Understanding Large Volumes of Interconnected Individuals by Visual Exploration

(System Description)

Olaf Noppens and Thorsten Liebig

Institute of Artifical Intelligence
Ulm University
Ulm, Germany
{olaf.noppens|thorsten.liebig}@uni-ulm.de

Abstract. Ontologies are now used within an increasing number of real-world applications. So far, significant effort has been spend in building tools to support users in creating, maintaining, and browsing the terminological part of an ontology. On the other hand, only little work has been done in supporting the user to explore the manifold interconnected assertional knowledge in order to analyze, visualize, and understand this network of individuals. In this paper, we present a new efficient visualization and editing approach which allows to investigate relationships within large volumes of interlinked individuals in order to grasp the structure of the assertional knowledge more easily.

1 Motivation

An OWL ontology typically consists of two parts [2]: The terminological or schema part introduces concepts and properties and gives structure to them in terms of axioms using the available OWL Lite or OWL DL language constructs (also called TBox in Description Logics). The assertional or data part defines concrete individuals and relationships between those individuals (also called ABox) utilizing the concepts and properties of the terminology.

As an example, consider the ontology given by the widely known Lehigh University Benchmark (LUBM) [6]. This test suite has become a de-facto standard for measuring reasoning performance and consists of an ontology covering basic elements in the domain of tertiary education. For instance, its TBox defines terms like `Department`, `Student`, and `Course` as well as properties like `subOrganisationOf`, `takesCourse`, and `headOf`. A corresponding ABox covers virtual universities, faculties, research groups, etc. which are randomly populated with individuals such as `FullProfessor3`, `GraduateStudent27`, `GraduateCourse6`. These individuals are related via property instantiations, e. g. `GraduateStudent27 takesCourse GraduateCourse6`. A huge ontology of this kind obviously is characterized by a large number of individuals (i. e. students, departments, etc.), property instantiations (course attendees, employees, etc.) and a constant number of terms within the terminological schema. In fact, within a typical real-world application, the TBox is assumed to be much smaller in comparison to the amount of assertions in the ABox.

New reasoning systems as well as recent system optimizations have shown significant increase in speed for answering conjunctive queries with respect to LUBM test cases containing several hundred thousands of individuals [13,14,18]. Even if these results rely on synthetically generated data produced by a relatively simple benchmark generator such as the LUBM, they clearly demonstrate promising progress in the development of scalable reasoning systems.

On the other hand, little has yet been done to support ontology users or developers to visually edit or explore such large volumes of interrelated individuals. There is currently no representation approach or even a tool to gradually inspect an individual with respect to its direct and indirect fillers regarding a transitive property. For instance, within the university domain one could be interested in stepping through the `subOrganisationOf` relationship up to a specific institute in order to expand all the persons which are advised by the corresponding chair. Or within the Gene Ontology (GO) [4], which contains millions of individuals, users are presumably interested in investigating which specific DNA binding product interacts with which kind of receptors for example.

Our approach utilizes a user-driven visualization strategy, making use of animated expansion steps, clustering techniques, and different levels of detail views. It allows for operations on a single individual or on sets of individuals and takes property features such as transitivity, symmetry, etc. into account.

1.1 Visualizing Entailed Assertions

Note that our visualization approach inherently requires reasoning. Since OWL allows for property hierarchies as well as symmetrical, transitive, or functional properties, a reliable reasoning system such as RacerPro [7] is needed to make implicit property instantiations explicitly available. Without reasoning feedback an ABox visualizer would degrade to a syntax imaging tool, probably hiding the most important correlations. For instance, in case of `has-child` being a subproperty of a transitive property `has-descendant`, there implicitly exists a `has-descendant` link between an individual and all its children. Another example deals with the fact that in OWL individuals are not disjoint by default. As a consequence two fillers of a functional property (a property with at most one filler per source) will be merged to one individual by the inference engine. Thus, an ABox visualization should also render them as one object bearing two identifiers.

Therefore, the underlying presentation principle of our ABox visualizer is to always render the semantical implied relationships between individuals in order to make effectively visible what is entailed in the ontology. We argue that any serious ontology authoring or browsing tool should allow users to explore implicitly modeled as well as explicitly given information, while being able to distinguish between both. In this sense this component is a direct extension of our OWL TBox authoring tool OntoTrack [12].

1.2 Related Work

As mentioned before, there is poor tool support in browsing and editing ABox data. There are some OWL editors which allow for inspection of single individuals with help of selection lists and standard form elements such as Protégé [10] or SWOOP [8]. However, they only provide one level of detail and have no interface for exploring the fillers of more than one property in parallel. Moreover, list-style and table-based approaches are not adequate techniques to analyze large amounts of data and are less practical in showing a chain of property fillers in a clear and concise manner (e. g. to follow the transitivity of properties).

Tools such as GrOWL [11] try to offer a graph based interface which uses graphical icons to depict different types of nodes (class, property, or individual) as well as language constructs (negation, union, etc.). GrOWL provides a spring layout which dynamically adds nodes to the graph on user demand. However, this functionality is not sufficiently fine-graded enough for the task of gradually inspecting property fillers. In addition, the resulting graph is somewhat verbose and may distract the user due to frequent layout changes or node agglutination.

On the triple-based representation level of RDF, individuals are just as any other resources. Furthermore, since OWL is layered on top of RDF Schema, there is a lack of expressivity in RDF needed to capture the semantics of an OWL ABox appropriately. As said before, the graph representation of the RDF data model is conceptually different from the graph structure of an ABox even without considering entailed statements.

Nevertheless, RDF Gravity [5] effectively supports a user in filtering a RDF graph by means of selecting resources or specifying RDQL queries but is not ideal with respect to a user-friendly layout.

2 Interactive Exploring of Individuals

Exploring Large Volumes of Interrelated Individuals. When visualizing large data volumes there always is a trade-of between detailedness and overview. One lesson learnt from the visual analysis of large data sets in general is that it is not advisable to arbitrarily visualize all dependencies, particulars etc. at any time [9]. Hence, our approach tries to provide detail information on user demand while offering an overview at the same time. Here we adopt the single-view visualizing paradigm enabling selective detailed views which has turned out to be adequate for visualizing concept and property hierarchies as invented by our ontology authoring framework OntoTrack.

Our visualization paradigm is based on a user-directed exploration of interrelated individuals which does consider an individual as a first-class element but also allows to group them within clusters as following. Starting with a user selected individual, one can interactively exploit the property fillers (respectively datatype values in the case of datatype properties) of each property the individual is related to in a step-wise fashion. For instance, the LUBM ontology defines a property named `takesCourse` which relates students with courses, e. g. `GraduateStudent27 takesCourse Course10` as well as `Course21`.

Exploring on User-Demanded Expansion. Figure 1 shows that the root is related via the property `subOrganizationOf` with exactly one other individual. When hovering over an individual with the mouse pointer and clicking the middle mouse button, the tool offers a preview depicting all the properties together with their number of fillers (as long as they have at least one). From our experience, it is often the case that a "natural" exploration of an ontology will typically not only include a directed exploration with respect to a property but also its inverse direction even if there is no inverse property explicitly defined (to provide a bidirectional exploration). For instance, when having all courses a student takes one would like to explore other students of a specific course. In Figure 1 there are filler of four other properties to which the actual root is related to in an inverse fashion. In order to denote the different directions a small arrow indicates whether the actual root is the source of the property (right arrow) or a filler (left arrow).

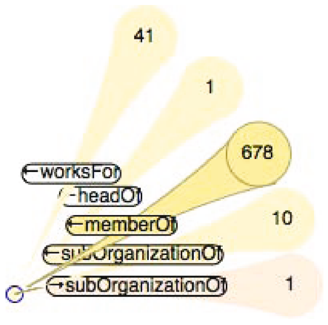


Fig. 1. Preview context menu for expanding property fillers

After expansion, all fillers with respect to the selected property are grouped within a so-called *property filler cluster* which will be drawn as a a club originating from the individual which is considered as the source of the expansion (e. g. `takesCourse` in Figure 2). Each individual within the cluster are rendered as circles whose labels are accessible via mouse-over tooltips. Due to a standardized package algorithm the clusters diameter approximates the number of individuals and allows to easily compare different filler sets by their rendering size.

At any time the user can guide his exploration by choosing a follow up root from the individuals within the currently visible clusters or may branch by selecting other properties for further expansion. To distinguish different properties different colors are used. In addition each club carries its corresponding property label. One can optionally switch of in-view label rendering. Then there will be a list of colored property names in order to denote which property is represented by which color. However, in both cases, the color of the property club and the property label is always the same for clubs which are based on the same property. In a similar manner, different colors are uses for the different types of fillers (i. e. individuals versus data values).

Figure 2 shows an example snapshot of a partially expanded LUBM ontology containing five universities (approx. 60 thousand overall individuals). Following the first expansion level at the very bottom of Figure 2 it is easy to perceive that the root individual (an undergraduate student) takes two courses. Furthermore, one of this courses has more than 30 other participants from which one takes four courses. The root individual also is member of an university with over 40 employees (see → `memberOf` ← `worksFor` expansion). Any changes in the layout such as (de-)expanding clusters are animated to easily grasp the differences between two exploration states. Beyond that the whole layout can continuously be zoomed or paned simply by mouse-down movements.

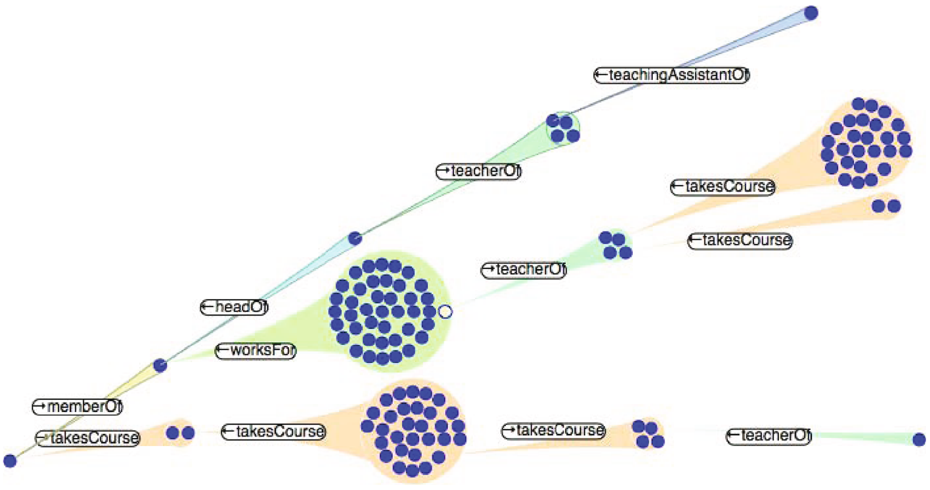


Fig. 2. Partial expanded property fillers

Close-By Detail Information. In our first prototype [15] we followed exactly the *detailed view* mode paradigm of OntoTrack to provide further information for individuals, such as name, via mouse-wheel usage. However, displaying individuals names within clusters results in diluting the correlation between the property filler cluster's diameter and its number of fillers. We therefore decided to provide an additional area for detailed information instead. When hovering over a property filler cluster with the mouse pointer, all contained individuals are displayed in the detailed view area at the right hand side of the visualization as shown in Figure 3. Here, further detailed levels can be activated or de-activated using the mouse-wheel, e. g. to display (in a non-graph-based way) all direct classes the selected individual is instance of, or even to display told information.

Identifiers, Labels, and Names. When showing the name of an individual the question arises what does the name mean and where it comes from. Our experience with our first prototype attested the feeling that for exploring an ontology more naturally, names as typically provided with a RDF label annotation is better suited. However, as these labels are just labels and does not carry any semantics or constraints there are not always used. For instance, to express that each individual which is instance of `Person` should have a name, this is typically modeled as restrictions with respect to a property `hasName`. Therefore, our visualization offers the possibility not only to use the local name of the URI or the label (if any) but also to specify a (datatype) property whose filler is used for labeling the graphical representation. Note that there must exist such a filler.

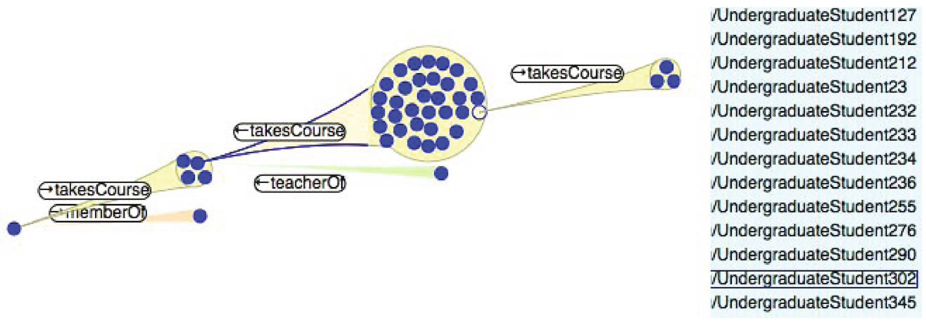
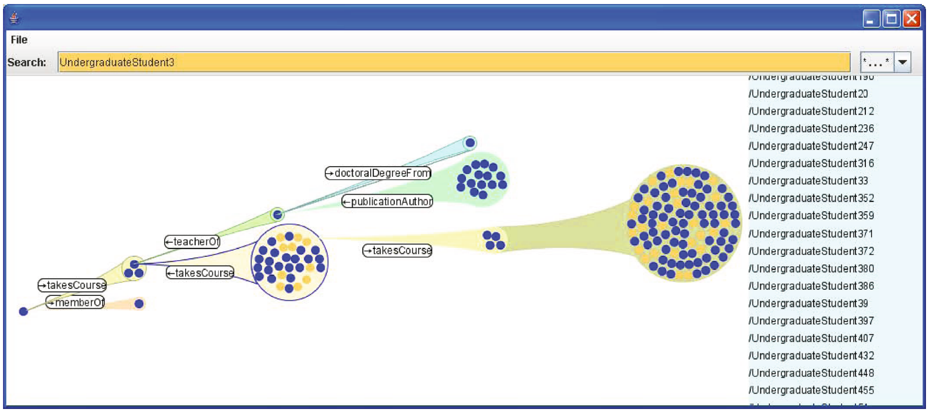


Fig. 3. Sampled detailed view for the individuals of the second property filler cluster `takesCourse`

Cloning in Favor of a Concise Visualization. From a logical perspective, one and the same individual can be related to another individual via different properties and may appear multiple times within the expansion path of a specific property, e. g. due to cycles or inverse exploration. A visualization can either use one single graphical representation for each individual or has to use a cloned graphical representation for different occurrences of the same individual. We decided to allow for clones representations in clusters containing one an the same individual because otherwise a path-directed expansion would no longer be possible. However, if a user hovers with the mouse pointer over an individual, all its visible representations are highlighted simultaneously.

Exploring at Different Levels of Granularity. In addition to expand a single individual it is also possible to expand the fillers of a whole cluster with respect to a property as shown in Figure 4. This can be done by choosing the club itself (border or background) rather than a specific individual as expansion source. The emerging filler cluster then contains the union of all fillers of the predecessor cluster. In case of already expanded clubs (with respect to that specific property), all their individuals move into the new cluster in an animated manner which may serve as a simple visual explanation of the underlying action semantics.

Distinguishing Between Inferred and Told Information. According to the semantics of a *transitive property* an individual is related to all directly or indirectly related fillers reachable via this property. Therefore, when expanding the fillers of a transitive property with respect to an individual (or cluster of



Search: UndergraduateStudent3

/UndergraduateStudent20
/UndergraduateStudent212
/UndergraduateStudent236
/UndergraduateStudent247
/UndergraduateStudent316
/UndergraduateStudent33
/UndergraduateStudent352
/UndergraduateStudent359
/UndergraduateStudent371
/UndergraduateStudent372
/UndergraduateStudent380
/UndergraduateStudent386
/UndergraduateStudent39
/UndergraduateStudent397
/UndergraduateStudent407
/UndergraduateStudent432
/UndergraduateStudent448
/UndergraduateStudent455

Fig. 4. Cluster expansion and instant highlighting of search matches

individuals) the transitive closure of related fillers are shown. One can, however, distinguish between directly and indirectly related fillers by expanding the next level. The first expansion will then only contains the *direct* property fillers whereas all other fillers are transferred into the subsequent property cluster in an animated fashion. For instance, the upper part of Figure 5 shows the first expansion level of the transitive property `subOrganizationOf`, which contains

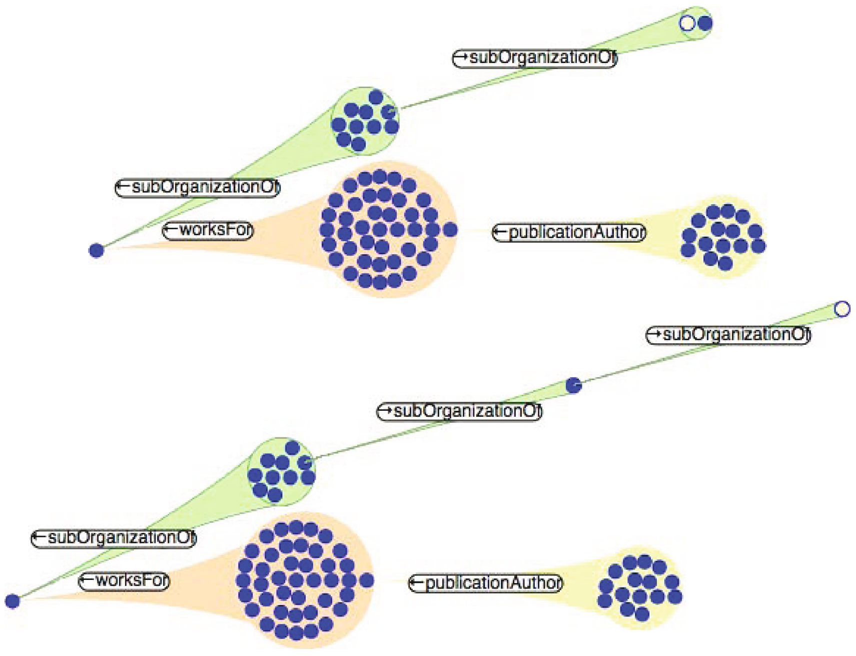


Fig. 5. Follow up expansion of a transitive property

two individuals which are reachable from the selected individual in the inverse expanded property filler cluster. After expanding the next level (lower part of Figure 5) it becomes apparent that one of the individuals now has moved to the next expansion level.

Searching and Editing Features. Our ABox visualizer also supports OntoTrack's search features which are based on dynamic queries [17] which also can be found in tools like SpaceTree [16]. For instance, during a string based search process the user starts typing an individual or property name and all matching entities are instantly highlighted. Each additional character or deletion in the search string directly results in an updated highlighting of the matching individuals as shown in Figure 4. In addition to browsing, the ABox visualizer also supports rudimentary editing features such as the removal and addition of property fillers. One can add an individual to a filler set by using the drag 'n drop paradigm. For instance, a user can drag an individual out of a cluster into another one or onto the work space. The result of the former is a changed filler set membership, the latter will remove the instance from the original filler set. However, removing an individual from all filler sets does not remove the individual from the underlying ontology. For those kind of editing actions additional authoring features still have to be developed.

3 Implementation and Current Work

During our implementation of a first prototype [15] we discovered several problems with respect to the visualization paradigm and the communication performance with our external reasoning system. These experiences were carefully considered when designing the current revision of our ABox visualizer. As a recent analysis has shown, ontologies typically consist of several hundreds of individuals [19]. Obviously, scalability and performance are very important issues for user-friendly tools. From a graphical perspective, we address scalability and performance with our decision for the Piccolo framework [3] because of our positive experience with large numbers of graphical objects in the ontology authoring framework OntoTrack.

Following from our description above, our ABox tool is linked with an OWL inference engine in order to be able to visualize entailed knowledge. Querying for all entailed knowledge at start up is obviously not a feasible solution when dealing with large volumes of data. As we are currently not able to use the standard DIG 1.1 interface [1] for reasoner communication because it only supports very basic ABox queries, we use in the meantime the RacerPro reasoner via its native syntax and query interface over TCP [7]. The use of the nRQL query language gives us also the possibility for an query optimization: we query for the first property fillers in advance which are successors of the current visible clusters[1]. The querying of the next answer tuples is then organized following the *last come,*

[1] The first ones are those which are "easily" to compute by the reasoner. For more information to that subject please refer to Racer's nRQL language.

first serve principle. If the user exploits other branches of the property fillers those will be computed first. The graphical representation also gives feedback about the process of querying, i. e. the question mark symbol is shown in the middle of a property filler cluster when some individuals are still missing until the reasoner answered the query for the remaining fillers. Note that the upcoming new version of DIG, namely DIG 2.0, will support more fine-graded and more expressive queries.

Current work deals with focus and thumbnail techniques in order to preserve an overview of the data during exploration operations. Whenever a cluster would occupy to much screen space, i. e. when the size will exceed a certain value, it will automatically switch to a more compact visualization using a thumbnail representation. We also plan to optimize the layout algorithm with respect to individual or club placing and better visualization of the starting points of property filler clusters. Our experience with the first prototype showed us some improvements we have implemented and therefore plan to conduct a user study in order to gain real-user feedback.

4 Summary and Outlook

In this paper we presented a novel approach for exploring large volumes of individuals within ontology languages such as OWL. In contrast to other visualization tools we focus on a visualization of semantically interrelated individuals. The system therefore incorporates a reasoning system to discover, for instance, direct and indirect fillers, symmetric, functional and transitive properties to provide a semantically correct visualization and not only told information. Our implementation allows for incremental inspection of filler sets, selective browsing and applies several abstractive visualization techniques not found in current tools. First experiences with volumes containing several thousands of individuals such as in the synthetical LUBM benchmark ontologies are encouraging.

References

1. Sean Bechhofer. The DIG Description Logics Interface: DIG/1.1. Technical report, University of Manchester, 2003.
2. Sean Bechhofer, Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L. McGuinness, Peter Patel-Schneider, and Lynn Andrea Stein. OWL Web Ontology Language Reference. W3C Recommendation, February 2004.
3. Ben Bederson, Jesse Grosjean, and Jon Meyer. Toolkit Design for Interactive Structured Graphics. Technical Report CS-TR-4432, University of Maryland, January 2002.
4. The Gene Ontology Consortium. Gene Ontology: tool for the unification of biology. *Nature Genetics*, 25:25–29, May 2000.
5. Sunil Goyal and Rupert Westenthaler. RDF Gravity. Salzburg Research, 2004. http://semweb.salzburgresearch.at/apps/rdf-gravity/.
6. Yuanbo Guo, Zhengxiang Pan, and Jeff Heflin. LUBM: a benchmark for OWL knowledge base systems. *Journal of Web Semantics*, 3(2):158–128, July 2005.

7. Volker Haarslev and Ralf Möller. Racer: A Core Inference Engine for the Semantic Web. In *Proc. of the 2nd Int. Workshop on Evaluation of Ontology-based Tools (EON 2003)*, pages 27–36, 2003.
8. Aditya Kalyanpur, Bijan Parsia, Evren Sirin, Bernardo Cuenca Grau, and James Hendler. Swoop: A Web Ontology Editing Browser. *Journal of Web Semantics*, 4(2), 2006.
9. Daniel A. Keim. Visual exploration of large data sets. *Communications of the ACM*, 44(8):38–44, 2001.
10. Holger Knublauch, Ray W. Fergerson, Natalya F. Noy, and Mark A. Musen. The Protégé OWL Plugin: An Open Development Environment for Semantic Web Applications. In *Proc. of the 3rd Int. Semantic Web Conference (ISWC 2004)*, pages 229 – 243, 2004.
11. Sergey Krivov, Ferdinando Villa, and Rich Williams. GrOWL, visual browser and editor for OWL ontologies. *Journal of Web Semantics*, 2006.
12. Thorsten Liebig and Olaf Noppens. OntoTrack: A semantic approach for ontology authoring. *Journal of Web Semantics*, 3(2):116 – 131, 2005.
13. Ralf Möller, Volker Haarslev, and Michael Wessel. On the scalability of Description Logic instance retrieval. In *Proc. of the 29th German Conf. on Artificial Intelligence*, LNAI, pages 171–184, Bremen, Germany, June 2006. Springer.
14. Boris Motik and Ulrike Sattler. A comparison of reasoning techniques for querying large Description Logic ABoxes. In *Proc. of the 13th Int. Conf. on Logic Programming Artificial Intelligence and Reasoning (LPAR'06)*, volume 4246 of *LNCS*, pages 227–241, Phnom Penh, Cambodia, November 2006. Springer.
15. Olaf Noppens and Thorsten Liebig. Interactive Visualization of Large OWL Instance Sets. In *Proc. of the 3rd Semantic Web User Interaction Workshop (SWUI 06) at the ISWC'06*, 2006.
16. Catherine Plaisant, Jesse Grosjean, and Benjamin B. Bederson. SpaceTree: Supporting Exploration in Large Node Link Tree, Design Evolution and Empirical Evaluation. In *Proc. of the IEEE Symposium on Information Visualization (INFOVIS 2002)*, pages 57 – 64, Boston, USA, October 2002.
17. Ben Shneiderman. Dynamic queries for visual information seeking. *IEEE Software*, 11(6):70–77, 1994.
18. Evrin Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. Pellet: A practical OWL DL reasoner. *Journal of Web Semantics*, 2006. To appear.
19. Taowei David Wang, Bijan Parsia, and James Hendler. A Survey of the Web Ontology Landscape. In *Proc. of the 5th Int. Semantic Web Conference (ISWC 2006)*, 2006.