A SPARQL Query Engine over Web Ontologies using Contextual Logic Programming

Nuno Lopes and Salvador Abreu

Universidade de Évora

Abstract. Querying is one of the key aspects in the Semantic Web. SPARQL, a W3C recommendation, attempts to become the standard Web query language. The XPTO system is capable of representing and querying ontologies described in the OWL language using Contextual Logic Programming. Here is presented a component of the XPTO system that enables answering SPARQL queries.

1 Introduction

The XPTO[1] system, enables accessing OWL [DSB+04] (Web Ontology Language) ontologies from within a Contextual Logic Programming environment, namely GNU Prolog/CX. It also allows to integrate these ontologies in the running program enabling using them as a part of the computation. The XPTO system is further detailed in [FLA07,LFA07].

Here is described a component of the system that enables it to answer queries formulated using the SPARQL query language and thus presenting the possibility of making the system visible to the World Wide Web though a Web Service.

2 SPARQL Query Engine

The presented component is dedicated to SPARQL query resolution: it allows for the possibility of querying the internal representation of the ontology using the SPARQL query language. It is split into 3 parts: the parser, the query resolution and the returning of the results as XML. The implemented SPARQL parser follows the specifications of the language defined in [PS06] and the results are returned in XML as specified in [BB06].

The SPARQL query is parsed to produce a GNU Prolog/CX context representing the query that is then activated to calculate the output and display the resulting XML.

[1] XPTO is a recursive acronym that stands for *XPTO Prolog Translation of Ontologies*.

2.1 Representation of a SPARQL query

The query representation process consists of a SPARQL parser that converts a query defined in the SPARQL syntax [PS06] into a GNU Prolog/CX context. This context represents the entire query and can be used to return the results. The execution of the generated context, triggered by a default message, that will bind the variables present in the query and show the results.

Element representation The representation of query elements, such as SPARQL variables and resources, is presented next.

Variables The SPARQL variables are represented as Prolog variables. Thus, once the result is calculated, the query resolution system simply binds the corresponding variable to return the results.

There are some other structures needed to display the results: it is necessary to store the name of the variable in the SPARQL query in order to return it in the results. To achieve this, all the variables in the SPARQL query are stored in a list that will be the argument of the unit `vars/1`. The elements of this list are in the format `SparqlVariableName = PrologVariable`. `SparqlVariableName` corresponds to the name of the variable in the SPARQL query and `PrologVariable` is the Prolog variable assigned to represent it. `PrologVariable` will start unbound and, as the context is resolved, will be instantiated with the solutions it may have. SPARQL variables appear in the generated context for the query using the `PrologVariable` representation, enabling a simple access to the value of the variable or direct instantiation of an unbound variable. This representation can be seen in the GNU Prolog/CX context shown in Figure 2.

Resources Resources are represented using Prolog terms or atoms. If the resource is an absolute IRI (delimited by `'<'` and `'>'`) it is represented as an atom containing the entire IRI. If it corresponds to a prefixed name (a prefix label and a local part separated by a colon `':'`) it is represented as Prolog compound term of arity 2 with the functor ':'. The arguments of the term are the prefix name and the local part respectively. If the prefix name is empty the atom `''` will be used to represent it.

Query representation A SPARQL query is represented as GNU Prolog/CX context whose structure is similar to the structure of the query. The elements of the query can be clearly identified in the representation: `select`, `where` as well as the *Modifiers* (if there are any present in the query).

The example query presented in Figure 1 is a `select` query containing two basic graph patterns with a shared variable: `?t` and the context produced by the parser is shown in Figure 2.

A context is represented by a Prolog list containing unit names. The first element of the list will be the unit that first tries to evaluate the goal upon execution. The individuals and property values are gathered from the units in

```
SELECT
       ?flavor ?color
WHERE {
   ?t :hasFlavor    ?flavor  .
   ?t :hasColor      ?color  .
}
```

Fig. 1. Query example (simple select)

```
[ where([set([
             triple(A,hasFlavor,B),
             triple(A,hasColor,C) ])
       ]),
 select([flavor=B,color=C]),
 vars([flavor=B,color=C,t=A]) ]
```

Fig. 2. Generated context (partial) for the query in Figure 1

a higher position in the context. This way in the final positions of the list are found the units `select/1` (in the case of a select query) and `vars/1`. These units contain in their arguments a list of variables and will allow any unit in the context to access either all the variables in the context or the selected variables.

2.2 SPARQL resolution system

The core unit in the query resolution process is the `triple/3` unit, which is responsible for instantiating the variables in the query by accessing the data.

This unit can be redefined in order to access data available from different sources. It generates one query to the XPTO system for each property that appears in the SPARQL query. The pattern in line 2 of Figure 2 (page 3) will generate the following query:

```
/> property(hasFlavor,F) :> item(I).
```

The argument of the `item/1` goal will be instantiated with the name of the individual. The arguments of the unit `property/2` are the name of the property being queried and the value of that property for the returned individual. Using the `property` unit to query the internal representation has the advantage of being able to perform the query using a Prolog variable in the position of the property name, thus enabling to return all the properties of the individual or querying the property name based on the property value.

3 Conclusion

The developed component acts as a translator: mapping SPARQL queries to a representation of the query that is based on CxLP units. In this representation each operator and each part of the SPARQL query corresponds to a unit occurring in the context and the complete query is represented by a context that combines the available units.

There are still further improvements necessary such as:

Complete the SPARQL support: Currently not all of the SPARQL constructors are implemented

Adopt the latest SPARQL specifications: The SPARQL system was developed against the specifications of 6 April 2006 in which SPARQL was considered W3C Candidate Recommendation.

References

[BB06] D. Beckett and J. Broekstra. SPARQL Query Results XML Format. W3C recommendation, W3C, April 2006. Available at: http://www.w3.org/TR/2006/CR-rdf-sparql-XMLres-20060406/.

[DSB+04] M. Dean, G. Schreiber, S. Bechhofer, Frank van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein. OWL web ontology language reference. W3C recommendation, W3C, Feb 2004. http://www.w3.org/TR/owl-ref/.

[FLA07] Cláudio Fernandes, Nuno Lopes, and Salvador Abreu. On querying ontologies with contextual logic programming. In Christine Golbreich, Aditya Kalyanpur, and Bijan Parsia, editors, *OWL: Experiences and Directions 2007*, volume 258 of *CEUR Workshop Proceedings ISSN 1613-0073*, June 2007.

[LFA07] Nuno Lopes, Cláudio Fernandes, and Salvador Abreu. Contextual logic programming for ontology representation and querying. In Axel Polleres, David Pearce, Stijn Heymans, and Edna Ruckhaus, editors, *2nd International Workshop on Applications of Logic Programming to the Web, Semantic Web and Semantic Web Services*, September 2007.

[PS06] Eric Prud'hommeaux and Andy Seaborne. SPARQL Query Language for RDF. Technical report, W3C, 2006. Available at: http://www.w3.org/TR/2006/CR-rdf-sparql-query-20060406/.