended the Manchester OWL Syntax [8] with the variable symbols `?X` and `?Y`. Both are symbols that are intended to represent *classes*. Any OWL class axiom in the extended Manchester OWL syntax represents a *relational pattern definition*. We permit degenerate patterns in which only one or no variable symbol occurs. Similarily, both `?X` and `?Y` can be filled by the same class name. Any pattern can be applied to two OWL class descriptions to yield an OWL axiom. The OWL axiom is derived by substituting `?X` with the first and `?Y` with the second OWL class description.

For example, we can define the pattern **CC-part-of** as `?X SubclassOf: part-of some ?Y` where **part-of** is an OWL object property. Then we can apply the **CC-part-of** pattern to the primitive classes A and B, **CC-part-of**(A, B). The resulting OWL axiom is derived by substituting `?X` with A and `?Y` with B: `A SubclassOf: part-of some B`. In such a scenario, the patterns are never directly used within OWL for reasoning. Instead, the patterns provide a template for asserting OWL axioms.

More complex ontology design patterns[5] can be asserting using different relational pattern definitions. Table 1 lists a translation of the relational patterns in the RO to relational pattern definitions.

The approach can be restricted to unary or extended to n-ary relational patterns. Unary patterns require a single variable symbol, while n-ary relational patterns use the variable symbols `?X1`, `?X2`, `?X3`, etc. For such an application, the OBO Flatfile Format would have to be extended to accommodate n-ary relation.

4 Open patterns and meta-properties

The patterns themselves contain the free variables `?X1`,...,`?XN`. In some cases, it may be useful to use open patterns themselves as meta-properties of an OWL ontology. In such a case, the open variables are universally quantified. However, the common interpretation of quantification over classes is second order, where the quantifier ranges over the powerset of the universe. This results in undecidability of class satisfiability. We provide a decidable semantics for open relational pattern definitions, where the quantifier ranges only over *named classes* in the signature of the OWL ontology. We use description logic syntax [2] to formalize this approach.

The semantics of a description logic theory over a signature $\Sigma = (C, R, A)$, with C a set of concept symbols (including $\top$ and $\bot$), R a set of relation symbols and A a set of individual symbols, is given by an interpretation $\mathcal{I}$. The interpretation $\mathcal{I}$ consists of a non-empty set $U^{\mathcal{I}}$ and an interpretation function δ, such that for every $C_i \in C$, $\delta(C_i) \subseteq U^{\mathcal{I}}$, $\delta(R_i) \subseteq U^{\mathcal{I}} \times U^{\mathcal{I}}$ for every $R_i \in R$ and $\delta(a) \in U^{\mathcal{I}}$ for every $a \in A$. The interpretation function is inductively extended in the usual way. Using standard description logic notation [2], examples of these inductive definitions include:

$$\top^{\delta} = U^{\mathcal{I}}$$

$$(C \sqcap D)^{\delta} = C^{\delta} \cap D^{\delta}$$

$$(\exists R.C)^{\delta} = \{a \in U^{\mathcal{I}} | \exists b.(a, b) \in R^{\delta} \wedge b \in C^{\delta}\}$$

Using a higher-order logic, the interpretation of free class variables such as `?X` and `?Y` will map to a subset of the powerset of $U^{\mathcal{I}}$: $\delta(?X) \in \mathcal{P}(U^{\mathcal{I}})$. Universal quantification over these free variables would then range over the full powerset of $U^{\mathcal{I}}$. In particular, satisfiability of terminological axioms[6] that contain class expressions involving `?X` or `?Y` would have to consider the powerset of $U^{\mathcal{I}}$.

For the relational pattern definitions, we adopt a different approach where it is not necessary to use the full powerset in the interpretations of the class variables. Instead, the variable symbols `?X1`,..., `?XN` are interpreted with an extension of one of the atomar classes from the signature Σ. If Σ is finite, then

[5] http://ontologydesignpatterns.org

[6] Terminological axioms in description logics are of the form $C \sqsubseteq D$ or $C \equiv D$ with C and D being concept expressions, or $R \equiv S$ with R and S being relationship (role) expressions.

satisfiability of terminological axioms in OWL extended with `?X` and `?Y` will be trivially decidable.

Formally, let T be a description logic theory over the signature $\Sigma = (C \cup \{?X1, ..., ?XN\}, R, A)$ and $\mathcal{I}$ be an interpretation with the interpretation function δ and a domain $U^{\mathcal{I}}$, and $P^-(U^{\mathcal{I}}) = \{C_i^{\delta} | C_i \in C\}$. Then $\delta(?X1) \in P^-(U^{\mathcal{I}})$,..., $\delta(?XN) \in P^-(U^{\mathcal{I}})$.

This restriction leads to decidability of the satisfiability problem for terminological axioms (if Σ is finite): satisfiability of a terminological axiom involving `?X1`,..., `?XN` can be decided by verifying the satisfiability of the terminological axioms that arise through substituting `?X`,..., `?XN` with all atomar concept symbols in Σ. Since the signature $\Sigma = (C, R, A)$ is finite, $|C|^N$ terminological axioms must be verified for satisfiability to decide the satisfiability of one open relational pattern definition involving `?X1`,..., `?XN`.

5 Application to the OBO Flatfile Format

5.1 OWLDEF method

Due to the decidability of satisfiability of terminological axioms, the definition schema for relations in the OBO Flatfile Format can be employed in two directions: from OBO to OWL and from OWL to OBO. We have already described how relations can be defined and translated to OWL using a relational pattern definition. Based on these definitions, new relations between categories in the OBO Flatfile Format can be extracted from an OWL knowledge base. Therefore, these definitions can also serve as a method for an extended form of reasoning using the OBO Flatfile Format.

The OBO Flatfile Format provides a means to express relations between classes, yet it does not provide a way to define the relations themselves. We use relational pattern definitions in our extended syntax to define relations between classes in the OBO Flatfile Format. For this purpose, we extend the `Typedef` environment in the OBO Flatfile Format to include the definition of relations. For example, to define the relation **CC-has-part**, we would use the following `Typedef` statement in the OBO Flatfile Format:

```
[Typedef]
id: CC-has-part
name: has-part
owldef: ?X SubClassOf: has-part some ?Y
```

According to our semantics, every *use* of the relation **CC-has-part** in the OBO Flatfile Format is expanded to an OWL axiom in which the variables are filled by the classes between which the relation was asserted. For example, the statement that every mouse body has some tail as part in the OBO Flatfile Format would be:

```
[Term]
id: Mouse_body
relationship: has-part Tail
```

Using the OWLDEF method, *Mouse_body* and *Tail* fill ?X and ?Y, respectively. The resulting OWL axiom would be

```
Mouse_body SubClassOf: has-part some Tail
```

Although most relations in biomedical ontologies follow an existential all-some pattern, some relations must be formalized differently. In particular the relation **integral-part-of** from the RO [15] cannot be formalized using a standard existential pattern. A class C is an **integral-part-of** a class D if and only if C is a **CC-part-of** D and D **CC-has-part** C. These two statements do not directly translate into a single OWL axiom. Therefore, we performed a transformation into a single axiom. This axiom states that it is not possible (subclassOf Nothing) that some entity is an instance of ?X and not the part of some ?Y, and neither is it possible that some entity is an instance of ?Y and has no ?X as part.

```
(?X and not (part-of some ?Y)) or
(?Y and not (has-part some ?X))
             subclassOf Nothing
```

This statement is logically equivalent to asserting both axioms `?X SubclassOf: part-of some ?Y` and `?Y SubclassOf: has-part some ?X`.

The patterns we define can not only be used to expand relations between classes into complex OWL statements, but also to convert a complex OWL ontology into a set of relations between classes. For this purpose, let L be the set of named classes in the signature of an OWL ontology. Then, for each pair (x, y) of classes $x, y \in L$, we replace `?X` with x and `?Y` with y in the relational pattern definitions, and use OWL reasoning to verify whether the resulting OWL axiom is true in the OWL ontology. If the resulting axiom is true in the OWL ontology, the relation between the classes x and y holds and we can add this information to the OBO Flatfile Format. This approach is superior to syntactic transformations of OWL to OBO, because it accounts for the semantics of the relations, and makes the inferences that can be drawn from them available in the OBO format.

5.2 Implementation and evaluation

We implemented the expansion and the contraction of relational patterns in two separate software libraries and applications. The first Java library is an extension of the Manchester OWL API [7] and provides functionality to convert OBO Flatfile Format ontologies to OWL using OWLDEF relational pattern definitions. This extension replaces the Manchester OWL API's OBO Flatfile parser. Our OBO parser reads the `owldef` definitions for the relations and adds the derived axioms to the OWL ontology. Instead of the transformation of `relationship` statements based on the OWL API's parser, we use a custom transformation based on the `owldef` definitions. For each term definition containing a `relationship` statement that refers to a relation with an `owldef`

Relationship	OWLDEF Pattern
part-of	?X subclassOf part-of some ?Y
develops-from	?X subclassOf develops-from some ?Y
integral-part-of	(?X and not (part-of some ?Y)) or (?Y and not (has-part some ?X)) subclassOf Nothing
realized-by	?X subclassOf realized-by only ?Y
realizes	?X subclassOf realizes some ?Y
lacks-part	?X subclassOf not (has-part some ?Y)
has-function	?X subclassOf has-function some ?Y
lacks-function	?X subclassOf not (has-function some ?Y)
has-function-realized-by	?X subclassOf has-function some (realized-by only ?Y)

Table 1. OWLDEF implementation of selected relations.

definition, `?X` and `?Y` from the `owldef` statement are replaced by the corresponding term names (see 5.1). After the replacement, we use the OWL API's Manchester syntax inline axiom parser to generate an OWL axiom from the resulting statement. Each axiom is added to the OWL ontology in addition to the axioms generated by the transformation from OBO to OWL implemented in the Manchester OWL API.

Second, we provide a prototypical implementation to extract relational patterns from an OWL ontology so that they can be converted back to the OBO Flatfile Format. For this purpose, an OWL ontology is read using the Manchester OWL API [7]. Based on a list of relational patterns such as those in table 1 and the list of all class names in the loaded OWL ontology, binary relations between classes are generated as OWL axioms: each class name in the signature of the OWL ontology is used to replace "?X" in the pattern and then combined with all class names to replace "?Y" in the same pattern. Consequently, all combinations of named classes are generated to fill variables in the relation patterns, leading to a list of OWL axioms. Using the Hermit OWL reasoner [12], we attempt to prove each of these OWL axioms and keep track of those that the reasoner could infer from the axioms asserted in the ontology. As a consequence, we obtain a list of theorems that hold in the ontology and can be added back to an OBO file.

To evaluate our method we applied it to the Celltype Ontology (CL) [3]. The CL uses two relations, **is-a** and **develops-from**. The pattern for **is-a** is `?X SubClassOf: ?Y` and the pattern for develops-from is `?X SubClassOf: develops-from some ?Y`.

The CL contains 1253 **is-a** and 275 **develops-from** statements, i.e., 1528 axioms that restrict CL categories using one of these two relations. We classify the generated OWL ontology using the Hermit OWL reasoner. Based on the classified OWL ontology, we attempt to prove the two patterns for each pair of named classes in the ontology. We use the Hermit reasoner to perform these inferences. Using this approach, we identify 9,497 **is-a** and 124,420 **develops-from** statements that we add to the OBO Flatfile represenation of the CL. This shows that OWL reasoning can be used to infer new relations in OBO ontologies.

Since the CL only uses **is-a** and **develops-from**, our conversion is similar to other OBO-to-OWL conversion methods. Therefore, we further evaluated our method with the Malaria Ontology and the Sequence Ontology (SO) [5]. The Malaria Ontology uses the **realized-by** relation three times in its axioms, and, using OWL reasoning, we infer 56 **realized-by** relations between classes from the three assertions. Further, we added one **integral-part-of** relation to the SO (*exon* **integral-part-of** *transcript*). From this assertion, we could infer that all exons are **part-of** some transcript, and conversely that all transcript have an exon as part. We provide the results at our website at http://bioonto.de/obo2owl.

6 Discussion

6.1 Comparison to other approaches

There are several methods and tools available to convert ontologies in the OBO Flatfile Format to OWL [11, 13]. Some tools and methods are capable of converting OWL to OBO [16]. At least one semantics is proposed for the OBO Flatfile Format that uses an interpretation of OBO in OWL [9]. The commonly used conversion tools for OBO to OWL have in common that they interpret a relation R between two classes C and D as an existential statement: `C SubClassOf: R some D`.

Although this pattern is appropriate for a majority of currently used relations in OBO and OBO Foundry ontologies, it fails in several cases. Table 1 lists several such cases. In particular, the **integral-part-of** and **has-integral-part** relations in the RO require a different translation to OWL. Further relations that are used in OBO ontologies include the **realized-by** relation between a function or disposition and a process. Several complex relations such as **has-function-realized-by** [6] also require a more expressive translation to OWL.

To the best of our knowledge, there are no conversion tools available that are compatible with the RO in that they apply the definition patterns of the RO in the conversion. Similarily, the OWL implementation of the RO does not coincide with the definitions of the RO relations in first order logic. We are also not aware of an implementation of the RO in OWL that implements or approximates the definition patterns the RO attempts to provide.

Our prototypical implementations serve to demonstrate our method. In the future, we plan to use the OPPL formalism [10] to formalize relational patterns. OPPL is a pattern processing language for OWL, and OPPL version 2[7] permits the definition of complex patterns similar to relational pattern definitions.

6.2 Limitations

The OWLDEF method provides a flexible way to define relations using complex OWL statements. However, it interferes with other parts of the OBO Flatfile

[7] http://oppl2.sourceforge.net

Format. In particular disjointness, intersection and union statements do not interoperate well with the OWLDEF method. To illustrate the problem, consider the following definition of a category in the OBO Flatfile Format:

```
[Term]
id: ID:1
intersection_of: ID:2
intersection_of: integral-part-of ID:3
```

The difficulty is that `integral-part-of ID:3` is not a class description when the OWLDEF method is used. Instead, `ID:1 integral-part-of ID:3` would translate into one OWL axiom but axioms cannot be disjoint from classes (`ID:2`) in OWL. This shows a fundamental limitation of the OBO Flatfile Format, because it is not obvious what an intersection statement together with an **integral-part-of** relation is intended to mean.

However, the current translations of the OBO Flatfile Format to OWL do not provide an adequate semantics for this statement either, because the relation **integral-part-of** is not appropriately translated. One possible solution would be to disallow the use of relational statements in intersection, disjointness or union statements, and allow only class names as arguments. However, it is subject to future research to provide a semantics for these statements in combination with the OWLDEF method.

6.3 Application to RDF and Linked Data

OWL relational pattern definitions can be applied to unstructured RDF data to provide a semantic layer and an interpretation of relations used in RDF stores. One application would be to apply our method to Linked Data [4]. Linked Data is a web of data where URIs denote things and links between URIs are expressed using RDF and represent relations between things. At least a fragment of the Linked Data cloud contains URIs of *classes*, not individuals, and relations between these classes are expressed in RDF. Similar to the OBO Flatfile Format, the semantics of the relation between classes is not made explicit. OWL relational pattern definitions can provide a method to convert some parts of the Linked Data cloud from RDF to OWL, utilizing the expressive semantics of OWL to formalize, structure and verify pieces of data. An implementation and evaluation of applying OWL relational pattern definitons to RDF and Linked Data is subject to future work.

7 Conclusions

We developed a novel semantics for a fragment of the OBO Flatfile Format by explicitly incorporating relational definition patterns in the OBO Flatfile Format. A definition pattern is an OWL axiom with variables for OWL classes. Our approach leads to a flexible OWL-based semantics for several biomedical ontologies. Motivated by the problem of finding an adequate semantics for the

OBO Flatfile Format, we developed an extension to OWL that is general enough to be applicaple in many domains. It provides a means to incorporate ontology design patterns in the ontology development process, leading to an intuitive interface to otherwise complex logical statements.

8 Acknowledgements

We thank three anonymous reviewers for valuable comments on this manuscript.

We acknowledge funding from the Institute for Medical Informatics, Statistics and Epidemiology at the University of Leipzig, the Max Planck Institute for Evolutionary Anthropology and the European Bioinformatics Institute.

References

1. Ashburner, M., et al.: Gene ontology: tool for the unification of biology. the gene ontology consortium. Nat Genet 25(1), 25–29 (May 2000)
2. Baader, F.: The Description Logic Handbook : Theory, Implementation and Applications. Cambridge University Press (January 2003)
3. Bard, J., Rhee, S.Y., Ashburner, M.: An ontology for cell types. Genome Biology 6(2) (2005)
4. Christian Bizer, T.H., Berners-Lee, T.: Linked data – the story so far. International Journal on Semantic Web and Information Systems (2009), in press
5. Eilbeck, K., Lewis, S.E., Mungall, C.J., Yandell, M., Stein, L., Durbin, R., Ashburner, M.: The sequence ontology: A tool for the unification of genome annotations. Genome Biology 6(R55) (2005)
6. Hoehndorf, R., Ngomo, A.C.N., Kelso, J.: Applying the functional abnormality ontology pattern to anatomical functions. Journal for Biomedical Semantics (2010)
7. Horridge, M., Bechhofer, S., Noppens, O.: Igniting the owl 1.1 touch paper: The owl api. In: Proceedings of OWLEd 2007: Third International Workshop on OWL Experiences and Directions (2007)
8. Horridge, M., et al.: The Manchester OWL Syntax. Proc. of the 2006 OWL Experiences and Directions Workshop (OWL-ED2006) (2006)
9. Horrocks, I.: OBO Flat File Format Syntax and Semantics and Mapping to OWL Web Ontology Language. Tech. rep., University of Manchester (March 2007), http://www.cs.man.ac.uk/ horrocks/obo/
10. Iannone, L., Rector, A., Stevens, R.: Embedding knowledge patterns into owl. In: Proceedings of European Semantic Web Conference 2009. pp. 218–232 (2009)
11. Moreira, D.A., Musen, M.A.: OBO to OWL: a protege OWL tab to read/save OBO ontologies. Bioinformatics 23(14), 1868–1870 (July 2007)
12. Motik, B., Shearer, R., Horrocks, I.: Hypertableau Reasoning for Description Logics. Journal of Artificial Intelligence Research 36, 165–228 (2009)
13. Mungall, C.: Mapping obo to owl. http://www.godatabase.org/dev/doc/mapping-obo-to-owl.html (2005)
14. Schulz, S., Stenzhorn, H., Boeker, M., Smith, B.: Strengths and limitations of formal ontologies in the biomedical domain. RECIIS – Electronic Journal in Communication, Information and Innovation in Health 3(1), 31–45 (2009)
15. Smith, B., et al.: Relations in biomedical ontologies. Genome Biol 6(5) (2005)
16. Tirmizi, S.H., Miranker, D.: Obo2owl: Roundtrip between obo and owl. Tech. rep., University of Texas (2006), ftp://ftp.cs.utexas.edu/pub/techreports/tr06-47.pdf