SPARQLAS - Implementing SPARQL Queries with OWL Syntax

Mark Schneider

WeST — Institute for Web Science and Technologies
University of Koblenz-Landau
Universitaetsstrasse 1, Koblenz 56070, Germany
mschneider@uni-koblenz.de

Abstract. Using SPARQL for writing queries on OWL-DL ontologies can be complicated due to its RDF triple semantics. Translating OWL expressions to this semantics is a non-trivial problem. We present SPARQL-DL Abstract Syntax (SPARQLAS), a proper subset of SPARQL, to solve this issue by employing the OWL Functional-Style Syntax to compose queries. We illustrate how the translation is made possible by a model-to-model transformation and how a query can be connected to an UML model. Finally, we evaluate SPARQLAS against SPARQL and show the efforts of SPARQLAS for queries on OWL-DL ontologies.

1 Introduction

With the W3C recommendation for SPARQL [1] as the query language for RDF, querying OWL-DL ontologies can be quite complicated. Having OWL Functional-Style Syntax [2] recommended to write new ontologies, it becomes more difficult than necessary to query such ontologies using SPARQL. The main problem lies in SPARQL and its RDF triple semantics. If a user wants to query an OWL-DL ontology, he needs to translate his accustomed OWL syntax to the proper representation in RDF triples. This translation, if done manually, can be time consuming and is prone to syntactic as well as to semantic errors.

Existing approaches, which do not rely on RDF triples to specifically query OWL-DL ontologies, are lacking in different aspects. Terp [3] uses SPARQL as its query language and allows Manchester Syntax constructs to simplify the query. However, Terp is dependent on Pellet as its SPARQL processing engine and is consequently not applicable to other engines. SAIQL [4] introduces a new OWL-DL query language that solves the RDF triples issue, but it is not compatible to SPARQL. jOWL's implementation of SPARQL-DL [5] resolves the problem by applying OWL Functional-Style Syntax like axioms for querying, yet it only covers a restricted set of OWL-DL expressions.

Therefore, we introduce SPARQL-DL Abstract Syntax (SPARQLAS) that combines employing SPARQL to query OWL-DL ontologies and using OWL Functional-Style Syntax to compose queries. With SPARQLAS, the user does not have to translate between two languages, but is still able to utilize any

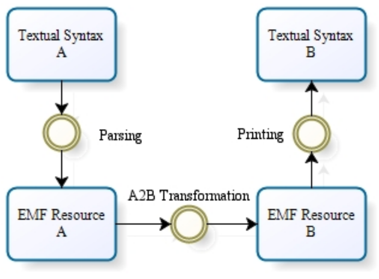

Fig. 1. Conceptual Approach

SPARQL processing engine that supports the OWL-DL entailment regime. The optional application of a shortened version of the OWL Functional-Style Syntax simplifies queries even more.

These advantages of SPARQLAS are accomplished by creating a new and more readable syntax out of SPARQL and the OWL Functional-Style Syntax and by transforming SPARQLAS to SPARQL. The transformation itself makes a special SPARQLAS processing engine needless and assures an easy embedding of SPARQLAS into running IDEs.

In the following, we first present in Section 2 the implementation of SPARQLAS at the TwoUse Toolkit. There we elaborate the conceptual approach of how a SPARQLAS query is parsed, transformed and printed to a SPARQL query before discussing the realization of the textual syntax and the core transformation. In Section 3 we show a concrete application example and in Section 4 we evaluate the efforts of SPARQLAS against SPARQL. Finally in Section 5, we conclude and discuss some future work.

The example throughout this paper deals with the commonly known wine ontology [6] and is querying for the region in which German wine is located.

2 Implementation

2.1 Conceptual Approach

To get a SPARQLAS query translated to a SPARQL query, we first need to define a meta model for each language so that a model transformation can be performed upon. For writing a SPARQLAS query and a subsequent use of a SPARQL query, those meta models have to be linked to a textual syntax. Fig. 1 illustrates the conceptual approach and shows what steps need to be realized for the complete translation.

Both textual syntaxes to SPARQLAS and SPARQL are created by using the Eclipse plug-in EMFText[1]. EMFText is able to generate a textual syntax derived from a meta model in Ecore [7] and a grammar in EBNF [8] notation. It also provides the parser to achieve an EMF Resource from a SPARQLAS

[1] http://www.emftext.org/index.php/EMFText

query and the printer to compose a SPARQL query out of an EMF Resource. The actual translation between both EMF Resources is performed by an ATL[2] model-to-model transformation.

2.2 Textual Syntax

The first action for implementing SPARQLAS is developing its textual syntax with EMFText, since a textual syntax for SPARQL is already available at the EMFText Concrete Syntax Zoo [9]. SPARQLAS is based on the theoretical concepts of SPARQL-DL [10] and adopts the framework of SPARQL, but replaces the RDF triples inside its *Where* clause by the OWL Functional-Style Syntax or a shortened version of it. There are all four kinds of queries supported, namely *Select*, *Construct*, *Ask* and *Describe*. One of SPARQLAS' goals is to make queries more readable, thus some SPARQL constructs are either simplified or completely omitted. For example there is no difference between *BASE* and *PREFIX* anymore, there is only the possibility to declare a *Namespace* with or without a prefix. The complete SPARQLAS grammar can be found within the online representation of the TwoUse project [11]. By means of these simplifications, the example SPARQL query

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX : <http://www.w3.org/TR/2003/PR-owl-guide-20031209/wine#>
SELECT DISTINCT ?x
WHERE {
 :GermanWine owl:equivalentClass [
  rdf:type owl:Class ;
  owl:intersectionOf [
   rdf:first :Wine ;
   rdf:rest [
    rdf:first [
     rdf:type owl:Restriction ;
     owl:onProperty :locatedIn ;
     owl:hasValue ?x ] ;
    rdf:rest rdf:nil ] ] ] }
```

can be reduced to this SPARQLAS query:

```
Namespace
   (=<http://www.w3.org/TR/2003/PR-owl-guide-20031209/wine#>)
Select
Where(EquivalentClasses(GermanWine And(Wine Has(locatedIn ?x))))
```

[2] http://www.eclipse.org/m2m/atl/

	Prologue	**From (Named)**	**Query Syntax**
SPARQL	Prefix, Base	yes	RDF Triple
SPARQLAS	Namespace	no	OWL-DL Expression
	Optional	**Graph**	**Union**
SPARQL	yes	yes	yes
SPARQLAS	no	no	no
	Filter	**Order**	**Limit**
SPARQL	yes	yes	yes
SPARQLAS	no	no	no

Table 1. Differences between SPARQL and SPARQLAS constructs

2.3 Model-to-Model Transformation

The ATL model-to-model transformation can be divided into two parts, the prologue and the query itself. The prologue consists of the *Namespace* declaration and will be transformed to the equivalent *PREFIX* notation. During the query part, the query form will be identified first and transformed to accordingly. If it is a *Select* query, the option *DISTINCT* will be set additionally. Then the main contribution of each transformation starts, translating an OWL expression to its equivalent RDF triple(s). These translations are based on specific OWL to RDF mapping rules [12, 13] and also finalize the transformation.

This way the expressiveness of SPARQLAS queries is equivalent to SPARQL queries using only RDF triples of the aforementioned mapping rules, which makes SPARQLAS a subset of SPARQL. Further, SPARQLAS is even a proper subset of SPARQL since not every SPARQL construct can be expressed by SPARQLAS, e.g. *FILTER* or *OPTIONAL* conditions. These constructs are omitted with regard to simplicity and readability. The complete list of differences between SPARQLAS and SPARQL can be found in Table 1.

This specific SPARQLAS to SPARQL transformation is implemented at the TwoUse Toolkit[3] and can be executed by opening the context menu on a SPARQLAS file under the entry *TwoUse* → *Transform to SPARQL*.

3 Application

As a running application example, we present SPARQLAS for UML. Fig. 2 depicts a sample UML diagram according to the example query annotated with stereotypes from the UML Profile for RDF and UML Profile for OWL [14]. Out of this UML model, an ontology can be produced by another transformation implemented at the TwoUse Toolkit. This ontology can then be queried using SPARQLAS.

We developed a context menu function on UML Operations via *SPARQLAS for UML* → *Add SPARQLAS Query*, which creates an

[3] http://code.google.com/p/twouse/

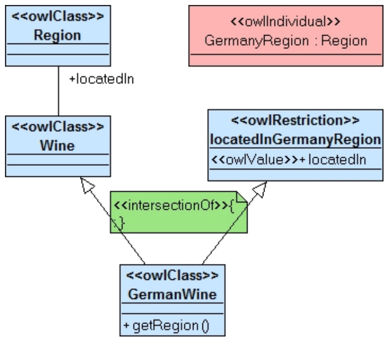

Fig. 2. SPARQLAS for UML example

empty SPARQLAS *Select* query with the given *Namespace* from the UML profile. This query file will be named after the convention *fileName.className.operationName*.sparqlas4uml, e.g. according to Fig. 2 it will be named *Wine.uml.GermanWine.getRegion*.sparqlas4uml. Strictly speaking this is a SPARQLAS4UML query, but it imports the whole SPARQLAS language and so shall indicate that a SPARQLAS query is linked to an actual UML file. Furthermore, this link between UML file and query will also be expressed at the UML file itself as the EMF Resource will be added to the model and so the query can be directly accessed through the UML model. This way it is possible to automatically generate SPARQLAS queries from UML diagrams. For this example the following empty *Select* query will be constructed:

```
Namespace
   ( = <http://www.w3.org/TR/2003/PR-owl-guide-20031209/wine#> )
Select
Where ( )
```

4 Evaluation

At this point we would like to evaluate SPARQLAS against SPARQL. For this purpose we examine 60 different test queries in two different ways with regard to complexity. First, we compare the underlying meta model of SPARQLAS and SPARQL. Due to EMFText every test query in textual syntax has an EMF Resource representation, therefore we can count each instantiation of a meta model class in these EMF Resources. To accelerate this counting process for a total of 120 queries, we make use of another plug-in of the TwoUse Toolkit called *OWLizer*, which can create ontologies out of Ecore based languages, e.g. EMF Resources. These ontologies consist of the desired individuals that we need to count and compare to achieve one indicator of complexity. After this procedure, we obtain a result that implies a substantial reduction in complexity of SPARQLAS queries against SPARQL queries. SPARQLAS queries reduce the count of individuals in average by 82% with a minimum of 71% and a maximum of 88% reduction with a standard deviation of 4%. The absolute numbers of individuals range from 5-32 for SPARQLAS queries and 31-178 for SPARQL queries.

The second evaluation kind deals with the length of each query, but instead of counting every single character, we compare the count of OWL expressions in each SPARQLAS query to the count of RDF triples an equivalent SPARQL query needs. This method ensures that the effort of using OWL expressions rather than RDF triples can be evaluated. The results of this procedure suggest that the efforts may not be as predominant as with the other evaluation scenario, yet the average reduction is by 50% with a maximum of 82% but having a standard deviation of 31%. This high standard deviation is explained by the fact that there are actual queries with no effort, i.e. one OWL expression is equivalent to one RDF triple, e.g. in case of *ClassAssertion*. However, there is only no effort in simple cases, more complex queries, especially nested queries, are showing more and more reduction. The absolute numbers of OWL expressions resp. RDF triples of our test queries range from 1-5 resp. from 1-23.

5 Conclusion and Future Work

In this paper we have presented the implementation of a novel query language for OWL-DL ontologies, called SPARQL-DL Abstract Syntax (SPARQLAS), that combines SPARQL and the OWL Functional-Style Syntax or a shortened version of it. The possible application of two different syntaxes with the same underlying meta model indicates that the specific querying syntax is easily changeable. This language can be employed just like a SPARQL query after the execution of a model-to-model transformation from SPARQLAS to SPARQL. We have demonstrated in a running application example, SPARQLAS for UML, how SPARQLAS queries can be connected to UML models and performed on ontologies created from these models by applying the UML Profile for RDF and UML Profile for OWL. Further, we have shown by means of an evaluation that SPARQLAS is effectively reducing the complexity of a query on an OWL-DL ontology, especially for nested queries.

In our future work, we will enhance SPARQLAS to support the Manchester Syntax and discuss the usage of other syntaxes as well. This way the user will be able to choose his preference among these syntaxes. Additionally, we will create a new application example to connect queries with Ecore models, SPARQLAS for Ecore, since the OWLizer plug-in is capable of generating ontologies from such Ecore models.

Acknowledgments

The work presented in this paper is supported by EU STReP-216691 MOST.

References

1. Prud'hommeaux, E., Seaborne, A.: SPARQL Query Language for RDF (15.01.2008) http://www.w3.org/TR/rdf-sparql-query/.
2. Motik, B., Patel-Schneider, P.F., Parsia, B.: OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (27.10.2009) http://www.w3.org/TR/owl2-syntax/.
3. Bulka, B.: Pellet 2.1: Introducing Terp (01.04.2010) http://clarkparsia.com/weblog/2010/04/01/pellet21-terp/.
4. Kubias, A., Schenk, S., Staab, S., Pan, J.: OWL SAIQL - An OWL DL Query Language for Ontology Extraction. In: OWLED 2007 OWL: Experiences and Directions Third International Workshop. (06.2007)
5. Decraene, D.: jOWL - SPARQL-DL Test Suite (03.2009) http://jowl.ontologyonline.org/SPARQL-DL.html.
6. W3C: Wine Ontology (10.02.2004) http://www.w3.org/TR/2004/REC-owl-guide-20040210/wine.rdf.
7. Steinberg, D., Budinsky, F., Paternostro, M., Merks, E.: EMF: Eclipse Modeling Framework. 2nd ed., rev. and updated. edn. The eclipse series. Addison-Wesley, Upper Saddle River, NJ (2009) 17-23.
8. International Organization for Standardization: ISO/IEC 14977:1996: Information technology - Syntactic metalanguage - Extended BNF (1996) http://standards.iso.org/ittf/PubliclyAvailableStandards/index.html.
9. EMFText: EMFText Concrete Syntax Zoo SPARQL (20.02.2010) http://www.emftext.org/index.php/EMFText_Concrete_Syntax_Zoo_SPARQL.
10. Sirin, E., Parsia, B.: SPARQL-DL: SPARQL Query for OWL-DL. In: 3rd OWL: Experiences and Directions Workshop (OWLED2007). (2007)
11. Schneider, M.: SPARQLASGrammar (13.04.2010) http://code.google.com/p/twouse/wiki/SPARQLASGrammar.
12. Patel-Schneider, P.F., Horrocks, I.: OWL Web Ontology Language Semantics and Abstract Syntax: Section 4. Mapping to RDF Graphs (06.02.2004) http://www.w3.org/TR/owl-semantics/mapping.html.
13. Patel-Schneider, P.F., Motik, B.: OWL 2 Web Ontology Language Mapping to RDF Graphs (28.10.2009) http://www.w3.org/TR/owl2-mapping-to-rdf/.
14. Object Management Group: Ontology Definition Metamodel (05.2009) 131-179, http://www.omg.org/spec/ODM/1.0/.