Multi-view Ontology Visualization

Julia Dmitrieva, Fons J. Verbeek
Imaging & BioInformatics, LIACS, Universiteit Leiden, The Netherlands

Abstract

Available visualization tools are mostly devoted to representation of an ontology only in one certain geometry and in one chosen dimension. We present our ontology visualization approach where the reasoner is used to transform an ontology to the graph structure. This graph structure can be visualized with 5 *different geometrical representations, with different level of depth, and with a subset of properties of the user choice. Augmented with navigation functionalities our visualization becomes a dynamic ontology browser.*

1 Introduction

With ontologies researchers can represent, share and integrate knowledge. Currently, there are different tools available for modeling and editing ontologies, e.g. Protégé[6] and OntoEdit[3]. Besides modeling and editing also visualization methods are needed, where a knowledge base can be inspected at different levels of granularity by means of interaction and navigation. From the most comprehensive analysis[8] we conclude that there are no visualization tools available that can combine 2D and 3D dimensional views, and which can explore an ontology in different geometries. In each tool the visualization is realized in one particular geometry and in one particular dimension. With our work we introduce the ontology visualization approach where an ontology can be represented as a graph structure and visualized by means of different geometrical models.

2 Graph Generation

Ontologies can be very diverse, some are very straightforward, and only represent the taxonomy. Other ontologies are very complex and based on rich constructors of Description Logic [7]. In NCI-THESAURUS ontology concepts are frequently defined as union of intersections of other concepts, e.g. $C \sqsubseteq A \sqcap ((\exists R_1.B_1 \sqcap B_2) \sqcup (C_1 \sqcap \exists R_2.C_2))$. Because of the diversity and complexity of constructors, we need to find a way to represent both the taxonomy as well as other kinds of *connections* between the concepts. In our approach we represent an ontology as a graph structure. We use a reasoner services to generate a taxonomy and infer another relationships between concepts. The transformation to graph structure happens as follows:

1. The reasoner finds all subclasses of the given concept C: If $B \sqsubseteq C$ then C is connected to B with a red edge.

2. The reasoner finds all superclasses of the given concept C: If $C \sqsubseteq B$ then C is connected to B with a green edge.

3. Inferring that the C and E are related to each other via R requires extraction of properties from axioms of the concept. Then the reasoner can be asked whether the concept $C \sqcap \neg\exists R.E$ is satisfiable, if not then it is necessary for the class C to have the property R with the filler E.

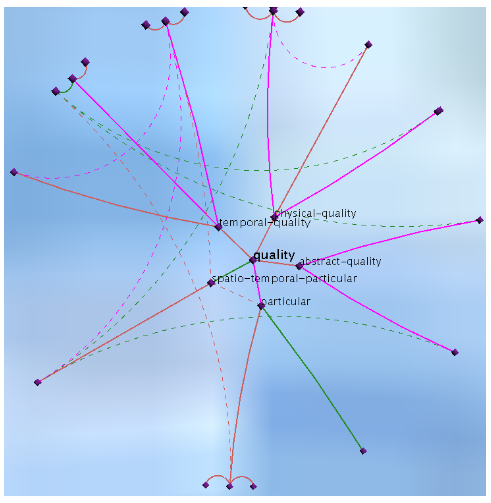
This procedure begins with the root concept and recursively calls itself till the certain level of depth is reached. This level of depth is a parameter of the user interface.

3 Visualization

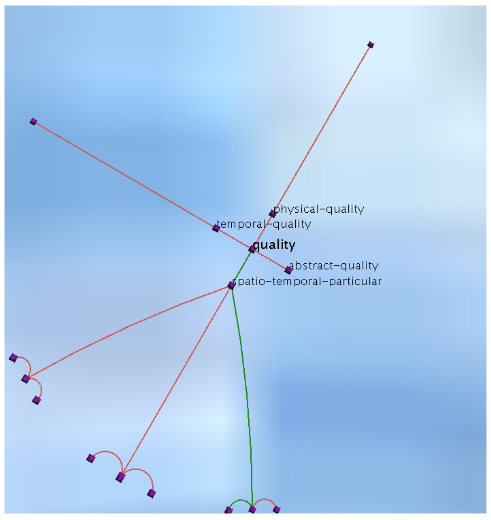
The visualized graph data-structure is directly generated from the ontology (OWL/OBO file) by means of the transformation process. The graph can be generated for different levels of depth. Level "1" means that only direct sub/super classes and the concepts that are directly related to the root concept are represented. In level ”2” the children concepts of the level ”1” concepts are generated. In the visualization a user can navigate through the graph and expand each concept. In addition, a simple query interface is provided, where the user can retrieve the concepts which are related to the given query term. With a *right-mouse click* the definition of a concept can be retrieved; the definition is an annotation property and has no semantic meaning, it contains the specific domain knowledge and can be of interest to the specialist.

3.1 Representations of Views Based on Different Relations

In knowledge bases the concepts are interconnected to each other with different kinds of relations. The visualization of the graph with all possible relations can be too complicated for the user; consequently, the complex web of edges may hide important information. Moreover, the user can be interested only in a particular part of ontology where the concepts are related via a subset of the properties. This representation is referred as a view. The main idea of the view generation was based on the work of Noy and Musen[11], in which a *Traversal View* was defined as a self-contained portion of ontology. We elaborate on this idea for our visualization, with that difference that in place of extraction of self-contained portion of an ontology we visualize a graph structure that represents this part of the ontology (cf. Figure 1). In the current version of our application we provide only the global level of traversal depth, this means that all the properties will be traversed with the same depth. From the list of properties, the user can select a subset of interest. On basis of this selection the graph structure will be generated only for a subset of the ontology. This subset contains the concepts that are reachable from the central concept by the defined set of properties and the defined depth of the traversal.



(a) Visualization of the concept ”quality” from Dolce-Lite ontology with all relations

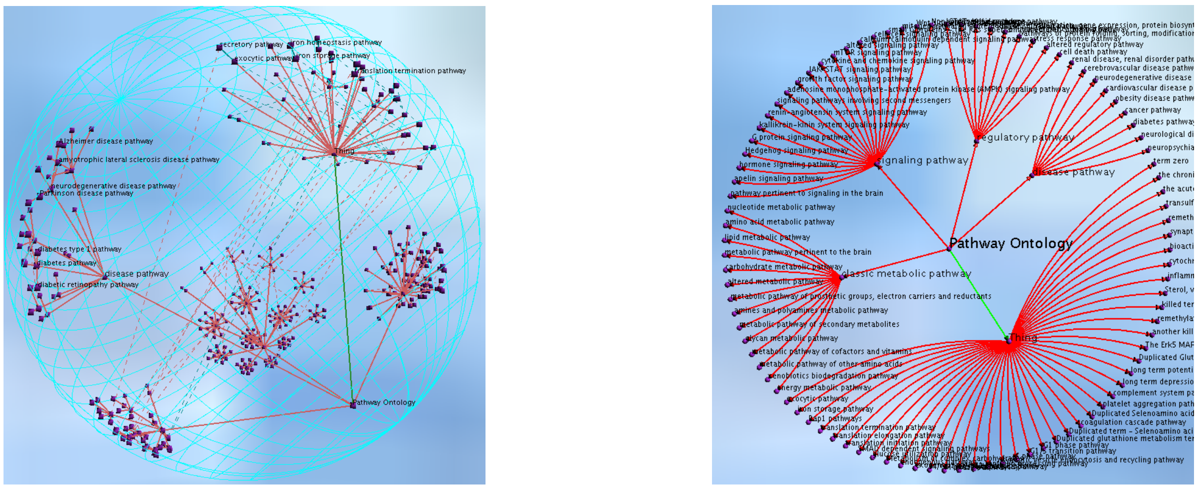


(b) The concept ”quality” with all properties and with the level of depth = 2

Figure 1: Visualization of the concept ”quality” from Dolce-Lite ontology

3.2 Representations of Views Based on Different Geometry

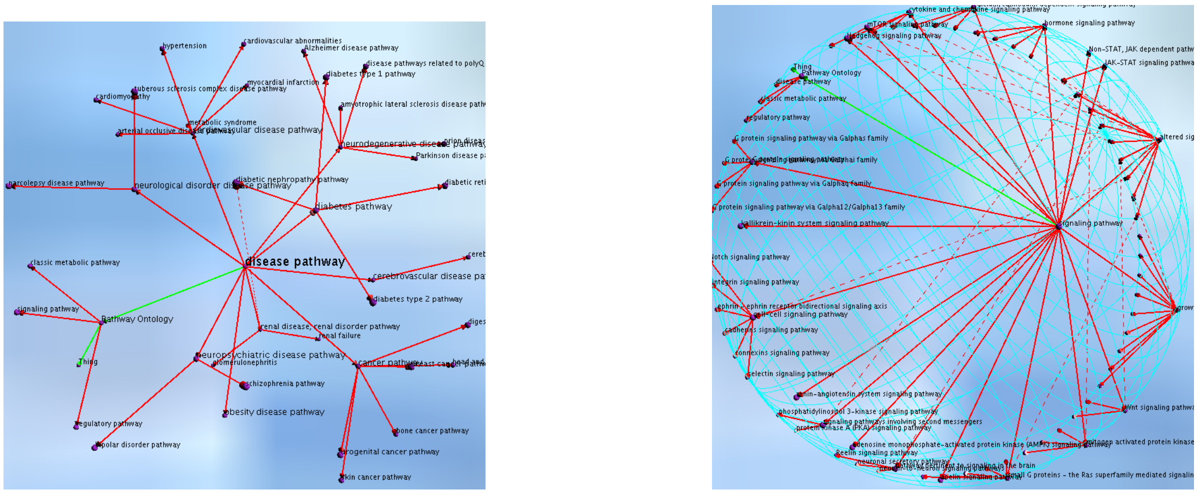
We have implemented two Euclidean views; i.e. *Sphere* and *Disk*. In the *Sphere* visualization, the root concept is placed in the center of the visualization window. The sub/super classes and related concepts surround the root concept and are evenly spaced on the imaginary sphere surface (cf. Figure 3). Two other representations, i.e. the Klein Model, and the Poincaré Disk model (cf. Figure 2), are based on the hyperbolic geometry and realize the so called ”focus + context” techniques[9, 10]. In addition, we have also implemented the *Stereographic* view (cf. Figure 3), where the graph structure is laid out at the surface of a sphere. All the views are augmented with the corresponding geometrical transformations. In Euclidean view the Euclidean transforms are implemented, whereas for the Klein model the hyperbolic transforms[12] are applied, and in Poincaré model the Möbius transformations are used.



(a) Visualization of Pathway ontology with Klein Model

(b) Visualization of Pathway ontology with Poincaré Model

Figure 2: PATHWAY concept with the hyperbolic geometry



(a) Visualization of Pathway ontology with Sphere Model

(b) Visualization of Pathway ontology with Stereographic Model

Figure 3: PATHWAY concept with different geometries

4 Implementation

The implementation was realized in Java. To generate the *virtual* graph structure, on basis of defined depth and properties, we have used two additional API's. The OWL-API [4] is used to parse the ontology and, more specifically, to extract the concept definitions containing the restriction axioms. The Pellet reasoner [5] is used to extract the inferred hierarchy. All visualization components are based on the Java-3D API [1].

5 Conclusions and Discussion

In this paper we have presented our approach on information visualization from ontologies. In our visualization method we represent an ontology as a graph structure. This graph structure is based on inferred hierarchy that is calculated by the reasoner.

Because the visualization of all the connections between the concepts can be confusing, we provide the user the possibility to filter the properties and get the requested part of the ontology. In order to explore an ontology, different geometrical representations can be generated. This multi-view approach can be helpful for the user, because by different geometrical representation an ontology can be shown with emphasizing of different part of the structure. For example, the hyperbolic views are suitable to represent a global structure of an ontology, while the Euclidean views are good to visualize a local structure. Our visualization methodology[2] is on-line for probing and inspection as a Java Web Start application.

At present time we are working on transformation of our application to Protégé [6] plugin.

References

[1] Java 3d api. `http://java.sun.com/products/java-media/3D/`.

[2] Multi-view ontology visualization. `http://www.liacs.nl/~jdmitrie/ontVis/OntologyVisualization.html`.

[3] Ontoedit an ontology engineering environment. `http://www.ontoknowledge.org/tools/ontoedit.shtml`.

[4] The owl-api. `http://owlapi.sourceforge.net/index.html`.

[5] Pellet. `http://pellet.owldl.com/`.

[6] Protégé, ontology editor and knowledge-base framework. `http://protege.stanford.edu/`.

[7] *The Description Logic Handbook.* Cambridge University Press, Cambridge, 2002.

[8] A. Katifori and C. Halatsis. Ontology visualization methods - a survey. *AMC Computing Surveys*, 39(4), 2007.

[9] J. Lamping and R. Rao. The hyperbolic browser: A focus + context technique for visualizing large hierarchies. *Journal of Visual Languages and Computing*, 7:33–35, 1996.

[10] T. Munzner. H3: laying out large directed graphs in 3d hyperbolic space. *Information Visualization, IEEE Symposium on*, 1997.

[11] N. F. Noy and M. A. Musen. Specifying ontology views by traversal. *LNCS*, 3298:713–725, 2004.

[12] M. Phillips and C. Gunn. Visualizing hyperbolic space: Unusual uses of 4×4 matrices. 1991.