A Protégé Plugin for Querying and Reasoning on Persistent OWL Ontologies

Maria del Mar Roldan-Garcia, Joaquin J. Molina-Castro and Jose F. Aldana-Montes

University of Malaga, Computer Languages and Computing Science Department
Malaga 29071, Spain,
(mmar,jmolina, jfam)@lcc.uma.es,
WWW home page: http://khaos.uma.es

1 Introduction

The OWL language [1] is being widely used to define ontologies in the Web. This language provides three increasingly expressive sublanguages, namely OWL Lite, OWL-DL and OWL Full. The OWL-DL XML based syntax together with its correspondence with Description Logics (DL) [2], make it a good candidate to be the standard language for defining ontologies used by Semantic Web applications. However, there are still relatively few tools that allow us to manipulate, store and query ontologies defined using this language. Furthermore, Semantic Web applications, such as biological tools, use large ontologies, that is, ontologies with a large number (millions) of instances. Description logic based tools allow us to manage OWL-DL ontologies, but not very large ontologies. The development of tools for storing and querying OWL ontologies is currently under investigation. In this paper we present a plugin for the Protégé [3] environment for reasoning and querying OWL-DL ontologies stored in a relational database. We define a relational schema for storing OWL ontologies and a query/reasoning language, which supports reasoning on both, ontology structure and ontology instances. However, the main feature of this plugin is that developers of OWL-DL reasoners can customize it in order to use their own database and their own query language by means of several configurations files.

2 Architecture of the Plugin

In case that a reasoner developer wants to customize the plugin for building his or her own tool, he or she has to provide two different kinds of information. On the one hand, information about how to access the ontology information stored in the database. This includes functions to obtain information about the ontology structure, such us properties domain and range, all classes and properties in the ontology, etc. These functions are used for showing the ontology information in a graphical way. On the other hand, information about the query and reasoning language implementation. That is, the operators of the query mechanism and both Tbox and Abox reasoning that the specific language supported. Information

about the implementation of the language for one or more specific repositories is also needed. All this information is provided by means of several configuration files. These files are XML files. We have chosen XML because there exists many java libraries to manage XML. The use of RDF or RDFS could also be an option, but there are more tools for managing XML than RDF. We will consider changing to RDF in the later versions of the tool.

Finally, using all this information, the query and reasoning tool generator customizes a Protege generic graphical interface, which shows the information about the ontology using the ontology information configuration files, and the specific reasoning supported by the specific query language.

Once the tool is generated, it allows users to make queries using the specific operators defined in the query and reasoning configurations files. These queries will be performed in a persistent repository. Note that the tool does not implement the query language. It is the language developer who has to give an implementation for each element of the language.

3 Developing a Plugin for Relational Databases

We have developed our plugin for querying and reasoning on OWL ontologies stored in a relational database. Specifically, it implements the ECQ (Extended Conjunctive Queries) language [4] which supports reasoning defined in the context of the system DBOWL [5]. It uses Oracle as Database Management System. We will briefly describe the configuration files for this particular language. The aim of this section is not to present the query language, but to present how, and how easy, it is to use and to configure our Plugin. We are supposing that the ontology is already stored in an Oracle database. The schema of the database is described in [5].

3.1 Configuration Files

The file *ontinformation.xml* contains the functions for obtaining the ontology information from the relational database. For example, we define a function for obtaining all the features of a specific property or a function for obtaining all the restrictions for a specific class. Due to these functions are trivial, we will describe only the query and reasoning configurations files.

The *queryLanguage* element of the *queryLanguages.xml* file contains the following information. The name of the language is *RECQ*, from Reasoning and Extended Conjunctive Queries. It is the language developer who chooses the name. The storage model is relational and the DBMS used for implementing it is Oracle. The file *RECQQueries.xml* defines the operators for the query mechanism. In our case, allowed operators are and, or, all, max and min. The *RECQTbox.xml* and *RECQAbox.xml* files specify respectively, the Tbox and the Abox reasoning supported. We only have to specify name, description and parameters names for each inference. Finally, the *RECQoracle.xml* file defines the implementation for the queries (which will use the query operators and the Abox reasoning) and

also the implementations for the Tbox inferences. Note that the first rule in the file implements the connection to the Oracle DBMS.

Once the configuration files are defined, the use of the tool is really easy. First, the user must select the ontology. The content of the *queryLanguages.xml* file is used in order to allow the user to select the query language he/she wants to use. Once the language is selected, using the configuration files, the tool customizes the generic graphical interface for the selected language. Only reasonings supported by it are enabled, and the panel for building the queries is provided only to the supported operators. A window asking the login and password for connecting to the database is also shown. In order to see the description of a specific language or reasoning, the user only has to place the mouse on the corresponding element and the tool shows its description. The user is now able to make queries over the selected repository.

4 Conclusions

This paper presents a Protégé Plugin that helps developers to build query and reasoning tools for OWL ontologies, which are persistently stored in a database. It is possible to customize the Plugin to different query languages according to the needs of the users. In order to do this, some configuration files are defined. These files define the query mechanism and specify the Tbox and Abox reasonings supported by the specific query language. Using this Plugin it is possible to implement the same query language using different storage models and different DBMS easily, and to use a common graphical interface to perform queries and reasoning. We have also presented an example of how to use this tool for querying and reasoning on an OWL ontology which is stored in a relational database.

5 Acknowledgements

This work has been supported by the Spanish MEC Grant (TIN2005-09098-C05-01).

References

1. OWL Web Ontology Language Reference. W3C Working Draft, 10 February 2004. http://www.w3.org/TR/owl-ref/, 2004.
2. Borgida, A., Lenzerini, M. and Rosati, R.. The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press, 2003.
3. The Protégé Ontology Editor and Knowledge Base Acquisition System. http://protege.stanford.edu/
4. Roldan-Garcia, M.M., Molina-Castro, Joaquin J., Aldana-Montes, J.F. ECQ: A Simple Query Language for the Semantic Web. Proceedings of DEXA 2008. Turin, Italy. 2008.
5. Roldan-Garcia, M.M., Aldana-Montes, J.F. DBOWL: Towards a Scalable and Persistent OWL reasoner. Proceedings of the Third International Conference on Internet and Web Applications and Services. Athens, Greece. 2008.