XSD To OWL: A Case Study

Aaron Wheeler, Jim Dike, and Michael Winburn
3 Sigma Research, Indialantic, Florida, USA

Abstract – *This paper addresses the challenge of applying Semantic Web technologies like OWL-DL reasoners to XML documents to find implied relationships not stated directly in the documents. The challenge comes because the structure of XML schema often contains implicit assumptions about taxonomy and relationships. This makes it difficult to implement a completely general and automated mechanism to transform XML schemas to OWL ontologies. We describe our efforts to transform XML schemas to OWL ontologies and discuss our rationale for particular mappings. This work contributes to this area of research by providing new interpretations of XSD terminology in the context of OWL and shows how to map XSD structures to more complex OWL structures.*

Keywords: knowledge representation, ontologies, semantics, OWL, XML

1 Introduction

In this paper we address the challenge of applying Semantic Web technologies like OWL-DL reasoners to XML documents to find implied relationships not stated directly in the documents. The challenge comes because the structure of XML schema often contains implicit assumptions about taxonomy and relationships [1]. This makes it difficult to implement a completely general and automated mechanism to transform XML schemas to OWL ontologies.

We describe our efforts to transform XML schemas to OWL ontologies and discuss our rationale for particular mappings. This work contributes to this area of research by providing new interpretations of XSD terminology in the context of OWL and shows how to map XSD structures to more complex OWL structures.

Section 2 of this paper provides background information about relevant structures in both the XML Schema language and in OWL. Section 3 describes examples of related work, our approach, and how our work differs from previous efforts. We also illustrate a how the proposed transformation fits in to a larger system architecture for automated reasoning about XML documents. Section 4 concludes with a brief description of a current project that we intend to apply these transformations, and also a discussion of future research directions.

2 Background

The Extensible Markup Language (XML) provides language for describing the structure of information to support automated processing [2]. The XML Schema Definition (XSD) language contains type and element definitions that describe characteristics of well-formed elements and attributes of XML documents [3]. However, the XSD language does not express semantics [1] and so creates difficulties for Semantic Web technologies.

An ontology provides a formal description of a domain of discourse [4] and represents a core component of the Semantic Web. The World Wide Web Consortium (W3C) has recommendations for expressing ontologies using RDF [5], RDFS [6], and OWL [9].

The Resource Description Framework (RDF) represents a W3C standard for the Semantic Web that makes statements using subject-predicate-object triples [5]. RDF Schema (RDFS) extends RDF by allowing for the definition of classes and properties that describe other classes and properties [6].

Both RDF and the OWL 2 Web Ontology Language provide a formal semantics for interpreting ontology structures. OWL extends RDF and RDFS with terminology to express ontologies using description logic, a decidable fragment of first order logic [7]. OWL DL ontologies belongs to subset of OWL ontologies that satisfy the expressive requirements of a description logic. The expressiveness, completeness, and decidability of OWL DL makes it possible for automated reasoning engines to discover new information implied by ontology structures. OWL 2 has terminology to express ontologies semantically equivalent to the SROIQ description logic [8].

2.1 XML Schema Definition (XSD) Language

The XSD language has a large vocabulary for describing the structure of valid XML documents. However, we do not need necessarily all of these descriptors to make semantic inferences about the documents. In this section we make note of those XSD entities that we choose to use for automated reasoning. See [3] for more details.

The xsd:attribute defines attributes of elements using build-in types like xsd:integer and xsd:string. The xsd:simpleType construct includes built-in types like string

and integer, as well as derived types. We can restrict values using xsd:enumeration, xsd:minInclusive, xsd:minExclusive, xsd:maxInclusive, and xsd:maxExclusive.

The xsd:complexType element may have any combination of attributes, elements, and content. Complex types with simple content have attributes and content, with complex content have attributes and elements, and with mixed content have attributes, elements, and content. We can extend the content of complex types or place restrictions on existing content.

The xsd:element declares elements used to create entities. The substitutionGroup attribute for elements indicates that all elements belonging to this group may replace one another.

Several elements describe collections of elements. The xsd:group element allows for easy reference of collections of elements, while xsd:attributeGroup does the same for collections of attributes. For defining a particular ordering of child nodes we use xsd:sequence, for an exclusive choice we use xsd:choice, and for no particular ordering we use xsd:all.

Lastly, the elements xsd:annotation and xsd:documentation provide human-readable descriptions of XSD constructs.

2.2 Web Ontology Language (OWL)

In OWL we declare sets of individuals with owl:Class. We relate individuals to data values with owl:DatatyeProperty and to other individuals with owl:ObjectProperty. We specify the equivalence of classes with owl:equivalentClass and superclass/subclass relationships with rdfs:subClassOf. OWL allows for set definitions through the existential (some) and universal (only) quantifiers owl:someValuesFrom and owl:allValuesFrom. We represent intersections and unions of set restrictions with, respectively, owl:intersectionOf and owl:unionOf. We restrict datatype property values in our set definitions using owl:oneOf for enumerations; and owl:minInclusive, owl:minExclusive, owl:maxInclusive, and owl:maxExclusive for numerical ranges [9].

3 XSD to OWL

In this section we introduce similar work done by others to transform XSD to OWL, and then describe our methodology in detail.

3.1 Related Work

Table 1 shows the mapping between XSD and RDF/OWL proposed by Ferdinand et al [10]. They also use an intersection of owl:allValuesFrom and a cardinality restriction (owl:minCardinality or owl:maxCardinality) to express local XSD entities having a type and cardinality. They do not handle complex types defined by simple type restrictions[10]. Bohr et al [11] have a similar mapping (Table 1).

Table 1: Comparison of XSD to OWL mappings.

XSD	Ferdinand et al [10]	Bohr et al [11]	Garcia et al [13]	3SR
@fixed				owl:oneOf
@mixed				owl:DatatypeProperty
@type			rdfs:range	rdfs:subClassOf; owl:subPropertyOf; rdfs:range
all	owl:intersectionOf	owl:intersectionOf		owl:intersectionOf
attribute	owl:DatatypeProperty	owl:DatatypeProperty	owl:DatatypeProperty; rdf:Property	owl:DatatypeProperty
attributeGroup	owl:Class		owl:Class	owl:Class; rdfs:subClassOf; owl:ObjectProperty
choice	owl:intersectionOf, owl:unionOf, owl:complementOf	owl:intersectionOf, owl:unionOf, owl:complementOf	owl:unionOf	owl:unionOf
complexType	owl:Class; owl:ObjectProperty	owl:Class	owl:Class	owl:Class; owl:ObjectProperty
element		owl:Class; owl:ObjectProperty	owl:DatatypeProperty; owl:ObjectProperty; rdf:Property	owl:Class; owl:ObjectProperty; owl:DatatypeProperty
enumeration				owl:oneOf
extension	rdfs:subClassOf		rdfs:subClassOf	rdfs:subClassOf
group	owl:Class	owl:DatatypeProperty	owl:Class	owl:Class; owl:ObjectProperty
maxOccurs		owl:maxCardinality	owl:maxCardinality	
minOccurs		owl:minCardinality	owl:minCardinality	
restriction	rdfs:subClassOf		rdfs:subClassOf	rdfs:subClassOf
sequence	owl:intersectionOf	owl:intersectionOf	owl:intersectionOf	owl:intersectionOf
simpleContent				owl:DatatypeProperty
simpleType	owl:DatatypeProperty	owl:DatatypeProperty		owl:DatatypeProperty
substitutionGroup	owl:subPropertyOf		rdfs:subPropertyOf	
see text				owl:equivalentClass
see text				owl:someValuesFrom

We need to preserve as much as possible the semantic relationships expressed in the XML schema when we convert the XSD to OWL. We found a tool that comes close to our needs contained in the XSD2OWL package in the ReDeFer Project from the Rhizomik Initiative led by researchers from the GRIHO (Human-Computer Interaction and Data Integration) research group at the University of Lleida in Spain [12]. Table 1 shows the XSD to OWL mappings used by ReDeFer. The transformations may produce an OWL-Full ontology because it creates an rdf:Property for elements with both datatype and object ranges [13].

The commercial product TopBraid Composer [14] has an Eclipse plugin called XsdImport that converts XSD to OWL. XsdImport converts complex and simple types, attributes, attribute groups, as well as anonymous and named

declarations to OWL classes and properties. However, XsdImport does not fully support OWL restrictions [15].

None of the efforts described above seem to provide us with sufficient support to create OWL representations of XML schema that we need for our particular automated reasoning applications. For example, we would like more automated support to create class restrictions from attributes and sub-elements for entities in the XSD. Our methodology, that we describe in the next section, for mapping XSD to OWL differs from others in at least two significant respects. Our methodology differs first in our handling of datatype properties and second in our use of restrictions for class definitions.

3.2 Methodology

In this section we describe our mapping strategy from XSD to OWL. The inspiration for our mapping strategy comes from the kinds of competency questions [16] we would like to answer with SPARQL queries over the inferred relationships generated by an OWL-DL reasoner.

Generally, if an XML document has particular a kind of element with some attribute value or some sub-element then we want to infer that this element belongs to some other class or has some particular property value or relation to some other specific entity or type of entity. Note that our strategy focuses on a transformation that allows for automated reasoning about the semantics of the XML documents and not on an equivalent representation of all document structure described in the schema. This means we can select from the schema only those aspects that we need for reasoning and leave out those related to validation. We assume that we reason only on valid XML documents.

Essentially, the entity in an XML document represents an individual. Attributes become datatype property relations and sub-elements become object property relations. Attributes may have a fixed value or enumerated values.

Table 1 shows our proposed mapping from XSD to RDF/OWL. The following sections provide details about particular decisions we made for the mappings.

owl:DatatypeProperty

We create DatatypeProperty definitions for element content and for attributes. Elements and complex types with mixed content also have a datatype property relation to the content. When a complex type allows for content, we create an owl:DatatypeProperty for this content. Complex types and elements with only simple content also receive owl:DatatypeProperty definitions.

owl:ObjectProperty

The XSD entities attributeGroup, group, complexType, and element that produce OWL class definitions also generate OWL object property definitions.

owl:Class

We create OWL classes for the XSD entities attributeGroup, group, complexType, and element because all of these could contain attributes and/or sub-elements. An attributeGroup represents a class with restrictions on the OWL datatype properties for each of the attributes in the group. We transform the group entity similarly, but using the OWL object property.

Complex type and element entities with only name and type attributes we could interpret as datatype properties. Instead, for consistent treatment of complex types and elements in general, we define these as classes having a datatype property relation to the type.

We model xsd:complexType in OWL as owl:Class defined with an owl:equivalentClass expressed as an owl:intersectionOf restrictions on the XSD sub-components listed in the complex type declaration.

In one respect that our approach differs from that implemented by ReDefer [12] lies in the interpretation of the XSD element. ReDeFer maps element to owl:DatatypeProperty, owl:ObjectProperty, or rdf:Property depending on the details of the element description. We interpret the XSD element as an owl:Class with an owl:DatatypeProperty relation to its content (if any).

owl:equivalentClass

We use an equivalent class of the intersection of restrictions on elements and attributes contained in the entity. The XSD terms all and sequence generate an intersection of all entities they contain. The any and choice entity produces an union of its component entities.

Attributes and sub-elements in an equivalent class definition allows us to infer that individuals of any class having these property relations also belong to this class.

owl:someValuesFrom

This restriction applies to enumerated datatype properties and object properties. We chose to express these properties with existential rather than universal restrictions because universal restrictions prevent a reasoner from inferring class membership of other classes and individuals not also having the universal restriction. This applies in particular to individuals of unknown type but with certain known properties.

Elements with attributes and content have relations to attribute data and content data, so in OWL they become classes defined as an equivalent class to the intersection of datatype properties for the attributes and content. Elements that can contain other elements have object property restrictions to these other elements.

We want to make explicit the relationships between classes and their property relations to other objects and datatype values. We do this with an owl:equivalentClass using an owl:intersectionOf of all owl:Restriction declarations created from the attributes and elements in the class.

rdfs:subClassOf and owl:subPropertyOf

Both the base and type attributes result in subclass and subproperty statements in the OWL representation.

xsd:choice

Other approaches to mapping XSD to OWL kept the exclusive-or interpretation of the XSD choice entity. We believe this creates combinatorial issues should a type have many choice elements with many alternatives. We choose instead to use a logical union represented by owl:unionOf on all the sub-entities. This interpretation of xsd:choice makes explicit in OWL the possibilities so that later one might add explicitly more specific exclusive-or restrictions for some combinations.

3.3 System Architecture

Figure 1 shows a system architecture for ontological reasoning about XML document semantics. One or more data sources provide data components that get collected into an XML document that validates to an XML schema(s). This XML schema we have previous transformed to an OWL ontology using the transformation rules described in the preceding section. We transform the XML document to an OWL individual and load it into a reasoning engine, such as Pellet [17], FaCT++ [18], or HermiT [19]. The reasoning engine generates an ontology model that contains the inferred relationships. We apply SPARQL [20] queries to this model to retrieve facts explicit in or implied by the data. These facts then become components of new data products.

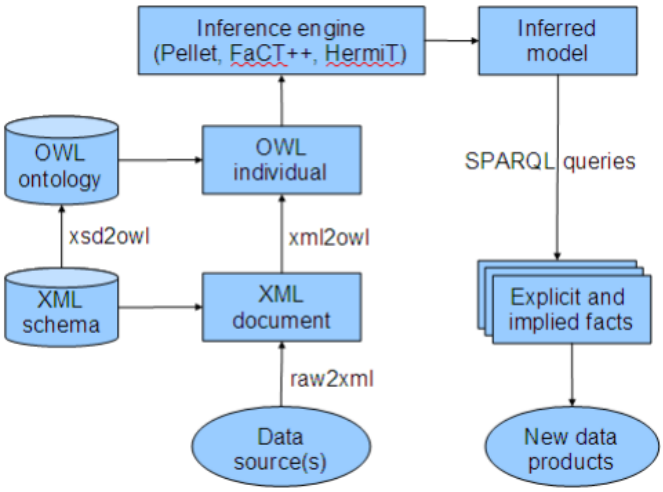

Figure 1. System architecture for ontological reasoning about document semantics.

4 Conclusions

We have shown a way to interpret key components of the XSD language in a way that allows their semantic representation as an OWL ontology. Our strategy for mapping XSD to OWL seems to offer to us better handling of datatype values, element content, and more elaborate restrictions on class definitions.

The representation of XML schema as ontologies makes it possible to apply automated reasoning engines infer additional relationships implied by the XML structures expressed in the schema.

We will continue to develop and refine the mapping in several ways. For example, we\do not currently create owl:minCardinality or owl:maxCardinality restrictions on property relations. However, we expect that a need for inference on cardinality will arise.

5 References

[1] OWL Web Ontology Language Overview. W3C Recommendation 10 February 2004. http://www.w3.org/TR/2004/REC-owl-features-20040210/

[2] Extensible Markup Language (XML) 1.1 (Second Edition). W3C Recommendation 16 August 2006, edited in place 29 September 2006. http://www.w3.org/TR/2006/REC-xml11-20060816/

[3] XML Schema Part 0: Primer Second Edition. W3C Recommendation 28 October 2004. http://www.w3.org/TR/xmlschema-0/

[4] T.R. Gruber. A translation approach to portable ontology specifications. Knowledge Acquisition 5(2):199–220, 1993. http://tomgruber.org/writing/ontolingua-kaj-1993.pdf

[5] RDF Primer. W3C Recommendation 10 February 2004. http://www.w3.org/TR/2004/REC-rdf-primer-20040210/

[6] RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation 10 February 2004. http://www.w3.org/TR/rdf-schema/
[7] From SHIQ and RDF to OWL: The Making of a Web Ontology Language by Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. Journal of Web Semantics, 1(1):7-26, 2003. http://www.comlab.ox.ac.uk/people/ian.horrocks/Publications/download/2003/HoPH03a.pdf
[8] OWL 2 Web Ontology Language Document Overview. W3C Recommendation 27 October 2009. http://www.w3.org/TR/owl2-overview/
[9] OWL 2 Web Ontology Language Primer. W3C Recommendation 27 October 2009. http://www.w3.org/TR/owl2-primer/
§[10] Matthias Ferdinand, Christian Zirpins, D. Trastour. Lifting XML Schema to OWL. Web Engineering - 4th International Conference, ICWE 2004, Munich, Germany, July 26-30, 2004. http://vsis-www.informatik.uni-hamburg.de/getDoc.php/publications/204/fzt-lxs-04.pdf
[11] Hannes Bohring and Soren Auer. Mapping XML to OWL Ontologies. Leipziger Informatik-Tage, volume 72, 2005. pp. 147-156. http://www.informatik.uni-leipzig.de/~auer/publication/xml2owl.pdf
[12] ReDeFer. http://www.rhizomik.net/html/redefer/
[13] Roberto Garcia (2007). A Semantic Web Approach to Digital Rights Management. Dissertation. Department of Technologies, Universitat Pompeu Fabra, Barcelona, Spain. http://www.rhizomik.net/html/~roberto/thesis/
[14] TopBraid Composer. http://www.topquadrant.com/products/TB_Composer.html
[15] XsdImport – Convert XSD schemas to OWL. http://www.incunabulum.de/projects/it/xsdimport
[16] N.F. Noy and D.L. McGuinness. Ontology Development 101: A Guide to Creating Your First Ontology. http://protege.stanford.edu/publications/ontology_development/ontology101-noy-mcguinness.html
[17] Pellet: OWL 2 Reasoner for Java. http://clarkparsia.com/pellet
[18] FaCT++. http://owl.man.ac.uk/factplusplus/
[19] Hermit OWL Reasoner. http://hermit-reasoner.com/
[20] SPARQL Query Language for RDF. W3C Recommendation 15 January 2008. http://www.w3.org/TR/rdf-sparql-query/