Support for Interoperability between OWL based and XML Schema based Applications

Chrisa Tsinaraki, Stavros Christodoulakis
TUC/MUSIC (Technical University of Crete / Lab. of Distributed Multimedia Information Systems & Applications)
{chrisa, stavros}@ced.tuc.gr

Abstract

We present in this paper a framework that provides support for interoperability between XML Schema based and OWL based applications. In particular, we describe how the information exchange between such applications is achieved, through the transformations of XML documents to OWL/RDF descriptions and of OWL/RDF descriptions to (parts of) valid XML documents. This functionality is built on top of OWL ontologies that fully capture the semantics of the XML Schemas. These ontologies are the outcome of the application of the XS2OWL mapping model that we have developed on an XML Schema. This way, the work reported here integrates and extends our previous work on the XS2OWL mapping model to take into account, in addition to the transformation of XML Schemas to OWL-DL ontologies, the transformation of XML documents to OWL/RDF descriptions and vice versa.

Categories and Subject Descriptors

H.3 [Information Storage and Retrieval]: [H.3.1 Content Analysis and Indexing] Dictionaries, Indexing methods. [H.3.5 Online Information Services] Data sharing, Web-based services. [H.3.7 Digital Libraries] Standards.

General Terms [consult http://www.acm.org/class/1998/ for details]

Algorithms, Standardization, Languages.

Keywords

Interoperability, Standards, XML/XML Schema, Ontologies, Semantic Web, OWL

1 Introduction

The *eXtensible Markup Language (XML)* [Bray et al. 2004] is the dominant standard for information exchange in the Web today. The *XML Schema Language* [Fallside 2001] is the preferred syntax for structuring XML documents, because of its structural capabilities and its central role in the data exchange in the Internet. As a consequence, several standards in different application domains (e.g. multimedia, e-learning, digital libraries, chemistry etc.) have been expressed in XML Schema syntax. This way, the XML/XML Schema framework allows for syntactic and structural interoperability support in the Internet today.

The development of the Semantic Web, on the other hand, has resulted in tools and methodologies that support semantic interoperability. The semantic interoperability allows applications of the same domain, possibly based on different standards, to exchange information through the utilization of domain knowledge that is expressed in the form of domain ontologies. An *ontology* is a logical theory accounting for the intended meaning of a formal vocabulary, i.e. its ontological commitment to a particular conceptualization of the world [Guarino 1998]. The dominant standard for ontology description is the *Web Ontology Language (OWL)* [McGuinness and van Harmelen 2004]. As a consequence, several OWL domain ontologies have been specified and are being specified. In addition, many OWL-based tools and applications have been developed and are being developed, which allow advanced semantic processing (including reasoning). This way, enrichment of the existing information with automatically extracted knowledge is possible.

The advances in the Web technology described in the previous paragraphs have resulted in the development of both OWL and XML based applications for several application domains. Consider, as an example, the multimedia domain applications. Several applications of the multimedia domain, such as automatic knowledge extraction from multimedia content, would benefit from advanced semantic processing and usually prefer working in an OWL environment. On the other hand, several other multimedia applications, like for example video segmentation, may utilize the MPEG-7 [Chang, Sikora and Puri 2001] XML Schema based syntax, since MPEG-7 is the dominant standard in multimedia content description. These two types of multimedia applications may need to interoperate in some usage scenarios (consider, for example, a segmentation application that imports the segment content descriptions).

The above example shows that, in the working environment formed in the Web today, the need for interoperability between XML Schema based applications and OWL based applications is apparent. This interoperability can be achieved if the XML Schema based and the OWL based applications of the same domain can exchange information. The information exchange can be accommodated if the XML documents that are valid according to a given XML Schema (e.g. valid MPEG-7 documents in the previous example) can be transformed in OWL/RDF descriptions that capture the knowledge encoded in the XML documents and vice versa, OWL/RDF descriptions can be transformed in XML documents (or document fragments) that are valid according to the XML Schemas used by other applications.

The existing research in this area is very limited. The transformation of XML documents in OWL/RDF descriptions has been carried out in [García and Celma 2005]. This approach is based on the ontologies that are automatically produced from XML Schemas according to the methodology described in [García and Celma 2005]. The major shortcoming of this approach is that the information captured in the ontologies does not allow the transformation of the OWL/RDF descriptions to (parts of) valid XML documents.

The framework presented in this paper offers the functionality required in order to allow for interoperability between XML Schema based and OWL based applications, since it allows transforming XML documents to OWL/RDF descriptions and OWL/RDF descriptions to (parts of) valid XML documents. This work builds on our previous research in the XS2OWL mapping model [Tsinaraki C. and Christodoulakis S. 2007a; Tsinaraki C. and Christodoulakis S. 2007b] and extends it. The XS2OWL mapping model supports the automatic transformation of XML Schemas in OWL. The ontologies resulting from the application of the XS2OWL mapping model to an XML Schema are utilized in the transformations described in this paper. The work reported here integrates and extends our previous research conducted in the XS2OWL mapping model [Tsinaraki C. and Christodoulakis S. 2007a; Tsinaraki C. and Christodoulakis S. 2007b] to take into account, in addition to the transformation of XML Schemas to OWL-DL ontologies, the transformation of XML documents to OWL/RDF descriptions and vice versa, the transformation of OWL/RDF descriptions to XML documents.

The rest of the paper is organized as follows: The necessary background information is provided in section 2, including descriptions of XML Schema and OWL. An overview of the XS2OWL mapping model is presented in section 3. The transformation of XML documents to OWL/RDF descriptions and of OWL/RDF descriptions to (parts of) valid XML documents are described in sections 4 and 5 respectively. The paper concludes in section 6, where our future research directions are also outlined.

2 Background

In this section we provide the necessary background information. In particular, we present the XML Schema language in subsection 2.1 and the Web Ontology Language (OWL) in subsection 2.2.

2.1. The XML Schema Language

The *XML Schema Language* [Fallside 2001] allows the definition of classes of XML documents using XML syntax and provides datatypes and rich structuring capabilities. An XML document is composed of *elements*, with the root element delimiting the beginning and the end of the document.

The XML Schema elements belong to XML Schema *types*, specified in their “type” attribute, and are distinguished into complex and simple elements, depending on the kind (simple or complex) of the types they belong to. Reuse of element definitions is supported by the *substitutionGroup* attribute, which states that the current element is a specialization of another element. The elements may either have a predefined order (forming XML Schema *sequences*) or be unordered (forming XML Schema *choices*). The main difference between sequences and choices is that all the sequence items must appear within the containing sequence in their specified order, while the choice items may appear at any order. Both sequences and choices may be nested. The minimum and maximum number of occurrences of the elements, choices and sequences are specified, respectively, in the “minOccurs” and “maxOccurs” attributes (absent “minOccurs” and/or “maxOccurs” correspond to values of 1). Reusable complex structures, combining sequences and choices, may be defined as *model groups*.

The *simple XML Schema types* are usually defined as restrictions of the basic datatypes provided by XML Schema (i.e. strings, integers, floats, tokens etc.). Simple types can neither contain elements nor carry attributes. The *complex XML Schema types* represent classes of XML constructs that have common features, represented by their elements and *attributes*. The attributes describe features with values of simple type and may form *attribute groups* comprised of attributes that should be used simultaneously. The elements represent features of the complex XML Schema types with values of any type. Default and fixed values may be specified for both attributes and simple type elements, in the *default* and *fixed* attributes respectively. Inheritance is supported for both simple and complex types, and the base types are referenced in the “base” attribute of the type definitions.

All the XML Schema constructs may have textual annotations, specified in their “annotation” element. The top-level XML Schema constructs (attributes, elements, simple and complex types, attribute and model groups) have unique *names* (specified in their “name” attribute). The nested elements and attributes have unique names in the context of the complex types in which they are defined, while the nested (complex and simple) types are unnamed. All the XML Schema constructs may have unique identifiers (specified in their “id” attribute). The top-level constructs may be referenced by other constructs using the “ref” attribute.

2.2. The Web Ontology Language (OWL)

The *Web Ontology Language (OWL)* [McGuinness and van Harmelen 2004] is the dominant standard in ontology definition. OWL has been developed according to the description logics paradigm and uses *RDF (Resource Description Framework)*/*RDFS (Resource Description Framework Schema)* [Manola and Milles 2004; Brickley and Guha 2004] syntax. Three OWL species of increasing descriptive power have been specified: (a) *OWL-Lite*, which is intended for lightweight reasoning but has limited expressive power; (b) *OWL-DL*, which provides the description logics expressivity and guarantees computational completeness and decidability of reasoning; and (c) *OWL-Full*, which has more flexible syntax than OWL-DL, but does not guarantee computational completeness and decidability of reasoning.

The basic functionality provided by OWL is: *(a) Import of XML Schema Datatypes* that extend or restrict the basic datatypes (e.g. ranges etc.). The imported datatypes have to be declared (using the rdfs:Datatype construct), as *RDFS datatypes*, in the ontologies they are used; *(b) Definition of OWL Classes* (using the owl:Class construct), organized in subclass hierarchies (using the rdfs:subClassOf construct), for the representation of sets of individuals sharing some properties. Complex OWL classes can be defined via *set operators* (using the owl:intersectionOf, owl:unionOf and owl:complementOf constructs) or via *direct enumeration* of their members (using the owl:oneOf construct); *(c) Definition of OWL Individuals*, essentially instances of the OWL classes, following the restrictions imposed on the class in which they belong; and *(d) Definition of OWL Properties*, which may form property hierarchies (using the rdfs:subPropertyOf construct), for the representation of the features of the OWL class individuals. Two kinds of properties are provided by OWL: *(i) Object Properties*, defined using the owl:ObjectProperty construct, which relate individuals of one OWL class (the property domain, defined using the rdfs:domain construct) with individuals of another OWL class (the property range, defined using the rdfs:range construct); and *(ii) Datatype Properties*, defined using the owl:DatatypeProperty construct, which relate individuals belonging to one OWL class (the property domain) with values of a given datatype (the property range). Restrictions may be defined on OWL class properties (using the owl:Restriction construct), including type (using the owl:allValuesFrom construct), cardinality (using the owl:minCardinality, owl:maxCardinality and owl:cardinality constructs), and value (using the owl:hasValue construct) restrictions. OWL classes, (object and datatype) properties and individuals are identified by unique identifiers, that are specified in the "rdf:ID" attribute. They may also have labels, defined using the rdfs:label construct, and textual descriptions, defined using the rdfs:comment construct.

3 XS2OWL Overview

We present in this section an overview of the XS2OWL mapping model, which allows the automatic transformation of XML Schemas to OWL-DL constructs. We decided to use the OWL-DL specie of OWL, since it provides the description logics expressivity and guarantees computational completeness and decidability of reasoning.

The XS2OWL transformation model, which is outlined in Figure 3-1, takes an XML Schema as input and transforms it into:

(a) A *main* OWL-DL ontology that directly captures the XML Schema semantics.
(b) A *datatypes* XML Schema, which contains the simple XML Schema datatypes defined in the source XML Schema and are used in the main ontology.
(c) A *mapping* OWL-DL ontology that:
- Keeps the mapping of the rdf:IDs of the OWL constructs of the main ontology with the names of the XML Schema constructs. This information is necessary, since in a valid OWL ontology the different constructs should have unique rdf:IDs, while the XML Schema Language allows different constructs to have the same name.
- Systematically captures the semantics of the XML Schema constructs that cannot be directly captured in the main ontology, since they cannot be represented by corresponding OWL constructs.

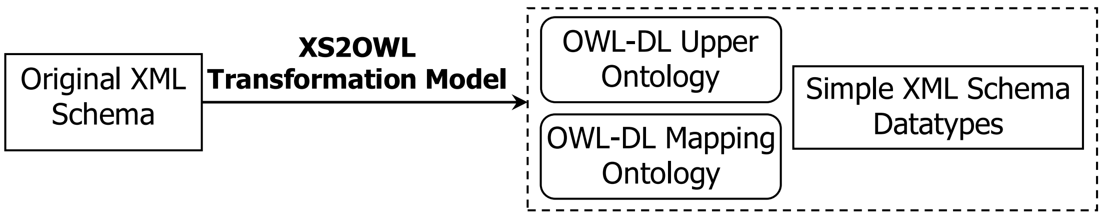

Figure 3-1: The XS2OWL Transformation Model

Thus, for every input XML Schema, the XS2OWL mapping model produces and ontological infrastructure that can support interoperability between XML Schema based applications and OWL based applications. This interoperability is achieved through the exchange of information between the XML Schema based and the OWL based applications of the same domain. The details of the utilization of the produced ontological infrastructure in order to allow information exchange between these applications are described in sections 4 and 5.

The rest of this section includes the description of the structure and the semantics of the mapping ontologies in subsection 3.1 and the presentation of the XS2OWL mappings in subsection 3.2.

3.1. Mapping Ontologies

We present in this subsection the structure and the semantics of the mapping ontologies produced using the XS2OWL mapping model. The mapping ontologies follow a model that allows keeping the mapping of the corresponding OWL and XML Schema constructs and captures the XML Schema semantics that cannot be represented in OWL. This model is expressed as an OWL-DL ontology, the *OWL2XMLRules Ontology*, which is extended with individuals defined in the mapping ontologies. The classes of the OWL2XMLRules ontology are listed below.

- The *DatatypePropertyInfoType* class, which keeps the mapping of the OWL datatype properties with the corresponding XML Schema constructs and specifies the possible default value and the kind (i.e. attribute, element or simple type extension) of the XML Schema construct represented by a datatype property.

- The *ElementInfoType* class, which captures information about the XML Schema elements that cannot be directly represented by OWL constructs (like, for example, sequence element order) and keeps the mapping of the XML Schema elements with the corresponding OWL constructs.
- The *ComplexTypeInfoType* class, which captures information about the complex XML Schema types that cannot be directly represented by OWL constructs and keeps the mapping of the complex XML Schema types with the corresponding OWL constructs.
- The *ChoiceType* and *SequenceType* classes, which capture information about the complex XML Schema sequences and choices that cannot be directly represented by OWL constructs.

3.2. The XS2OWL Mappings

In this section we present the mappings comprising the XS2OWL mapping model that allows the transformation of the XML Schema constructs to OWL-DL constructs.

The XS2OWL mappings are presented in Table 1. The first column of Table 1 contains the XML Schema constructs, while the second column contains the OWL constructs that represent them in the main ontology. The third column provides the mapping ontology constructs that represent the semantics of the XML Schema constructs that cannot be expressed directly in OWL and the fourth column presents the contents of the datatypes XML Schema.

Table 1. Overview of the XS2OWL Mappings

XML Schema Construct	**OWL-DL Representation**		
	Main Ontology	**Mapping Ontology**	**Datatypes**
Complex Type	Class	*ComplexTypeInfoType* individual	
Simple Datatype	Datatype Declaration		Simple Type
Element	(Datatype or Object) Property	*ElementInfoType* individual	
Attribute	Datatype Property	*DatatypePropertyInfoType* individual	
Sequence	Unnamed Class - Intersection	*SequenceInfoType* individual	
Choice	Unnamed Class - Union	*ChoiceInfoType* individual	
Annotation	Comment		

According to Table 1, the direct mappings of the XML Schema constructs to OWL constructs in the main ontologies are the following:

- The complex XML Schema types are mapped to OWL classes, since both the complex XML Schema types and the OWL classes represent sets of entities with common features.
- The simple XML Schema datatypes are mapped to datatype declarations. This is due to the fact that OWL does not directly support the definition of simple datatypes, but it only allows using simple XML Schema datatypes that have been declared in the OWL ontologies.
- The XML Schema attributes are mapped to OWL datatype properties, since both the XML Schema attributes and the OWL datatype properties represent simple type features.
- The (simple and complex type) XML Schema elements are mapped to OWL (datatype and object) properties, since both the XML Schema elements and the OWL properties represent features.
- The XML Schema sequences and choices are represented by OWL unnamed classes formed using set operators and cardinality restrictions on the sequence/choice items, since the XML Schema sequences and choices describe the feature cardinalities and how the entity features are combined.
- The annotations of the XML Schema constructs are mapped to OWL comments, since both the XML Schema annotations and the OWL comments are textual descriptions of the different language constructs.

As an example, consider the XML Schema person.xsd, which describes the structure of person descriptions and is presented in Figure 3-2.

```
<xs:schema ...>
  <xs:element name="Persons" type="PersonsType"/>
  <xs:complexType name="PersonsType">
```

```
    <xs:sequence>
      <xs:element name="Person" type="PersonType" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="PersonType">
    <xs:sequence>
      <xs:element name="Name" type="xs:string"/>
      <xs:element name="Age" type="validAgeType"/>
    </xs:sequence>
    <xs:attribute name="prefix" type="xs:string"/>
  </xs:complexType>
  <xs:simpleType name="validAgeType">
    <xs:restriction base="xs:float">
      <xs:minInclusive value="0.0"/>
      <xs:maxInclusive value="150.0"/>
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```

Figure 3-2: XML Schema for the Description of Persons

The root element of the XML Schema of Figure 3-2 is the "`Persons`" element, of type "`PersonsType`". "`PersonsType`" essentially is a sequence of "`Person`" elements, of type "`PersonType`". The "`PersonType`" instances have the "`prefix`" attribute, of string type, and the elements "`Name`" and "`Age`", of string and "`validAgeType`" type respectively. "`validAgeType`" is a simple XML Schema datatype that represents real numbers in the range [0-150], which could be valid person ages.

```
<rdf:RDF ...>
  <owl:Ontology rdf:about=""/>
  <rdfs:Datatype rdf:about="&datatypes;validAgeType">
    <rdfs:isDefinedBy rdf:resource="&datatypes;"/>
    <rdfs:label>validAgeType</rdfs:label>
  </rdfs:Datatype>
  <owl:Class rdf:ID="PersonsType">
    <rdfs:label>PersonsType</rdfs:label>
  </owl:Class>
  <owl:ObjectProperty rdf:ID="Person__PersonType">
    <rdfs:domain rdf:resource="#PersonsType"/>
    <rdfs:range rdf:resource="#PersonType"/>
    <rdfs:label>Person</rdfs:label>
  </owl:ObjectProperty>
  <owl:Class rdf:ID="PersonType">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#prefix__xs_string"/>
        <owl:maxCardinality rdf:datatype="&xsd;integer">1</owl:maxCardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Class>
        <owl:intersectionOf rdf:parseType="Collection">
          <owl:Restriction>
             <owl:onProperty rdf:resource="#Name__xs_string"/>
             <owl:cardinality rdf:datatype="&xsd;integer">1</owl:cardinality>
          </owl:Restriction>
          <owl:Restriction>
             <owl:onProperty rdf:resource="#Age__validAgeType"/>
             <owl:cardinality rdf:datatype="&xsd;integer">1</owl:cardinality>
          </owl:Restriction>
        </owl:intersectionOf>
      </owl:Class>
    </rdfs:subClassOf>
    <rdfs:label>PersonType</rdfs:label>
  </owl:Class>
  <owl:DatatypeProperty rdf:ID="prefix__xs_string">
    <rdfs:domain rdf:resource="#PersonType"/>
    <rdfs:range rdf:resource="&xs;string"/>
    <rdfs:label>prefix</rdfs:label>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="Name__xs_string">
    <rdfs:domain rdf:resource="#PersonType"/>
    <rdfs:range rdf:resource="&xs;string"/>
    <rdfs:label>Name</rdfs:label>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="Age__validAgeType">
    <rdfs:domain rdf:resource="#PersonType"/>
    <rdfs:range rdf:resource="&datatypes;validAgeType"/>
    <rdfs:label>Age</rdfs:label>
  </owl:DatatypeProperty>
  <owl:ObjectProperty rdf:ID="Persons__PersonsType">
    <rdfs:range rdf:resource="#PersonsType"/>
    <rdfs:label>Persons</rdfs:label>
  </owl:ObjectProperty>
</rdf:RDF>
```

Figure 3-3: Main Ontology resulting from the application of the XS2OWL Mapping Model on the XML Schema of Figure 3-2

The application of the XS2OWL mapping model on the XML Schema of Figure 3-2 results in the main ontology of Figure 3-3. The main ontology of Figure 3-3 directly captures the semantics of the XML Schema of Figure 3-2 in OWL-DL and consists of:

- The "`validAgeType`" datatype declaration, which allows using the "`validAgeType`" datatype defined in the XML Schema referenced in the "`&datatypes;`" XML entity.
- The "`PersonsType`" and "`PersonType`" classes, which represent the "`PersonsType`" and "`PersonType`" complex types respectively.
- The "`Name__xs_string`", "`Age__validAgeType`" and "`prefix__xs_string`" datatype properties, which represent the "`Name`" and "`Age`" simple type elements and the "`prefix`" attribute respectively.
- The "`Persons__PersonsType`" and "`Person__PersonType`" object properties, which represent the "`Persons`" and "`Person`" complex type elements respectively.

4 Transformation of XML Documents to OWL/RDF Descriptions

We present in this section the transformation of XML documents to OWL/RDF descriptions. The transformation process is outlined in Figure 4-1.

As shown in Figure 4-1, the transformation algorithm takes as input an XML document and produces an OWL/RDF description using the information captured in the main ontology, the mapping ontology and the simple XML Schema datatypes that represent in OWL the semantics of the XML Schema to which the input XML document obeys. The produced OWL/RDF description is comprised of individuals that belong to the classes of the main ontology that captures the semantics of the XML Schema. As a consequence, the transformation process can be applied only to XML documents that obey to XML Schemas on which the XS2OWL mapping model has been applied.

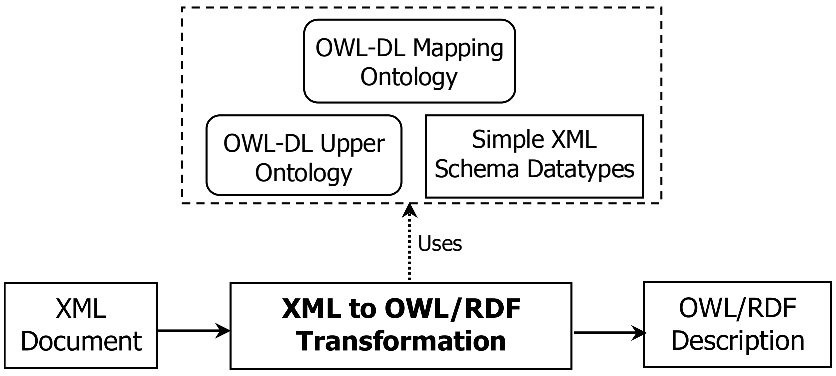

Figure 4-1: Transformation of XML Documents to OWL/RDF Descriptions

The algorithm transformXMLdocument that transforms XML documents to OWL/RDF descriptions is presented in Figure 4-2.

```
algorithm transformXMLdocument(XMLdocument)
  define an individual that represents the document using owl:Thing
  root_element = root element of XMLdocument
  call transformXMLelement(root_element)
end algorithm

algorithm transformXMLelement(XMLelement)
  if XMLelement is of simple type
    call transformSimpleXMLelement(XMLelement)
  else
    call transformComplexXMLelement(XMLelement)
  end if
end algorithm

algorithm transformSimpleXMLelement(XMLelement)
  dp_id = rdf:ID of the datatype property corresponding to XMLelement (found
through the mapping ontology)
  dp = new instance of the datatype property with rdf:ID = dp_id
  dp value = XMLelement content
end algorithm

algorithm transformComplexXMLelement(XMLelement)
  op_id = rdf:ID of the object property corresponding to XMLelement (found
through the mapping ontology)
  op = new instance of the object property with rdf:ID = op_id
  call defineIndividual(XMLelement)
end algorithm

algorithm defineIndividual(XMLelement)
  t = XMLelement type
  cid = rdf:ID of the class corresponding to t (found through the mapping
ontology)
  if t extends a simple type
    st = the simple type extended by t
    base_dp_id = concatenate(content, st@name)
    base_dp = new instance of the datatype property with rdf:ID = base_dp_id
    base_dp value = content of XMLelement
  end if
  indiv = new individual of the class with rdf:ID = cid
  for each attribute of t with default value that is not defined in XMLelement
    a = current attribute
    define an instance of a having the default value
  end for
  for each simple type element of t with default value that is not defined in
XMLelement
    e = current element
    define an instance of e having the default value
  end for
  for each attribute of XMLelement
    a = current attribute
    call transformXMLattribute(a)
  end for
  for each element of XMLelement
    e = current element
    call transformXMLelement(e)
  end for
end algorithm

algorithm transformXMLattribute(XMLattribute)
  dp_id = rdf:ID of the datatype property corresponding to XMLattribute (found
through the mapping ontology)
  dp = new instance of the datatype property with rdf:ID = dp_id
  dp value = XMLattribute value
end algorithm
```

Figure 4-2: Algorithm that transforms XML documents to OWL/RDF descriptions

As an example, consider the XML document of Figure 4-3, which is a valid XML document structured according to the XML Schema of Figure 3-2 and describes a set of persons containing one person. The algorithm presented in Figure 4-2 transforms this XML document to the OWL/RDF description of Figure 4-4, which is based on the main ontology that captures the semantics of the XML Schema of Figure 3-2.

```
<Persons ...>
  <Person prefix="Ms">
    <Name>Chrisa Tsinaraki</Name>
    <Age>35</Age>
  </Person>
</Persons>
```

Figure 4-3: XML Document, valid according to the XML Schema of Figure 3-2, that describes a set of persons containing one person

```
<owl:Thing>
  <person:Persons__PersonsType>
    <person:PersonsType>
      <person:Person__PersonType>
        <person:PersonType>
          <person:prefix__xs_string>Ms</person:prefix__xs_string>
          <person:Age__validAgeType>35</person:Age__validAgeType>
          <person:Name__xs_string>Chrisa Tsinaraki</person:Name__xs_string>
        </person:PersonType>
      </person:Person__PersonType>
    </person:PersonsType>
  </person:Persons__PersonsType>
</owl:Thing>
```

Figure 4-4: OWL/RDF description of a set of persons containing one person, which is equivalent with the XML document of Figure 4-3 and compliant to the main ontology of Figure 3-3

The transformation of XML documents to OWL/RDF descriptions is very important since it allows importing in OWL/RDF the knowledge encoded in existing XML descriptions and utilize it during the reasoning process. This is extremely important if the XML Schema obeyed by the XML documents represents a standard, because in this case a large number of descriptions are expected to exist.

5 Transformation of OWL/RDF Descriptions to XML Documents

In this section we describe the transformation of OWL/RDF descriptions to XML documents (or XML document fragments). The transformation process is depicted in Figure 5-1.

As shown in Figure 4-1, the transformation algorithm takes as input an OWL/RDF description and outputs an XML document (or XML document fragment). The transformation algorithm uses the information captured in the main ontology, the mapping ontology and the simple XML Schema datatypes that capture the semantics of the XML Schema to which the output XML document obeys. The input OWL/RDF description is comprised of individuals that belong to the classes of the main ontology that captures the semantics of this XML Schema.

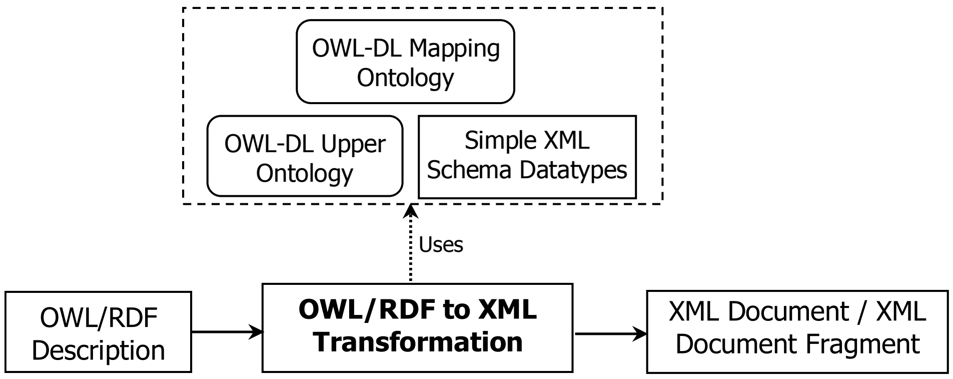
OWL-DL Mapping Ontology
OWL-DL Upper Ontology
Simple XML Schema Datatypes
Uses
OWL/RDF Description
OWL/RDF to XML Transformation
XML Document / XML Document Fragment

Figure 5-1: Transformation of OWL/RDF Descriptions to XML Documents (or XML Document Fragments)

The algorithm transformOWLRDFdescription that transforms OWL/RDF descriptions to XML documents (or XML document fragments) is presented in Figure 5-2.

```
algorithm transformOWLRDFdescription(OWLRDFdescription)
  if the description contains an object property corresponding to the root
element
    call transformOWLRDFdescription2Doc(OWLRDFdescription)
  else
    call transformOWLRDFdescription2Fragment(OWLRDFdescription)
  end if
end algorithm

algorithm transformOWLRDFdescription2Doc(OWLRDFdescription)
  define a root element with all the necessary namespaces
  root_property = the object property instance of OWLRDFdescription that
corresponds to the root element
  call transformObjectProperty(root_property)
end algorithm

algorithm transformOWLRDFdescription2Fragment(OWLRDFdescription)
  top_properties = the object property instances of OWLRDFdescription that
correspond to the element(s) closer to the root element
  for each object property in top_properties
    op = the current object property
    call transformObjectProperty(op)
  end for
end algorithm

algorithm transformObjectProperty(ObjectProperty)
  e_name = name of the element that corresponds to ObjectProperty (found through
the mapping ontology)
  e = a new instance of the element with name = e_name
  e_value = the individual that represents the value of ObjectProperty
  call transformIndividual(e_value)
end algorithm

algorithm transformDatatypeProperty(DatatypeProperty)
  kind = construct represented by DatatypeProperty
  if kind = "Attribute"
    a_name = name of the attribute that corresponds to DatatypeProperty (found
through the mapping ontology)
    a = a new instance of the attribute with name = a_name
    a_value = the value of DatatypeProperty
  else if kind = "Element"
    e_name = name of the element that corresponds to DatatypeProperty (found
through the mapping ontology)
    e = a new instance of the element with name = e_name
    e_value = the value of DatatypeProperty
  else
    output the value of DatatypeProperty
  end if
end algorithm

algorithm transformIndividual(Individual)
  dps = the datatype properties of Individual
  ops = the object properties of Individual ordered according to the information
in the mapping ontology
  for each datatype property in dps
    dp = the current datatype property
    call transformDatatypeProperty(dp)
  end for
  for each object property in ops
    op = the current object property
    call transformObjectProperty(op)
  end for
end algorithm
```

Figure 5-2: Algorithm that transforms OWL/RDF Descriptions to XML Documents (or XML Document Fragments)

As an example, the application of the algorithm presented in Figure 5-2 on the OWL/RDF description of Figure 5-3 results in the XML document fragment of Figure 5-4, which is valid according to the XML Schema of Figure 3-2.

```
<owl:Thing>
  <person:Person__PersonType>
    <person:PersonType>
      <person:prefix__xs_string>Prof</person:prefix__xs_string>
      <person:Age__validAgeType>59</person:Age__validAgeType>
      <person:Name__xs_string>Stavros Christodoulakis</person:Name__xs_string>
    </person:PersonType>
  </person:Person__PersonType>
</owl:Thing>
```

Figure 5-3: OWL/RDF description of a person

```
<Persons ...>
  <Person prefix="Prof">
    <Name>Stavros Christodoulakis</Name>
    <Age>59</Age>
  </Person>
</Persons>
```

Figure 5-4: XML Document Fragment that describes a person and is the result of the application of the algorithm of Figure 5-2 on the OWL/RDF description of Figure 5-3

The transformation of OWL/RDF descriptions to XML documents (or XML document fragments) is important, since it allows to pure XML Schema based applications to import descriptions enriched using through advanced semantic processing.

6 Conclusions – Future Work

We have presented in this paper a framework that provides support for interoperability between XML Schema based and OWL based applications. The framework presented in this paper allows these types of applications to exchange information, since it allows transforming XML documents to OWL/RDF descriptions and OWL/RDF descriptions to (parts of) valid XML documents. This work builds on our previous research in the XS2OWL mapping model [Tsinaraki C. and Christodoulakis S. 2007a; Tsinaraki C. and Christodoulakis S. 2007b] and extends it to take into account, in addition to the transformation of XML Schemas to OWL-DL ontologies, the transformation of XML documents to OWL/RDF descriptions and vice versa, the transformation of OWL/RDF descriptions to XML documents.

Our future research in this area includes the development of methodologies for the systematic integration of the main ontologies produced using the XS2OWL mapping model with widely accepted top-level ontologies like DOLCE [DOLCE] and SUMO [IEEE SUO WG].

References

Bray T., Paoli J., Sperberg-McQueen C. M., Maler E., Yergeau F. and Cowan J. (eds.) 2004. *Extensible Markup Language (XML) 1.1*. W3C Recommendation, http://www.w3.org/TR/xml11/.

Brickley D. and Guha R. V. (eds.) 2004. *RDF Vocabulary Description Language 1.0: RDF Schema*. W3C Recommendation, http://www.w3.org/TR/rdf-schema.

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

The DOLCE Ontology, http://www.loa-cnr.it/DOLCE.html

Fallside D. and Walmsley P. (eds.) 2001. *XML Schema Part 0: Primer*. W3C Recommendation, http://www.w3.org/TR/xmlschema-0/.

García R. and Celma O. 2005. *Semantic Integration and Retrieval of Multimedia Metadata*. In the proceedings of Knowledge Mark-up and Semantic Annotation Workshop, Semannot'05.

Guarino N. 1998. *Formal Ontology and Information Systems*. In Proc. of the 1st International Conference "Formal Ontology in Information Systems" (FOIS '98), pp. 3-15, June 6-8 1998

IEEE SUO WG, *Standard Upper Ontology Working Group (IEEE P1600.1)*, http://suo.ieee.org/

McGuinness D. L. and van Harmelen F. (eds.) 2004. OWL Web Ontology Language: Overview. W3C Recommendation, http://www.w3.org/TR/owl-features.

Manola F. and Milles E. (eds.) 2004. *RDF Primer*. W3C Recommendation, http://www.w3.org/TR/rdf-primer.

Tsinaraki C. and Christodoulakis S. 2007a. *XS2OWL: A Formal Model and a System for enabling XML Schema Applications to interoperate with OWL-DL Domain Knowledge and Semantic Web Tools*. In Proc. of the DELOS Conference, February 2007, Tirrenia, Italy.

Tsinaraki C. and Christodoulakis S. 2007b. *Interoperability of XML Schema Applications with OWL Domain Knowledge and Semantic Web Tools*. In Proc. of the ODBASE 2007, November 26-29 2007, Vilamoura, Portugal.