Interactive Visualization of Large OWL Instance Sets

Olaf Noppens and Thorsten Liebig

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

Abstract. The adaption of Semantic Web techniques in real-world applications showed that it becomes a more and more demanding issue to understand not only the conceptual knowledge of an ontology but also the highly dynamic knowledge consisting of individuals and relationships between them. In this paper, we present an visualization paradigm and a prototypical implementation which allows to interactively browse large sets of individuals and to discover relationships between them in an easy and animated manner. The approach is optimized for efficient navigation as well as manipulation of ontologies containing a large number of individuals.

1 Motivation

An OWL ontology typically consists of two parts [3]. The terminological part introduces concepts and properties and gives structure to them in terms of axioms using the available language constructs (also called TBox in Description Logics). The assertional part contains individuals and relationships between individuals (also called ABox).[1]

As an example, consider a social-network ontology whose TBox defines terms like Person and Company as well as properties like `has-friend`, `supervisorOf`, `has-child`, and `worksWith`. The corresponding ABox covers all individuals of the domain such as `Peter`, `Mary`, `Sue` and property instantiations, e. g. `Peter has-friend Sue`. A large social-network obviously is characterized by a large number of individuals and property instantiations rather than by many 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. Therefore, scalability with respect to queries dealing with very large volumes of instance data are one of the actual research challenges [10] in order to meet real-world requirements.

Actually, recent system optimizations have shown significant increases in speed in answering conjunctive queries with respect to ABoxes containing several hundred thousands of individuals [16,17]. Even if these test results rely on synthetically generated data by a relatively simple benchmark generator such as the Lehigh University Benchmark (LUBM) [8], they clearly demonstrate promising progress in the development of scalable reasoning systems.

[1] Note that there is no clear distinction between both parts, e.g. due to the presence of nominals in case of OWL DL.

On the other hand, little has yet been done to support ontology users or developers to visually edit or explore 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 a company social-network one could be interested in stepping through the supervisorOf relationship until a point where he wants to investigating all the co-workers of a specific person. Or within the Gene Ontology (GO) [5], which contains millions of individuals, users are presumably interested in investigating which specific DNA binding product interacts with which kind of receptors for example.

We argue that an interactive graphical visualization of individuals and relationships would help users to depict and analyze the larger structure of the ABox. In our opinion, such an approach has to be able to visualize huge sets of data by making use of elaborated abstraction and rendering techniques.

Note that our main aim is not to develop a graphical query language for retrieving data, which could also be addressed by using a language such as SPARQL [20]. Our aim is rather to provide a tool which allows to easily browse through large data repositories in order to be able to capture interesting details as well as to understand how the data is organized in general.

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. Our prototypical system implementation also allows for basic editing and is connected with an OWL reasoning system.

1.1 Visualizing Entailed Assertions

Note that virtually all of the above requires reasoning. Since OWL allows for property hierarchies as well as symmetrical, transitive, or functional properties, a reliable reasoning system such as RacerPro [9], Pellet [22], or FaCT++ [23] is needed to make implicit property instantiations explicity available for visualization. Without reasoning feedback an ABox visualizer would degrade to a syntax imaging tool, probably hiding the most important assertions. For instance, in case of `has-child` being a sub-property 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 semantically 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 explicity 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 [15].

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 [13] or SWOOP [11]. 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). In addition, an OWL ABox inherently builds a graph structure (which has nothing to do with the underlying RDF data model).

Tools such as GrOWL [14] try to offer such 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. Figure 1 shows a fraction of an ABox part of a social-network ontology rendered with GrOWL. The graph shows 26 individuals which are interrelated via two different properties.

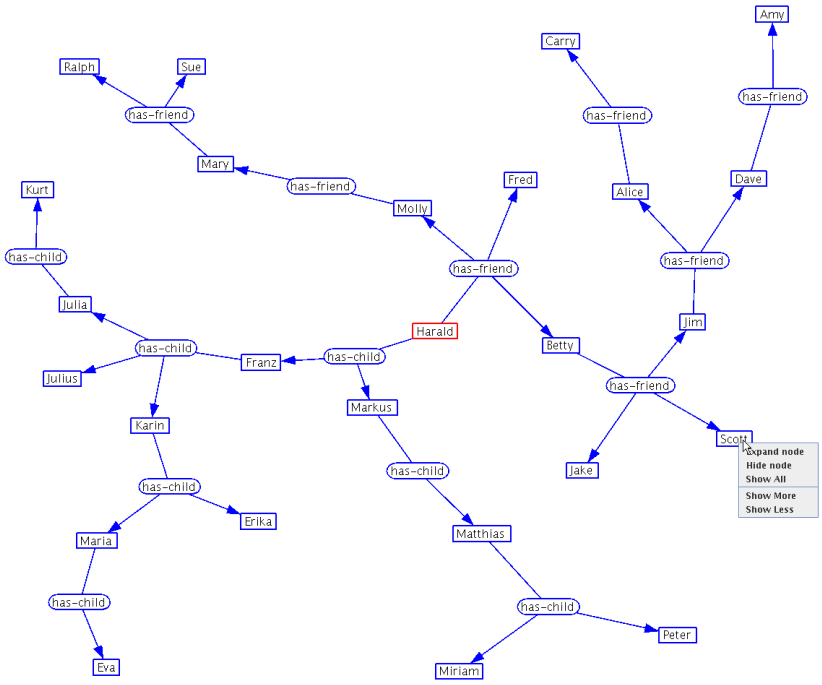

Fig. 1. GrOWL showing a fraction of a sample social-network ontology.

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. There even is a one to many mapping from OWL into the RDF data model. Therefore, none of the RDF visualizing tools are suitable to render sets of OWL individuals just due to the missing conceptual framework. Nevertheless, RDF Gravity [7] 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 layouting. Figure 2 again depicts our social-network ontology as rendered with RDF Gravity. Recall that any RDF renderer is not able to provide information about semantically entailed knowledge according to the OWL semantics.

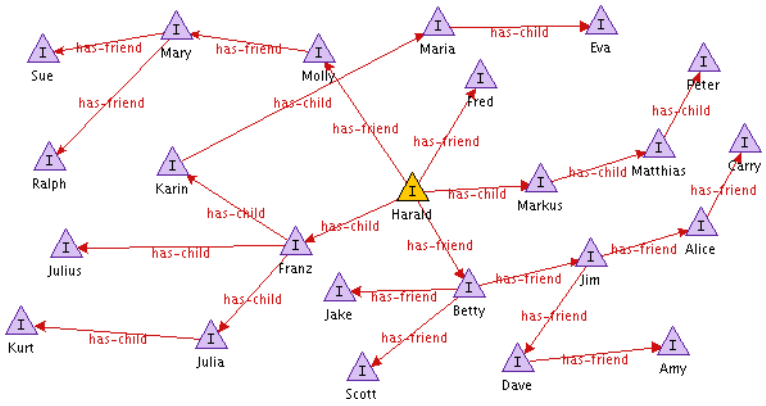

Fig. 2. RDF Gravity showing the same fraction as in Figure 1.

Other approaches like the self organized map (SOM) based view of [24] are able to visualize the quantitative differences with respect to classification of individuals with the downside of loosing access to specific individuals.

In the following we will present our new approach which allows one to interactively visualize and browse even large volumes of individuals.

2 Interactive Visualization of Individuals

2.1 Exploring Features

In contrast to other instance visualization techniques our approach does not aim at providing a diagrammatic understanding of class membership (instance classification) but of the relationships between individuals, i. e. of how individuals are related. It should be noticed that for visualizing class membership there already are adequate visualization

approaches (e. g. [6]) and our approach is complementary to these approaches, instead of replacing them.[2]

One lesson learnt from the visual analysis of large data sets in general, is that it is not advisable to visualize all dependencies, details etc. at any time [12]. Hence, our approach tries to provide detail information on user demand while offering an overview at the same time. Here we follow the single-view visualization paradigm offering selective detailed views which has turned out to be adequate for concept hierarchies as demonstrated by our ontology authoring framework ONTOTRACK.

The basic idea behind our visualization is a user-directed exploration of relationships between individuals. Starting with a user selected root instance, one can interactively exploit the property fillers (resp. datatype values in the case of datatype properties) of each property in a step-wise fashion. For instance, the tool offers a mouse over preview, showing the number of fillers within miniaturized extensions for each of the properties (which actually have fillers) on middle mouse click. For instance, Figure 3 shows that the actual root is related via three properties to 2, 3, resp. 53 other individuals. The user can choose one of the properties for further expansion.

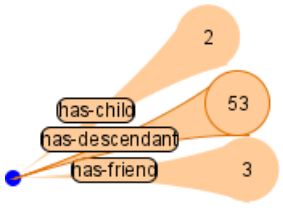

Fig. 3. Preview context menu for expanding property fillers.

After expansion all fillers with respect to the selected property are grouped in a so-called *property filler cluster* which will be drawn as a club originating from the root individual. By default, the individuals within this club are rendered as circles whose labels are accessible via mouse over tooltips. At any time the user can guide his exploration by choosing a follow up root for further expansion or may branch by selecting other properties. Different colors are used to distinguish different properties as well as different types of fillers (individuals vs. data values).

An example snapshot can be seen in the screen capture of our sample social-network ontology in Figure 4. It is easy to perceive that the root individual `Harald` is related via the property `has-friend` to three different individuals, via the property `has-child` to two individuals and via the transitive property `has-descendants` to many other individuals. 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.

[2] We use the term "individual" to denote a concrete ABox object and "instance of" to express a membership of a specific class extension.

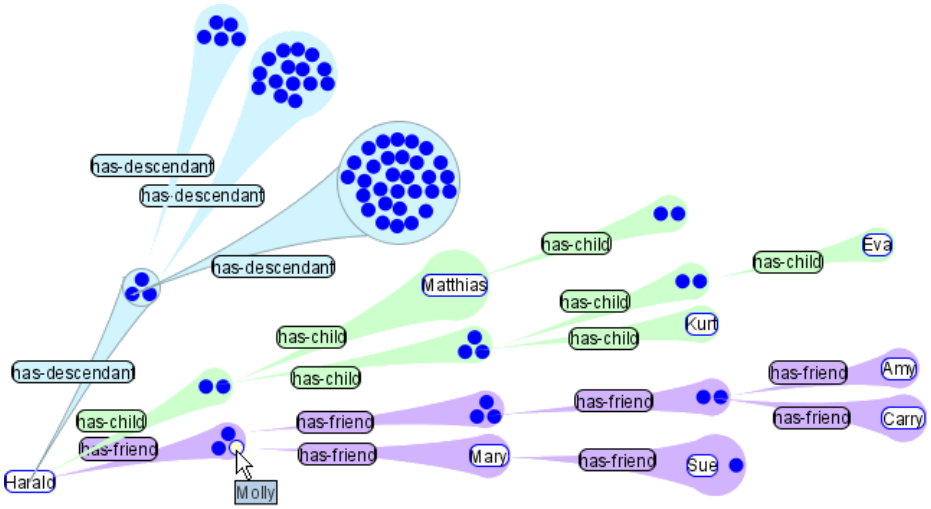

Fig. 4. Partial expanded property fillers.

Compare this graphical representation with those given by GrOWL and RDF Gravity in Figure 1 and 2. The lower two expansion trails for `has-friend` and `has-child` (green and purple colored) show the same 26 individuals which can be reached via these two properties from the individual `Harald`. In comparison to the former our approach is less verbose, because it clusters the fillers of each property, and allows selective detail information.

In addition to expanding a single individual it is also possible to expand the fillers of a whole cluster with respect to a property. 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 server as a simple visual explanation of the underlying action semantics.

One and the same individual can be related to another individual via different properties or may appear multiple times within the expansion path of a specific property (i. e. due to cycles). As a consequence, an individual can appear in multiple clusters and at different expansion levels. We decided to allow for a cloned graphical representation in each expanded cluster for one and the same individual. Otherwise a path-directed expansion would no longer be possible due to the occurrence of an individual in distant clusters. However, if a user hovers with the mouse pointer over a cloned individual, all its visible representations are highlighted simultaneously.

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 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. As with all other graphical changes this is done in an animated fashion in order to allow users to easily grasp the ratio of direct/indirect fillers. For instance, the upper part of Figure 5 shows the first expansion level of the transitive property `has-descendants`, which contains all individuals reachable from the root individual. After expanding the next level (lower part of Figure 5) it becomes apparent that the individuals `Hans`, `Birgit`, and `Jenny` are the only directly related fillers. Note that in case of a transitive property we currently only allow to either expand fillers which are direct property fillers or whole clusters. Otherwise one could expand an individual which may actually be part of a distant cluster.[3]

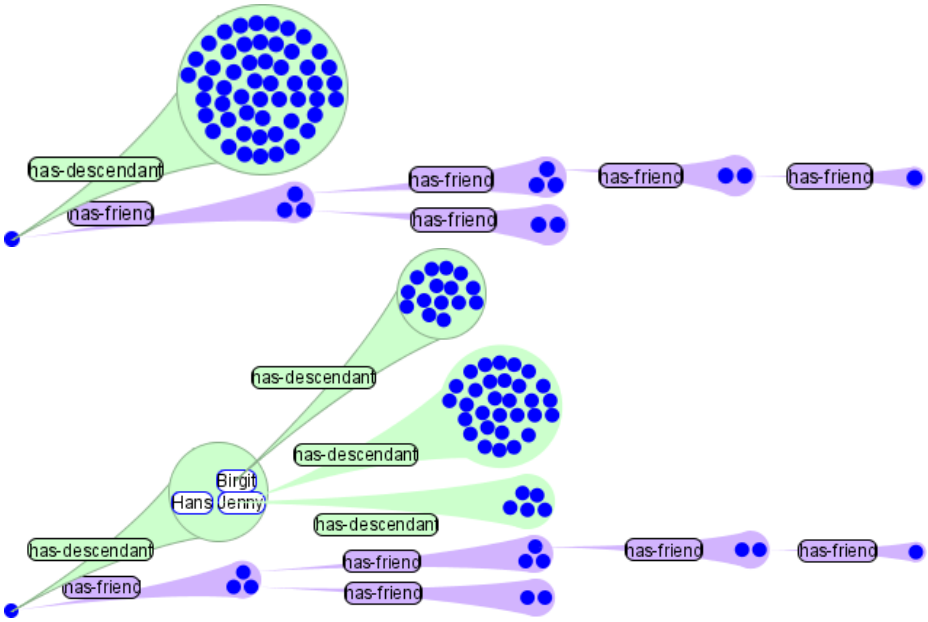

Fig. 5. Follow up expansion of a transitive property.

As mentioned before, the visualization utilizes the navigation and visualization principles of OntoTrack and therefore provides further information in its *detail view* mode. This mode is activated or de-activated for each instance separately using the mouse-wheel up- respectively downwards while hovering over the individual with the mouse pointer. As an example, in Figure 4 the individuals `Mary` and `Sue` show their full names as a result of being in the detail view mode. As mentioned before, a user can quickly access the name of an individual via the so-called tooltip pop-ups which automatically appear on mouse over action (see for example `Molly` in Figure 4). Fu-

[3] One could think of some feasible strategy which may provide a sensible result for such an action, too.

ture work will investigate the possibility to allow for more in-depth information within different levels of detail views (e. g. told information, class membership, etc.).

It is known that distortion techniques can support the visual data exploration process by preserving an overview of the data during exploration operations [12]. In this sense, 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.

2.2 Searching and Editing Features

Our ABox visualizer also supports ONTOTRACK's search features which are based on dynamic queries [21] which also can be found in tools like SpaceTree [19]. Currently only a string-based search is implemented. When one starts typing an individual or property name all matching entities are highlighted. Each additional character or deletion in the search string directly results in an updated highlighting of the matching individuals. In a future version we will extend the search facility with an option to expand all matching elements in a way which allows to grasp their relationships to the actual root.

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 Status and Current Work

The implementation of our ABox visualizer is still under development and the previously described features are the ones we have already implemented. A recent analysis of available ontologies has shown that they typically consist of several hundred of individuals [25]. Obviously, scalability and performance are very important issues for user-friendly tools. Due to our experience with large numbers of graphical objects in our ontology authoring framework ONTOTRACK we have chosen Piccolo [4] as our graphical framework. All these components as well as our ABox visualizer are implemented in Java. Even if our current implementation is a stand-alone system the internal ontology model as well as the graphical representation are based on core libraries and functionality of ONTOTRACK, as illustrated in the system architecture of Figure 6.

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 obviously is not a feasible solution when dealing with large volumes of data. However, in order to avoid delays due to awaiting reasoning results our tool employs a look ahead approach. It turned out to be a practical strategy to

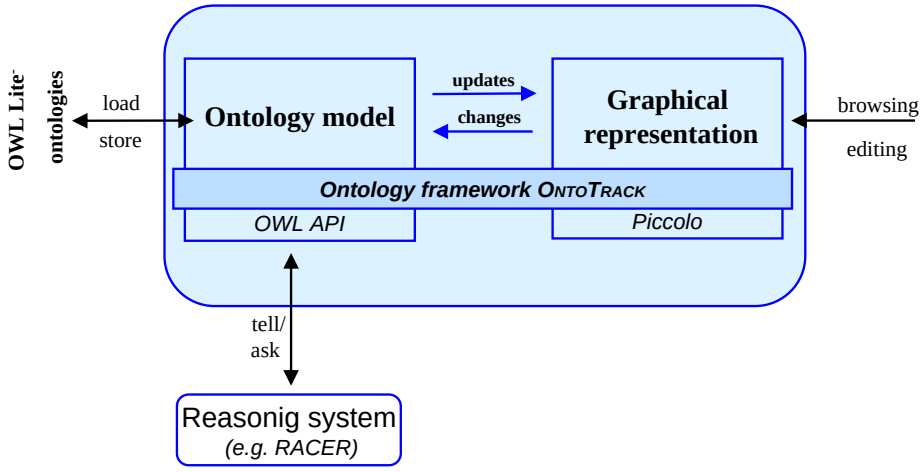

Fig. 6. System architecture based on OntoTrack technology.

query for those property fillers in advance, which are successors of the current visible clusters.

Unfortunately, we were not able to use the standard DIG 1.1 interface [1] for reasoner communication because it only supports very basic ABox queries. It also does not distinguish between direct and indirect property fillers which is needed for our visualization of transitive properties. However, the upcoming new version of DIG, namely DIG 2.0, will support more fine-graded and more expressive queries [2] suitable for our purpose. In the meantime, our system uses the RacerPro reasoner via its native syntax and query interface over TCP [9].

Current work deals with extending the browsing and visualization principle in a way that allows to explore property instantiation backwards (i. e. from filler to origin) for a bidirectional exploration. As an optional feature we plan to automatically collapse clusters if the distance from the actual focus and the focus' width falls below a user definable threshold. Such an approach could rely on user behavior and time as proposed in the context of UML [18] for example. 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. Furthermore we want to integrate our ABox visualizer into our ontology authoring tool OntoTrack more tightly to allow a seamless navigation and manipulation of concepts and individuals within one environment.

4 Summary and Outlook

This paper presents a novel approach for exploring large sets of individuals within ontology languages such as OWL. The prototypical implementation allows for incremental inspection of filler sets, selective browsing and applies several abstractive visualization techniques not found in current tools. First experiences with instance sets containing

hundreds of individuals are encouraging. We plan to test the implementation with synthetically generated data as well as available large volume ABoxes in order to make quantitative statements about scalability in comparison with reasoning benchmarks in the near future. Concerning the visualization and editing interface we also plan to conduct a user study and to add additional functionality as soon as the implementation has reached a sufficient degree of reliability.

References

1. Sean Bechhofer. The DIG Description Logics Interface: DIG/1.1. Technical report, University of Manchester, 2003.
2. Sean Bechhofer, Thorsten Liebig, Marko Luther, Olaf Noppens, Peter Patel-Schneider, Boontawee Suntisrivaraporn, Anni-Yasmin Turhan, and Timo Weithöner. DIG 2.0 – Towards a flexible interface for Description Logic reasoners. In *Proc. of the OWL Experiences and Directions Workshop (OWLED'06) at the ISWC'06*, 2006. To appear.
3. 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.
4. Ben Bederson, Jesse Grosjean, and Jon Meyer. Toolkit Design for Interactive Structured Graphics. Technical Report CS-TR-4432, University of Maryland, January 2002.
5. The Gene Ontology Consortium. Gene Ontology: tool for the unification of biology. *Nature Genetics*, 25:25–29, May 2000.
6. Christiann Fluit, Marta Sabou, and Frank van Harmelen. *Handbook on Ontologies in Information Systems*, chapter Supporting User Tasks through Visualisation of Light-weight Ontologies, pages 414–432. Springer, 2004.
7. Sunil Goyal and Rupert Westenthaler. RDF Gravity. Salzburg Research, 2004. http://semweb.salzburgresearch.at/apps/rdf-gravity/.
8. 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.
9. Volker Haarslev and Ralf Möller. Racer: A core inference engine for the Semantic Web Ontology Language (OWL). In *Proc. of the 2nd Int. Workshop on Evaluation of Ontology-based Tools (EON 2003)*, pages 27–36, 2003.
10. Ian Horrocks. Applications of description logics: State of the art and research challenges. In *Proc. of the 13th Int. Conf. on Conceptual Structures (ICCS'05)*, number 3596 in Lecture Notes in Artificial Intelligence, pages 78–90. Springer, 2005.
11. 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.
12. Daniel A. Keim. Visual exploration of large data sets. *Communications of the ACM*, 44(8):38–44, 2001.
13. Holger Knublauch, Fergerson Ray W., Natalya F. Noy, and Mark A Musen. The Protégé OWL Plugin: An Open Development Environment for Semantic Web Applications. In S.A. McIlraith, editor, *Proc. of the 3rd Int. Semantic Web Conference (ISWC 2004)*, pages 229 – 243, 2004.
14. Sergey Krivov, Ferdinando Villa, and Rich Williams. GrOWL, visual browser and editor for OWL ontologies. *Journal of Web Semantics*, 2006.
15. Thorsten Liebig and Olaf Noppens. ONTOTRACK: A semantic approach for ontology authoring. *Journal of Web Semantics*, 3(2):116 – 131, 2005.
16. Ralf Möller, Volker Haarslev, and Michael Wessel. On the scalability of Description Logic instance retrieval. In Ch. Freksa and M. Kohlhase, editors, *Proc. of the 29th German Conf. on Artificial Intelligence*, LNAI, pages 171–184, Bremen, Germany, June 2006. Springer.

17. Boris Motik and Ulrike Sattler. A comparison of techniques for querying large Description Logic ABoxes. In M. Hermann and A. Voronkov, editors, *Proc. of the 13th Int. Conf. on Logic Programming Artificial Intelligence and Reasoning (LPAR'06)*, LNCS, Phnom Penh, Cambodia, November 2006. Springer. To appear.
18. Benjamin Musial and Timothy Jacobs. Application of Focus + Context to UML. In J. Weckert, editor, *Proc. of the Austrialian Symposium on Information Visualization*, volume 1, 2003.
19. 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.
20. Eric Prud'hommeaux and Andy Seaborne. SPARQL Query Language for RDF. W3C Candidate Recommendation, April 2006.
21. Ben Shneiderman. Dynamic queries for visual information seeking. *IEEE Software*, 11(6):70–77, 1994.
22. 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.
23. Dimitry Tsarkov and Ian Horrocks. FaCT++ Description Logic Reasoner: System Description. In *Proc. of the Int. Joint Conference on Automated Reasoning (IJCAR'06)*, volume 4130 of *LNAI*, pages 292–297, 2006.
24. Kewei Tu, Miao Xiong, Lei Zhang, Haping Zhu, Jue Zhang, and Yong Yu. Towards Imaging Large-Scale Ontologies for Quick Understanding and Analysis. In *Proc. of 4th Int. Semantic Web Conference (ISWC 2005)*, volume 3729 of *LNCS*, pages 702 – 715. Springer, 2005.
25. Taowei David Wang, Bijan Parsia, and James Hendler. A Survey of the Web Ontology Landscape. In *Proc. of Int. Semantic Web Conference (ISWC 2006)*, 2006.