Metaphor-based Semantic Browsing in M-FIRE

Andrea Proli
DEIS, University of Bologna
Bologna, Italy

Stefano Rizzi
DEIS, University of Bologna
Bologna, Italy

Matteo Golfarelli
DEIS, University of Bologna
Bologna, Italy

Keywords

Semantic Web, RDF, Visual Interfaces

1. INTRODUCTION AND MOTIVATION

Though the syntax of RDF is designed to be human-readable, most end-users are not familiar with it. Thus, tools have been developed that (1) generate visual presentations of RDF statements and (2) translate user actions performed on those presentations into queries over the RDF knowledge base. An open problem in this field is to guarantee a satisfactory compromise between expressivity and domain-independence. The former is meant as the capability of delivering an intuitive visualization of knowledge and some tailored navigation primitives to end-users working in a given application domain, while the latter is aimed at accomplishing a high degree of reusability. Most existing tools, e.g. [2, 3], favor domain-independence by visually presenting constructs – such as classes and specializations – that are familiar to knowledge engineers but not to domain experts. The same holds for most Protégé plug-ins except Jambalaya [1], that allows users to associate custom semantics to the same graphical primitive, namely containment. Indeed, though domain-specific formalisms have a lower degree of reusability, they provide graphically richer constructs better understood by domain experts.

An approach to achieve a nice trade-off between reusability and expressivity is to decouple the mechanism for transforming RDF documents into an expressive visualization from the criteria that drive the transformation. In this demonstration we present M-FIRE (Metaphor-based Framework for Information Representation and Exploration), a configurable framework for semantic browsing of RDF-based knowledge, relying on the adoption of custom *metaphors*. Metaphors drive the process through which visual presentations are obtained for a given document, and define how queries are generated upon user actions. M-FIRE generalizes the approach pursued by current tools, which provide representations for *individuals* only, by allowing metaphors to specify the representation of more complex information patterns, namely *sets of statements*. The demonstration will be focused on (1) showing how users can perform semantic browsing by relying on domain-specific and intuitive visualizations of concepts, thus interacting in a simple manner with complex knowledge, and (2) illustrating how flexibility and reusability are effectively achieved in our framework by the use of metaphors.

Demos and Posters of the *3rd European Semantic Web Conference (ESWC 2006),* Budva, Montenegro, 11th - 14th June, 2006

2. APPROACH OVERVIEW

The overall functional architecture of our framework is sketched in Figure 1. First of all, we rely on an *RDF server* that, given a SPARQL query, returns a result as an RDF document (from now on, the *source*). We will use the Jena engine to this end. The *metaphor selector* takes the source and returns the best suited metaphor for its fruition, according to the vocabulary of the source. The metaphor consists of a *representation metaphor* and a *navigation metaphor*. The representation metaphor is given as input to the *representer*, which applies the directives contained in it to generate a *representation* describing in an abstract form, independently of any implementation detail, how concepts will be visualized. Then, a properly chosen *encoder* translates the representation into a concrete form, called *encoding* (e.g., an HTML document), which can be given as input to the end-user's rendering program (e.g., a Web browser). The choice of the best suited encoder for a given representation is carried out by the *encoder selector*, again based on the representation vocabulary.

Once rendering has been completed by the rendering program, end-users are allowed to interact with it. Events generated by user actions are captured by the *controller*, which creates an *event description* in the form of an OWL document describing the occurred event (for instance, a user's double click on an icon representing a soccer player). The event description is then given as input to the *navigator* together with the chosen navigation metaphor. In the same way as the representation metaphor tells the representer which representation must be produced for a given source, the navigation metaphor tells the navigator which SPARQL query must be formulated for a given event. The resulting query is then forwarded to the RDF server, and the process is repeated.

Some details on the different phases are given in the following subsections.

2.1 Representation

Representation is the process of obtaining a document in which certain graphical drawings are associated to certain (kinds of) statements belonging to the source, according to the directives contained in the representation metaphor. In order to provide a general solution, representation is split into two phases, namely *enrichment* and *mapping*, and the representation metaphor is split accordingly, namely into an *enrichment metaphor* and a *mapping metaphor*.

Enrichment exploits the enrichment metaphor to augment the source with new classifications and concept definitions. This is done by launching the Pellet reasoner (which sup-

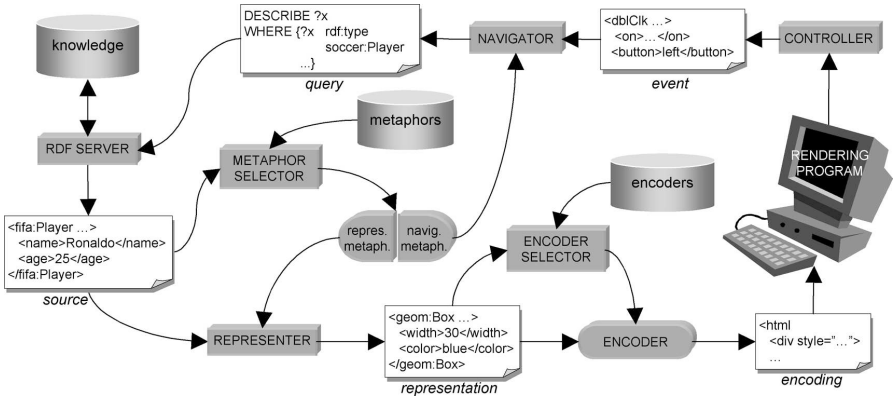

Figure 1: Overall functional architecture for M-FIRE

Figure 2: Two representations obtained by applying different metaphors to the same source

ports OWL-DL reasoning) on the merge between the source and the enrichment metaphor. Mapping interprets the directives contained into the mapping metaphor to carry out the association of particular visual items and graphical styles to certain kinds of statements in the document resulting from enrichment. This is done by translating the mapping metaphor into a set of SPARQL CONSTRUCT queries, executed by the ARQ SPARQL engine, whose results are then properly merged to create the representation. Notably, a crisp separation between enrichment and mapping gives metaphor designers a better control over the kind of inferences and classifications that are performed, also increasing the flexibility and the modularity of design.

2.2 Encoding

Encoding consists in translating the representation into a document that can be parsed by a proper program to produce a graphical rendering. Many formats could be used to this end: for instance, two circles connected by a line could be encoded as both a GraphML document and an SVG document; a table containing names and photos could be encoded as an SVG document as well as an HTML document. We developed two different encoders, one for HTML and one for GraphML. Figure 2 shows the HTML rendering of two representations obtained by applying different representation metaphors to the same RDFS source containing information about soccer players. To the left, soccer teams are the focus of interest and they are rendered as a list of players; to the right, goalscorers are shown together with their score.

A relevant issue concerning encoding is *semantic annotation*, that traces a correspondence between the graphical items used to represent a given set of statements and the represented statements themselves. Such correspondence is first established at a conceptual level by the representer, then embedded into the encoding. There, such annotations can be used by the end-user's rendering program to integrate graphical information with semantic information.

2.3 Navigation

Navigation is the interaction schema triggered by a user action upon the visual presentation of a given piece of knowledge, and it consists in translating that action into a query over the underlying knowledge base. In order to enhance flexibility, the directives for this translation are contained in a *navigation metaphor*. Remarkably, the representation and the navigation metaphors are independent of each other: e.g., a navigation metaphor could state that a double click on an item representing a soccer player should trigger a query for retrieving the soccer team in which that player is enrolled, whatever its visual representation is.

Navigation is carried out in two steps. First, the *controller* captures the actions performed by the user on the current rendering and describes them in the form of an OWL document, by relying on the semantic annotations in the encoding. The controller needs to be tightly integrated with the rendering program; we developed two controllers: one is a simple HTML viewer that uses the Microsoft WebBrowser ActiveX control for rendering, the other is a plug-in for Protégé that uses the yFiles library for rendering GraphML documents. Then, the *navigator* parses a set of directives contained in the navigation metaphor to produce, from the event description, a DESCRIBE SPARQL query which is then sent to the RDF server for execution.

3. REFERENCES

[1] M. Storey et al. Jambalaya: Interactive visualization to enhance ontology authoring and knowledge acquisition in Protégé. In *Proc. Workshop on Interactive Tools for Knowledge Capture*, 2001.

[2] F. Van Harmelen et al. Ontology-based information visualisation. In *Proc. Workshop on Visualization of the Semantic Web*, pages 546–554, 2001.

[3] R. Volz et al. KAON SERVER - a semantic web management system. In *Proc. WWW*, 2003.