Round-tripping between XML and RDF[*]

Steve Battle
Hewlett-Packard Labs,
Bristol, UK
steve.battle@hp.com

Abstract

This paper addresses the issue of lifting XML data into a common representational framework, namely RDF. The context of this work is the modeling of HP web-services, including message content exchanged with the provider agent. This is not Yet Another Serialization Syntax for RDF, but an approach to using RDF in a way that is complementary with XML.

1 Introduction

XML continues to be the primary data format for e-commerce. To gain independence from any particular serialization we prefer to model the data with RDF. While it is straightforward to describe ad-hoc XSLT transformations from XML into RDF/XML, the reverse mapping is problematic because there is no canonical RDF/XML serialization.
The round-tripping tool described in this paper allows us to directly interpret an XML document with an RDF model – i.e. not passing indirectly through RDF/XML. This is the so-called problem of *lift*, where different, possibly heterogeneous, data sources are mapped into a common representational framework for ease of processing. This complements the task of mapping the XML schema to OWL[1]. More than that, it allows us to take suitably constructed RDF models and output them directly as XML. We use XML schema as the basis for describing how XML is mapped into RDF and back.
The trajectory of this process is for incoming messages to be lifted into RDF. Conversely, new outgoing messages are created in RDF through the application of rules. This outgoing message content is lowered into XML prior to transmission. The benefits of this approach over working directly with XSLT is that XML schema are declarative, supporting the round-trip, whereas the form of the XSLT transform depends on the direction of the mapping.
The challenge is to name the schema components (with URIs) so as to preserve the local scope of elements and attributes within their immediately enclosing type.

[*] Work carried out as part of EU funded Semantic Web enabled Web Services (SWWS IST-2002-37134).

2 Simple and complex type mappings

The central idea is that every element and attribute name maps to an RDF property[2], viewing the XML structure as a relational model between parent nodes and their children. Alternative approaches have been proposed; Melnik[3] in particular has suggested using the element names to classify the content of the element, and only attributes would be identified with RDF properties. Indeed, this is quite often the intended meaning of the XML. Melnik is interested in a generic mapping between XML and RDF whereas we assume that an XML schema is available to guide us.
OWL distinguishes between object and data-type properties. Our first intuition may be that this distinction depends upon whether they are derived from an attribute or element. However, literal content is found in both elements and attributes. XML schema describes them in terms of simple and complex types. Simple data-types are mapped onto a relation between a resource and a typed literal. Untyped simple types are interpreted as plain literals. For derived simple-types the property definition uses only to the base-type from which it is derived. The content of enumerations can be preserved as property restrictions.
XML schema ComplexTypes are described using *sequence*, *all* and *choice* compositors. For each complex type we create a new resource that represents the class and use it to define the *rdf:type* of the product information. Complex types may also support mixed content models with interspersed text. In this case the *rdf:value* property is used where a singular literal won't do.
XML is not strictly a tree, but a tree with pointers as described by IDs and IDREFs. Rather than preserve the IDREF datatype, we interpret it as a URI defined relative to the document base. IDREFs are an exception where a simple type does not refer to a literal value.
We preserve the partitioning of the schema namespace by appending a schema component designator to the schema base. Like XPaths, these designators can identify the scope of an element or attribute definition even where the enclosing type is unnamed. Attributes are distinguished from elements by their preceding '@'. We also introduce a '~' prefix for global named types. These features can be seen in the namespace declarations of figures 1 and 2.

3 Sequence

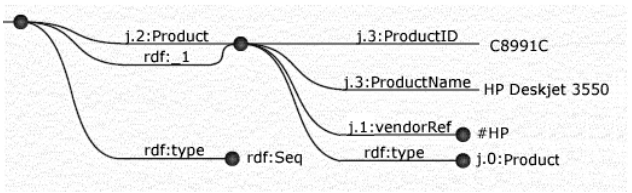
A naive translation of XML into RDF has the side effect of losing the sequencing implicit in the XML. Patel-Schneider et al[4,5] make up for this deficit at a semantic level so that XML may be read directly as RDF. Our approach is more modest, using existing RDF constructs in conjunction with a defined mapping. We note that even in XML, sequencing is often redundant. XML schema is the cue for when sequence is important, with its *sequence* compositor. 'Lift' is a function of both the XML input and its schema.
There are many approaches we could take to sequencing. RDF lists, in particular empty lists with properties, are problematic. We adopt a simple solution based on RDF containers. RDF sequences get a bad press but provide a low overhead solution (few added triples). We classify the container as an *rdf:Seq*.
A problem we encounter is how to describe this in OWL. In particular, container membership properties fall foul of the need to discriminate between Object and Datatype properties in OWL DL. In fact we face the same problems whether we select containers or collections. We treat sequencing as a data structuring issue rather than as ontological. Our approach allows the relational model and ordering to be separated without losing connectivity. The ontology, containing only the relational structure remains coherent.

4 XML to RDF

The XML fragment below is based on real messages exchanged with a web-service for online purchasing. *Product* is a relationship between the outer context and a complex typed product node, and is defined to be in sequence (alphabetical order). *ProductID* and *ProductName* are relationships between this product node and string typed data. The *vendorRef* is a reference (IDREF) to another part of the document where vendor information is collated.

```
<Product vendorRef="HP">
  <ProductID>C8991C</ProductID>
  <ProductName>HP Deskjet 3550
  </ProductName>
</Product>
```

The figure below illustrates the underlying tree structure of the XML in graphical form[6]. Namespace prefixes preserve the distinction between complex-types, attributes, and elements; defined globally and locally.



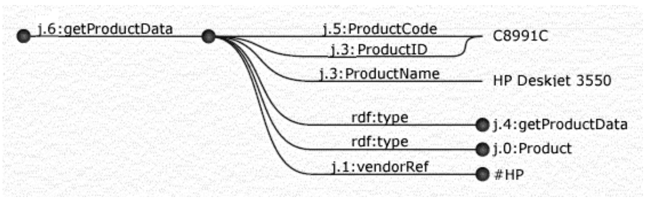
j.0= http://example.com/productList.xsd/~
j.1= http://example.com/productList.xsd/~Product/@
j.2= http://example.com/productList.xsd/Product_List/Products/
j.3= http://example.com/productList.xsd/~Product/

Fig. 1: simple and complex type properties

5 RDF to XML

The aim here is to construct an XML document by manipulating the RDF model. We will create a follow-up product information request. The schema for this output message is simple, comprising a *getProductData* element containing a single *ProductCode*.

Little work is needed to generate a model for this message as we already have one in figure 1; in the node of type Product. We assert OWL equivalences to define that a *ProductCode* is (property) equivalent to a *ProductID*, allowing us to use a different property name. Similarly we establish (class) equivalence between the existing *Product* and the required *getProductData* message. Finally we insert a *getProductData* relationship representing the new document element.



j.0 to j.3 as in fig.1
j.4= http://example.com/getProductData.xsd/~
j.5= http://example.com/getProductData.xsd/~getProductData/
j.6= http://example.com/getProductData.xsd/

Fig. 2: results of alignment

The product information is now a mixture of the old and the new. The XML production process is seeded with a starting node, the root node of fig. 2 above. Given a schema, it tries to match the available content to the schema. Because it is selective, it ignores the older information, using only information in the j.3 and j.4 namespaces, emitting the following XML.

```
<getProductData>
  <ProductCode>C8991C</ProductCode>
</getProductData>
```

References

[1] Klein, M., Fensel, D., von Harmelen, F., Horrocks, I., The relation between ontologies and XML schemas, http://www.cs.man.ac.uk/~horrocks/Publications/download/2001/etai01.pdf
[2] Trastour, D., Ferdinand, M. Zirpins, C., Pragmatic Reasoning-Support for Web-Engineering: Lifting XML-Schema to OWL, ICWE 2004, Munich, Germany.
[3] Melnik, S, Bridging the Gap between RDF and XML, http://www-db.stanford.edu/~melnik/rdf/fusion.html
[4] Patel-Schneider, P. Simeon, J., Building the Semantic Web on XML, ISCW2002.
[5] The Yin/Yang Web: A Unified Model for XML Syntax and RDF semantics, IEEE Transactions on Knowledge and Data Engineering, 15(3), July/Aug 2003, pp797-812.
[6] Sayers, C., Node-centric RDF Graph Visualization, http://www.hpl.hp.com/techreports/2004/HPL-2004-60.html