StarLion: Auto-Configurable Layouts for Exploring Ontologies

Stamatis Zampetakis, Yannis Tzitzikas, Asterios Leonidis, and Dimitris Kotzinos

Computer Science Department, University of Crete, GREECE, and
Institute of Computer Science, FORTH-ICS, GREECE
{zabetak,tzitzik,leonidis,kotzino}@ics.forth.gr

Abstract. The visualization of ontologies is a challenging task especially if they are large. We will demonstrate `StarLion`, a system providing exploratory visualizations which enhance the user understanding. `StarLion` combines many of the existing visualization methods with some novel features for providing better 2D layouts. Specifically, one distinctive feature of `StarLion` is the provision of Star-like graphs of variable radius whose layout is derived by a Force Directed Placement algorithm (*FDPA*) specially adapted for RDF Schemas. This approach enables users to gradually explore and navigate through the entire ontology without overloading them. `StarLion` can also handle multiple namespaces, a very useful feature for assisting the understanding of interdependent ontologies. Another distinctive characteristic of `StarLion` is the provision of a novel method for configuring automatically the *FDPA* parameters based on layout quality metrics, and the provision of an interactive configuration method offered via an intuitive tool-bar.

1 Introduction

The understanding of an ontology with many classes and properties represented as a directed graph (Figure 1(b)), is a hard and time consuming task. Our objective is to alleviate this problem by providing 2D visualizations that could aid users in tasks like: selection of a suitable ontology from a corpus of ontologies, understanding the structure of one particular ontology, and understanding a number of interrelated ontologies.

The field of graph drawing and visualization is very broad. There are many works using *FDP* algorithms and some of them also support star-like views with variable radius. Most of these works refer to general (plain) graphs and they are not RDF-specific. RDF graphs contain more information than plain graphs and have more visualization needs (e.g subclass hierarchies must be vertical). For this reason we did not rely on such algorithms but we designed a dedicated force directed algorithm which combines the *spring-model*([1–3]) with the *magnetic-spring model*([4, 5]). Apart from this, the notion of namespaces does not exist in plain graphs, while in RDF graphs plays an important role. Finally, and since different graphs exhibit different graph features the layout algorithm must be

configurable (ideally auto-configurable) for deriving aesthetically pleasing layouts. We will demonstrate `StarLion`[1] a system for ontology visualization, and we will focus on (a) the support of real-time exploration through star-like graphs of variable radius since this allows users to explore large schemas while controlling the amount of displayed information on the basis of user preferences or screen-size constraints, (b) the visualization of multiple (dependent) ontologies since every ontology usually extends and reuses elements from other ontologies, and (c) the manual (interactive) and the automatic (based on quality metrics) configuration of the *FDPA* layout algorithm.

2 The StarLion Approach

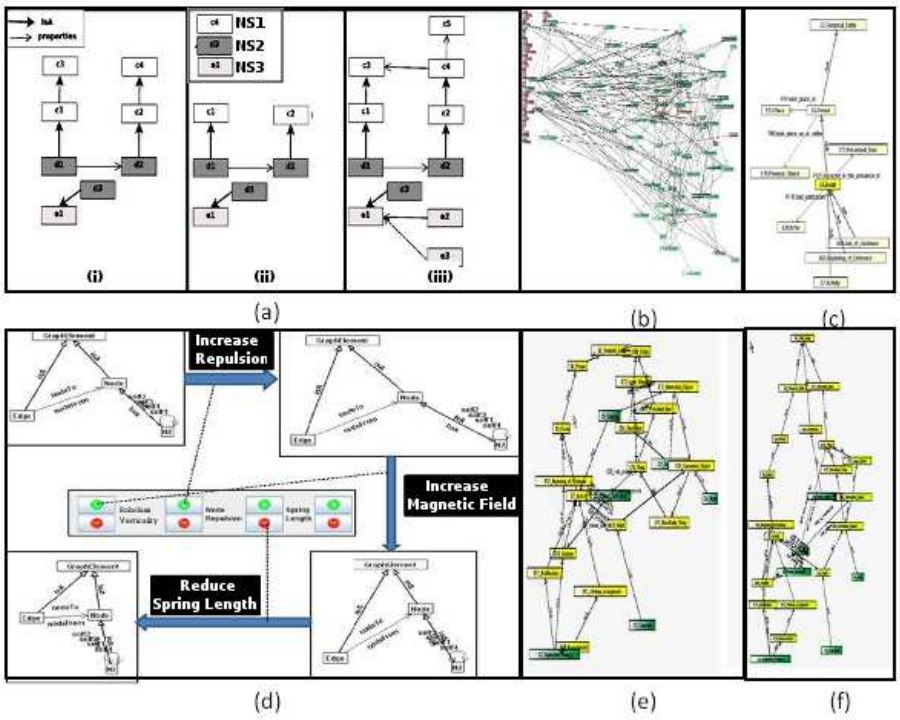

Fig. 1. StarLion distinctive features

Dependent Namespaces. It is often useful to load along with a schema the schemata (Namespaces) on which it depends. This enhances and completes the

[1] Implemented in Java using the jgraph library, for more see http://www.ics.forth.gr/~tzitzik/starlion

understanding by the user for the schema who is interested in, but also multiplies the visualization difficulties and makes it a cumbersome task even for experienced users. To address this problem, we propose a feature where upon loading a specific namespace NS, the user is able to see all namespaces upon which NS depends on, and select those to be visualized, while each namespace's classes are drawn using a different color. `StarLion` offers 3 options for the visualization of namespaces. Transitive option (Fig1(a)(i)) loads a namespace (e.g NS1) plus all superclasses of the classes of NS1 that belong to different namespaces. Direct option (Fig1(a)(ii)) loads the namespace (e.g NS1) and only the directly connected elements from the other namespaces, while full option (Fig1(a)(iii)) loads all dependent namespaces completely.

StarView. During *StarView* interaction, the user selects one class c as focal point. Class c is visualized along with all related classes in distance (radius) k from c, creating Star-like graphs. Figure1(c) shows the *StarView* with k=2. The focal point can be changed by clicking on any of the visualized elements.

Layout Algorithm. For deriving automatically the 2D layout we view the graphs as mechanical systems. We adopt the force model that was proposed in [6] for visualizing E-R diagrams. Specifically, that model combines the *spring-model* (proposed and developed in [1–3]) with the *magnetic-spring model* (proposed in [4, 5]). In our case we apply them on RDFS graphs. Nodes (in our case classes) are viewed as equally charged particles which *repel* each other. Edges (i.e. RDF properties and isA relationships) are viewed as springs that *pull* their adjacent nodes. Moreover, we assume that the springs that correspond to isA links are all magnetized and that there is a global magnetic field that acts on these springs. Specifically, this magnetic field is parallel (i.e. all magnetic forces operate in the same direction) and the isA springs are magnetized unidirectionally, so they tend to align with the direction of the magnetic field, here upwards. This is because the classical way (in semantic web, object-oriented class diagrams, Formal Concept Analysis, Hasse diagrams in discrete mathematics, etc) of presenting specialization/generalization relationships is to put the superclass above the subclass.

FDPA Configuration. Repulsion strength (K_e), spring length (L) and magnetic strength (K_m), are the three parameters that affect the result of the algorithm, and their configuration is one of the crucial issues in *FDPA* algorithms. Since every graph has different features, we have to configure these parameters to get a satisfying layout and this is not easy for a casual user by entering explicitly numeric values. For this reason we added a tool-bar (as shown in Figure 1(d)) which allows the user to relatively increase/decrease each parameter with a few clicks. To further improve the layout and reduce the clicks on that tool-bar, we devised a novel method for configuring automatically these parameters based on a number of quality metrics. This functionality is offered through a, so called "magic" button, that initiates the quality measurement, applies the proper configuration to the parameters and executes the *FDPA* taking away the burden from the user of knowing anything about the parameters. Quality is measured according to two metrics, one measuring how vertical the subclass relationships

are, and another measuring the density of the layout. We have conducted a user study which showed that this method does improve the layout.

3 Demonstration Scenario

During the live demonstration the audience will see `StarLion` in action over `CIDOC CRM`[2] ontology (Fig1(b)). Despite the big size (number of nodes and edges) of that ontology and the large number of inevitable edge crossings, the modified *FDPA* can give us a good overview. Then we will show how the user can start the exploration through *StarView* (Fig1(c)) for visualizing incrementally parts of the ontology. Someone who wants to obtain information about a specific class in the ontology usually wants to see all super-classes, subclasses and properties related to this class. We will present how this can be done with *StarView* and how easy is to continue this procedure for another class in the neighborhood. Next we are going to visualize `CIDOC CRM Digital` ontology which is an extension of `CIDOC CRM` with six new classes and a dozen of new properties. `CIDOC CRM Digital` obviously depends on `CIDOC CRM` and for this reason we are going to use the dependent name-spaces feature discussed earlier. Finally we will see how the *FDPA* can be configured through the tool-bar (Fig1(d)) to enhance the layout and some results from using the auto-configuration feature. Figure 1(e) shows the layout of a schema after an application of the *FDPA* with default parameters and Figure 1(f) shows the same layout after the auto-configuration of the parameters.

4 Related Work

One of the most popular open source tools is the *Jambalaya* plug-in of Protégé (http://protege.stanford.edu), which offers a set of different algorithms (trees, radial, grids) and allows the user to select the type of objects he wants to visualize (e.g. classes only, etc.), but offers no automatic method for restructuring the layout. *Touchgraph*[3] is a commercial product that offers a Star-like view where the selected node is automatically located at the center of the screen with only its directly connected nodes visible. *Welkin*[4] provides a layout algorithm based on a force directed model but it limits user interaction to configuration of the layout and presentation parameters only. *ISWIVE* [7], incorporates the topic features from Topic Maps into RDF and thus into the corresponding visualizations. Finally, *RDF-Gravity* provides a standard but non-configurable force directed layout with zooming facility, while the exploration is achieved only through filtering and textual information presentation.

Compared to ours the aforementioned tools do not offer methods for configuring automatically the layout and most of them lack the ability to extend visualizations beyond radius 1. The combination of the RDFS-adapted *FDPA*

[2] The international standard (ISO 21127:2006) for controlled exchange of cultural heritage information. `http://cidoc.ics.forth.gr/`

[3] http://www.touchgraph.com/navigator.html

[4] http://simile.mit.edu/welkin/

with *StarView* of variable radius and dependent namespaces, offers a powerful exploration method that is unique.

5 Conclusion

`StarLion` is a new visualization and exploration tool of ontologies, aiming at enhancing their understanding by the ordinary user; especially for those ontologies that are either large or interdependent (depending on one another) or both. As also verified by a user study (whose results are not reported here for reasons of space) the exploration through star-like graphs and the layout improvement through the interactive/automatic configuration methods results in a flexible and intuitive interaction and highly improves users' understanding of the ontologies at hand.

References

1. Eades, P.: A heuristic for graph drawing. Congressus Numerantium vol. 42, pp. 146–160. (1984)
2. Kamada, T.: On Visualization of Abstract Objects and Relations. PhD thesis, Dept. of Information Science, Univ. of Tokyo (1988)
3. Fruchterman, T., Reingold, E.: Graph drawing by force-directed placement. Software - Practice and Experience vol. 21, pp. 1129–1164. (1991)
4. Sugiyama, K., Misue, K.: Graph drawing by magnetic-spring model. Journal on Visual Lang. Comput. vol. 6, pp. 217–231. (1995)
5. Sugiyama, K., Misue, K.: A simple and unified method for drawing graphs: Magnetic-spring algorithm. In: Graph Drawing, pp. 364–375. Springer Berlin/Heidelberg (1994)
6. Tzitzikas, Y., Hainaut, J.: How to tame a very large er diagram (using link analysis and force-directed placement algorithms). In: Conceptual Modeling - ER2005, pp. 144–159. Springer Berlin/Heidelberg (2005)
7. Chen, X., Fan, C., Lo, P., Kuo, L., Yang, C.: Integrated visualization for semantic web. In: Intern. Conf.on Advanced Information Networking and Applications. pp. 701–706. (2005)