SPARQL Back-end for Contextual Logic Agents

Cláudio Fernandes and Salvador Abreu

Universidade de Évora

Abstract. XPTO is a contextual logic system that can represent and query OWL ontologies from a contextual logic programming point of view. This paper presents a prototype of a SPARQL component for that system which is capable of mapping Prolog/CX to SPARQL queries.

1 Introduction

We present a back-end that aims to transparently merge the reasoning of the XPTO [1] [FLA07] [LFA07] internal knowledge base with external OWL [BvHH+05] ontologies, more exactly its Lite and DL sub languages, available from third parties, by means of the SPARQL [PS06] query language. To achieve this, we developed a system that provides functions for communicating with Web SPARQL agents for ontology querying purposes. It provides the system with the ability to pass a SPARQL query to an arbitrary SPARQL Web agent and get the solution, encapsulating the results as bindings for logic variables.

The presented back-end grants XPTO with capabilities for writing Prolog/CX [AD03] programs to reason simultaneously over local and external Web ontologies.

2 Mapping Prolog to SPARQL Queries

Although it can be viewed as a single independent component, the back-end purpose is to allow the XPTO-using programmer to query external and internal ontologies using the same query syntax and declarative context mechanics as the XPTO internal system. This will allow to transparently query internal and external ontologies and merge their results in the same program.

To achieve this level of functionality, we developed a Prolog/CX to SPARQL engine that satisfies the following requirements:

- Translate a partially bound Prolog/CX goal into SPARQL;
- Send the SPARQL query to the specified Semantic Web SPARQL service;
- Fetch the XML result file, parse it and return the solutions as Prolog variable bindings using the Prolog/CX backtrack mechanism to iterate over sets of answers;

A SPARQL query in the back-end environment is a Prolog/CX context execution. Figure 1 illustrates a definition of a back-end query.

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

```
QUERY := sparql(URI) /> P1 ... Pn :> ITEM

     URI  := URL
     P    := property(VALUE) or where(PROP, VALUE)
     ITEM := item(INDIVIDUAL)
```

Fig. 1. Back-End Query Definition

On the left side of the defined operator `'/>'` is specified the external agent and on the right side are the goals and query restrictions. The right side of the operator encodes the query that must be mapped to SPARQL. We translate that information into RDF triples, much in the same way a database is translated into triples, i.e, for each of the `n` stated properties about an individual, the back-end must translate it to `(n-1)` triples. The triples are extracted by the union of each **property** term of the right side and the **item** term, which represents the subject of the triple.

3 Examples and Query solutions

We now present an example. We will use the *XAK* - XML Army knife [Dod06] SPARQL service which implements the SPARQL Protocol for RDF and provides a SPARQL query engine for RDF data available on the Internet.

The *Wine* OWL DL ontology [2] is a sample ontology used in the OWL specification documents and will serve as the use case ontology in this paper. Among others, the `IceWine` class present in the ontology defines two properties: `hasBody` and `hasColor`.

Figure 2 shows an example of a back-end query that asks *XAK* to search the *Wine* ontology for all the individuals that have both of these properties.

```
1 ?- sparql('http://xmlarmyknife.org/api/rdf/sparql/') />
2    hasBody(A) :> hasColor(B) :> item(IND).
```

Fig. 2. Back-end Prolog/CX query to *XAK*

The Prolog/CX query in Figure 2 has no ground Prolog atoms besides the **url** that identifies *XAK*. It includes two specified properties, thus originating two RDF triples, one for each property. Figure 3 shows the correspondent SPARQL generated code.

[2] The ontology is accessible in http://www.w3.org/TR/owl-guide/wine.rdf

```
SELECT ?id ?hasColor ?hasBody
WHERE {
?id :hasColor ?hasColor.
?id :hasBody ?hasBody.
}
```

Fig. 3. Generated SPARQL for the query in Figure 2

After the SPARQL generation, the code is sent to *XAK*. In order to successfully communicate with it, the back-end must first encode the query as specified in the SPARQL Protocol for RDF [Cla06] and establish the values of a few parameters like the default graph to be queried. (Figure 4 shows the generated string that is sent over to *XAK*).

```
GET http://xmlarmyknife.org/api/rdf/sparql/query?default-graph-uri
=http://www.w3.org/2001/sw/WebOnt/guide-src/wine.owl&query=
PREFIX+:+<http://www.w3.org/2001/sw/WebOnt/guide-src/wine%23>
+select+?id+?hasColor+?hasBody+where+{?id+:hasColor+?hasColor+.+
?id+:hasBody+?hasBody}
```

Fig. 4. Back-end encoded query example

If a successful query response code is returned, a file with the solutions is received. This file is in the SPARQL Query Results XML Format [BB06] and includes one solution. This XML file is then parsed and the solution values are returned as bindings for Prolog variables as illustrated by the last lines in Figure 5.

```
?- sparql('http://xmlarmyknife.org/api/rdf/sparql/') />
   hasBody(A) :> hasColor(B) :> item(IND).

A ='http://www.w3.org/2001/sw/WebOnt/guide-src/wine#Medium'
B ='http://www.w3.org/2001/sw/WebOnt/guide-src/wine#White'
IND ='http://www.w3.org/2001/sw/WebOnt/guide-src/wine#SelaksIceWine' ? ;
(4 ms) no
```

Fig. 5. Prolog/CX query to *XAK* and the returned solution

The solution presents only one individual, **SelaksIceWine**, and the values **Medium** and **White** for properties *hasBody* and *hasColor* respectively. This means the whole ontology only has one individual that has those two properties defined.

4 Initial Assessment and Conclusions

The component presented in this paper is still work in progress. With the current capabilities, one can use the expressiveness of Logic Programming to perform basic queries to an ontology via a third party SPARQL Web Service. These capabilities can then be combined with other Prolog/CX data access forms for reasoning over different data repositories. For example, an application can indifferently use local data provided by the XPTO engine, external data through the SPARQL back-end and data residing in a relational data base accessed using ISCO [AN06].

Although no proper benchmarks were defined yet, the experimental work revealed no particular performance issues on the back-end side, which means that practically only the *XAK* connection will introduce some latencies. Note, however, that the generation of SPARQL is currently done in a *per-query* basis. One important feature to be implemented as future work is to allow the generation of SPARQL code for a composite (e.g. conjunction) of Prolog/CX queries.

References

[AD03] Salvador Abreu and Daniel Diaz. Objective: in Minimum Context. In Catuscia Palamidessi, editor, *Logic Programming, 19th International Conference, ICLP 2003, Mumbai, India, December 9-13, 2003, Proceedings*, volume 2916 of *Lecture Notes in Computer Science*, pages 128–147. Springer-Verlag, 2003. ISBN 3-540-20642-6.

[AN06] Salvador Abreu and Vítor Nogueira. Using a Logic Programming Language with Persistence and Contexts. In Masanobu Umeda and Armin Wolf, editors, *Declarative Programming for Knowledge Management*, volume 4369 of *LNCS*, Fukuoka, Japan, 2006. Springer.

[BB06] Dave Beckett and Jeen Broekstra. SPARQL Query Results XML Format. Candidate recommendation, World Wide Web Consortium, 25 December 2006. http://www.w3.org/TR/rdf-sparql-XMLres/.

[BvHH+05] Sean Bechhofer, Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. Owl web ontology language reference. Recommendation, World Wide Web Consortium, 19 October 2005. http://www.w3.org/TR/2004/REC-owl-ref-20040210/.

[Cla06] Kendall Grant Clark. SPARQL Protocol For RDF. Candidate recommendation, World Wide Web Consortium, 6 October 2006. http://www.w3.org/TR/rdf-sparql-protocol/.

[Dod06] Leigh Dodds. XML Army Knife. http://xmlarmyknife.org/api/rdf/sparql/query, 5 December 2006.

[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. Candidate recommendation, World Wide Web Consortium, 25 July 2006. http://www.w3.org/TR/2006/CR-rdf-sparql-query-20060406/.