Querying and Reasoning with RDF(S)/OWL in XQuery*

Jesús M. Almendros-Jiménez

Dpto. de Lenguajes y Computación,
Universidad de Almería, Spain
jalmen@ual.es

Abstract. In this paper we investigate how to use the XQuery language for querying and reasoning with RDF(S)/OWL-style ontologies. Our proposal allows the handling of RDF(S)/OWL triples by means of a XQuery library for the Semantic Web, and it encodes RDF(S)/OWL reasoning by means of XQuery functions. We have tested and implemented the approach.

1 Introduction

The *Semantic Web* framework [9,14] proposes that *Web data* represented by *HMTL* and *XML* have to be enriched by means of *meta-data*, in which modeling is mainly achieved by means of the *Resource Description Framework (RDF)* [19] and the *Web Ontology Language (OWL)* [20]. OWL is based on the so-called *Description Logic (DL)* [6], which is a family of logics (i.e. *fragments*) with different expressive power. On the other hand, *XQuery* [11] is a typed functional language devoted to express queries against XML documents. It contains *XPath 2.0* [8] as a sublanguage. *XPath 2.0* supports navigation, selection and extraction of fragments from XML documents. *XQuery* also includes *flowr* expressions (i.e. *for-let-orderby-where-return* expressions) to construct new XML values and to join multiple documents.

In this paper we investigate how to use the XQuery language for querying and reasoning with RDF(S)/OWL-style ontologies. The main features of our proposal can be summarized as follows:

– XQuery has been developed for querying XML documents, however, Web data can be also represented by means of RDF(S) and OWL. Therefore, XQuery should support the simultaneous querying of Web data by both its structure and by its associated meta-data given in form of RDF(S)/OWL-style ontologies. This is an important problem for supporting data discovery tasks against collections of Web data. The proposed approach allows to query XML/RDF(S)/OWL documents and to obtain as output the same kind of documents.

* This work has been partially supported by the Spanish MICINN under grant TIN2008-06622-C03-03.

X. Du et al. (Eds.): APWeb 2011, LNCS 6612, pp. 450–459, 2011.

- RDF(S) and OWL querying should be combined with reasoning. RDF(S) and OWL allows to express and infer complex relationships between entities which should be exploited by means of a query language. The proposed approach is able to use semantic information inferred from RDF(S) and OWL resources.
- Finally, we will propose an implementation of the approach. Firstly, RDF(S)-/OWL triples can be handled by means of a XQuery library providing support to the traversal of triples, and the access to triples components. Secondly, RDF(S)/OWL reasoning is *encoded* by means of XQuery functions which facilitates the use of the ontology in queries. In addition, such an encoding allows to reason with the elements of the ontology.

A great effort has been made for defining query languages for RDF(S)/OWL documents (see [7] for a survey about this topic). The proposals mainly fall on extensions of *SQL*-style syntax for handling the *triple-based RDF structure*. In this line the most representative language is *SPARQL* [21]. Some authors [10,15,1] have already investigated how to combine XML and RDF(S)/OWL querying. RDF(S)/OWL can be combined with XML query languages, for instance, by encoding RDF(S)/OWL into XML data and by encoding SPARQL in XQuery. Alternatively, in [13], SPARQL and XQuery are combined in order to provide an unified framework for RDF and XML. We have considered in our approach a different point of view. XQuery can be equipped with a library for the handling of RDF(S)/OWL. And also, the ontologies can be encoded by means of XQuery functions. The advantage of our approach is that XQuery functions are able to reason about the ontology, that is, they are able to infer new information from the **TBox** and **ABox**.

OWL/DL reasoning is a topic of research of increasing interest in the literature. Most of DL reasoners (for instance, *Racer* [16], *FaCT++* [23], *Pellet* [22]) are based on tableaux based decision procedures. We can distinguish two main reasoning mechanism in this context. The first mechanism focuses on checking *consistence* of the ontology. Consistence means that the ontology has a model. The second mechanism focuses on retrieving *logic consequences* from the ontology. Such logic consequence includes the transitive closure of the subclass and subproperty relationships, the membership of individuals to classes, and the relations between individuals, which are not explicitely declared in the ontology.

In our proposal, we have studied a fragment of Description Logic/OWL which can be easily encoded in XQuery. Such an encoding takes Description Logic formulas and encodes them by means of XQuery functions. XQuery is based on the use of *flowr* expressions for querying XML documents. In our case, we use *flowr* expressions to query OWL documents, and it allows to encode OWL ontologies by means of XQuery. Such an encoding allows to *reason* about the elements of the **ABox** of the ontology: we can infer the *membership of individual to classes* and the *roles that two individuals play*. In addition, we are interested in the use of XQuery for querying the meta-data of the **TBox**. In this case, the XQuery library can be used for *retrieving the elements of the* **TBox**.

OWL is a language with different fragments restricted in expressive power (i.e. they consider different subsets of the OWL vocabulary and define restrictions in the use of some of them) in order to retain reasoning capabilities (tractability, scalability, complexity, etc) and decidability. Recently, three fragments of the new OWL 2 have been proposed: OWL EL, OWL RL and OWL QL [20]. In particular, OWL QL is a fragment OWL 2 allowing efficient querying. With respect to the fragment of DL considered in our proposal, we have taken a fragment which is expressive enough that includes the basic elements of OWL. It is a subset of the $\mathcal{ALC}$ fragment with some restrictions incorporating some relations between properties. One key point of our fragment is that the **TBox** has to be acyclic. This condition is required in order to be supported by XQuery. However, we will discuss in the paper the adaptation of our proposal to the recently proposed OWL QL.

We would like to remark that our work continues previous work about XQuery. In [2], we have studied how to define an extension of XQuery for querying RDF(S)/OWL documents. Similarly to the current proposal, such an extension allows to query XML and RDF(S)/OWL resources with XQuery, however, the proposed extension of the quoted papers is based on the implementation of XQuery in a logic language like Prolog (see [5,4,3] for more details).

Finally, we have tested and implemented the proposal of this paper. The XQuery library for handling RDF(S)/OWL together with an example of ontology encoded in XQuery can be downloaded from `http://indalog.ual.es/XQuerySemanticWeb`.

The structure of the paper is as follows. Section 2 will present the kind of ontologies we consider in our framework. Section 3 will describe the XQuery library for RDF(S)/OWL. Section 4 will show the encoding of ontologies in XQuery. Section 5 will give some examples of ontology querying and finally, Section 6 will conclude and present future work.

2 Ontology Representation

In this section we will define the fragment of OWL of our framework.

Definition 1. *An ontology $\mathcal{O}$ in our framework contains a* **TBox** *including a sequence of definitions of the form:*

$C \sqsubseteq D$	(`rdfs:subClassof`)	$P \equiv Q^-$	(`owl:inverseOf`)
$E \equiv F$	(`owl:equivalentClass`)	$P \equiv P^-$	(`owl:SymmetricProperty`)
$P \sqsubseteq Q$	(`rdfs:subPropertyOf`)	$\top \sqsubseteq \forall P^-.D$	(`rdfs:domain`)
$P \equiv Q$	(`owl:equivalentProperty`)	$\top \sqsubseteq \forall P.D$	(`rdfs:range`)

where C is a class (i.e. concept) description of left-hand side type *(denoted by $C \in \mathcal{L}$), of the form: $C ::= C_0 \mid \neg C \mid C_1 \sqcup C_2 \mid C_1 \sqcap C_2 \mid \exists P.C \mid \forall P.C$ and D is a* class (i.e. concept) description of right-hand side type *(denoted by $D \in \mathcal{R}$), of the form: $D ::= C_0 \mid D_1 \sqcap D_2 \mid \forall P.D$. In all previous cases, C_0 is an atomic class (i.e. class name) and P, Q are* property (i.e. role) names. *Formulas of equivalence $E \equiv F$ are syntactic sugar for $E \sqsubseteq F$ and $F \sqsubseteq E$. In addition, the* **ABox** *contains a sequence of definitions of the form: $P(a,b) \mid C_0(a)$ where P is*

TBox	
(1) $Man \sqsubseteq Person$	*(2)* $Woman \sqsubseteq Person$
(3) $Person \sqcap \exists author\_of.Manuscript \sqsubseteq Writer$	*(4)* $Paper \sqcup Book \sqsubseteq Manuscript$
(5) $Manuscript \sqcap \exists reviewed\_by.Person \sqsubseteq Reviewed$	
(6) $Manuscript \sqcap \neg \exists\ rating.Score \sqsubseteq Unrated$	
(7) $Manuscript \sqsubseteq \forall rating.Score$	*(8)* $Manuscript \sqsubseteq \forall of\_topic.Topic$
(9) $author\_of \equiv writes$	*(10)* $average\_rating \sqsubseteq rating$
(11) $authored\_by \equiv author\_of^{-}$	*(12)* $\top \sqsubseteq \forall\ author\_of.Manuscript$
(13) $\top \sqsubseteq \forall\ author\_of^{-}.Person$	*(14)* $\top \sqsubseteq \forall\ reviewed\_by.Person$
(15) $\top \sqsubseteq \forall\ reviewed\_by^{-}.Manuscript$	*(16)* $friendOf^{-} \equiv friendOf$
ABox	
(1) *Man*(“*Abiteboul*”)	
(2) *Man*(“*Buneman*”)	*(3)* *Man*(“*Suciu*”)
(4) *Book*(“*Data on the Web*”)	*(5)* *Book*(“*XML in Scottland*”)
(6) *Paper*(“*Growing XQuery*”)	*(7)* *Person*(“*Anonymous*”)
(8) *author_of*(“*Abiteboul*”, “*Data on the Web*”)	*(9)* *authored_by*(“*Data on the Web*”, “*Buneman*”)
(10) *author_of*(“*Suciu*”, “*Data on the Web*”)	*(11)* *author_of*(“*Buneman*”, “*XML in Scottland*”)
(12) *writes*(“*Simeon*”, “*Growing XQuery*”)	*(13)* *reviewed_by*(“*Data on the Web*”, “*Anonymous*”)
(14) *reviewed_by*(“*Growing XQuery*”, “*Almendros*”)	*(15)* *rating*(“*XML in Scottland*”, “*excellent*”)
(16) *average_rating*(“*Growing XQuery*”, “*good*”)	*(17)* *of_topic*(“*Data on the Web*”, “*XML*”)
(18) *of_topic*(“*Data on the Web*”, “*Web*”)	*(19)* *of_topic*(“*XML in Scottland*”, “*XML*”)
(20) *friendOf*(“*Simeon*”, “*Buneman*”)	

Fig. 1. An Example of Ontology

a property name, C_0 *is a class name, and* a, b *are* individual names. *We require that the* **TBox** *does not include concepts and roles that depend themself, that is, the* **TBox** *is acyclic.*

Our fragment of OWL is equipped with the basic elements of OWL: equivalence and inclusion of classes, and equivalence and inclusion of properties. In addition, it allows the specification of inverse and symmetric properties, and the domain and range of properties. We have restricted our approach to *object properties*, it forces to consider properties on *datatypes* as object properties. Let us remark that in right hand-sides of inclusion relations is not allowed to specify either $\neg C$ or $D_1 \sqcup D_2$ or $\exists P.C$. The reason for that is the encoding in XQuery of the ontology. Basically, the encoding allows to query the ontology and to use inclusion and equivalence for reasoning. Finally, we require that the **TBox** is acyclic. The fragment is basically $\mathcal{ALC}$ with restrictions in right hand-sides of class axioms, adding some property relations. Let us see now an example of a such a DL ontology (see Figure 1).

The ontology of Figure 1 describes, in the **TBox**, meta-data in which the elements of *Man* and the elements of *Woman* are elements of *Person* (axiom (1) and (2)); and the elements of *Paper* and *Book* are elements of *Manuscript* (axiom (4)). In addition, a *Writer* is a *Person* who is the *author_of* a *Manuscript* (axiom (3)), and the class *Reviewed* contains the elements of *Manuscript reviewed_by* a *Person* (axiom (5)). Moreover, the *Unrated* class contains the elements of *Manuscript* which have not been rated (axiom (6)). The classes *Score* and *Topic* contain, respectively, the values of the properties *rating* and *of_topic* associated to *Manuscript* (axioms (7) and (8)). The property *average_rating* is a subproperty of *rating* (axiom (10)). The property *writes* is equivalent to *author_of*

(axiom (9)), and *authored_by* is the inverse property of *author_of* (axiom (11)). Finally, the property *author_of*, and conversely, *reviewed_by*, has as domain a *Person* and as range a *Manuscript* (axioms (12)-(15)). Axiom (16) defines *friend_of* as a symmetric relation. The **ABox** describes data about two elements of *Book*: "Data on the Web" and "XML in Scottland" and a *Paper*: "Growing XQuery". It describes the *author_of* and *authored_by* relationships for the elements of *Book* and the *writes* relation for the elements of *Paper*. In addition, the elements of *Book* and *Paper* have been reviewed and rated, and they are described by means of a topic.

3 A Semantic Web Library in XQuery

Now, we will describe the Semantic Web library defined in XQuery. In summary, we have to handle the triples of the **TBox** and **ABox** of the given ontology. For simplicity we will assume that the elements of the **ABox** and the **TBox** has been declared in separated *RDF* items, *one for each formula*. Firstly, in order to handle the **ABox**, we have defined the following XQuery function:

```
declare function sw:abox($doc as node()) as node()*{$doc//owl:Thing[@rdf:about]};
```

which returns the sequence (i.e. by means of the XQuery "`node()*`" type) of triples of the **ABox** included in the document `$doc`. In order to access to the components of each triple of the **ABox** we have defined XQuery functions `sw:subject`, `sw:object` and `sw:property`. For instance,

```
declare function sw:subject($triple as node()) as xs:string{ string($triple/@rdf:about)};
```

Now, the **TBox** can be also handled by means of the library. For instance, we can obtain the classes in the **TBox** which are equivalent to a given class as follows:

```
declare function sw:equivalentClass($doc as node(),$class as node()) as node()*{
(for $x in $doc//owl:Class[owl:equivalentClass/@rdf:resource]
        where sw:eq($x/@rdf:about,$class/@rdf:about)
return sw:toClass($x/owl:equivalentClass/@rdf:resource)) union
(for $x in $doc//owl:Class[owl:equivalentClass/@rdf:resource]
        where sw:eq($x/owl:equivalentClass/@rdf:resource,$class/@rdf:about)
return sw:toClass($x/@rdf:about)) union
(for $x in $doc//owl:Class[owl:equivalentClass/owl:intersectionOf]
        where sw:eq($x/@rdf:about,$class/@rdf:about)
return $x/owl:equivalentClass/owl:intersectionOf ) union
(for $x in $doc//owl:Class[owl:equivalentClass/owl:Restriction]
        where sw:eq($x/@rdf:about,$class/@rdf:about)
return $x/owl:equivalentClass/owl:Restriction)
};
```

Let us remark that a class E can be equivalent by means $E \equiv F$, $E \equiv D_1 \sqcap D_2$ and $E \equiv \forall P.D$. The library handles the equivalence formula $E \equiv F$ as not oriented, and then returns E as equivalent to F, and F as equivalent to E. The previous function `sw:toClass` represents a class name in OWL format. The previous `sw:eq` is a boolean function to check equality between *RDF* identifiers.

Similarly, we can get the subclasses or superclasses of a given class in the **TBox**, by means of the functions `sw:subClassOf` and `sw:superClassOf`, respectively. In the library, there are also XQuery functions for querying about properties. For instance, the following function returns the domain of a property:

```
declare function sw:domain($doc as node(),$property as node()) as node()*{
for $x in $doc//owl:ObjectProperty where sw:eq($property/@rdf:about,$x/@rdf:about)
return sw:toClass($x/rdfs:domain/@rdf:resource)
};
```

There are also functions `sw:inverseOf`, `sw:equivalentProperty`, `sw:subPropertyOf`, `sw:superPropertyOf` and `sw:range`, and a function `sw:SymmetricProperty` to retrieve the symmetric properties. Finally, we can query the elements of the ontology like classes, properties and individuals. For instance, the following function returns the classes of the **TBox**:

```
declare function sw:Classes($doc as node()) as node()*{
for $x in $doc//owl:Class[@rdf:about] return sw:toClass($x/@rdf:about)};
```

In the library, there are also functions `sw:Properties`, `sw:Individuals`, `sw:Unions`, `sw:Intersections`, `sw:ClassesofIntersections`, `sw:ClassesofUnions`, `sw:Restrictions` (i.e. formulas built from $\forall$ and $\exists$). Also, there are also functions `sw:oftype`, `sw:type`, `sw:ofrole` and `sw:role`, retrieving the individuals of a certain type, the types of an individual, and the same for roles. Finally, given a property filler we can query the related individuals by means of two functions: `sw:About` and `sw:Resource`.

Now, using the Semantic Web library we can write queries for retrieving the elements of the ontology, expressing the result in XML format. For instance, we can get the atomic superclasses of a each class:

```
<classes>{
for $x in sw:Classes(doc('example.owl'))
return <class> string($x/@rdf:about) </class>
union
<superclasses>{
for $y in sw:subClassOf(doc('example.owl'),
sw:toClass(string($x/@rdf:about)) where $x/@rdf:about
return string($y/@rdf:about)
} </superclasses>
} </classes>
```

obtaining w.r.t. the running example the XML document:

```
<classes>
       <class>#Man</class>
          <superclasses>#Person</superclasses>
       <class>#Woman</class>
          <superclasses>#Person</superclasses>
....
</classes>
```

In summary, the proposed Semantic library of XQuery allows to query the elements of the **TBox** and the **ABox**. However, for reasoning with the **TBox** and the **ABox**, we have to encode the **TBox** by means of XQuery functions.

4 Encoding of Ontologies by Means of XQuery

Now, we present an example of encoding of an ontology by means of XQuery. The encoding allows to obtain the logic consequences from the **TBox** about the elements **ABox**. For instance, the class *Writer* of the running example can be defined in XQuery as follows:

```
declare function ex:writer($doc as node()) as node()*{
sw:oftype($doc,"Writer") union(
for $x in ex:author_of($doc) where
       (some $z in ex:person($doc) satisfies
       sw:eq(sw:About($z),sw:About($x)))
       and (some $u in ex:manuscript($doc) satisfies
       sw:eq(sw:About($u),sw:Resource($x)))
return sw:toIndividual(sw:About($x)))
};
```

The previous encoding defines the class *Writer* as the union of: the elements of the **ABox** of type *Writer*, and the "*Person*'s which are *author_of* a *Manuscript*", following the **TBox** formula $Person \sqcap \exists author\_of.Manuscript \sqsubseteq Writer$. The elements of the **ABox** of type *Writer* are retrieved by means of a call to the function `sw:oftype` of the Semantic library. The elements of the class *Person* are computed by means of the call to the function `ex:person`, which is defined similarly to `ex:writer`. The same can be said for `ex:author_of` and `ex:Manuscript`. The intersection is encoded by means of the `some-satisfies` construction of XQuery, and the same can be said for the existential quantifier. By calling `ex:writer(doc('example.owl'))`, we would obtain:

```
<owl:Thing rdf:about="#Abiteboul"/>  <owl:Thing rdf:about="#Buneman"/>
<owl:Thing rdf:about="#Suciu"/>      <owl:Thing rdf:about="#Simeon"/>
```

Basically, the idea of the encoding is as follows. Each class C and property P of the ontology defines a function called C and P, respectively. In the running example we would have the functions: `ex:person`, `ex:man`, etc. in the case of classes, and `ex:author_of`, `ex:authored_by`, etc. in the case of properties. Now, the functions for classes are defined as the union of the elements of the **ABox** of the given class, and the elements of subclasses of the given class in the **TBox**. In the case of properties, the functions are defined as the union of: the elements of the **ABox** full-filling the property, and the elements of subproperties of the given property in the **TBox**. In the running example, *author_of* property is encoded as follows:

```
declare function ex:author_of($doc as node()) as node()*{
(sw:ofrole($doc,"author_of")) union
(for $y in sw:ofrole($doc,"writes")
return sw:toFiller(sw:About($y),"author_of",sw:Resource($y))) union
(for $z in sw:ofrole($doc,"authored_by")
return sw:toFiller(sw:Resource($z),"author_of",sw:About($z)))
};
```

Now, by calling `ex:author_of(doc('example.owl'))`, we would obtain:

```
<owl:Thing rdf:about="#Abiteboul">
<author_of rdf:resource="#DataontheWeb"/>
</owl:Thing>
<owl:Thing rdf:about="#Suciu">
<author_of rdf:resource="#DataontheWeb"/>
</owl:Thing>
<owl:Thing rdf:about="#Buneman">
<author_of rdf:resource="#XMLinScottland"/>
</owl:Thing>
```

```
<owl:Thing rdf:about="#Simeon">
<author_of rdfs:resource="#GrowingXQuery"/>
</owl:Thing>
<owl:Thing rdf:about="#Buneman">
<author_of rdfs:resource="#DataontheWeb"/>
</owl:Thing>
```

Let us remark that in both examples, the encoding has used the ontology for reasoning about class assertions and property fillers. In the first case *Abiteboul*, *Suciu*, *Buneman* and *Simeon* are members of the class *Writer* because they are obtained from the **ABox**, and from the **TBox** by means of the reasoning with the formulas $author\_of \equiv writes$ and $authored\_by^{-} \equiv author\_of$ (i.e. `sw:author_of` returns also elements related by means of *authored_by* and *writes* relationships). The elements of the classes *Paper* and *Book* are elements of *Manuscript* by $Paper \sqcup Book \sqsubseteq Manuscript$.

5 Querying an Ontology by Means of XQuery

Now, we will show some examples of querying and reasoning w.r.t. the running example. Such examples have been tested and can be downloaded from `http://indalog.ual.es/XQuerySemanticWeb`.

The first example retrieves the *"Authors of a Manuscript"*. Let us remark that the relation *"author_of"* is equivalent a *"writes"* and the inverse of *"authored_by"*. In addition, the class *"Manuscript"* includes the elements of the classes *"Paper"* and *"Book"*. The query can be expressed in our proposal as follows:

```
<manuscripts>{
for $x in ex:author_of(doc('example.owl')) return
<item> <author> { sw:subject($x) } </author>
<manuscript> { sw:object($x) } </manuscript> </item>
}</manuscripts>
```

obtaining the answer in XML format as follows:

```
<manuscripts>
   <item><author>#Abiteboul</author> <manuscript>#DataontheWeb</manuscript> </item>
   <item><author>#Suciu</author> <manuscript>#DataontheWeb</manuscript></item>
   <item> <author>#Buneman</author> <manuscript>#XMLinScottland</manuscript> </item>
   <item> <author>#Simeon</author> <manuscript>#GrowingXQuery</manuscript> </item>
   <item> <author>#Buneman</author> <manuscript>#DataontheWeb</manuscript> </item>
</manuscripts>
```

The second example retrieves the *"Authors of reviewed manuscripts"* in which the class *Reviewed* includes the manuscripts for which a *"reviewed_by"* relation exists. The query can be expressed in our framework as follows:

```
<manuscripts>{
for $y in ex:author_of(doc('example.owl'))
       where some $x in ex:reviewed(doc('example.owl')) satisfies
       sw:eq(sw:subject($x),sw:object($y))
return <item> <author> { sw:subject($y) } </author>
<manuscript> { sw:object($y) } </manuscript> </item>
}</manuscripts>
```

obtaining as answer:

```
<manuscripts>
   <item><author>#Abiteboul</author> <manuscript>#DataontheWeb</manuscript> </item>
   <item><author>#Suciu</author> <manuscript>#DataontheWeb</manuscript></item>
   <item> <author>#Simeon</author> <manuscript>#GrowingXQuery</manuscript> </item>
   <item> <author>#Buneman</author> <manuscript>#DataontheWeb</manuscript> </item>
</manuscripts>
```

Let us remark that *"XML in Scottland"* is not a reviewed manuscript.

6 Conclusions and Future Work

In this paper we have investigated how to use the XQuery language for querying and reasoning with RDF(S)/OWL-style ontologies. As future work we would like to study how to adapt our proposal to other ontology languages, in particular, how to extend to the recently proposed OWL QL, the W3C proposal for querying OWL ontologies. We have to solve some troubles to cover with OWL QL. Firstly, OWL QL is based on a different fragment of OWL. It is, for instance, restricted to existential quantification of properties, but on the other side it assumes cyclic ontologies. Moreover, it is focused on conjunctive queries. In order to handle conjunctive queries in OWL QL a rewriting query algorithm has been proposed in [12]. In summary, the adaptation of OWL QL to our framework is not a trivial task. Firstly, we have to implement a fix point operator in XQuery in order to cover with cyclic ontologies. In addition, OWL QL assumes the use of conjunctive queries and query rewriting. In this case, we need a formalism for expressing conjunctive queries in XQuery and a mechanism for rewriting. We are investigating how to use SWRL [17] and RIF [18] for expression conjunctive queries and the XQuery itself for query rewriting.

References

1. Akhtar, W., Kopecký, J., Krennwallner, T., Polleres, A.: XSPARQL: Traveling between the XML and RDF worlds – and avoiding the XSLT pilgrimage. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 432–447. Springer, Heidelberg (2008)
2. Almendros-Jiménez, J.M.: An RDF Query Language based on Logic Programming. Electronic Notes in Theoretical Computer Science 200(3) (2008)
3. Almendros-Jiménez, J.M.: An Encoding of XQuery in Prolog. In: Bellahsène, Z., Hunt, E., Rys, M., Unland, R. (eds.) XSym 2009. LNCS, vol. 5679, pp. 145–155. Springer, Heidelberg (2009)
4. Almendros-Jiménez, J.M., Becerra-Terón, A., Enciso-Baños, F.J.: Integrating XQuery and Logic Programming. In: Seipel, D., Hanus, M., Wolf, A. (eds.) INAP-WLP 2007. LNCS, vol. 5437, pp. 117–135. Springer, Heidelberg (2009)
5. Almendros-Jiménez, J.M., Becerra-Terón, A., Enciso-Baños, F.J.: Querying XML documents in logic programming. TPLP 8(3), 323–361 (2008)
6. Baader, F., Calvanese, D., McGuinness, D.L., Patel-Schneider, P., Nardi, D.: The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge Univ. Press, Cambridge (2003)

7. Bailey, J., Bry, F., Furche, T., Schaffert, S.: Web and Semantic Web Query Languages: A Survey. In: Eisinger, N., Małuszyński, J. (eds.) Reasoning Web. LNCS, vol. 3564, pp. 35–133. Springer, Heidelberg (2005)
8. Berglund, A., Boag, S., Chamberlin, D., Fernandez, M.F., Kay, M., Robie, J., Siméon, J.: XML path language (XPath) 2.0. In: W3C (2007)
9. Berners-Lee, T., Hendler, J., Lassila, O., et al.: The Semantic Web. Scientific american 284(5), 28–37 (2001)
10. Bikakis, N., Gioldasis, N., Tsinaraki, C., Christodoulakis, S.: Semantic based access over XML data. In: Lytras, M.D., Damiani, E., Carroll, J.M., Tennyson, R.D., Avison, D., Naeve, A., Dale, A., Lefrere, P., Tan, F., Sipior, J., Vossen, G. (eds.) WSKS 2009. LNCS, vol. 5736, pp. 259–267. Springer, Heidelberg (2009)
11. Boag, S., Chamberlin, D., Fernández, M.F., Florescu, D., Robie, J., Siméon, J., Stefanescu, M.: XQuery 1.0: An XML query language. In: W3C (2004)
12. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The DL-Lite family. Journal of automated reasoning 39(3), 385–429 (2007)
13. Droop, M., Flarer, M., Groppe, J., Groppe, S., Linnemann, V., Pinggera, J., Santner, F., Schier, M., Schöpf, F., Staffler, H., et al.: Embedding XPATH Queries into SPARQL Queries. In: Proc. of the 10th International Conference on Enterprise Information Systems (2008)
14. Eiter, T., Ianni, G., Krennwallner, T., Polleres, A.: Rules and ontologies for the semantic web. In: Baroglio, C., Bonatti, P.A., Małuszyński, J., Marchiori, M., Polleres, A., Schaffert, S. (eds.) Reasoning Web. LNCS, vol. 5224, pp. 1–53. Springer, Heidelberg (2008)
15. Groppe, S., Groppe, J., Linnemann, V., Kukulenz, D., Hoeller, N., Reinke, C.: Embedding SPARQL into XQUERY/XSLT. In: Proceedings of the 2008 ACM symposium on Applied computing, SAC 2008, pp. 2271–2278. ACM, New York (2008)
16. Haarslev, V., Möller, R., Wandelt, S.: The revival of structural subsumption in tableau-based description logic reasoners. In: Proceedings of the 2008 International Workshop on Description Logics (DL 2008), CEUR-WS, pp. 701–706 (2008)
17. Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B., Dean, M.: SWRL: A Semantic Web Rule Language Combining OWL and RuleML. W3C Member Submission (May 21, 2004), `http://www.w3.org/Submission/SWRL/`
18. Kifer, M.: Rule interchange format: The framework. In: Calvanese, D., Lausen, G. (eds.) RR 2008. LNCS, vol. 5341, pp. 1–11. Springer, Heidelberg (2008)
19. Klyne, G., Carroll, J.J.: Resource Description Framework (RDF): Concepts and Abstract Syntax. Technical report (2004), `http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/`
20. Motik, B., Patel-Schneider, P.F., Parsia, B., Bock, C., Fokoue, A., Haase, P., Hoekstra, R., Horrocks, I., Ruttenberg, A., Sattler, U., et al.: OWL 2 web ontology language: Structural specification and functional-style syntax. In: W3C (2008)
21. PrudHommeaux, E., Seaborne, A., et al.: SPARQL query language for RDF. In: W3C (2008)
22. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. Web Semantics: Science, Services and Agents on the World Wide Web 5(2), 51–53 (2007)
23. Tsarkov, D., Horrocks, I.: FaCT++ Description Logic Reasoner: System Description. In: Furbach, U., Shankar, N. (eds.) IJCAR 2006. LNCS (LNAI), vol. 4130, pp. 292–297. Springer, Heidelberg (2006)