Interpreting XML via an RDF Schema

Michel Klein[1]

Abstract. One of the major problems in the realization of the vision of the "Semantic Web" is the transformation of existing web data into sources that can be processed and used by machines. This paper presents a procedure that can be used to turn XML documents into knowledge structures, by interpreting them as RDF data via an RDF-Schema specification. This allows semantic annotation of XML documents via external RDF-Schema specifications. This procedure could potentially multiply the availability of semantically annotated data.

1 Introduction

In the vision of the "Semantic Web", computers will be able to use the data on the web not just for display purposes, but for automation, integration and reuse of data across various applications[2]. This will enable several new types of applications, ranging from advanced information retrieval systems, to e-business integration, to automatically configured web services. In general, this requires two things. First, the data should somehow be *structured*, in order to allow machines to distinguish and identify pieces of data. Second, it is necessary that these pieces of data are described in such a way that the *meaning* of it can be exploited by machines. This meaning is often described by relating data to general concepts, which are, in their turn, characterized by their properties and relations to other concepts. These specifications of relevant concepts and their relations are often called "ontologies".[3]

When we look at the current status of the Web, we can make several observations. First, there is a lot of data that is only slightly structured. One could thing of the many HTML pages that contain information which is only structured for rendering purposes. In such pages, pieces of text are structured by labelling them with their role in the document, for example *title*, *paragraph* or *list item*. This often implies that the structure has a very coarse granularity. A second observation is that XML is becoming very popular. XML is a general mechanism that can be used to add user-defined tags to textual data. There are already a few web-sites represented in XML and formatted via an XSL stylesheet[4]. Also, what is more significant, XML is being used by industry. Many companies publish product information in XML or use XML for the communication in their business processes. All together, this gives us a vast amount of quite well structured data. As a third observation, we see that — as a result of the enthusiasm about the idea of a Semantic Web — the interest in the use and development of ontologies is increasing. The concept of ontology is becoming clear to more and more people and tool support is growing, too. Ontology development is slowly evolving from an expert-task to a task that can be performed by arbitrary web publishers. However, our fourth observation gives a less positive perspective: although there are already a number of ontologies, there is not yet much data on the web that is related to it. RDF Schema [3] and DAML+OIL [4] — examples of languages to specify ontologies on the internet — rely on the data model of the Resource Description Framework (RDF) for the representation of instance data. RDF data is not yet abundantly available, and if available, it is often very coarse-grained, for example, describing complete web pages instead of specific pieces of data on it.

Based on these observations, we can conclude that there is an important link missing: the connection between the existing (structured) data on the web and the domain knowledge as represented in ontologies. In this paper, we will present a procedure to interpret general XML documents as RDF data via an RDF Schema, i.e. an ontology. This allows people to describe the content of structured documents via an ontology. Machines can than interpret the XML document unambiguously as a set of statements in the RDF data model.

The rest of this paper is organized as follows. In the next section, we will explain the goal and benefits of our procedure. Section 3 will explain the procedure itself and illustrate it with an example. Section 4 discusses some open issues and compares our approach to other approaches that relate XML with RDF. Finally, section 5 concludes the paper.

2 Using an ontology to interpret XML

The main purpose of XML is to provide a mechanism that can be used to mark-up and structure documents. This allows machines to identify pieces of data in a document by their label. For example, an XML document might contain the markup `<price currency="EUR">45,38</price>`. This string assigns a label "price" with the text "45,38" and a label "currency" with the text "EUR". However, these labels themselves don't bear any meaning with them.

Connecting these mark-up labels with meaning is thus important when the data should be used on the "Semantic Web". A common misconception is that XML-Schema [7] documents can be used to add meaning to XML documents. As XML is mainly a structuring mechanism, the goal of XML Schema is to provide structure prescriptions for XML documents. That is, XML Schema can be used to specify restrictions of what can be in between the labels in XML documents, e.g. they can specify that the content of a "price" label should be a rational number. It cannot be used to specify the meaning of tags, despite the feature to build hierarchies of *element types*. This hierarchy doesn't contain conceptual knowledge, but only functions as a syntactic shortcut to allow reuse of complex definitions. A more elaborate comparison of XML Schema with conceptual hierarchies

[1] Vrije Universiteit Amsterdam, De Boelelaan 1081a, 1081 HV Amsterdam, the Netherlands, Michel.Klein@cs.vu.nl

[2] As explained at http://www.w3.org/2001/sw/

[3] Note that we use the word "meaning" here in an informal way: we refer to the interpretation of data, not to the formal semantics of the ontology.

[4] See http://www.w3.org/Style/XSL/.

can be found in [8].

To associate some meaning with XML documents, it is necessary to relate the labels with something that carries meaning. Concepts and properties in ontologies can be used for that, because ontologies formally specify the understanding of concepts in a particular domain. The naive way of relating ontologies with XML documents is to match the labels in an XML document syntactically with the names of concepts and properties in the ontology that is associated with it. However, this only solves part of the problem, because it is not clear what role the data in the XML document fulfils. For example, if we have a concept "price" defined in an ontology and we consider again `<price currency="EUR">45,38</price>`, it is not yet clear whether "price" is the data type of "45,38", or that the *value* of price is "45,38". The interpretation of the meaning of nested labels is even more complicated. In [6, chapter 6] is shown that there are several possible interpretations of a nesting relation in an XML document, for example, a concept-instance relation, an aggregation, or an object-attribute relation.

For reliable use on the Semantic Web, it is necessary that the data can be unambiguously interpreted. The RDF data model is used to provide such an unambiguous interpretation of data. This data model is quite simple: basically, it consist of statements of the form `subject, predicate, object`, where statements can be linked by using the object of one statement as the subject of another. The things that are being described (as subject of statements) are called *resources*. The object of a statement can either be another resource or a literal string value. The RDF data model makes it possible to distinguish between the different interpretations that we showed in the previous paragraph.

subject	predicate	object
`<price>45,38</price>`	rdf:type	Price
Price	value	"45,38"

Our goal is to use a domain ontology to associate meaning with an ambiguous XML document, so that it results in an unambiguous set of RDF triples. Therefore, we need an algorithm that determines how the ontology specifies an interpretation of XML labels and their data. Once we have such an algorithm, we can specify the *interpretation* of XML data unambiguously by an ontology. This will allow us to describe the meaning of XML data remotely, resulting in data that is useable on the Semantic Web. This procedure to annotate syntactic data doesn't require any change or addition to XML or RDF itself. Therefore, it can be used immediately.

3 Specification of the procedure

The procedure that we propose uses an ontology that is represented in RDF Schema. The ontology specifies which concepts are relevant, how they are related, and which properties they might have. The baseline of our procedure is that we use the ontology to specify *which* labels in the XML document are interesting, and what role they have, i.e., whether they specify a property or a class. This implies that we do not convert every syntactic structure into an statement in the RDF data model. This is intended: our experiences show that many syntactic structures do not have much meaning from a human point of view.

The nesting relation between two elements in an XML document is not automatically interpreted as a named property between resources. Instead, the ontology decides whether a label is interpreted as the name of a class, or as the name of a property. Based on this, our procedure specifies which RDF statements should be created. Our algorithm has to take care of a few special cases. One of them is the situation when two directly connected labels in XML are both identified as classes. In this case, a new property has*Classname* is defined to connect the two in the RDF data. This means that our default interpretation of nesting relation is the "object-attribute" relation [6]. Another special case is the treatment of XML *attributes*. Our procedure treats them in exactly the same way as sub-elements: this is in agreement with the data model of XML, which is a ordered tree in which attributes and subelements are equivalent (except for the order: this is not important for attributes) [2]. Finally, our procedure ignores comments.

We have introduced two RDF properties in our procedure. The first one is rdfx:describes, which is used to connect the resource of the XML document itself (its URI) to the classes that are used to describe its own properties. This property is not required when the top-level elements are identified as properties instead of classes. For example, if there is an element `<date>01-01-2002</date>` that tells something about the document, we create a statement that connects document and date directly:

subject	predicate	object
document_URI	date	"01-01-2002"

The benefit of having a specific property in this case is that we can define a fixed meaning for it: the object of a statement with rdfx:describes as a predicate tells something about the subject. Applications can exploit this fixed meaning, e.g. for retrieval. Dynamically generated properties (the has*Classname* properties explained above) do not have a fixed meaning; therefore, they cannot be exploited directly by general applications.

The second property that we introduced is rdfx:value. This one is used to connect a literal string value to an element that is matched to a class.

3.1 Detailed description

The procedure is described in detail below. To make clear which classes and properties are defined in the ontology that describes the document, we prepend these names with the semi namespace-qualifier "onto:". We prepend the two properties that we introduced with "rdfx".

1. make the document itself the "active resource", i.e., the URI of the document will be used as subject for the first statement;
2. traverse all elements (including attributes) of the XML document tree in a depth-first order, starting form the root element;
3. for each element, look in the describing ontology for a property or class definition with a syntactically equal name; if it is not found, continue with the next element;
4. when the element is associated with a property Prop
   (a) if there is no "unfinished statement" (a statement of which only two of the three elements are filled in), create the first part of a triple (i.e. an unfinished statement):

subject	predicate	object
"active_resource"	onto:Prop	...

   (b) else, if there is an unfinished statement, finish it with a new anonymous resource and create a new unfinished statement:

subject	predicate	object
...	...	*anon_1*
anon_1	onto:Prop	...

5. if the element is associated with a class Cls

(a) if there is an unfinished statement, finish it and create the following new triple:

subject	predicate	object
...	...	*anon_1*
anon_1	rdf:type	onto:Cls

and make *anon_1* the new "active resource".

(b) else, if the document itself is the active resource, create the following triples:

subject	predicate	object
URI_of_document	rdfx:describes	*anon_1*
anon_1	rdf:type	onto:Cls

and make *anon_1* the new "active resource".

(c) else, define a property has*Cls* (based on the name of the class) and create the following triples:

subject	predicate	object
active_resource	onto:hasCls	*anon_1*
anon_1	rdf:type	onto:Cls

and make *anon_1* the new "active resource".

6. if the element has textual content (called PCDATA or CDATA in XML)

(a) if there is an unfinished statement, finish it by giving it the content as *literal* value:

subject	predicate	object
...	...	*"textual_content"*

(b) else, add the following statement:

subject	predicate	object
active_resource	rdfx:value	*"textual_content"*

7. continue traversing the XML tree; when backtracking behind an element that has matched a class, make the previous active resource the active resource.

3.2 Example of use

To show how this procedure can be used to transform XML data into RDF statements, we will now give an example. We will describe the data in a piece of XML by an ontology and show how our procedure transforms this in a set of useful RDF statements. We use a part of a cXML purchase order. cXML[5] is an XML langauge for the communication of data related to electronic commerce in B2C situations.

A sample cXML purchase order might look like this:

```
<cXML>
  <IndexItem>
    <ItemDetail>
      <UnitPrice>
        <Money currency="USD">1000</Money>
      </UnitPrice>
      <Description xml:lang="en">
        Armada M700 PIII 500 12GB
      </Description>
      <UnitOfMeasure>EA</UnitOfMeasure>
      <Classification domain="SPSC">
        C43171801
      </Classification>
      <ManufacturerPartID>
        140141-002
      </ManufacturerPartID>
      <ManufacturerName>Florsheim</ManufacturerName>
      <URL>http://www.compaq.com</URL>
    </ItemDetail>
    <IndexItemDetail>
      <LeadTime>10</LeadTime>
      <ExpirationDate>2000-06-01</ExpirationDate>
      <EffectiveDate>2000-01-01</EffectiveDate>
      <SearchGroupData>
        <Name xml:lang="en">Notebook</Name>
        <SearchDataElement name="Processor Speed"
           value="500MHZ"/>
      </SearchGroupData>
      <TerritoryAvailable>USA</TerritoryAvailable>
    </IndexItemDetail>
  </IndexItem>
</cXML>
```

The ontology in which we described our interpretation of the meaning of this data is given below (for reasons of space, we do not use the RDF Schema syntax in the example, but use a trivial textual syntax instead). This ontology describes the most important concepts and properties in an intuitive way, using the tag names that are used in the XML document.

```
Class ItemDetail
Class Money
Class Classification
Property UnitPrice
Property currency
 - rdfs:domain Money
Property description
Property domain
 - rdfs:domain Classification
```

When we now use this ontology to apply our procedure to the sample XML document, we will get the set of RDF triples that is shown below. A graph-representation of these triples is shown in Figure 1. In both representations, we assume that the document is identified with the URI "http://mybus.com/order#1223".

subject	predicate	object
...#1223	rdfx:describes	*anon_1*
anon_1	rdf:type	onto:ItemDetail
anon_1	onto:UnitPrice	*anon_2*
anon_2	rdf:type	onto:Money
anon_2	onto:currency	"USD"
anon_2	rdfx:value	"1000"
anon_1	onto:Description	"M700 PIII 500 12GB"
anon_1	onto:hasClasssification	*anon_3*
anon_3	rdf:type	onto:Classsification
anon_3	onto:domain	"SPSC"
anon_3	rdfx:value	"C43171801"

The resulting RDF statements represents meaningful information that can be accessed via the ontology. This accessibility can be further improved by extending the ontology. For example, to make clear that what is called "ItemDetail" in the example should be considered as a "Product", one could state in the ontology that "ItemDetail" is a subclass of "Product".

The procedure can also help to harmonize differently structured information. For example, in xCBL[6], price information is encoded as follows:

```
<ProductPrice>
  <Amount>1000</Amount>
  <Currency>
    <CurrencyCoded>USD<CurrencyCoded>
  </Currency>
</ProductPrice>
```

[5] http://www.cxml.org/

[6] XML Common Business Library. Another set of specifications of XML business document and their components, but is targeted at B2B e-commerce. See http://www.xcbl.org/.

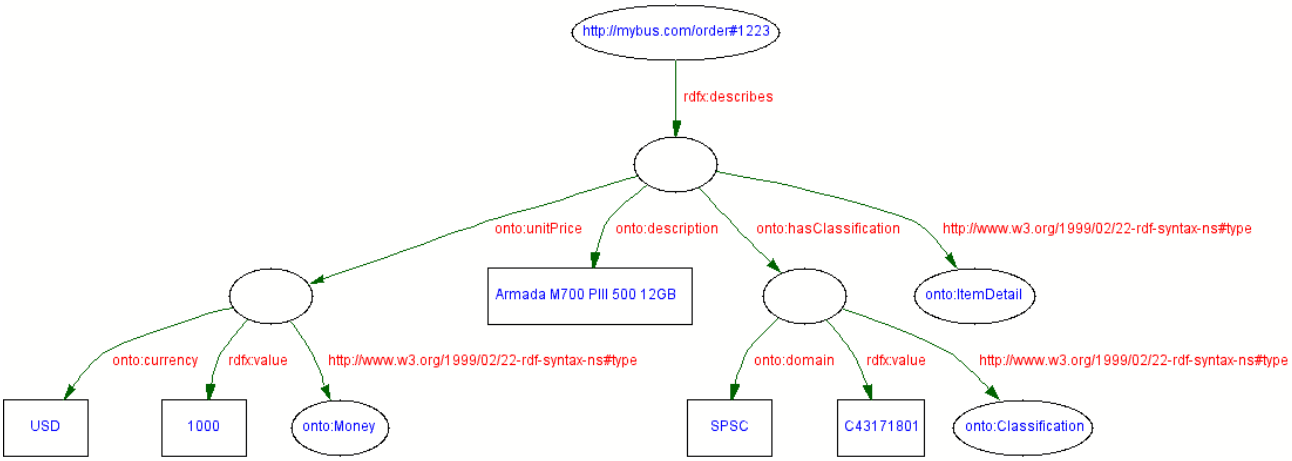

Figure 1. The graph representation of the RDF statements that resulted from the procedure

This is structurally different from the cXML encoding, although it represents the same information. If we apply our procedure to this piece of XML with an "ontology" consisting of a class ProductPrice and the two properties amount and currencyCoded, it results in the following triple-set:

subject	predicate	object
anon_1	rdf:type	onto:ProductPrice
anon_1	onto:amount	"1000"
anon_1	onto:currencyCoded	"USD"

This triple-set is completely equivalent with the corresponding triples from the cXML encoding. The only difference is the name of the properties and classes, but this can easily be solved by defining equivalent properties and classes.[7] Thus, when the meaning of different XML representations is specified via associated ontologies, information can be integrated via mapping ontologies. The normalization is already done by the procedure that interprets the documents via the ontology. Of course, the creation and specification of mapping ontologies far from trivial. However, a discussion if this is beyond the scope of this paper.

4 Discussion

In this section, we will discuss some of the choices we made in the procedure, and compare our approach with other approaches that relate ontologies and XML, or XML and RDF.

First, in our procedure, we deliberately chose to transform only those elements into RDF statements that match with a construct (class or property) in the ontology. This results in a smaller set of triples than when all structural constructs are transformed into statements. We made this choice because we think that it depends on the *use* whether structural constructs are meaningful enough to be retained. Moreover, this also allows us to steer the transformation of the data. An example is the transformation of "ProductPrice" in the previous paragraph. The triples would not have been similar to the cXML result if we were not able to ignore the `<Currency>` tag.

Second, there is still an important problem left: how to transform knowledge that is entirely encoded in the *content* of elements or attributes. An example of this type of knowledge is shown in the depicted cXML purchase order:

```
<SearchDataElement name="Processor Speed"
   value="500MHZ"/>
```

This element has two attributes of which the *values* represent the name and value of the property. Ideally, this would be transformed in a statement with a predicate Processor_Speed and object "500MHZ". To specify this, we could use "PropertyIdentifier" and "ValueIdentifier" pairs in our ontology. These pairs then specify which textual content should be interpreted as name of a property, and which content as the value of that property. However, this would break the feature that we use a pure domain ontology for the interpretation of data. Things like "PropertyIdentifier" and "ValueIdentifier" specify information about the *representation*, not about the domain.

A third point to discuss is the choice for RDF Schema, Of course, other ontology languages than RDF Schema could be used to describe the meaning of the data, too. However, we took RDF Schema as an example because it is currently the most prominent ontology language on the web. Also, there is already some tool support for it available.[8] Even this basic ontology language is already powerful enough to be used for our purpose.

Our approach is notably different from other approaches that relate XML and RDF. Generally speaking, there are two approaches. The first provides a new XML syntax for RDF data, preferable simpler than the current one.[9]. This approach uses XML to define a language to represent the triples. An example is the "strawman syntax" of Tim Berners-Lee[10], or Jonathan Borden's syntax[11]. It should be clear that our procedure doesn't provide an XML syntax for triples, but uses an ontology to extract RDF statements from XML documents.

A second approach comes closer to our method. This approach "harvests" RDF triples from general XML documents. For example, Sergey Melnik has described such a procedure.[12]. However, these procedures transform every element into RDF statements, and do not exploit ontological information about the domain. This thus results

[7] In RDFS, however, equivalent statements are not supported.

[8] See for example `http://protege.stanford.edu` and `http://img.cs.man.ac.uk/oil/`.

[9] Note that we didn't show the official RDF syntax in this paper; we only showed tables with statements

[10] `http://www.w3.org/DesignIssues/Syntax`

[11] `http://www.openhealth.org/RDF/RDFSurfaceSyntax.html`

[12] `http://www-db.stanford.edu/~melnik/rdf/fusion.html`

in statements that closely follow the structure and sometimes have only little meaning.

In the C-Web project[13], an approach with goals comparable to ours is developed [1]. They created a mapping language to map XPath location paths to schema paths in the ontology. This approach allows to make more specific mappings, but requires an additional resource, i.e. the specification of rules. Our approach only depends on XML source and the ontology.

Finally, there are also methods to relate ontologies to XML. In [5], for example, a procedure is described to derive DTD's (document structure prescriptions) for XML documents. In [8], a similar procedure is described to derive XML Schema prescription from a ontology. Both procedures start with an ontology and result in an XML structure to represent instance data. The method described in this paper starts from the XML document and uses the ontology to describe what is represented.

5 Conclusion

In this paper, we pointed out the problem that much data on the web is, or will be, represented in XML, but that this data cannot directly be used on the Semantic Web. XML provides structure, but leaves the interpretation open. To use the XML data in a Semantic Web, we proposed a procedure that transforms ambiguous XML data into useful RDF statements. This procedure depends on an ontology that describes the meaning of the data.

Our procedure has several advantages. First of all, it can directly be used without any alteration or addition to either XML, RDF, or RDF Schema. Also, all languages are used in agreement with their original purpose: XML is used to allow syntactic structuring of data, RDF to provide a uniform underlying data model, and RDF Schema to specify conceptual knowledge about data. Moreover, with our procedure, it is possible to describe the data remotely, i.e., no write-access to the XML document is needed. This has the additional advantage that different people can access the data via different viewpoints. The ontology is used to superimpose a specific perspective on the data, which results in a particular set of statements.

When this procedure will be implemented in automated "XML-to-RDF" interpreters, a lot of existing syntactic data will be made available as knowledge structures for the Semantic Web.

REFERENCES

[1] B. Amann, I. Fundulaki, M. Scholl, C. Beeri, and A.M. Vercoustre, 'Mapping xml fragments to community web ontologies', in *Proceedings Fourth International Workshop on the Web and Databases (WebDB'2001)*, (2000).

[2] Tim Bray, Jean Paoli, and C. M. Sperberg-McQueen. Extensible markup language (XML) 1.0 W3C recommendation. `http://www.w3.org/TR/REC-xml`, 1998.

[3] D. Brickley and R. V. Guha, 'Resource Description Framework (RDF) Schema Specification 1.0', Candidate recommendation, World Wide Web Consortium, (March 2000).

[4] Dan Connolly, Frank van Harmelen, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein, 'DAML+OIL (March 2001) Reference Description', W3C Note, World Wide Web Consortium, (December 2001).

[5] Michael Erdmann and Rudy Studer, 'How to structure and access XML documents with ontologies', *Data & Knowledge Engineering*, **36**(3), 317–335, (March 2001).

[6] Michael Erdmann, *Ontologien zur konzeptuellen Modellierung der Semantik von XML*, Ph.D. dissertation, University of Karlsruhe, Norderstedt, 2001. Book on Demand.

[7] David C. Fallside. XML schema part 0: Primer. `http://www.w3.org/TR/xmlschema-0/`, May 2001. W3C Recommendation.

[8] Michel Klein, Dieter Fensel, Frank van Harmelen, and Ian Horrocks, 'The Relation between Ontologies and XML Schemas', *Linköping Electronic Articles in Computer and Information Science*, **6**(4), (2001).

[13] See `http://cweb.inria.fr`