SPARQL2XQuery 2.0: Supporting Semantic-based Queries over XML Data

Ioannis Stavrakantonakis[#], Chrisa Tsinaraki[+], Nikos Bikakis[‡], Nektarios Gioldasis[#], Stavros Christodoulakis[#]

[#] *MUSIC/TUC, Technical University of Crete, Greece*

[+] *Department of Information Engineering and Computer Science, University of Trento, Italy*

[‡] *Knowledge and Database Systems Laboratory, National Technical University of Athens, Greece*

gstavrak@ced.tuc.gr, chrisa@disi.unitn.it, bikakis@dblab.ntua.gr, nektarios@ced.tuc.gr, stavros@ced.tuc.gr

Abstract

The dominant standards in multimedia content and service description, namely the MPEG-7 and the MPEG-21, have been expressed in XML Schema syntax. In addition, ontologies that capture the semantics of these standards have been developed using the semantic web languages. Since different communities in the industry and the academia work and are familiar with the Semantic Web environment or the XML environment, a Schema mapping framework and a Query mapping framework are needed that will allow querying multimedia content and service descriptions in a uniform way in both the XML and Semantic Web environments. We present in this paper SPARQL2XQuery 2.0, a framework that allows expressing semantic queries on top of XML data through the translation of SPARQL queries in XQuery syntax. SPARQL2XQuery 2.0 may work with both existing ontologies and with automatically produced ones, formed according to our XS2OWL 2.0 transformation model. XS2OWL 2.0 exploits the OWL 2.0 semantics and supports the new XML constructs introduced by XML Schema 1.1.

1. Introduction

The flexibility and the advanced structure-description capabilities of the XML Schema language [2] have made it a de facto standard in metadata description. This is the case in several application domains like, for example, the multimedia domain, where both the MPEG-7 [20] and MPEG-21 [21] standards (used, respectively, for multimedia content and service description) have been expressed using XML Schema syntax.

The development of the Semantic Web, on the other hand, and the advanced semantic processing capabilities offered by the Semantic Web languages have led to the development of ontologies [23][22][24][7] capturing the semantics of the standards. This way, the multimedia descriptions are expressed using Semantic Web language syntax and may be enriched through inferencing. These descriptions are then stored in RDF repositories, accessed using the SPARQL query language. There are however communities both in the academia and the industry that have based their work on XML Schema. These groups work with XML descriptions, stored in XML repositories accessed through the XQuery language.

A similar situation exists in the cultural heritage domain: There are several standards expressed in XML Schema syntax, like the TEI [26], the EAD [27] and several others, which are used by the cultural heritage institutions (libraries, archives, museums, etc.). On the other hand, the CIDOC/CRM standard [25] has been developed, which essentially is an ontology, expressed in OWL/RDF syntax, which subsumes the semantics of these standards.

The above discussion shows that in both the multimedia and the cultural heritage domains a Schema mapping framework and a Query mapping framework are needed that will allow querying content and service descriptions in a uniform way in both the XML and Semantic Web environments.

In this paper, we present the SPARQL2XQuery 2.0 framework, which allows SPARQL queries to be answered over XML data using the XQuery query language. To accomplish this, mappings between ontologies and XML Schemas are defined that allow our framework to translate SPARQL queries in XQuery syntax. SPARQL-2XQuery 2.0 may work with both existing ontologies and with automatically produced ones, formed according to our XS2OWL 2.0 transformation model. The SPARQL-2XQuery 2.0 framework extends our previous work in the SPARQL2XQuery 1.0 framework [19] so as to exploit the OWL 2.0 [1] semantics and the new constructs introduced by XML Schema 1.1 [2].

The rest of the paper is structured as follows: The related work is presented in Section 2, an overview of the SPARQL2XQuery 2.0 framework is provided in Section 3, the XS2OWL 2.0 transformation model is presented in Section 4, the SPARQL-to-XQuery translation is outlined in Section 5, the application of our framework in the multimedia and cultural heritage domains is discussed in Sec-

tion 6 and the paper concludes in Section 7, where our future research directions are also outlined.

2. Related Work

Several approaches have been proposed in the literature, trying to bridge the gap between the XML and the Semantic Web environments.

Transforming XML Schema to OWL and through this, XML data to RDF, is a field that has been extensively investigated recently [3][4][5][6][7][8].

Another important research issue is that of mapping existing ontologies which are described with XML Schemas to semantic language representations, in order to transform XML data to RDF data based on manual mappings [9][10].

Recently, a combination of Semantic Web (SPARQL) and XML (XQuery, XPath, XSLT) technologies [15][16][17] has been exploited in order to transform XML data to RDF and vice versa.

Compared to SPARQL2XQuery, the above approaches focus on data transformation, they do not provide a solution for integrating and querying the existing XML data from the Semantic Web environment and they do not deal with the problem of "relating" the existing XML data with the Semantic Web data.

Moreover, in [15][16][17], which are closer to our approach, the user has to (a) interface with more than one data models and query languages; (b) be aware of the syntax and the semantics of each of these approaches in order to express his queries, since every approach has adopted its own syntax and semantics by modifying and merging the standard technologies; and (c) be aware of the underlying XML Schema in order to create his retrieval query accordingly (using XQuery or XSLT).

In our work, the user is not expected to know the underlying XML Schema or even the existence of XML data; he expresses his query only in standard SPARQL, in terms of the ontology that he is aware of, and he is able to transparently retrieve the XML data in his favored format.

In some older approaches, mappings between XML Schemas (or DTDs) and ontologies are established in order to support data integration [11][12][13][14]. These approaches do not support the standard technologies (like XML Schema, OWL, RDF, SPARQL, etc.).

Finally, compared to our previous work in the SPARQL2XQuery 1.0 framework [19], the SPARQL2XQuery 2.0 framework extends it in order to exploit the OWL 2.0 semantics and the new constructs introduced by XML Schema 1.1. In addition, it utilizes the XS2OWL 2.0 transformation model for the automatic transformation of XML Schemas in OWL syntax, which also exploits the latest features of the standards that allow the automatic specification of accurate mappings between the XML Schema and the OWL syntax. In particular, the XML Schema identity constraints can now be accurately represented in OWL 2.0 syntax (which was not possible in OWL 1.0 syntax), thus overcoming the most important limitation of the XS2OWL 1.0 framework [3][4] and, consequently, of the SPARQL2XQuery 1.0 framework.

3. SPARQL2XQuery 2.0 Overview

In this section we present an overview of the SPARQL2XQuery 2.0 framework. The framework architecture is shown in Figure 1.

The SPARQL2XQuery 2.0 framework supports two scenarios:

1. **Querying XML data based on an automatically generated OWL ontology.** In this scenario, the following actions take place:
   a) Using the XS2OWL 2.0 framework, the XML Schema according to which the XML data are structured, is automatically expressed in OWL 2.0 syntax.
   b) The SPARQL2XQuery framework takes as input the XML Schema and the ontology generated by XS2OWL 2.0 and automatically generates and maintains the mappings between the ontology and the XML Schema.
   c) The SPARQL queries posed over time by the users that see the generated ontology are translated to XQuery expressions.
   d) The query results are transformed into the desired format (SPARQL Query Result XML Format or RDF) and returned to the users.
2. **Querying XML data based on an existing OWL ontology.** In this scenario, the following actions take place:
   a) An existing OWL ontology is manually mapped by a domain expert to the XML Schema.
   b) The SPARQL queries posed over the ontology are translated to XQuery expressions.
   c) The query results are transformed into the desired format (SPARQL Query Result XML Format or RDF).

In both scenarios, the Semantic Web users and the applications that pose SPARQL queries over the ontology are not expected to know the underlying XML Schemas or even the existence of XML data. They express their queries only in standard SPARQL, in terms of the ontology that they are aware of, and they are able to retrieve the underlying data in their favored format.

The SPARQL2XQuery 2.0 has been implemented as an extension of the SPARQL2XQuery 1.0 framework, using Java related technologies (Java 2SE, Axis2 and Jena) and the Oracle Berkeley DB XML database. The XS2OWL 2.0 transformation model has been implemented as an extension of the XS2OWL 1.0 framework, using XSLT.

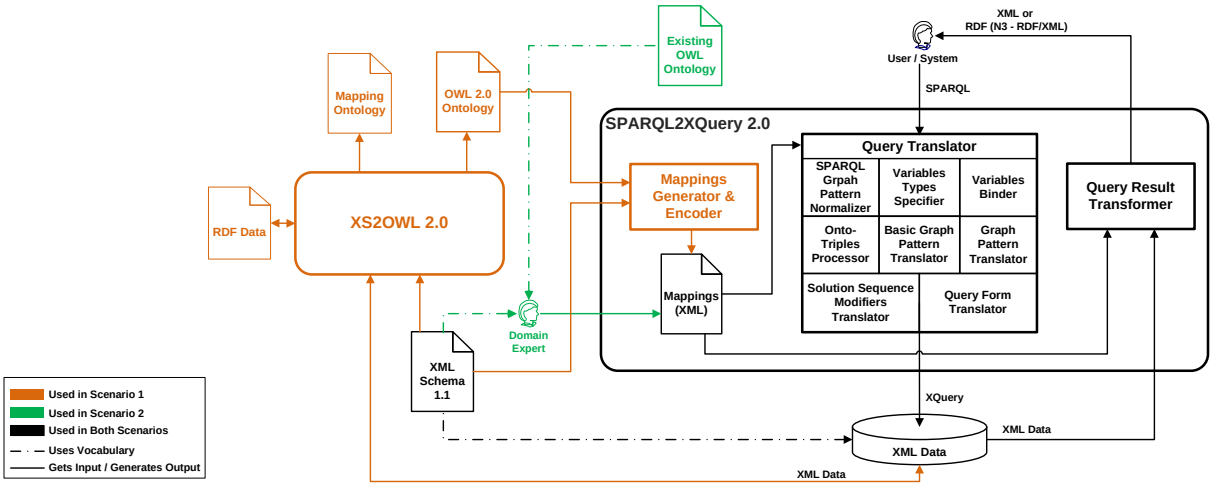

Figure 1. The SPARQL2XQuery 2.0 Framework, which allows the evaluation of SPARQL Queries over XML Data. If the queries are posed on top of existing ontologies, mappings between the ontologies and the underlying XML Schema(s) should be manually specified. These mappings play a significant role in the SPARQL query translation in XQuery syntax. If there is not used an existing ontology, the XS2OWL 2.0 Transformation Model is applied on the XML Schema syntax, expresses it in OWL 2.0 syntax and the mappings are automatically generated.

4. The XS2OWL 2.0 Transformation Model

This section describes the XS2OWL 2.0 transformation model, which is the basis for the representation of XML Schemas in OWL syntax. The XS2OWL 2.0-based transformation process generates two ontologies: (a) A *main ontology* that represents the XML Schema constructs using OWL constructs and (b) A *mapping ontology* that associates the names of the XML Schema constructs with the IDs of the equivalent main ontology constructs and captures any information present in the XML Schema that cannot be captured in the main ontology due to the expressivity limitations of the OWL 2.0 syntax. The mapping ontology keeps information that is not usable by the Semantic Web tools, but can be of use in other applications like, for example, the transformation of RDF data structured according to the main ontology in XML syntax compliant with the original XML Schemas.

XS2OWL 2.0 is an extension and update of our previous work with the XS2OWL 1.0 framework. XS2OWL 2.0 exploits the OWL 2.0 semantics (the OWL 2.0 RL profile is used), in order to achieve a more accurate representation of the XML Schema constructs in the main ontology, and supports the new XML constructs introduced by XML Schema 1.1. In particular, XS2OWL 2.0 allows, in addition to the XS2OWL 1.0 support, the representation of: (a) The XML simple datatypes (see Subsection 4.1 for details); (b) The XML Schema identity constraints – i.e. key, keyref and unique – (see Subsection 4.2 for details); and (c) The XML constructs introduced by the XML Schema 1.1 – i.e. Error, Substitution Group 1.1, Alternative, Assert and Override – (see subsection 4.3 for details). A detailed comparison of XS2OWL 1.0 and XS2OWL 2.0 is presented in Table 1 and an overview of the XS2OWL 2.0 transformation model is provided in Table 2.

Table 1. XS2OWL 1.0 – XS2OWL 2.0 Comparison
(Legend: ✓ supported ✖ not supported ❖ mapping ontology only)

	XML Construct	XS2OWL 1.0	XS2OWL 2.0
XML Schema 1.0	Complex Type	✓	✓
	Attribute	✓	✓
	Element	✓	✓
	Attribute	✓	✓
	Annotation	✓	✓
	Sequence	✓	✓
	Choice	✓	✓
	Substitution Group	✓	✓
	Extension	✓	✓
	(Nested) Simple Type	❖	✓
	Key	✖	✓
	Keyref	✖	✓
	Unique	✖	✓
	Redefine	✖	❖
XML Schema 1.1	Error	✖	✓
	Substitution Group 1.1	✖	✓
	Alternative	✖	❖
	Assert	✖	❖
	Override	✖	❖

Table 2. The XS2OWL 2.0 Transformation Model

XML Schema Construct	OWL 2.0 Construct
Complex Type	Class
Simple Datatype	Datatype Definition
Element	(Datatype or Object) Property
Attribute	Datatype Property
Sequence	Unnamed Class – Intersection
Choice	Unnamed Class – Union
Annotation	Comment
Extension, Restriction	subClassOf axiom
Unique (Identity Constraint)	HasKey axiom
Key (Identity Constraint)	HasKey axiom – ExactCardinality axiom
Keyref (Identity Constraint)	HasKey axiom – ExactCardinality axiom
Substitution Group	SubPropertyOf axioms
Alternative	On Mapping Ontology
Assert	On Mapping Ontology
Override, Redefine	On Mapping Ontology
Error	Datatype

4.1. Simple Type Representation

We describe here how XS2OWL 2.0 handles simple type definitions. Simple types can be either built-in XML Schema types, such as xsd:string, or user-defined simple types. Since OWL 2.0 introduced the *DatatypeDefintion* axiom for datatype definition, this axiom is used for the representation of the user-defined simple types in the main ontology generated according to XS2OWL 2.0. In the following paragraphs we describe how the simple types defined using the different XML Schema constructs (restriction, union and list) and the unnamed simple types are handled by XS2OWL 2.0.

Restriction. The XML Schema restricted simple types are formed from the restriction of an existing (built-in or user-defined) simple type, the *base type*. The base type is specified in the *base* attribute of the *restriction* element or in the *simpleType* element (a sub-element of *restriction*) in case of an unnamed *base* element. The restricted simple types are represented in the main ontology by a *DatatypeDefintion* axiom (implemented in the OWL 2.0 RDF syntax using *rdfs:Datatype*) that contains an *EquivalentClass* axiom and in the mapping ontology by a *SimpleTypeInfoType* individual. The major difference with XS2OWL 1.0 is that, since OWL 1.0 only allowed the declaration of datatypes defined in XML Schemas, in XS2OWL 1.0 we had only datatype declarations of externally defined types while in XS2OWL 2.0 we have datatype definitions. An example of a restricted simple XML Schema type is the *"ValidAgeType"*, shown in Figure 2. The representation of *"ValidAgeType"* after the application of the XS2OWL 2.0 transformation model in the automatically generated main and mapping ontologies are shown, respectively, in Figure 3 and Figure 4.

```
<xs:simpleType name="ValidAgeType">
 <xs:restriction base="xs:float">
  <xs:minInclusive value="0.0"/>
  <xs:maxInclusive value="150.0"/>
 </xs:restriction>
</xs:simpleType>
```

Figure 2. Restricted Simple Type Example

```
<rdfs:Datatype rdf:ID="ValidAgeType">
<owl:equivalentClass>
 <rdfs:Datatype>
  <owl:onDatatype rdf:resource="&xsd;#float"/>
  <owl:withRestrictions rdf:parseType="Collection">
   <rdf:Description>
    <xsd:maxInclusive rdf:datatype="&xsd;#float"> 150.0
    </xsd:maxInclusive>
   </rdf:Description>
   <rdf:Description>
    <xsd:minInclusive rdf:datatype="&xsd;#float"> 0.0
    </xsd:minInclusive>
   </rdf:Description>
  </owl:withRestrictions>
 </rdfs:Datatype>
</owl:equivalentClass>
</rdfs:Datatype>
```

Figure 3. Representation of the Restricted Simple Type of Figure 2 in the automatically generated Main Ontology

```
<ox:SimpleTypeInfoType rdf:ID="ValidAgeType_si">
 <ox:classID>validAgeType</ox:classID>
 <ox:typeID>validAgeType</ox:typeID>
 <ox:definitionType>restriction</ox:definitionType>
</ox:SimpleTypeInfoType>
```

Figure 4. Representation of the Restricted Simple Type of Figure 2 in the automatically generated Mapping Ontology

Union. The XML Schema union simple types are formed from the union of existing types. Union members are specified in the *memberTypes* attribute of the *union* element or in the *simpleType* sub-elements of *union* (in case of unnamed union members). The union simple types are represented in the main ontology by a *DatatypeDefintion* axiom that contains a *unionOf* axiom and in the mapping ontology by a *SimpleTypeInfoType* individual.

List. The XML Schema list simple types are comprised of a list of values of a specific datatype. The type of the list members is specified in the *itemType* attribute of the *list* element or in the *simpleType* sub-elements of *list* (in case of unnamed list members). The list simple types are represented in the main ontology by a *Datatype* axiom that contains an *EquivalentClass* axiom and in the mapping ontology by a *SimpleTypeInfoType* individual.

Unnamed Simple Types. An unnamed XML Schema simple type is a simple type the definition of which is nested in the declaration of an XML construct (element, group, attribute group, attribute, simple type and alternative). It may be defined using any of the restriction, union and list XML Schema constructs and is valid in the scope of the XML construct it is nested in. The unnamed XML

Schema simple types are represented in the same way with the named ones, but the IDs of the constructs representing them in the main ontology and the mapping ontology are automatically decided, following a set of naming convention rules, in order to be unique. Since the unnamed simple types may be nested in other unnamed simple types, a recursive algorithm is used for the generation of the constructs that represent them in both the main and the mapping ontology as was done in the XS2OWL 1.0. The major difference is that in XS2OWL 1.0 in the main ontology were stored only declarations of externally defined simple types, while in XS2OWL 2.0 the datatype definitions are stored in the main ontology.

4.2. Identity Constraint Representation

Two types of identity constraints are supported by XML Schema: (a) The constraints imposed by attributes of type *ID*, *IDREF* and *IDREFS*; and (b) The constraints imposed by the *unique*, *key* and *keyref* elements. The former have been treated by XS2OWL 1.0 and, since they are attributes of built-in XML Schema types, they are represented, in both XS2OWL 1.0 and 2.0, in the same way with the other attributes. The second type of identity constraints could not be accurately represented using OWL 1.0 constructs and were not taken into account in XS2OWL 1.0. In XS2OWL 2.0 they are represented using the *HasKey* axiom of OWL 2.0.

The identity constraints of the second type contain: (a) A *selector* element, which specifies the XML Schema elements on which the identity constraint is applied; and (b) One or more *field* elements, where the XML Schema constructs (elements or attributes) that form the constraint value are specified. Both the *selector* and *field* elements specify the construct(s) they refer to in their *xpath* attribute. The *xpath* attribute uses a set of XPath expressions, which should be evaluated over the XML Schema in order to locate the XML constructs they refer to.

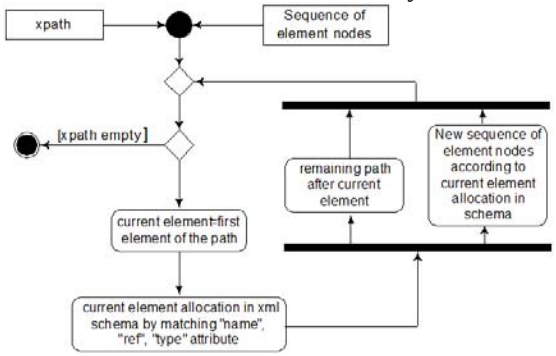

Figure 5. The "XPathEvaluator" Algorithm, which evaluates the XPath expressions

Since the XPath expressions do not refer to the node hierarchy of the XML Schema but in the node structure of the XML data following it, the "XPathEvaluator" algorithm (see the activity diagram of Figure 5) has been developed for XPath expression evaluation. The XPath expression evaluation returns a set of XML Schema constructs, which are represented as (object or datatype) properties in the main ontology. Then, depending on the constraint type, the constraints themselves are expressed in OWL 2.0 syntax, as is explained in the following paragraphs.

Unique. In a *unique* identity constraint *U(S, F)* the *selector S* represents the XML Schema element that has a unique combination of values of the constructs specified in the *field* element(s) *F* of *U*.

For the representation of *U* in the main ontology, the following actions are performed: (a) An OWL class C_F is defined, which has a set ***P*** of properties that represent the XML constructs specified in *F*; (b) An object property P_{CF} is defined, having as domain the class C_S, which represents the type of *S* in the main ontology, and C_F as range; and (c) A *HasKey* axiom is defined from the class C_S on the P_{CF} property.

U is represented in the mapping ontology by an *IdentityConstraintInfoType* individual. An example of a *unique* identity constraint is shown in Figure 6, and its representations in the main ontology and the mapping ontology are shown, respectively, in Figure 7 and Figure 8.

```
<xs:element name="Persons" type="PersonsType">
 <xs:unique name="NameAddrUnique">
  <xs:selector xpath="Person"/>
  <xs:field xpath="Name"/>
  <xs:field xpath="Address/@city"/>
 </xs:unique>
</xs:element>
```

Figure 6. Unique Identity Constraint Example

```
<owl:Class rdf:ID="Persons_NameAddrUnique_PersonsType">
 <owl:hasKey rdf:parseType="Collection">
  <rdf:Description rdf:about="FirstName_xs_string"/>
  <rdf:Description rdf:about="city_addressGroup__xs_string"/>
 </owl:hasKey>
</owl:Class>
<owl:ObjectProperty rdf:ID="Persons__PersonsType">
 <rdfs:domain
       rdf:resource="#Persons_NameAddrUnique_PersonsType"/>
 <rdfs:range rdf:resource="#PersonsType"/>
 <rdfs:label>Persons</rdfs:label>
</owl:ObjectProperty>
```

Figure 7. Representation of the Unique Identity Constraint of Figure 6 in the Main Ontology

```
<ox:IdentityConstraintInfoType
rdf:ID="Persons_NameAddrUnique_PersonsType_ui">
 <ox:restrictionID>
  Persons_NameAddrUnique_PersonsType
 </ox:restrictionID>
 <ox:selectorPath>Person</ox:selectorPath>
 <ox:fieldPath>Name/FirstName</ox:fieldPath>
 <ox:fieldPath>Address/@city</ox:fieldPath>
 <ox:constraintType>unique</ox:constraintType>
</ox:IdentityConstraintInfoType>
```

Figure 8. Representation of the Unique Identity Constraint of Figure 6 in the Mapping Ontology

Key. The *key* identity constraint has the same semantics with the *unique* with the additional requirement that the XML Schema constructs of the *field* are mandatory. Thus, the representation of the *key* identity constraint is almost the same with the one of *unique*. The only difference is that the additional requirement is satisfied with the definition of *ObjectExactCardinality* and *DataExactCardinality* axioms on the object properties and the datatype properties of C_F respectively.

Keyref. The semantics of the *keyref* identity constraint are the same with the semantics of the *key* identity constraint; The only difference is that *keyref* refers to an existing *key* definition. Thus, the representation of the *keyref* identity constraint is the same with that of the *key* identity constraint in both the main ontology and the mapping ontology.

4.3. Representation of the XML Schema 1.1 Constructs

We present in this subsection the representation of the constructs introduced or substantially modified in XML Schema 1.1. In particular, in the following paragraphs will be discussed the representation of the modified substitution group and of the now introduced alternative, override, assert and error constructs.

Substitution Group. The syntax and the semantics of the substitution group construct, which allows in XML Schema 1.0 the use of a specific element structure using a specific name, have been modified in XML Schema 1.1. In particular, the same name may be used for several element structures in XML Schema 1.1.

Since the elements are represented in the main ontology by (object or datatype) properties, in XS2OWL 1.0 we have represented every substitution group *sg* using the *subPropertyOf* axiom on the property *p*, which represents the element that substitutes *sg*. In XS2OWL 2.0 we tokenize the list of the element names that appear in the *substitutionGroup* attribute of *sg* and then we individually cope with each of them as we did in XS2OWL 1.0.

Alternative, Override and Assert. Since the syntax of OWL 2.0 does not support the definition of structures with semantics equivalent or similar to the semantics of alternative, override and assert, these constructs are not represented, according to XS2OWL 2.0, in the main ontology but only in the mapping ontology by instances of the classes *AlternativeInfoType*, *OverrideInfoType* and *AssertInfoType* respectively.

Error. The XML Schema 1.1 built-in datatype *error* is used in conditional type assignment, in order to inform the XML Schema validator when an error message should be issued. The XS2OWL 2.0 transformation model treats *error* in the same way with any other built-in datatype.

5. SPARQL to XQuery Translation

In this section, we provide an overview of the SPARQL-to-XQuery translation supported by the SPARQL2XQuery 2.0 framework.

The *SPARQL2XQuery 2.0 Query Translator* component comprises of the following sub-components:

- The *SPARQL Graph Pattern Normalizer*, which rewrites the Graph-Pattern (*GP*) of the input SPARQL query in an equivalent normal form, based on equivalence rules. This makes the *GP* translation process simpler and more efficient.
- The *Variable Type Specifier*, which identifies the types of the variables in order to detect any conflict arising from the syntax provided by the user as well as to identify the form of the results for each variable. Moreover, the variable types are used by the *Onto-triples Processor* and the *Variable Binder* sub-components.
- The *Onto-Triples Processor*, which processes onto-triples (actually referring to the ontology structure and/or semantics) against the ontology and, based on this analysis, binds the correct XPaths to variables contained in the onto-triples. These bindings are going to be used in the next steps as input to the *Variable Binder* sub-component.
- The *Variable Binder*, which is used in the translation process for the assignment of the correct XPaths to the variables referenced in a given Basic Graph Pattern (*BGP*, a sequence of triple patterns and filters), thus enabling the translation of *BGPs* to XQuery expressions.
- The *Basic Graph Pattern Translator*, which performs the translation of a *BGP* into semantically equivalent XQuery expressions, thus allowing the evaluation of a *BGP* on a set of XML data. The translation is based on the *BGP2XQyuery* algorithm, which takes as input the mappings between the ontology and the XML schema, the *BGP*, the determined variable types and the variable bindings and generates XQuery expressions.
- The *Graph Pattern Translator*, which translates a *GP* into semantically equivalent XQuery expressions. The concept of a *GP* is defined recursively. The *Basic Graph Pattern Translator* sub-component translates the basic components of a *GP* (i.e. *BGPs*) into semantically equivalent XQuery expressions, which however have to be properly associated in the context of a *GP*. This means to apply the SPARQL operators (i.e. AND, OPT, UNION and FILTER) among them using XQuery expressions and functions.
- The *Solution Sequence Modifiers Translator*, which translates the SPARQL solution sequence modifiers using XQuery clauses (*Order By*, *For*, *Let*, etc.) and XQuery built-in functions. Solution Modifiers are applied on a solution sequence in order to create another, user desired, sequence. The modifiers supported by

SPARQL are *Distinct*, *Reduced*, *OrderBy*, *Limit*, and *Offset*.

- The *Query Forms Translator*, which is responsible for the final step of the SPARQL query translation in XQuery expressions. SPARQL has four forms of queries (*Select*, *Ask*, *Construct* and *Describe*). According to the query form, the structure of the final result is different. In particular, after the translation of any solution modifier is done, the generated XQuery is enhanced with appropriate expressions in order to achieve the desired result structure (e.g. to construct an RDF graph, or a result set) according to the query form.

We have extended the SPARQL-to-XQuery translation presented in [20] for the SPARQL2XQuery 1.0 framework, in order to exploit the OWL 2.0 semantics and the new constructs introduced by XML Schema 1.1. In particular, the SPARQL-to-XQuery translation has been extended in order to support the XML Schema datatypes and the XML Schema Identity constraints.

5.1. XML Schema datatypes

The SPARQL query language supports queries, where datatype references could be exploited in order to define the type of a literal. For example, consider the literal "42". Using the syntax "42"^^xsd:integer, "42" is stated to be an integer, while with the syntax "42"^^xsd:string, "42" is stated to be a string and with the syntax "42"^^ns:ValidAgeType "42" is stated to be of a user-defined type (ValidAgeType). According to the SPARQL specification, literals and datatype references could appear in the object part of a *Triple Pattern* or in a *Filter Expression* of an SPARQL query.

Using the XS2OWL 2.0 transformation model, the XML Schema simple datatypes are represented using the OWL 2.0 semantics as presented in Section 4. Exploiting the information of the generated mappings between the ontology produced by XS2OWL 2.0 and the initial XML Schema, the SPARQL2XQuery framework can handle queries that include datatype references in their literals.

5.2. XML Identity constraints

Since OWL 2.0 allows the representation of the XML Schema identity constraints and the XS2OWL 2.0 transformation model supports their representation, the SPARQL-to-XQuery translation of the SPARQL2XQuery 1.0 framework has been extended in SPARQL2XQuery 2.0 to exploit the XML Schema identity constraints during the translation. The identity constraints can be exploited in queries which contain the same variables between more than one *Triple Patterns* in a SPARQL query *Graph Pattern*. The SPARQL2XQuery 2.0 framework can handle this class of SPARQL queries, since it exploits the identity constraint information and the mappings between the generated ontology and the XML Schema.

6. Application in the Multimedia and Cultural Heritage Domains

We demonstrate in this section how our framework can be used in real-world applications of the multimedia and the cultural heritage domains.

Multimedia Domain. As already mentioned, the dominant standards for content and service description (MPEG-7 and MPEG-21 respectively) in the multimedia domain have been expressed in XML Schema syntax. As a consequence, several groups have been working with these standards and a great number of MPEG-7 and MPEG 21 descriptions have been created. The development of the Semantic Web, though, has made many research groups to adopt the Semantic Web technologies, develop ontologies that capture (fully or partially) the semantics of the standards and work with OWL/RDF descriptions formed according to the ontologies. Since there exist standard-based XML descriptions as well as groups working with the XML Schema based syntax of the standards, the capability of transparently posing queries on both the RDF and the XML Schema repositories is necessary. This can be achieved using the SPARQL2XQuery 2.0 framework in two different usage scenarios:

(a) An existing ontology like [22], which captures the semantics of the standard(s), is used and mappings between the ontology and the XML Schemas are manually defined. Then, the end-users pose their SPARQL queries over the ontology and the SPARQL2XQuery 2.0 framework expresses the queries in XQuery syntax, evaluates them and returns the query results.

(b) The XS2OWL 2.0 framework is used to automatically express the semantics of the standards in OWL 2.0 syntax. Then, the SPARQL2XQuery 2.0 framework automatically specifies the mappings between the generated ontology and the XML Schema(s) and may support user queries expressed in SPARQL syntax over the generated ontology in the same way it supports the queries of scenario (a).

Cultural Heritage Domain. There are several standards (over 100) for content description in the cultural heritage domain expressed in XML Schema syntax, like the TEI, the EAD and several others. On the other hand, the CIDOC/CRM standard has been developed, which essentially is an ontology, expressed in OWL/RDF syntax, that subsumes the semantics of the above-referred standards. Since the cultural heritage institutions have invested a great amount of time in the specification of descriptions that are formed according to the XML-based standards and they may have even developed software that manages them, the SPARQL2XQuery 2.0 framework can be used in order to make their contents accessible to users aware of the CIDOC/CRM without having to

change their working environment. In particular, mappings between the CIDOC/CRM ontology and the XML Schemas of the standards should be manually defined. Then, the end-users may pose their SPARQL queries over the CIDOC/CRM ontology and the SPARQL2XQuery 2.0 framework expresses the queries in XQuery syntax, evaluates them and returns the query results.

7. Conclusions and Future Work

We have presented the SPARQL2XQuery 2.0 framework that we have developed [28], which allows SPARQL queries to be answered over XML data using the XQuery query language. To accomplish this, mappings between ontologies and XML Schemas are defined that allow our framework to translate SPARQL queries in XQuery syntax. SPARQL2XQuery 2.0 may work with both existing ontologies and with automatically produced ones, formed according to our XS2OWL 2.0 transformation model. XS2OWL 2.0 exploits the OWL 2.0 semantics and supports the new XML constructs introduced by XML Schema 1.1.

The functionality offered by SPARQL2XQuery 2.0 is important, among other application domains, for the multimedia domain, since it allows multimedia applications of the Semantic Web and the XML environments to interoperate.

The SPARQL2XQuery 2.0 framework is going to be integrated in an ontology-based mediator [18] [19] framework that we are developing now and is going to provide semantic interoperability and integration between distributed heterogeneous sources using the standard Semantic Web and XML technologies.

8. References

[1] Motik B., Schneider P.F.P., Parsia B. (eds.): "OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax", W3C Recommendation, 27 Oct. 2009, http://www.w3.org/TR/owl2-syntax/.

[2] Gao S., Sperberg-McQueen C. M., Thompson H.S. (eds.) "W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures", W3C Working Draft, 3 Dec. 2009, http://www.w3.org/TR/xmlschema11-1/

[3] Tsinaraki C., Christodoulakis S., "Interoperability of XML Schema Applications with OWL Domain Knowledge and Semantic Web Tools". In Proc. of ODBASE 2007.

[4] Tsinaraki C., Christodoulakis S., "Support for Interoperability between OWL based and XML Schema based Applications". In the Proc. of DELOS Conference II, 2007.

[5] Ferdinand M., Zirpins C., Trastour D.: "Lifting XML Schema to OWL". In Proc. of ICWE 2004

[6] Bohring H., Auer S.: "Mapping XML to OWL Ontologies" Leipziger Informatik-Tage 2005: 147-156

[7] Garcia R., Celma O.: "Semantic integration and retrieval of multimedia metadata". In Proc. of SemAnnot 2005

[8] Bedini I., Gardarin G., Nguyen B. "Deriving Ontologies from XML Schema". In Proc. of EDA 2008 Vol. B-4, 3-17.

[9] Rodrigues T., Rosa P., Cardoso J.: "Mapping XML to Exiting OWL ontologies", International Conference WWW/Internet 2006

[10] Cruz C., Nicolle C.: "Ontology Enrichment and Automatic Population from XML Data", In Proc. of ODBIS 2008.

[11] Cruz I., Huiyong X., Hsu F.: "An Ontology-Based Framework for XML Semantic Integration". In Proc of IDEAS 2004

[12] Lehti P., Fankhauser P. "XML data integration with OWL: Experiences and challenges". In Proc. of SAINT 2004

[13] Bernd A., Beeri C., Fundulaki I., Scholl M.: "Ontology-Based Integration of XML Web Resources". In Proc. of ISWC 2002

[14] Christophides V., Karvounarakis G., et.al. "The ICS-FORTH Semantic Web Integration Middleware (SWIM)". IEEE Data Eng. Bull. 26(4):11-18 (2003)

[15] Groppe S., Groppe J., Linnemann V., Kukulenz D., Hoeller N., Reinke C., "Embedding SPARQL into XQuery/XSLT". In Proc. of ACM SAC 2008

[16] Droop M., Flarer M., et.al. "Embedding XPATH Queries into SPARQL Queries". In Proc. of ICEIS 2008

[17] Akhtar W., Kopecký J., et.al. "XSPARQL: Traveling between the XML and RDF Worlds and Avoiding the XSLT Pilgrimage". In Proc. of ESWC 2008

[18] Makris K., Bikakis N., Gioldasis N., Tsinaraki C., Christodoulakis S.: "Towards a Mediator based on OWL and SPARQL". In Proc. of WSKS 2009

[19] Makris K., Gioldasis N., Bikakis N., Christodoulakis S.: "Ontology Mapping and SPARQL Rewriting for Querying Federated RDF Data Sources". In Proc. of ODBASE 2010.

[20] Bikakis N., Gioldasis N., Tsinaraki C., Christodoulakis S.: "Querying XML Data with SPARQL". In Proc. of DEXA 2009.

[21] Chang, S.F., Sikora, T., Puri, A.: Overview of the MPEG-7 standard. In IEEE Transactions on Circuits and Systems for Video Technology 11:688–695, 2001.

[22] Pereira, F.: The MPEG-21 standard: Why an open multimedia framework?. In the Proc. of the 8th IDMS, 2001

[23] Tsinaraki C., Polydoros P., Christodoulakis S.. Interoperability support between MPEG-7/21 and OWL in DS-MIRF. In Transactions on Knowledge and Data Engineering (TKDE), Special Issue on the Semantic Web Era, 19(2):219–232, 2007.

[24] Hunter J.: Adding Multimedia to the Semantic Web - Building an MPEG-7 Ontology. In Proc. of ISWC 2001.

[25] Troncy R., Bailer W., Hausenblas M., Hofmair P., Schlatte R.: Enabling Multimedia Metadata Interoperability by Defining Formal Semantics of MPEG-7 Profiles. In Proc. of SAMT 2006.

[26] ISO 21127:2006 Information and documentation – A reference ontology for the interchange of cultural heritage information (CIDOC/CRM)

[27] The TEI site, http://www.tei-c.org /

[28] Official EAD Version 2002 Web Site, http://www.loc.gov/ead/

[29] Stavrakantonakis I.: “Interoperability support between OWL 2.0 and XML environments”, Diploma Thesis, Technical University of Crete, Dept. of Electronics and Computer Engineering, 2010 (to appear).