XS2OWL: A Formal Model and a System for enabling XML Schema Applications to interoperate with OWL-DL Domain Knowledge and Semantic Web Tools

Chrisa Tsinaraki[1] and Stavros Christodoulakis[1]

[1]TUC/MUSIC, Technical University of Crete Campus, 73100 Kounoupidiana, Crete, Greece
{chrisa, stavros}@ced.tuc.gr

Abstract. The domination of XML in the Internet for data exchange has led to the development of standards with XML Schema syntax for several application domains. Advanced semantic support, provided by domain ontologies and semantic Web tools like logic-based reasoners, is still very useful for many applications. In order to provide it, interoperability between XML Schema and OWL is necessary so that XML schemas can be converted to OWL. This way, the semantics of the standards can be enriched with domain knowledge encoded in OWL domain ontologies and further semantic processing may take place. In order to achieve interoperability between XML Schema and OWL, we have developed XS2OWL, a model and a system that are presented in this paper and enable the automatic transformation of XML Schemas in OWL-DL. XS2OWL also enables the consistent transformation of the derived knowledge (individuals) from OWL-DL to XML constructs that obey the original XML Schemas.

Keywords: Interoperability, Standards, XML Schema, OWL, Ontologies

1. Introduction

Web applications and services have formed an open environment, where the applications developed by different vendors interoperate on the basis of the emergent standards. The dominant data exchange standard in the Internet today is the *eXtensible Markup Language (XML)* [2]. The XML documents are usually structured according to schemas expressed in *XML Schema Language* [5] syntax. XML Schema uses XML syntax, supports very rich structures and datatypes for XML documents and plays a central role in the data exchange in the Internet. As a consequence, important standards in different application domains have been specified in XML Schema such as the MPEG-7 [4] and the MPEG-21 [14] for multimedia, the IEEE LOM [10] and SCORM [1] in e-learning, the METS [9] for Digital Libraries etc.

Advanced semantic support, though, would be very useful for several standard-based applications that need to integrate domain knowledge expressed in domain ontologies and perform semantic processing (including reasoning) within the constructs of the standards. As an example, consider the MPEG-7 based multimedia applications. MPEG-7 provides rich multimedia content description capabilities and has been specified using XML Schema syntax, like many other standards. MPEG-7 based services (e.g. retrieval, filtering etc.) would benefit from domain knowledge integration. MPEG-7 provides general-purpose constructs that could be used for domain knowledge description [18], but the developers that are going to integrate domain knowledge in MPEG-7 are likely to be more familiar with the *Web Ontology Language (OWL)* [13] than with the domain knowledge description mechanisms of MPEG-7. In addition, some applications of MPEG-7, like the (semi-)automatic multimedia content annotation may greatly benefit from using logic-based reasoners for OWL. As a consequence, the capability to work with the semantics of MPEG-7 expressed in OWL and integrated with OWL domain ontologies is beneficial for such applications. Since other MPEG-7 applications may work with the XML Schema version of MPEG-7, the derived knowledge should be converted back to standard MPEG-7/XML constructs.

We present in this paper the XS2OWL transformation model that allows to transform the XML Schema constructs in OWL, so that applications using XML Schema based standards will be able to use the Semantic Web methodologies and tools. XS2OWL also supports the conversion of the OWL-based constructs back to the XML Schema based constructs in order to maintain the compatibility with the XML schema versions of the standards. XS2OWL has been implemented as an *XML Stylesheet Transformation Language (XSLT)* [7] stylesheet and transforms every XML Schema based standard in an OWL-DL *Main Ontology*. This way, the constructs of the standard become first class Semantic Web objects and may be integrated with domain knowledge expressed as OWL domain ontologies. In addition, all the OWL-based Semantic Web tools, including reasoners, can be used with the standard-based descriptions. In addition, a *Mapping Ontology* is generated for each XML Schema, which allows encoding all the knowledge needed to transform the individuals generated or added later on to the main ontology back to XML syntax valid according to the original XML Schema.

The research conducted in the support of interoperability between XML Schema and OWL is limited. We had observed the need for such support for the MPEG-7 standard in the context of the DS-MIRF framework [16, 17, 18]. In order to achieve it, we first defined manually an Upper OWL-DL ontology capturing the *MPEG-7 Multimedia Description Schemes (MDS)* [12] and the *MPEG-21 Digital Item Adaptation (DIA) Architecture* [11]. This way, domain knowledge expressed in OWL domain ontologies could be integrated with the semantics of the standards captured in the Upper ontology, as was done with ontologies for soccer and Formula 1. Finally, we developed a set of transformation rules for transforming the OWL individuals that describe the multimedia content and have been defined using the Upper ontology and the domain ontologies back to the original MPEG-7/21 constructs. The transformation rules rely on a mapping ontology that systematically captures the semantics of MPEG-7/21 that cannot be captured in the Upper ontology. This work is an important motivating example for the need of the general-purpose mechanism described here.

The automatic transformation of XML Schema constructs to OWL constructs has been proposed in [6]. According to this methodology, an XML Schema is transformed to an OWL-Full ontology that partially captures the XML Schema semantics. This way, information is lost during the transformation from XML Schema to OWL, and no support is provided in order to transform OWL individuals obeying the ontologies produced back to XML syntax valid according to the original XML Schemas. Finally some XML Schema construct transformations of to OWL in [6] do not follow closely the XML Schema semantics. The XS2OWL model presented in this paper allows automatically transforming XML Schema constructs to OWL-DL constructs (not OWL-Full) without loosing any information. This way, computational completeness and decidability of reasoning are guaranteed in the OWL ontologies produced and back transformations are supported.

The rest of the paper is structured as follows: In section 2 we provide background information. The proposed model for transforming XML Schema constructs in OWL-DL is presented in section 3. The mapping ontologies that represent the XML Schema semantics that cannot be directly transformed in OWL-DL are described in section 4. In section 5 we present the realization of the XS2OWL model, so that the transformations are carried out automatically. The paper conclusions are presented in section 6.

2. Background

In this section we present the background information needed in other parts of the paper. In particular, we present in brief the *XML Schema Language* and the *Web Ontology Language (OWL)*.

The XML Schema Language. The *XML Schema Language* [5] 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. Reuse of the 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*). 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" attributes correspond to values of 1). Reusable complex structures, combining sequences and choices, may be defined as *model groups*.

The XML Schema language allows for the definition of both complex and simple elements. Complex elements belong to *complex types*, which may include other elements and carry *attributes* that describe their features. Simple elements belong to *simple types*, which 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 other elements nor carry attributes. Inheritance and constraints are supported for both simple and complex types. Sets of attributes that should be used simultaneously may form *attribute groups*. Default and fixed values may be specified for XML Schema attributes and simple type elements.

The top-level XML Schema constructs (attributes, elements, simple and complex types, attribute and model groups) have unique *names* (specified in their "name" attribute), while the nested types and groups 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.

The Web Ontology Language (OWL). The *Web Ontology Language (OWL)* [13] is the dominant standard in ontology definition. OWL has followed the description logics paradigm and uses *RDF (Resource Description Framework)*/*RDFS (Resource Description Framework Schema)* [8, 3] syntax. Three OWL species of increasing descriptive power have been specified: *OWL-Lite*, which is intended for lightweight reasoning but has limited expressive power, *OWL-DL*, which provides description logics expressivity and guarantees computational completeness and decidability of reasoning, and *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 represent *simple types* extending or restricting the basic datatypes (e.g. ranges etc.). The imported datatypes have to be declared, as *RDFS datatypes*, in the ontologies they are used; *(b) Definition of OWL Classes*, organized in subclass hierarchies, for the representation of sets of individuals sharing some properties. Complex OWL classes can be defined via *set operators* (intersection, union or complement of other classes) or via *direct enumeration* of their members; *(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, for the representation of the features of the OWL class individuals. Two kinds of properties are provided by OWL: *(i) Object Properties*, which relate individuals of one OWL class (the property domain) with individuals of another OWL class (the property range); and *(ii) Datatype Properties*, 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, including type, cardinality and value restrictions. OWL classes, properties and individuals are identified by unique identifiers specified in their "rdf:ID" attributes.

3. Transformation of XML Schema Constructs to OWL-DL

We present in this section a model for the direct transformation of the XML Schema constructs in OWL-DL. The result of the transformation of a source XML Schema is a *main ontology*, an OWL-DL ontology that captures the semantics of the XML Schema constructs. The transformations of the individuals XML Schema constructs are presented in the next paragraphs.

Simple XML Schema Datatypes. OWL does not directly support the definition of simple datatypes; it only allows importing simple datatypes. Existing XML Schema datatypes may be used in OWL ontologies if they have been declared in them. XS2OWL organizes all the simple XML Schema datatype definitions in the "datatypes" XML Schema and for each of them it generates an OWL datatype declaration. Let *st(name, id, body)* be an XML Schema simple datatype, where *body* is the body of the definition of *st*, *id* is the (optional) identifier of *st* and *name* is the name of *st*. *st* is transformed into: (a) The *st'(name', id, body)* simple datatype, which is stored in the "datatypes" XML Schema; and (b) the *dd(about, is_defined_by, label)* datatype declaration in the main ontology.

The *st'* simple type has the same *body* and *id* with *st*, while *name'* is formed as follows: If *st* is a top-level simple type, *name'* has the *name* value. If *st* is a simple type nested in the *ae* XML Schema construct (that may be an attribute or an element), *name'* has the value (a) *id* if *st* has a non-null identifier; and (b) the result of *concatenate(ct_name, '_', ae_name, '_UNType')* if *st* has a null identifier, where: (i) The *concatenate(...)* algorithm takes as input an arbitrary number of strings and returns their concatenation; and (ii) *ct_name* is the name of the complex type containing *ae*. If *ae* is a top-level attribute or element, *ct_name* has the 'NS' string as value.; and (iii) *ae_name* is the name of the property that represents *ae*.

The *dd* datatype declaration carries the following semantics: (a) *about* is the identifier referenced by the datatype declaration and is of the form *concatenate(url,name')*, where *url* is the URL of the "datatypes" XML Schema; (b) *is_defined_by* specifies where the datatype definition is located and has the *url* value; and (c) *label* is the label of *dd* and has *name'* as value.

As an example, consider the nested simple datatype of Fig. 1, which is defined in the "a1" attribute of the "ct1" complex type. It is transformed to the top-level simple datatype shown in Fig. 2, and the OWL datatype declaration shown in Fig. 3.

```
<xs:complexType name="ct1">
 <xs:simpleContent>
  <xs:extension base="xs:integer">
   <xs:attribute name="a1">
    <xs:simpleType>
     <xs:restriction base="xs:string"/>
    </xs:simpleType>
   </xs:attribute>
  </xs:extension>
 </xs:simpleContent>
</xs:complexType>
```

Fig. 1. Definition of a nested simple datatype

```
<simpleType name="ct1_a1_UNType">
 <restriction base="xs:string"/>
</simpleType>
```

Fig. 2. Top-level simple datatype representing the nested datatype of Fig. 1

```
<rdfs:Datatype rdf:about="&datatypes;ct1_a1_UNType">
 <rdfs:isDefinedBy rdf:resource="&datatypes;"/>
 <rdfs:label>ct1_a1_UNType</rdfs:label>
</rdfs:Datatype>
```

Fig. 3. OWL Declaration of the simple datatype of Fig. 2

Attributes. XML Schema attributes describe features with values of simple type. The OWL construct that can represent such features is the datatype property. Thus, XS2OWL transforms the XML Schema attributes into OWL datatype properties.

Let *a(name, aid, type, annot, ct_name, fixed, default)* be an XML Schema attribute, where *name* is the name of *a*, *aid* is the identifier of *a*, *type* is the type of *a*, *annot* is an (optional) annotation element of *a*, *ct_name* is the name of the complex XML Schema type *c_type* in the context of which *a* is defined (if *a* is a top-level attribute, *ct_name* has the null value), *fixed* is the (optional) fixed value of *a* and *default* is the (optional) default value of *a*. XS2OWL, transforms *a* into the OWL datatype property *dp(id, range, domain, label, comment)*, where: (a) *id* is the unique rdf:ID of *dp* and has *concatenate(name, '__', type)* as value; (b) *range* is the range of *dp* and has *type* as value; (c) *domain* is the domain of *dp* and has *ct_name* as value; (d) *label* is the label of *dp* and has *name* as value; and (e) *comment* is the textual description of *dp* and has *annot* as value. If any of the features of *a* is absent, the corresponding feature of *dp* is also absent. Note that: (a) If a fixed value of *a* is specified, it is represented as a value restriction in the definition of the OWL class *c* that represents *c_type*; and (b) If a default value of *a* is specified, it cannot be represented in the main ontology.

As an example, consider the "a1" attribute, shown in Fig. 1, which is transformed to the OWL datatype property shown in Fig. 4.

```
<owl:DatatypeProperty rdf:ID="a1__ct1_a1_UNType">
 <rdfs:domain rdf:resource="#ct1"/>
 <rdfs:range rdf:resource="&datatypes;ct1_a1_UNType"/>
 <rdfs:label>a1</rdfs:label>
</owl:DatatypeProperty>
```

Fig. 4. The OWL datatype property representing the "a1" attribute of Fig. 1

Elements. XML Schema elements represent features of complex XML Schema types and are transformed into OWL properties: The simple type elements are represented as OWL datatype properties and the complex type elements are represented as OWL object properties. Let *e(name, type, eid, annot, ct_name, substitution_group)* be an XML Schema element, where *name* is the name of *e*, *eid* is the identifier of *e*, *type* is the type of *e*, *annot* is an annotation element of *e*, *ct_name* is the name of the complex XML Schema type *c_type* in the context of which *e* is defined (if *e* is a top-level attribute, *ct_name* has the null value) and *substitution_group* is an (optional) element being extended by *e*. We represent *e* in OWL as a (datatype or object) property *p(id, range, domain, label, comment, super_property)*, where: (a) *id* is the unique rdf:ID of *p* and has *concatenate(name, '__', type)* as value; (b) *range* is the range of *p* and has *type* as value; (c) *domain* is the domain of *p* and has *ct_name* as value; (d) *label* is the label of *p* and has *name* as value; (e) *comment* is the textual description of *p* and has *annot* as value; and (f) *super_property* is the specification of the property specialized by *p* and has *substitution_group* as value.

```
<xs:element name="e" type="c_t2"/>
```

Fig. 5. Definition of the "e" element, nested in the complex type "c_t1"

```
<owl:ObjectProperty rdf:ID="e__c_t2">
 <rdfs:domain rdf:resource="#c_t1"/>
 <rdfs:range rdf:resource="#c_t2"/>
 <rdfs:label>e</rdfs:label>
</owl:ObjectProperty>
```

Fig. 6. The OWL object property representing the "e" element of Fig. 5

As an example, consider the "e" element, shown in Fig. 5, of type "c_t2", defined in the context of the complex type "c_t1". The "e" element is transformed to the OWL object property shown in Fig. 6.

Complex Types. The XML Schema complex types represent classes of XML instances that have common features, just as the OWL classes represent sets of individuals with common properties. Thus XS2OWL transforms the XML Schema complex types into OWL classes. Let *ct(name, cid, base, annot, attributes, sequences, choices)* be an XML Schema complex type, where: (a) *name* is the name of *ct*; (b) *aid* is the identifier of *ct*; (c) *base* is the (simple or complex) type extended by *ct*; (d) *annot* is an annotation element of *ct*; (e) *attributes* is the list of the attributes of *ct*; (f) *sequences* is the list of the *ct* sequences; and (g) *choices* is the list of the *ct* choices.

If *ct* extends a complex type, XS2OWL transforms it to the OWL class *c(id, super_class, label, comment, value_restrictions, cardinality_restrictions)*, where: (a) *id* is the unique rdf:ID of *c* and has *name* as value if *ct* is a top-level complex type. If *ct* is a complex type nested within the definition of an element *e*, *name* is a unique, automatically generated name of the form *concatenate(ct_name, '_', element_name, '_UNType')*, where *ct_name* is the name of the complex type containing *e* and *element_name* is the name of *e.* If *e* is a top-level element, *ct_name* has the 'NS' value; (b) *super_class* states which class is extended by *ct* and has *base* as value; (c) *label* is the label of *ct* and has *name* as value; (d) *comment* is the textual description of *ct* and has *annot* as value; (e) *value_restrictions* is the set of the value restrictions holding for the properties of *c*; and (f) *cardinality_restrictions* is the set of the cardinality restrictions assigned to the properties representing the *ct* attributes and the *ct* sequence/choice elements.

```
<owl:Class rdf:ID="ct1">
 <rdfs:subClassOf>
  <owl:Restriction>
   <owl:onProperty rdf:resource="#a1__ct1_a1_UNType"/>
   <owl:maxCardinality rdf:datatype="&xsd;integer">1</owl:maxCardinality>
  </owl:Restriction>
 </rdfs:subClassOf>
 <rdfs:subClassOf>
  <owl:Restriction>
   <owl:onProperty rdf:resource="#content__xs_integer"/>
   <owl:cardinality rdf:datatype="&xsd;integer">1</owl:cardinality>
  </owl:Restriction>
 </rdfs:subClassOf>
 <rdfs:label>ct1</rdfs:label>
</owl:Class>
<owl:DatatypeProperty rdf:ID="content__xs_integer">
 <rdfs:domain rdf:resource="#ct1"/>
 <rdfs:range rdf:resource="&xs;integer"/>
</owl:DatatypeProperty>
```

Fig. 7. OWL class representing the "ct1" complex type of Fig. 1

If *ct* extends a simple type, XS2OWL transforms it to the OWL class *c(id, label, comment, value_restrictions, cardinality_restrictions)*, with the same semantics with the classes representing complex types that extend complex types on the corresponding items. The extension of the simple type is represented by the datatype property *ep(eid, erange, edomain)* of cardinality 1, where: (a) *eid* is the unique rdf:ID of *ep* and has *concatenate(base, '_content')* as value; (b) *range* is the range of *ep* and has *base* as value; and (c) *domain* is the domain of *ep* and takes as value the *id* of *c*.

The attributes and the elements that are defined or referenced in *ct* are transformed to the corresponding OWL-DL constructs.

As an example, consider the complex type "ct1", shown in Fig. 1. The "ct1" complex type is represented by the "ct" OWL class, shown in Fig. 7, together with the "content__xs_integer" datatype property, which states that "ct1" is an extension of xs:integer.

Sequences and Choices. The XML Schema sequences and choices essentially are XML element containers, defined in the context of complex types and model groups. The main difference between sequences and choices is that the sequences are ordered, while the choices are unordered. XS2OWL transforms both the sequences and the choices to unnamed OWL-DL classes featuring complex cardinality restrictions on the sequence/choice items (elements, sequences and choices) and places them in the definition of the classes that represent the complex types where the sequences/choices are referenced or defined.

The lower bound of the minimum cardinality of the construct that represents a sequence/choice item has the value *i_min_occurs*s_min_occurs* and the upper bound of the construct maximum cardinality has the value *i_max_occurs*s_max_occurs*, where: (a) *i_min_occurs* is the value of the "minOccurs" attribute of the item; (b) *s_min_occurs* is the value of the "minOccurs" attribute of the sequence; (c) *i_max_occurs* is the value of the "maxOccurs" attribute of the item; and (d) *s_max_occurs* is the value of the "maxOccurs" attribute of the sequence. In addition, the cardinality of the sequence/choice items must always be a multiple in the range [*i_min_occurs – i_max_occurs*].

Sequence items must appear in their order. Thus, the sequences are transformed to unnamed classes, formed as the intersection of the cardinality restrictions of their items. Notice that the exact sequence cardinalities cannot be computed when a sequence item is contained in a sequence with unbounded maximum number of occurrences and the item has no maximum cardinality restriction. In addition, information regarding the sequence element ordering cannot be represented in OWL.

As an example, consider the sequence shown in Fig. 8, which is defined in the context of the complex type "c_t1". The sequence is represented, in the "c_t1" class definition, by the unnamed class shown in Fig. 9.

```
<xs:sequence minOccurs="2" maxOccurs="2">
 <xs:element name="e1" type="xs:string"/>
 <xs:element name="e2" type="xs:string" maxOccurs="3"/>
</xs:sequence>
```

Fig. 8. Sequence defined in the context of the Complex Type "c_type1"

```
<owl:Class>
 <owl:intersectionOf rdf:parseType="Collection">
  <owl:Restriction>
   <owl:onProperty rdf:resource="#e1__xs_string"/>
   <owl:cardinality rdf:datatype="&xsd;integer">2</owl:cardinality>
  </owl:Restriction>
  <owl:Restriction>
   <owl:onProperty rdf:resource="#e2__xs_string"/>
   <owl:minCardinality
rdf:datatype="&xsd;integer">2</owl:minCardinality>
  </owl:Restriction>
  <owl:Restriction>
   <owl:onProperty rdf:resource="#e2__xs_string"/>
   <owl:maxCardinality
rdf:datatype="&xsd;integer">6</owl:maxCardinality>
  </owl:Restriction>
 </owl:intersectionOf>
</owl:Class>
```

Fig. 9. OWL Representation of the sequence shown in Fig. 8

The choice items may appear at any order. Thus, the choices are transformed to unnamed classes, formed as the union of the allowed combinations of the cardinality restrictions of the choice elements. Notice that the exact choice cardinalities cannot be computed when a choice item is contained in a choice with unbounded maximum number of occurrences.

The unnamed classes that represent XML Schema sequences and choices are produced using the algorithms outlined above, that are available at [15]. It must be noted that, if the maximum number of occurrences of a sequence/choice has a large value (but is not unbounded), the manual generation of the restrictions is tedious and time-consuming and thus becomes error-prone and practically impossible.

References. XML Schema attributes, attribute groups, elements and model groups that are referenced in complex type definitions are transformed into OWL-DL datatype (if they are or contain attributes or simple type elements) or object (if they contain complex type elements) properties. Let *ref(ae)* be a reference, in a complex type *ct*, to the *ae* XML attribute or element. The reference is represented by the (datatype or object) property *rp(id, domain)*, where *id* is the rdf:ID of *rp* and has as value the value of the rdf:ID of the property that represents *ae*, and *domain* is the domain of *rp* and has the rdf:ID of the OWL class *c* that represents *ct* as value.

4. Mapping Ontologies

In section 3 we mentioned that some XML Schema semantics cannot be represented in OWL during the XML Schema to OWL transformation. These semantics do not affect the domain ontologies that may extend the main ontology and they are not used by the OWL reasoners; however, they are important when the individuals defined according to the main ontology have to be transformed back to valid XML descriptions compliant with the source XML Schema. In order to support this functionality, we have defined a model that allows transforming the OWL constructs back to XML Schema constructs. This model captures the XML Schema semantics that cannot be represented in OWL and is expressed as an OWL-DL ontology, the *OWL2XMLRules Ontology* (available at http://elikonas.ced.tuc.gr/ontologies/OWL2XMLRules/OWL2XMLRules). For a particular XML Schema that is transformed to OWL-DL, XS2OWL generates a *Mapping Ontology* that extends the OWL2XMLRules ontology with individuals and represents the semantics of the schema that are lost during the transformation to OWL.

In the following paragraphs, we present the classes of the OWL2XMLRules ontology as well as the model for the generation of individuals of the classes of the OWL2XMLRules ontology during the transformation of specific XML Schemas.

DatatypePropertyInfoType Class. It captures information about the datatype properties lost during the XML Schema to OWL transformation. This information includes the names of the XML constructs (elements, attributes) transformed to the datatype properties, the default values and the origins of the datatype properties, since an OWL datatype property may be the result of the transformation of an attribute, an element or it may state that a complex type extends a simple type.

Let *ae(name, ae_id, c_type, default)* be an attribute or a simple type element, where *name* is the name of *ae*, *ae_id* is the identifier of *ae*, *c_type* is the complex type in which *ae* has been defined and *default* is the default value of *ae*. *ae* is transformed into the *DatatypePropertyInfoType* individual *dpi(id, did, xml_name, dpi_type,* *def_val)*, where: (a) *id* is the unique rdf:ID of *dpi* and has *concatenate(ct_name, '_', name, '__', type)* as value, where *ct_name* is the name of the class that represents *c_type* in the main ontology; (b) *did* is the rdf:ID of the *dp* datatype property that represents *ae* in the main ontology; (c) *xml_name* is the name of *ae* and has *name* as value; (d) *dpi_type* represents the construct which has been mapped to *dp* and has the value *'Attribute'* if *ae* is an attribute and the value and *'Element'* if *ae* is an element; and (e) *def_val* represents the default value of *ae* and has *default* as value.

If a datatype property *dp* states that a complex type extends a simple type, a *DatatypePropertyInfoType* individual *dpi(id, did, dpi_type)* is generated for *dp*, where *id* and *did* have the semantics defined above and *dpi_type* has the 'Extension' value.

ElementInfoType Class. It captures information about the XML Schema elements that is lost during the XML Schema to OWL transformation. This information includes the names of the elements and, if they are parts of sequences, their ordering.

Let *e(eid, name, c_type, default, min, max, pos)* be an element, where *name* is the name of *e*, *eid* is the identifier of *e*, *c_type* is the complex type in which *e* has been defined, *default* is the default value of *e, min* is the minimum number of occurrences of *e*, *max* is the maximum number of occurrences of *e* and *pos* is the position of *e* if *e* is a sequence element. *e* is represented in the mapping ontology by the *ElementInfoType* individual *ei(id, pid, xml_name, def_val, min_occ, max_occ, position)*, where: (a) *id* is the unique rdf:ID of *ei* and has *concatenate(ct_name, '_', name, '__', type)* as value, where *ct_name* is the name of the class that represents *c_type* in the main ontology; (b) *pid* is the rdf:ID of the *p* property that represents *e* in the main ontology; (c) *xml_name* is the name of *e* and has *name* as value; (d) *dpi_type* represents the construct which has been transformed to *p* and has the *'Element'* value; (e) *def_val* represents the default value of *e* and has *default* as value; (f) *min_occ* represents the minimum number of occurrences of *e* and has *min* as value; (g) *max_occ* represents the maximum number of occurrences of *e* and has *max* as value; and (h) *position* represents the position of *e* if *e* is a sequence element.

ComplexTypeInfoType Class. It captures information lost during the XML Schema to OWL transformation about a complex type that has *name* as name. This information includes information about the datatype properties associated with the corresponding OWL class in the main ontology and the cardinality and ordering of the elements contained in the complex type.

Let *ct(name, ct_id, att_list, seq_list, cho_list)* be a complex type, where *name* is the name of *ct*, *ct_id* is the identifier of *ct*, *att_list* is the list of the *ct* attributes, *seq_list* is the list of the *ct* sequences and *cho_list* is the list of the *ct* choices. *ct* is represented in the mapping ontology by the *ComplexTypeInfoType* individual *ct(id, type_id, dpi_list, container_list)*, where: (a) *id* is the unique rdf:ID of *ct* and has *name* as value; (b) *type_id* represents the identifier of the OWL class *c* that represents *ct* in the main ontology; (c) *dpi_list* is the list of the representations of the datatype properties of *c*; and (d) *container_list* is the list of the representations of the *sc* containers.

ChoiceType and SequenceType Classes. They capture, respectively, information about the exact cardinalities and the structure of XML Schema choices and sequences that is lost during the XML Schema to OWL transformation.

Let *sc(sc_id, c_type, min, max, elements)* be a sequence or choice, where *sc_id* is the identifier of *sc*, *c_type* is the complex type in which *sc* has been defined, *min* is the minimum number of occurrences of *sc*, *max* is the maximum number of occurrences of *sc* and *elements* is the list of the elements of *sc*. We represent *sc* in the mapping ontology by the (*SequenceType* if *sc* is a sequence, *ChoiceType* if *sc* is a choice) individual *st(id, min_occ, max_occ, e_rep)*, where: (a) *id* is the unique rdf:ID of *st* and has *concatenate(ct_name, '__', i)* as value, where *ct_name* is the name of the class that represents *c_type* in the main ontology and *i* is the index of *sc* in *c_type*; (b) *min_occ* represents the minimum number of occurrences of *sc* and has *min* as value; (c) *max_occ* represents the maximum number of occurrences of *sc* and has *max* as value; and (d) *e_rep* is the list of the representations of the *elements* of *sc*.

As an example, consider the complex type "ct1", shown in Fig. 1. *ct1* is represented in the mapping ontology as shown in Fig. 10.

```
<ox:XSDComplexTypeInfoType rdf:ID="ct1">
 <ox:typeID>ct1</ox:typeID>
 <ox:DatatypePropertyInfo>
  <ox:DatatypePropertyInfoType rdf:ID="ct1_a1__ct1_a1_UNType">
   <ox:datatypePropertyID>a1__ct1_a1_UNType</ox:datatypePropertyID>
   <ox:XMLConstructID>a1</ox:XMLConstructID>
   <ox:datatypePropertyType>Attribute</ox:datatypePropertyType>
  </ox:DatatypePropertyInfoType>
 </ox:DatatypePropertyInfo>
 <ox:DatatypePropertyInfoType rdf:ID="ct1_content__xs_integer">
  <ox:datatypePropertyID>content__xs_integer</ox:datatypePropertyID>
  <ox:datatypePropertyType>Extension</ox:datatypePropertyType>
 </ox:DatatypePropertyInfoType>
</ox:XSDComplexTypeInfoType>
```

Fig. 10. Representation of the complex type "ct" of Fig. 1 in the mapping ontology

5. Realization and Evaluation of the XS2OWL Model

We present in this section the design and implementation of the XS2OWL system, which transforms automatically XML Schemas into OWL-DL ontologies and generates their mapping ontologies. According to the XS2OWL model, an XML Schema is transformed into: (a) A *main* OWL-DL ontology that directly captures the XML Schema semantics using OWL-DL constructs; (b) A *mapping* OWL-DL ontology that systematically captures the semantics of the XML Schema constructs that cannot be captured in the main ontology; and (c) A *datatypes* XML Schema containing the simple XML Schema datatypes defined in the source XML Schema, which are imported in the main ontology.

The XS2OWL transformation model has been implemented as an XSLT stylesheet. The information flow during the transformation is shown in Fig. 11. As shown in Fig. 11, the source XML Schema and the XS2OWL stylesheet are given as input to an XSLT processor, and the output comprises of the main ontology, the mapping ontology and the datatypes XML Schema.

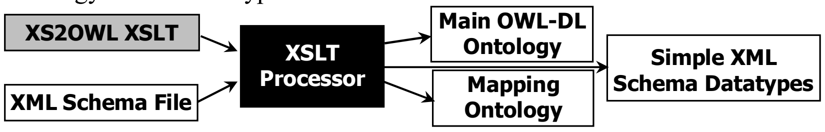

Fig. 11. The Information Flow in XS2OWL

In order to acquire extensive empirical evidence, we applied XS2OWL to several very large and well-accepted standards expressed in XML Schema: The MPEG-7 Multimedia Description Schemes (MDS) and the MPEG-21 Digital Item Adaptation (DIA) Architecture in the multimedia domain, the IEEE LOM and the SCORM in the e-learning domain and the METS standard for Digital Libraries. The XML Schema constructs of these standards have been automatically converted to OWL for each of those standards. We then produced individuals following the ontologies. Finally, we converted the individuals to XML syntax, valid with respect to the source XML Schemas. The transformations were successful for these standards and we found that in all cases the semantics of the standards were fully captured in the main and mapping ontologies generated by the XS2OWL system.

6. Conclusions

We have presented in this paper the XS2OWL formal model that allows to automatically transform XML Schemas into OWL-DL ontologies. This transformation allows domain ontologies in OWL to be integrated and logic-based reasoners to be used for various applications, as for example for knowledge extraction from multimedia data. XS2OWL allows the conversion of the generated OWL information back to XML. We have presented also the XS2OWL system that implements the XS2OWL model. We have used the implemented system to validate our approach with a number of well-accepted and extensive standards expressed in XML Schema. The automatically created ontologies have been found to accurately capture the semantics of the source XML Schemas.

Acknowledgments. The work presented here was partially funded in the scope of the DELOS II Network of Excellence in Digital Libraries (IST – Project Record #26059).

7. References

1. ADL Technical Team: Sharable Content Object Reference Model (SCORM), 2004.
2. Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., Yergeau, F., Cowan, J. (eds.): Extensible Markup Language (XML) 1.1. W3C Recommendation, 2006. (http://www.w3.org/TR/xml11/).
3. Brickley, D., Guha, R. V. (eds.): RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation, 2004. (http://www.w3.org/TR/rdf-schema).
4. 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.
5. Fallside, D., Walmsley, P. (eds.): XML Schema Part 0: Primer. W3C Recommendation, 2001. (http://www.w3.org/TR/xmlschema-0/).
6. García, R., Celma, O.: Semantic Integration and Retrieval of Multimedia Metadata. In the proceedings of the Semannot'05 Workshop, 2005.
7. Kay, M. (ed.) : XSL Transformations (XSLT) Version 2.0. W3C Recommendation, 2007. (http://www.w3.org/TR/xslt20/).
8. Manola, F., Milles, E. (eds.): RDF Primer. W3C Recommendation, 2004. (http://www.w3.org/TR/rdf-primer).

9. METS: Metadata Encoding and Transmission Standard (METS) Official Website. (http://www.loc.gov/standards/mets/).
10. IEEE LTSC 2002: IEEE 1484.12.1-2002 – Learning Object Metadata Standard. (http://ltsc.ieee.org/wg12/).
11. ISO/IEC: 21000-7:2004 – Information Technology – Multimedia Framework (MPEG-21) – Part 7: Digital Item Adaptation, 2004.
12. ISO/IEC: 15938-5:2003 – Information Technology –Multimedia content description interface – Part 5: Multimedia description schemes. First Edition, ISO/MPEG N5845, 2003.
13. McGuinness, D. L., van Harmelen, F. (eds.): OWL Web Ontology Language: Overview. W3C Recommendation, 2004. (http://www.w3.org/TR/owl-features).
14. Pereira, F.: The MPEG-21 standard: Why an open multimedia framework?. In the Proceedings of the 8th IDMS, LNCS 2158, Lancaster, September 2001, pp. 219–220.
15. Tsinaraki C. and Christodoulakis S. 2007. XS2OWL: A Formal Model and a System for enabling XML Schema Applications to interoperate with OWL-DL Domain Knowledge and Semantic Web Tools. Technical Report, http://www.music.tuc.gr/XS2OWL.pdf.
16. Tsinaraki C., Polydoros P. and Christodoulakis S.: Interoperability support for Ontology-based Video Retrieval Applications. In the Proceedings of the CIVR 2004, pp. 582-591.
17. Tsinaraki C., Polydoros P. and Christodoulakis S.: Integration of OWL ontologies in MPEG-7 and TVAnytime compliant Semantic Indexing. In Proc. of the Conference of Advanced Information Systems Engineering (CaiSE) 2004, pp. 398-413.
18. Tsinaraki, C., Polydoros, P., Kazasis, F., Christodoulakis S.: Ontology-based Semantic Indexing for MPEG-7 and TV-Anytime Audiovisual Content. In Multimedia Tools and Application Journal (MTAP), 26:299-325, 2005.