Towards a Mediator based on OWL and SPARQL

Konstantinos Makris, Nikos Bikakis, Nektarios Gioldasis, Chrisa Tsinaraki, Stavros Christodoulakis

Technical University of Crete, Department of Electronic and Computer Engineering
Laboratory of Distributed Multimedia Information Systems & Applications (MUSIC/TUC)
University Campus, 73100, Kounoupidiana Chania, Greece
{makris, nbikakis, nektarios, chrisa, stavros}@ced.tuc.gr

Abstract. We propose a framework that supports a federated environment based on a Mediator Architecture in the Semantic Web. The Mediator supports mappings between the OWL Ontology of the Mediator and the other ontologies in the federated sites. SPARQL queries submitted to the Mediator are decomposed and reformulated to SPARQL queries to the federated sites. The evaluated results return to the Mediator. In this paper we describe the mappings definition and encoding. We also discuss briefly the reformulation approach that is used by the Mediator system that we are currently implementing.

Keywords: Information Integration, Semantic Web, Interoperability, Ontology Mapping, Query Reformulation, SPARQL, OWL.

1 Introduction

The Semantic Web community has developed over the last few years standardized languages for describing ontologies and for querying OWL [1] based information systems. Database implementations are striving to achieve high performance for the stored semantic information. In the future large databases, RDF [2] data will be managed by independent organizations. Federated architectures will need to access and integrate information from those resources.

We consider in this paper a Mediator based architecture for integrating information from federated OWL knowledge bases. The Mediator uses mappings between the OWL Ontology of the Mediator and the Federated site ontologies. SPARQL [3] queries over the Mediator are decomposed and reformulated to be submitted over the federated sites. The SPARQL queries are locally evaluated and the results return to the Mediator site.

We describe in this paper the mappings supported by our architecture as well as the SPARQL reformulation algorithms supported by our system.

Ontology Mapping in general, is a topic that has been studied extensively in the literature [13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, and 24]. However, very few publications, in our knowledge, examine the problem of describing the mapping types that can be useful for the SPARQL query reformulation process and how they should be exploited. Only [7] deals with this subject but not directly, since it describes which mapping types cannot be used in the reformulation process.

Query Reformulation is a frequently used approach in *Query Processing* and especially in *Information Integration* environments. Much research has been done in the area of query reformulation; Up to now, though, limited studies have been made in the field of SPARQL query reformulation related to posing a query over different datasets. Some relevant work has been published for approximate query reformulation, based on an ontology mapping specification [12], using a part of OWL-DL, without specifying a particular query language. In addition, many approaches that deal with *optimization* [8, 9], *decomposition* [4, 5], *translation* [10, 11] and *rewriting* (in order to benefit from inference) [6] of a SPARQL query, have been published.

The rest of the paper is structured as follows: In section 2, we present a motivating example. Then, the patterns of the proposed correspondences and the mapping types are outlined in section 3, while the language that is used for mapping representation is discussed in section 4. The SPARQL query reformulation process is described in section 5 and the paper concludes in section 6.

2 Motivating Example

We present in this section a motivating example. In Fig. 1, we show the structure of two different ontologies. The source ontology describes a store that sells various products including books and cd's and the target ontology describes a bookstore.

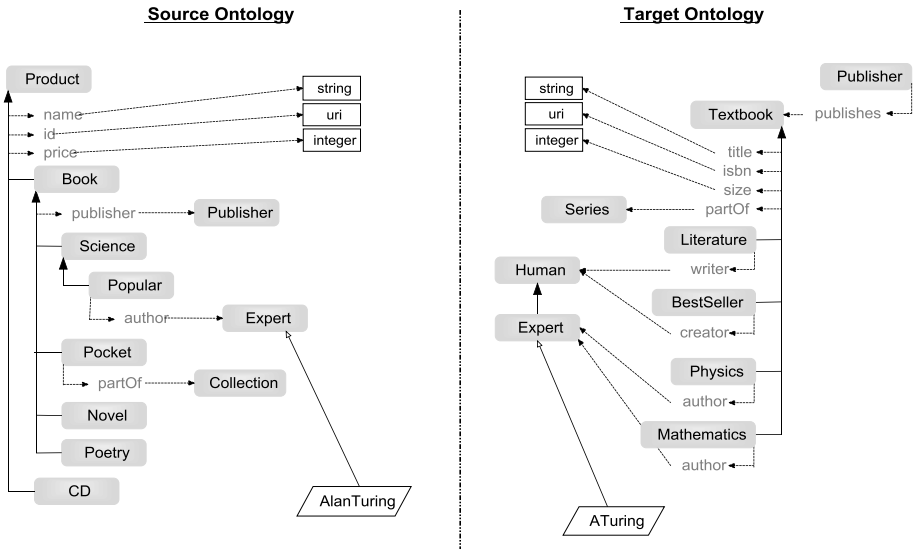

Fig. 1. Semantically Overlapping Ontologies. The notation is based on [24]. The rounded corner boxes represent the classes followed by their properties, the parallelogram boxes at the bottom express individuals, the rectangle boxes express the datatypes and finally, the arrows express the relationships between those basic constructs of OWL.

In Fig. 1, we observe some correspondences between the two ontologies. For example, the class “Book” in the source ontology seems to describe the same individuals with the class “Textbook” in the target ontology (equivalence relationship). In addition, correspondences like the one between the class “Collection” and the class “Series”, or the one between the datatype property “name” and the datatype property “title” can be thought as more general (subsumption relationships).

Apart from the obvious correspondences, we observe more complex ones such as those between the class “Science” and the union of the classes “Physics” and “Mathematics”, and the one between the class “Pocket” and the class “Textbook” restricted in its “size” property values.

3 Mapping Types

In this section we define the set of mapping types between the Mediator and the target ontologies. These mapping types are used for the SPARQL query reformulation process.

The basic concepts of OWL, whose mappings are useful for the reformulation process, are the classes (denoted as “*c*”), the object properties (denoted as “*op*”), the datatype properties (denoted as “*dp*”) and the individuals (denoted as “*i*”). Since we deal with SPARQL queries, some mapping types may not be useful for the query reformulation process. For example, a mapping between an individual of the source ontology and a concatenation of two different individuals of the target ontology would be meaningless, since they cannot be represented in SPARQL. Such types of mappings are described in [7] and many of them could be useful for processing the query results but not during the query reformulation and query answering process.

In order to define the mapping types that can be useful for the reformulation process, we use the set of symbols presented in Table 1.

Table 1. The notation used to define the different mapping types and concept/role expressions.

Symbol	Notation
⊑, ⊒	inclusion operators
≡	equality operator
⊓	intersection operator
⊔	union operator
¬	negation operator
∅	empty set
|	logical or
.	used to specify a sequence of binded concepts/roles
domain(c)	property domain restriction to the values of a class c
range(c)	property range restriction to the values of a class c
inverse(p)	inverse of a property p

We consider that a class expression (denoted as "*CE*") from the source ontology can be mapped to a class expression from the target ontology.

$$\textit{Class Expression Mapping:}\ \text{CE rel CE, rel:= } \equiv \mid \sqsubseteq \mid \sqsupseteq \qquad (\mathbf{1})$$

As a *class expression*, we denote any complex expression between two or more classes, using disjunctions, conjunctions or both. Any class that participates in the class expression can be restricted by the value of one or more properties, attached to it directly or indirectly (directly in some of its own properties, or indirectly in some property of an associated class) using a path (provided that a path connecting the class and the desired property already exists).

$$\text{CE::= c} \mid \text{c.R} \mid \text{CE} \sqcup \text{CE} \mid \text{CE} \sqcap \text{CE},$$

$$\text{R::= R'} \mid \neg\text{R'} \mid \text{R} \sqcup \text{R} \mid \text{R} \sqcap \text{R},$$

$$\text{R'::= P opr V, opr:= != } \mid = \mid \leq \mid \geq \mid < \mid >$$

R denotes the *restrictions* applied in a class, while *R'* stands for the restriction of a property path *P* in a possible value *V*. *V* can be either a data value or an individual. The operators that can be used in a property path restriction differ according to the type of *V*. In order to define a restriction on a data value all the above operators can be used (!=, =, ≤, ≥, <, >). But, in order to define a restriction on an individual only the !=, = operators can be used.

A *property path P* is a sequence of object properties (possibly empty) ending with a datatype/object property. It relates the class that we want to restrict with the property (object or datatype) in which the restriction should be applied.

$$\text{P::= P'} \mid \text{dp} \mid \text{P'.dp}$$

$$\text{P'::= } \emptyset \mid \text{op} \mid \text{P'.op}$$

Accordingly, an object property expression (denoted as "*OPE*") from the source ontology can be mapped to an object property expression from the target ontology.

$$\textit{Object Property Expression Mapping:}\ \text{OPE rel OPE} \qquad (\mathbf{2})$$

As an *object property expression*, we denote any complex expression between two or more object properties, using disjunctions, conjunctions or both. It is also possible for the inverse of an object property to participate in the object property expression of the target class. Any object property that participates in the object property expression can be restricted on its domain or range values, using the same type of restrictions with those described for the class expressions.

$$\text{OPE::= op} \mid \text{OPE} \sqcup \text{OPE} \mid \text{OPE} \sqcap \text{OPE} \mid \text{OPE} \sqcap \text{domain(CE)} \mid \text{OPE} \sqcap \text{range(CE)} \mid \text{inverse(OPE)}$$

Similarly, a datatype property expression (denoted as "*DPE*") from the source ontology can be mapped to a datatype property expression from the target ontology.

$$\textit{Datatype Property Expression Mapping:}\ \text{DPE rel DPE} \qquad (\mathbf{3})$$

As a *datatype property expression*, we denote any complex expression between two or more datatype properties, using disjunctions, conjunctions or both. Any datatype property that participates in the datatype property expression can be restricted on its domain values.

$$DPE ::= dp \mid DPE \sqcup DPE \mid DPE \sqcap DPE \mid DPE \sqcap domain(CE)$$

Finally, an individual from the source ontology (denoted as "i_s") can be mapped with an individual from the target ontology (denoted as "i_t").

$$\textit{Individual Mapping: } i_s \equiv i_t \quad (4)$$

We mention here that the equivalence between two different properties or property expressions denotes equivalence between the domains and ranges of those properties or property expressions. Similarly, the subsumption relations between two different properties or property expressions denote analogous relations between the domains and ranges of those two properties or property expressions.

4 Mapping Representation

The language that we use in order to represent the mappings between two overlapping ontologies has been defined in [20, 21]. It combines the alignment format [18], a format used to represent the output of ontology matching algorithms, and the OMWG mapping language [22] an expressive ontology alignment language. The expressiveness, the simplicity, the Semantic Web compliance (given its RDF syntax) and the capability of using any kind of ontology language are the key features of this language.

Below, we list a possible set of correspondences, using first-order logic expressions for the ontologies presented in Fig. 1 and afterwards we provide an example showing the mapping representation of a specific correspondence using the language that we mentioned above.

Type (1) mappings:

a. $\forall x, [Book(x) \Leftrightarrow Textbook(x)]$

b. $\forall x, [Publisher(x) \Leftrightarrow Publisher(x)]$

c. $\forall x, [Expert(x) \Leftrightarrow Expert(x)]$

d. $\forall x, [Product(x) \Rightarrow Textbook(x)]$

e. $\forall x, [Collection(x) \Leftarrow Series(x)]$

f. $\forall x, [Science(x) \Leftrightarrow (Physics(x) \vee Mathematics(x))]$

g. $\forall x, [Popular(x) \Leftrightarrow ((Physics(x) \vee Mathematics(x)) \wedge BestSeller(x))]$

h. $\forall x, [Pocket(x) \Leftrightarrow (Textbook(x) \wedge \exists y; [size(x, y) \wedge y \leq 14])]$

i. $\forall x, [(Novel(x) \vee Poetry(x) \Leftrightarrow Literature(x))]$

Type (2) mappings:

j. $\forall x, \forall y\ [publisher(x, y) \Leftrightarrow publishes(y, x)]$

k. $\forall x, \forall y\ [author(x, y) \Leftrightarrow (author(x, y) \wedge creator(x, y))]$ [1]

l. $\forall x, \forall y\ [partOf(x, y) \Leftrightarrow (partOf(x, y) \wedge \exists z;\ [size(x, z) \wedge z \leq 14])]$

Type (3) mappings:

m. $\forall x, \forall y\ [name(x, y) \Rightarrow title(x, y)]$

n. $\forall x, \forall y\ [id(x, y) \Rightarrow isbn(x, y)]$

Type (4) mappings:

o. AlanTuring = ATuring

The representation of mapping **f** (presented above) using the language that was discussed in this section is shown in Fig. 2.

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
…
<rdf:RDF xmlns="http://www.omwg.org/TR/d7/ontology/alignment/" …>
  <Alignment rdf:about="http://example.com/ ">
    <xml>yes</xml>
    …
    <onto1>
      <Ontology rdf:about="&Source;"/>
      …
    </onto1>
    <onto2>
      <Ontology rdf:about="&Target;"/>
      …
    </onto2>
    <map>
      …
      <Cell>
        <entity1>
          <Class rdf:about="&Source;Science"/>
        </entity1>
        <entity2>
          <Class>
            <or rdf:parseType="Collection">
              <Class rdf:about="&Target;Physics"/>
              <Class rdf:about="&Target;Mathematics"/>
            </or>
          </Class>
        </entity2>
        <relation rdf:resource="&omwg;equivalence"/>
      </Cell>
      …
    </map>
  </Alignment>
</rdf:RDF>
```

Fig. 2. The representation of mapping **f**.

[1] The object property "author" in the target ontology consists of two different domains. According to the semantics of OWL, this means that the domain of the object property "author" is actually the union of the classes "Physics" and "Mathematics".

5 SPARQL Query Reformulation

In this section we provide an overview of the SPARQL query reformulation process, using a predefined set of mappings that follows the different mapping types described in section 3.

The SPARQL query reformulation process is based on the query's *graph pattern* reformulation and is consequently independent of the query type (*Ask, Select, Construct, Describe*). The SPARQL solution modifiers (*Limit, Offset, Order By, Distinct, Reduce*) are not taken into consideration since they do not affect the reformulation process.

In order to reformulate the *graph pattern* of a SPARQL query using *1:N* mappings (mappings between a basic OWL concept of the source ontology and a complex expression, among basic OWL concepts, of the target ontology), the reformulation algorithm traverses the execution tree in a bottom up approach, taking each triple pattern of the *graph pattern* and checking for existing mappings of the subject, predicate and object part in the predefined mappings set. Finally, it reformulates the triple pattern according to those mappings.

In case of *N:M* mapping utilization (mappings between a complex expression of the source ontology and a complex expression of the target ontology), the total graph pattern must be parsed in order to discover the predefined complex mapping and then, the algorithm must produce the required combination of triple patterns, based on this mapping.

The SPARQL graph pattern operators (*AND, UNION* and *OPTIONAL*), do not result in modifications during the reformulation process.

The reformulation of the *FILTER* expressions is performed by reformulating the existing *IRIs* that refer to a class, property, or individual, according to the specified mappings. The SPARQL variables, literal constants, operators (*&&, ||, !, =, !=, >, <, >=, <=, +, -, *, /*) and built-in functions (e.g. *bound, isIRI, isLiteral, datatype, lang, str, regex*) that may occur in a *FILTER* expression remain the same during the reformulation process.

Finally, in case that more than one mappings are specified for a given class or property expression of the source ontology, the reformulation algorithm chooses the one that produces the most efficient reformulated query. Moreover, for efficiency reasons, a graph pattern normalization step is applied in parallel to the reformulation process, similarly with the one that is described in our SPARQL2XQuery [11] framework.

5.1 Reformulation Examples [2] [3]

We briefly present in this section the SPARQL reformulation process, using a set of examples due to the space limitation. We assume that an initial SPARQL query is posed over the source ontology presented in Fig. 1 and is reformulated to a semantically equivalent query in order to be posed over the target ontology of Fig. 1, using the mappings specified in section 4.

Example 1 : Consider the query posed over the source ontology: *"Return the titles of the pocket-sized scientific books"*. The SPARQL syntax of the source query and the reformulated query is shown in Fig.3. During the reformulation process the mappings *f)*, *h)* and *m)* from section 4 are used.

Source Query:

```
SELECT ?name
WHERE{
  ?x s:name ?name.
  ?x rdf:type s:Science.
  ?x rdf:type s:Pocket.
}
```

Reformulated - Target Query:

```
SELECT ?name
WHERE{
  {?x t:title ?name.}
  {{?x rdf:type t:Physics.}
   UNION
   {?x rdf:type t:Mathematics.}}
  {?x rdf:type t:Textbook.
   ?x t:size ?size.
   FILTER (?size ≤ 14)}
}
```

Fig. 3. The source and reformulated queries of Example 1.

Example 2 : Consider the query posed over the source ontology: *"Return the titles of books that belong to the poetry or novel category"*. The SPARQL syntax of the source query and the reformulated query is shown in Fig.4. During the reformulation process the mappings *i)* and *m)* from section 4 are used.

[2] The graph pattern normalization step is not included in the examples presented here, in order to make the reformulation process more easily understandable.

[3] For the SPARQL query examples presented in this subsection we use the following prefixes:

```
PREFIX s: <http://example.com/Source.owl#>
PREFIX t: <http://example.com/Target.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
```

Source Query:

```
SELECT ?name
WHERE{
  {?x s:name ?name.}
  {{?x rdf:type s:Poetry.}
   UNION
   {?x rdf:type s:Novel.}}
}
```

Reformulated - Target Query:

```
SELECT ?name
WHERE{
  ?x t:title ?name.
  ?x rdf:type t:Literature.
}
```

Fig. 4. The source and reformulated queries of Example 2.

6 Conclusions

In this paper we presented the formal definition and the encoding of the mappings of a semantic based mediation framework (based on OWL/RDF knowledge representations and SPARQL queries) that we are currently developing. The framework is based on a set of mapping types that can be useful in the context of SPARQL query reformulation. Thus, a SPARQL query that can be posed over a source ontology, is reformulated, according to the mappings, in order to be capable of being posed over a target ontology. We have also outlined the SPARQL query reformulation process and presented examples of query reformulation in a motivating example.

This work is part of a framework that we are pursuing, which aims to provide algorithms, proofs and middleware for the support of transparent access to federated heterogeneous databases across the web in the Semantic Web environment.

7 References

1. McGuinness D. L., van Harmelen F. (eds.): "OWL Web Ontology Language: Overview". W3C Recommendation, 10 Feb. 2004. (http://www.w3.org/TR/owl-features)
2. Manola F., Milles E. (eds.): "RDF Primer". W3C Recommendation, 10 Feb. 2004. (http://www.w3.org/TR/rdf-primer)
3. Prud'hommeaux E., Seaborne A. (eds.): "SPARQL Query Language for RDF". W3C Recommendation, 15 January 2008. (http://www.w3.org/TR/rdf-sparql-query/)
4. S.M. Benslimane, A. Merazi, M Malki, D. Amar Bensaber: "Ontology mapping for querying heterogeneous information sources". INFOCOMP (Journal of Computer Science) 7(2):44-51, ISSN 1807-4545. 2008
5. Bastian Quilitz, Ulf Leser: "Querying Distributed RDF Data Sources with SPARQL". In Proceedings of ESWC 2008
6. Yixin Jing, Dongwon Jeong, Doo-Kwon Baik: "SPARQL Graph Pattern Rewriting for OWL-DL Inference Query" Proceedings of the 2008 Fourth International Conference on Networked Computing and Advanced Information Management
7. Jérôme Euzenat, Axel Polleres, François Scharffe: "Processing ontology alignments with SPARQL" (Position paper). International Workshop on Ontology Alignment and Visualization, CISIS 2008, Barcelona, Spain, March 2008

8. Markus Stocker, Andy Seaborne, Abraham Bernstein, Christoph Kiefer, Dave Reynolds: "SPARQL basic graph pattern optimization using selectivity estimation" In Proceedings of WWW 2008
9. Olaf Hartig, Ralf Heese: "The SPARQL Query Graph Model for Query Optimization". In Proceedings of ESWC 2007
10. Christian Bizer, Richard Cyganiak: D2R Server. http://www4.wiwiss.fu-berlin.de/bizer/d2r-server/index.html
11. Nikos Bikakis, Nektarios Gioldasis, Chrisa Tsinaraki, Stavros Christodoulakis: "Querying XML Data with SPARQL". In Proceedings of the 20th International Conference on Database and Expert Systems Applications (DEXA '09)
12. J. Akahani, K. Hiramatsu, T. Satoh: "Approximate Query Reformulation for Ontology Integration". In Proc of the Semantic Integration Workshop Collocated with the Second International Semantic Web Conference (ISWC-03)
13. Namyoun Choi, Il-Yeol Song, Hyoil Han: A survey on ontology mapping. SIGMOD Record 35(3): 34-41 (2006)
14. Yannis Kalfoglou, W. Marco Schorlemmer: Ontology Mapping: The State of the Art. Semantic Interoperability and Integration 2005
15. Chiara Ghidini, Luciano Serafini: Mapping Properties of Heterogeneous Ontologies. AIMSA 2008: 181-193
16. Chiara Ghidini, Luciano Serafini: Reconciling Concepts and Relations in Heterogeneous Ontologies. ESWC 2006: 50-64
17. Chiara Ghidini, Luciano Serafini, Sergio Tessaris: On Relating Heterogeneous Elements from Different Ontologies. CONTEXT 2007: 234-247
18. Euzenat, J.: An API for ontology alignment. In Proc. 3rd International Semantic Web Conference, 2004: 678-712
19. François Scharffe, Jos de Bruijn: A language to specify mappings between ontologies. SITIS 2005: 267-271
20. Jérôme Euzenat, François Scharffe, Antoine Zimmermann: D2.2.10: Expressive alignment language and implementation. Knowledge Web EU-IST Project deliverable 2.2.10, 2007
21. Scharffe, F.: PhD thesis: "Correspondence Patterns Representation". Available at: http://www.scharffe.fr/pub/phd-thesis/manuscript.pdf
22. Scharffe, F., de Bruijn, J.: A language to specify mappings between ontologies. In: Proc. of the Internet Based Systems IEEE Conference (SITIS05). (2005)
23. François Scharffe. Omwg d7: Ontology mapping language. http://www.omwg.org/TR/d7/, 2007
24. J. Euzenat, P. Shvaiko: Ontology matching. Springer-Verlag, Heidelberg, 2007