Visualizing ontologies with AberOWL

Miguel Ángel Rodríguez-García[1], Luke Slater[1], Keiron O'Shea[1,2], Paul N Schofield[3], Georgios V Gkoutos[2], and Robert Hoehndorf[1]

[1] Computational Bioscience Research Center, King Abdullah University of Science and Technology, Thuwal 23955-6900, KSA
{miguel.rodriguezgarcia,luke.slater,keironteilo.oshea,robert.hoehndorf}@kaust.edu.sa
[2] Aberystwyth University, Aberystwyth, SY23 3DB, Wales, UK,
geg18@aber.ac.uk
[3] University of Cambridge, Downing Street, CB2 3EG, England, UK
pns12@hermes.cam.ac.uk

Abstract. Ontologies are formal theories that specify the kinds of entities and relations found in a domain. To quickly gain access to the content and structure of ontologies, ontology visualization techniques are commonly used. Visualization of ontologies often uses representations of hierarchical structures that are extracted from ontologies, most notably representations of the taxonomic relationships between classes. These graph-based representations can also be used to visualize structural changes in ontologies. We have developed a novel visualization environment for ontologies in which automated reasoning is used to generate a graph-based representation of an ontology's deductive closure, and subclass relations as well as description logic axioms that are entailed to hold between two classes are represented visually. The visualization environment can also be used to show differences between the entailed axioms of different ontology versions. The source code of the visualization environment is freely available, and we added our visualization environment to AberOWL (http://aber-owl.net), an ontology repository that contains over 400 ontologies, all of which can now be visually explored using our system.

Keywords: biomedical ontology, visualization, automated reasoning

1 Introduction

In recent years, a large number of ontologies has been developed across many scientific domains. These ontologies are often formalized in languages such as the Web Ontology Language (OWL) [5] or an OWL-compatible language such as the OBO Flatfile Format [12]. The major role of ontologies in biology and biomedicine is in data integration, as they formally describe the kinds of biological entities found within a domain, and their interrelations, and can therefore be used to provide semantic annotations that can be shared across databases.

Along with the increase in the number of ontologies, the need to develop tools that enable both ontology experts and domain experts to interact with ontologies has grown as well. One crucial aspect of interacting with ontologies is the ability to browse and visualize the content of ontologies. A widely used form of visualization for ontologies are graphs that represent classes and the axioms that hold between these classes. This form of representation is used in ontology editors such as Protégé [22] or the (now abandoned) OBO-Edit [24], as well as in ontology repositories such as BioPortal [21], OntoBee [31] or the Ontology Lookup Service (OLS) [2]. There are two key features based on which methods for visualizing ontologies as graph differ: the kind of 'relations' that are shown between classes as part of the graph structure, and whether only the asserted axioms are used to generate the graph structure or the inferences that can be drawn from these axioms.

Relations between classes [26] have traditionally been used for biological and biomedical ontologies, with the intention to represent *axiom patterns* that hold between two classes. In its simplest form, a relation *is-a* between two classes X and Y expresses a subclass axiom that holds between the two classes. However, many ontologies employ more complex axiom patterns, such as the *Part-of* pattern between two classes: X and Y are said to stand in the relation *Part-of* if and only if X is a subclass of `part-of some Y` [8]. In ontology repositories and ontology editors, these kinds of relations are rarely shown, with OLS and OBO-Edit as exceptions.

Another key distinguishing feature is whether only asserted axioms in the ontology are visualized or also inferred statements. In Protégé, and to some degree in OBO-Edit, it is possible to explore inferred relations between classes visually. In Protégé, these relations are limited to subclass axioms, while OBO-Edit is also able to show other kinds of relations.

Finally, visualization can also aid to structurally identify differences between ontologies, or between different versions of one ontology. Similarly, a key component in exploring and visualizing differences in ontology versions is whether only syntactic changes are identified or whether differences are also identified based on inferred axioms.

Here, we present an ontology visualization environment that provides a simple and intuitive way to represent classes in ontologies and their interrelations. The visualization environment employs an automated reasoner to identify axiom patterns that hold between two classes, thereby visualizing the inferences that can be drawn from an ontology, including complex patterns that represent more than simple subclass relations. The environment can also be used to visualize multiple ontologies at the same time, thereby enabling the exploration of differences between ontologies and ontology versions. The visualization environment we developed is integrated in the AberOWL ontology repository [9], available at http://aber-owl.net, which currently provides access to over 400 ontologies, and thereby allows exploring these ontologies, and their different versions, visually.

2 A brief overview of AberOWL

AberOWL is an ontology repository and framework for ontology-based data access. It allows access to hundreds of ontologies using automated reasoning through a web interface and a REST API. AberOWL is constituted of three main modules: the AberOWL server, the AberOWL synchronization service, and the AberOWL web repository.

The AberOWL server provides the core of the system. It uses the ELK reasoner [14], an OWL reasoner supporting the OWL EL profile [18], to ensure polynominal-time reasoning and querying. The ELK reasoner is fast enough for many practical uses even when applied to large ontologies [25]. The reasoner is used to classify each ontology and the server maintains a classified version of each ontology in memory. From there, it provides a JSON-based REST API for interacting with the ontologies loaded. In particular, the AberOWL server offers the possibility to query one or all ontologies by transforming a Manchester OWL Syntax [10] query string into an OWL class expression using the OWL API and retrieving its sub-, super- or equivalent classes. Additionally, the AberOWL server uses Apache Lucene [29] to create an index of all class and relation labels, synonyms, descriptions, and all other annotation properties, thereby allowing fast retrieval of classes and relations through substring-based search.

The AberOWL synchronization module integrates a service that monitors other ontology repositories for new ontologies as well as new versions of existing ontologies, and incorporates them into the AberOWL server. Currently, only the BioPortal repository [21] is monitored, which contains, amonst others, all the OBO Foundry ontologies [27].

The AberOWL web repository provides a web-based front end which constitutes the main user-interface for the repository. Its function is to allow users to interact with the AberOWL server, providing the possibility to query, browse, download, and visualize explore ontologies. It also makes a set of services built on top of AberOWL available to users, such as SPARQL query expansion or ontology-based PubMed searches.

3 Visualizing ontologies in AberOWL

We developed a visualization environment for ontologies in AberOWL that can visualize inferences drawn from ontologies, visualize both the subclass hierarchy as well as other types of relations between classes, and which can show the differences between the inferences drawn from different versions of an ontology. The aim is to provide an intuitive and easy-to-use method to explore the structure and inferences of ontologies in AberOWL, and visualization of the ontologies is done in real time using the AberOWL reasoning infrastructure.

In AberOWL, ontologies are visualized as directed graphs in which nodes represent classes and edges represent axioms that are inferred to hold between two classes. The subclass hierarchy of an ontology is always shown, and generated by dynamically using the AberOWL reasoning services to query for direct subclasses. A subclass edge is created between two nodes representing classes C and D in ontology O if and only if `C SubClassOf: D` can be inferred from the ontology O and there exists no other class E such that both `C SubClassOf: E` and `E SubClassOf: D`. The root of the subclass hierarchy is `owl:Thing`, and ontologies are initially visualized by querying for direct subclasses of `owl:Thing` using AberOWL. Whenever a user expands a node (by clicking on it) that represents class C, AberOWL is queried for direct subclass of C and the results of the query are generated dynamically as new nodes and linked to the node representing C through directed subclass edges.

To visualize axioms that represent more complex patterns, we follow the relational patterns proposed in the OBO Relation Ontology [26] and its corresponding approximation in OWL [8]. In particular, we identify the set of object properties that occur in an ontology O, and for each object property R in O, we generate a pattern of the type `X SubClassOf: R some Y`, where X and Y are variables standing for classes [8]. Given a node representing the class C in the ontology O, we dynamically generate an R-successor D of this node (with an R-labeled directed edge) if and only if D is a direct subclass of `R some C` in O. For example, to show `part-of` successors of the class *Apoptosis* in the Gene Ontology, we generate the class description `part-of some Apoptosis`, use AberOWL to query for the direct subclasses of `part-of some Apoptosis`, and dynamically generate a new node for each of the resulting classes together with a `part-of`-labeled edge from *Apoptosis* to this new node.

In AberOWL, we can also simultaneously visualize multiple ontologies within the same visualization environment. This is particularly useful to explore differences between multiple versions of the same ontology. AberOWL maintains older versions of ontologies in its repository; however, these versions are not, by default, accessible through automated reasoning. Therefore, when a user request is made to visualize an older version of an ontology, the AberOWL server will first classify this version so that queries can be answered using an automated reasoner. To allow faster subsequent queries to ontology versions, a classified model of these ontologies is kept in memory until it has not been queried for at least 90 minutes, at which time it is removed. We then use our visualization environment to show the subclass hierarchy as well as complex axiom patterns for two or more ontology versions simultaneously. If classes are shared between ontology versions, they are represented by the same node; if axiom patterns between two classes hold in two versions of an ontology, they are represented by the same edge. On the other hand, if axiom patterns or subclasses (of class descriptions) differ between versions, multiple different nodes and edges are created and visually distinguished through colors. This allows to visually explore differences in *the inferences* that can be drawn from different ontology versions.

To visually differentiate the origin of the each node (i.e., the ontology version in which it is present), we color-code ontology versions; we further color-code object properties. To further improve usability of the interface should multiple versions and object properties be selected, we add tooltips to nodes and edges that show the ontology version in which they appear and the kind of axiom pattern that is represented by the edge.

4 Implementation

Our visualization environment is implemented in JavaScript and utilizes the AberOWL reasoning services. Ontologies are visualized through several recursive functions, allowing accurate control over the growth of the tree. As inputs, the implementation of the algorithm requires:

- The root node of the ontology; by default, `owl:Thing` is used for all ontologies.
- The ID (or URI) of the ontology to visualize.
- A list of versions of the selected ontology that are visualized in parallel.
- A list of the object properties in the ontology, from which we generate axiom patterns and visualize them as additional edges.

Furthermore, the ontology visualization environment can be configured with additional parameters:

- The number of children that are shown for each ontology level; in case a node has many successors, only a subset of the successors is shown while the other nodes can be shown on request. This allows us to limit the number of classes that are shown in order to improve usability.
- The number of levels that are expanded through a single request; this allows us to show more than just direct successors of a node in a single request.
- The number of hierarchical levels that will be pre-loaded from the AberOWL server during the ontology visualization. The goal of this parameter is to optimize the load time when users expand additional nodes.

We use the JavaScript Promise pattern and AJAX together with the AberOWL REST API to generate new nodes and edges based on user requests, and to pre-load nodes that users may want to expand further. We use the D3js graph library to generate the resulting graphs. In particular, we use node-link diagrams, implemented in D3js, to represent the ontologies in AberOWL. Node-link diagrams can be used to visualize both acyclic and cyclic graphs and therefore allows us the flexibility to visualize multiple types of relational patterns between classes. Whenever a user changes the choice of which relational patterns or which ontology versions to display, the visualization environment interacts with the AberOWL server in order to regenerate the graph based on the user's selections.

5 Discussion

5.1 Comparison to related work

The significant increase in number of ontologies available online has stimulated the need among the research community to develop visualization tools which

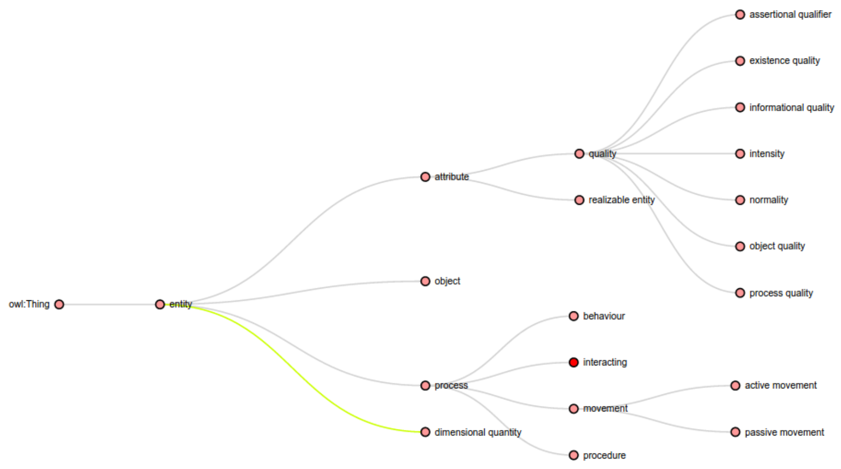

Fig. 1. The image shows an use case example of the Semanticscience Integrated Ontology (SIO) where one version and the object property `has unit` was selected. In the image we can differentiate two kind of edges: gray edges represent subclass relations between two classes, while the yellow edge represents a `has unit` association (i.e., `'dimensional quality'` is a direct subclass of `'has unit' some owl:Thing`).

support their navigation. However, the visualization of ontologies is not an easy task, since ontologies are expressed as formal theories (i.e., sets of axioms) from which inferences can be drawn, and visualizing the kind of inferences is challenging.

One way for classifying visualization methods is the number of dimensions used to represent the ontology. Visualization approaches such as OntoSphere [1] and Onto3DViz [6] use three dimensions to visualize ontologies, while methods employed in most ontology editors like Protégé [22] or OBO-Edit [24], and specialized visualization methods such as KC-Viz [19] , OWLViz [11] and GrOWL [17], utilize a two-dimensional representation.

The structure of ontologies can be visualized in two-dimensional space using several different methods [13]. However, the main aim of visualization of ontologies is often to effectively present hierarchical structures to users [30], and, consequently, the most widely used visualization forms are hierarchical graphs or treemaps [11, 3, 28].

The graph representation in ontologies can be a taxonomy (induced by subclass relations between classes in the ontology), or a representation of other types of relations (i.e., axiom patterns that hold) between classes [26, 8]. Strategies for visualizing ontologies also differ in the types of relations between classes that can be visualized. In ontology editors, for most parts the subclass relations in an ontology are shown while other types of axioms that hold between classes are rarely visualized. A prominent exception has been OBO-Edit [24], an ontology editor intended for use by biological domain experts and based on the OBO Flatfile Format, which could show different types of relations between classes beyond subclass relations. However, development on OBO-Edit has recently been abandoned in favor of ontology development environments that are based more on OWL, which rarely show relations other than subclass relations. Our approach can be used to generate graphs that represent any kind of axiom pattern in which two classes occur as variables.

A further distinction between visualization methods is whether they are able to visualize the asserted structure of an ontology or if they can also visualize the ontologies' inferred structure. The Protégé ontology editor, for example, is able to visualize both asserted subclass relations and inferred subclass relations. In AberOWL, only the inferred (subclass or other types of) relations between classes are visualized.

Table 1. Comparison of tools for visualizing ontologies

Tools	Dimensions	Visualization form	Diff techniques	Visualization technique	Visualization of axioms patterns
Aber-OWL [9]	2-Dimension	Node-link	✓	semantic	✓
Ecco [4]	2-Dimension	XSLT	✓	semantic and syntactic	×
GrOWL [17]	2-Dimension	Non-planar graph	×	semantic and syntactic	✓
KC-Viz [19]	2-Dimension	Tree	×	only syntactic	×
PROMPT-Viz [23]	2-Dimension	Horizontal tree layout Treemap layout	✓	only syntactic	×
Protégé [22]	2-Dimension	Tree	×	semantic and syntactic	×
OBO-Edit [24]	2-Dimension	Labeled hierarchical graph	×	semantic and syntactic	✓
Onto3DViz [6]	3-Dimension	3d-Tree	×	semantic	×
OntoSphere3D [1]	3-Dimension	Sphere	×	only syntactic	×
OntoView [15]	2-Dimension	RDF data model	✓	semantic and syntactic	×
OWLViz [11]	2-Dimension	Node-link	✓	semantic and syntactic	×

Ontologies are not static and will evolve due to extensions in their application domain or changes in the shared conceptualization, changes in the scientific knowledge of the domain, or correction of mistakes [16]. As a result of this evolution, different versions of the same ontology arise, and it is often useful to visualize the differences between different versions to understand the changes that may be necessary in applying the ontology within a use case. Research on ontology versioning and ontology evolution has focused on providing collaborative tools for editing ontologies. For instance, PromptDiff [20] is an ontology-versioning environment which, among others functions, is able to track structural changes of different versions of the ontology; OntoView [15] provides a methodology for ontology versioning which allows users specify relations between versions of ontologies; PROMPT-Viz [23] which is a Protégé plugin which provides advanced visualization of location, impact, type and extent of changes that have occurred between versions on an ontology; COntoDiff [7] tracks changes across multiple versions of ontologies; and the diff tool Ecco [4] that incorporates structural and semantic techniques that allows to distinguish effectual and ineffectual changes between ontologies. In AberOWL, we visualize the changes of different versions of ontologies across multiple versions, and using the AberOWL system for automated reasoning, we can visualize not only the direct syntactic changes to an ontology but also their impact of the inferences that can be drawn from them. Table 1 provides an overview over the main features of different ontology visualization approaches.

5.2 Conclusions

We developed a novel visualization environment for biological and biomedical ontologies, and integrated that environment in the AberOWL ontology repository. Using this visualization environment, it is possible to visualize the inferred structure of one ontology, including the structure induced by subclass relations as well as arbitrary axiom patterns that hold between classes [8]. Furthermore, we can visualize multiple versions of a single ontology at once, thereby allowing users to explore structural changes between ontology version. All structural relations between classes in the visualization environment are generated using an OWL reasoner, thereby allowing users to explore the inferences that can be drawn from the ontologies, or their different versions. Our work also demonstrates that the AberOWL system can be used as a service that enables the development of novel kinds of semantic applications using automated reasoning and semantic querying.

References

1. Bosca, A., Bonino, D., Pellegrino, P.: Ontosphere: more than a 3d ontology visualization tool. In: Swap (2005)
2. Cote, R., Jones, P., Apweiler, R., Hermjakob, H.: The ontology lookup service, a lightweight cross-platform tool for controlled vocabulary queries. BMC Bioinformatics 7(1), 97+ (2006), http://dx.doi.org/10.1186/1471-2105-7-97
3. García-Peñalvo, F.J., Colomo-Palacios, R., García, J., Therón, R.: Towards an ontology modeling tool. a validation in software engineering scenarios. Expert Systems with Applications 39(13), 11468–11478 (2012)
4. Gonçalves, R.S., Parsia, B., Sattler, U.: Ecco: A hybrid diff tool for owl 2 ontologies. In: OWLED (2012)
5. Grau, B., Horrocks, I., Motik, B., Parsia, B., Patelschneider, P., Sattler, U.: OWL 2: The next step for OWL. Web Semantics: Science, Services and Agents on the World Wide Web 6(4), 309–322 (November 2008), http://dx.doi.org/10.1016/j.websem.2008.05.001
6. Guo, S.S., Chan, C.W.: A tool for ontology visualizaiton in 3d graphics: Onto3dviz. In: Electrical and Computer Engineering (CCECE), 2010 23rd Canadian Conference on. pp. 1–4. IEEE (2010)

7. Hartung, M., Gross, A., Rahm, E.: COnto-Diff: Generation of Complex Evolution Mappings for Life Science Ontologies. Journal of Biomedical Informatics 46, 15–32 (2013)
8. Hoehndorf, R., Oellrich, A., Dumontier, M., Kelso, J., Rebholz-Schuhmann, D., Herre, H.: Relations as patterns: Bridging the gap between OBO and OWL. BMC Bioinformatics 11(1), 441+ (2010)
9. Hoehndorf, R., Slater, L., Schofield, P.N., Gkoutos, G.V.: Aber-OWL: a framework for ontology-based data access in biology. BMC Bioinformatics 16, 26 (2015), http://www.biomedcentral.com/1471-2105/16/26/abstract
10. Horridge, M., Drummond, N., Goodwin, J., Rector, A., Stevens, R., Wang, H.: The Manchester OWL Syntax. Proc. of the 2006 OWL Experiences and Directions Workshop (OWL-ED2006) (2006)
11. Horridge, M.: Owlviz (2004)
12. 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/
13. Katifori, A., Halatsis, C., Lepouras, G., Vassilakis, C., Giannopoulou, E.: Ontology visualization methodsa survey. ACM Computing Surveys (CSUR) 39(4), 10 (2007)
14. Kazakov, Y., Krötzsch, M., Simancik, F.: The incredible elk. Journal of Automated Reasoning 53(1), 1–61 (2014), http://dx.doi.org/10.1007/s10817-013-9296-3
15. Klein, M., Fensel, D., Kiryakov, A., Ognyanov, D.: Ontology versioning and change detection on the web. In: Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web, pp. 197–212. Springer (2002)
16. Klein, M.C., Fensel, D.: Ontology versioning on the semantic web. In: SWWS. pp. 75–91 (2001)
17. Krivov, S., Williams, R., Villa, F.: Growl: A tool for visualization and editing of owl ontologies. Web Semantics: Science, Services and Agents on the World Wide Web 5(2), 54–57 (2007)
18. Motik, B., Grau, B.C., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C.: Owl 2 web ontology language: Profiles. Recommendation, World Wide Web Consortium (W3C) (2009)
19. Motta, E., Mulholland, P., Peroni, S., dAquin, M., Gomez-Perez, J.M., Mendez, V., Zablith, F.: A novel approach to visualizing and navigating ontologies. In: The Semantic Web–ISWC 2011, pp. 470–486. Springer (2011)
20. Noy, N.F., Kunnatur, S., Klein, M., Musen, M.A.: Tracking changes during ontology evolution. In: The Semantic Web–ISWC 2004, pp. 259–273. Springer (2004)
21. Noy, N.F., Shah, N.H., Whetzel, P.L., Dai, B., Dorf, M., Griffith, N., Jonquet, C., Rubin, D.L., Storey, M.A.A., Chute, C.G., Musen, M.A.: Bioportal: ontologies and integrated data resources at the click of a mouse. Nucleic acids research 37(Web Server issue), W170–173 (July 2009), http://dx.doi.org/10.1093/nar/gkp440
22. Noy, N.F., Sintek, M., Decker, S., Crubezy, M., Fergerson, R.W., Musen, M.A.: Creating semantic web contents with Protege-2000. IEEE Intelligent Systems 16(2), 60–71 (March/April 2001)
23. Perrin, D.S.J.: Prompt-viz: Ontology version comparison visualizations with treemaps. Ph.D. thesis, Citeseer (2004)
24. Richter, J.D., Harris, M.A.A., Haendel, M., Lewis, S.: Obo-edit - an ontology editor for biologists. Bioinformatics (June 2007), http://dx.doi.org/10.1093/bioinformatics/btm112
25. Slater, L., Gkoutos, G., Schofield, P.N., Hoehndorf, R.: Using aber-owl for fast and scalable reasoning over bioportal ontologies. In: Proceedings of Interna-

tional Conference on Biomedical Ontologies (ICBO). pp. 72–76 (July 2015), http://icbo2015.fc.ul.pt/ICBO2015Proceedings.pdf

26. Smith, B., Ceusters, W., Klagges, B., Köhler, J., Kumar, A., Lomax, J., Mungall, C., Neuhaus, F., Rector, A.L., Rosse, C.: Relations in biomedical ontologies. Genome Biol 6(5), R46 (2005), http://dx.doi.org/10.1186/gb-2005-6-5-r46
27. Smith, B., Ashburner, M., Rosse, C., Bard, J., Bug, W., Ceusters, W., Goldberg, L.J., Eilbeck, K., Ireland, A., Mungall, C.J., Leontis, N., Serra, P.R., Ruttenberg, A., Sansone, S.A., Scheuermann, R.H., Shah, N., Whetzel, P.L., Lewis, S.: The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration. Nat Biotech 25(11), 1251–1255 (2007)
28. Storey, M.A., Musen, M., Silva, J., Best, C., Ernst, N., Fergerson, R., Noy, N.: Jambalaya: Interactive visualization to enhance ontology authoring and knowledge acquisition in protégé. In: Workshop on Interactive Tools for Knowledge Capture (K-CAP-2001). p. 93. Citeseer (2001)
29. The Apache Software Foundation: Apache Lucene. http://lucene.apache.org
30. Wang, T.D., Parsia, B.: CropCircles: topology sensitive visualization of OWL class hierarchies. Springer (2006)
31. Xiang, Z., Mungall, C.J., Ruttenberg, A., He, Y.: Ontobee: A linked data server and browser for ontology terms. In: Proceedings of International Conference on Biomedical Ontology. pp. 279–281 (2011)