Tim Berners-Lee
Date: September 1998. Last modified: $Date: 1998/10/14 20:17:13 $

Status: An attempt to explain the difference between the XML and RDF models. Editing status: Draft. Comments welcome!

Up to Design Issues

Why RDF model is different from the XML model

This note is an attempt to answer the question, "Why should I use RDF - why not just XML?". This has been a question which has been around ever since RDF started. At the W3C Query Language workshop, there was a clear difference of view between those who wanted to query documents and those who wanted to extract the "meaning" in some form and query that. This is typical. I wrote this note in a frustrated attempt to explain whatthe RDF model was for those who though in terms of the XML model. I later listened to those who thought in terms of the XML model, and tried to writ it the other way around in another note. This note assumes that the XML data model in all its complexity, and the RDF syntax as in RDF Model and Syntax, in all its complexity. It doesn't try to map one directly onto the other -- it expresses the RDF model using XML.

Let me take as an example a single RDF assertion. Let's try "The author of the *page* is *Ora*". This is traditional. In RDF this is a triple

```
triple(author, page, Ora)
```

which you can think of as represented by the diagram

How would this information be typically be represented in XML?

```
<author>
     <uri>page</uri>
     <name>Ora</name>
</author>
```

or maybe

```
<document href="page">
   <author>Ora</author>
</document>
```

or maybe

```
<document>
   <details>
    <uri>href="page"</uri>
    <author>
        <name>Ora</name>
    </author>
    </details>
</document>
```

or maybe

```
<document>
   <author>
    <uri>href="page"</uri>
    <details>
        <name>Ora</name>
    </details>
    </author>
</document>

<document href="http://www.w3.org/test/page" author="Ora" />
```

The XML Graph

These are all perfectly good XML documents - and to a person reading then they mean the same thing. To a machine parsing them, they produce different XML trees. Suppose you look at the XML tree

```
<v>
   <x>
    <y> a="ppppp"</y>
    <z>
        <w>qqqqq</w>
    </z>
   </x>
</v>
```

It's not so obvious what to make of it. The element names were a big hint for a human reader.

Without looking at the schema, you know things about the document structure, but nothing else. You can't tell what to deduce. You don't know whether *ppppp* is a *y* of *qqqqq*, or *qqqqq* is a *z* of *ppppp* or what. You can't even really tell what real questions can be asked. A source of some confusion is that in the xyz example above, there are lots of questions you *can* ask. They are questions like,

- Is there a w element within a details element?
- What is the content of the w element within the first x element?
- What is the content of the w element following the first y element which contains an x element whose a attribute is "pppp"?
- and so on.

These are all questions about the *document*. If you know the document schema (a big *if*) ,

and if that schema it only gives you a limited number of ways of expressing the same thing (another big *if*) , then asking these questions can be in fact equivalent to asking questions like

- What is the author of *page*?

This is hairy. It is possible because there is a mapping from XML documents to semantic graphs. In brief, it is hairy because

- The mapping is many to one
- You need a schema to know what the mapping is
- (The schemas we are talking about for XML at the moment do not include that anyway and would have to have a whole inference language added)
- The expression you need for querying something in terms of the XML tree is necessarily more complicated than the expression you need for querying something in terms of the RDF tree.

This last is a big one. If you try to write down the expression for the author of a document where the information is in some arbitrary XML schema, you can probably do it though it may or may not be very pretty. If you try to combine more than one property into a combined expression, (give me a list of books by the same author as this one), saying it in XML gets too clumsy to consider.

(Think of trying to define the addition of numbers by regular expression operations on the strings. Its possible for addition. When you get to multiplication it gets ridiculous - to solve the problem you would end up reinventing numbers as a separate type.)

Looking at the simple XML encoding above,

```
<author>
     <uri>page</uri>
     <name>Ora</name>
</author>
```

it could be represented as a graph

We can represent the tree more concisely if we make a shorthand by writing the name of each element inside its circle:

Of course the RDF tree which this represents (although it isn't obvious from the XML tree except to those who know) is

Here we have made a shorthand again by putting making the label for each part its URI.

The complexity of querying the XML tree is because there are in general a large number of ways in which the XML maps onto the logical tree, and the query you write has to be independent of the choice of them. So much of the query is an attempt to basically convert the set of all possible representations of a fact into one statement. This is just what RDF does. It gives you some standard ways of writing statements so that however it occurs in a document, they produce the same effect in RDF terms. The same RDF tree results from many XML trees.

Wouldn't it be nice if we could label our XML so that when the parser read it, it could find the assertions (triples) and distinguish their subjects and objects, so as to just deduce the logical assertions without needing RDF? This is just what RDF does, though.

The RDF Graph

In fact RDF is very flexible - it can represent this triple in many ways in XML so as to be able to fit in with particular applications, but just to pick one way, you could write the above as

```
<Description about="http://www.w3.org/test/page" Author ="Ora" />
```

I have missed out the stuff about namespaces. In fact as anyone can create or own the verbs, subjects and objects in a distributed Web, any term has to be identified by a URI somehow. This actual real example works out to in real life more like

```
<?xml version="1.0"?>
  <Description

          xmlns="http://www.w3.org/TR/WD-rdf-syntax#"
          xmlns:s="http://docs.r.us.com/bibliography-info/"

                  about="http://www.w3.org/test/page"
                  s:Author ="http://www.w3.org/staff/Ora" />
```

You can think that the "description" RDF element gives the clue to the parser as to how to find the subjects, objects and verbs in what follows.

This is pretty much the most shorthand way of using the base RDF in XML. There are others which are longer, but more efficient when you have, for instance, sets of many properties of the same object. The useful thing is that of course they all convey the same triple

It is a mess when you use questions about a document to try to ask questions about what the document is trying to convey. It will work. In a way. But flagging the grammar explicitly

(RDF syntax is a way of doing this) is a whole lot better.

Things you can do with RDF which you can't do with XML include

- You can parse the semantic tree, which end up giving you a set of (possibly mutually referential) triples and then you can use the ones you want ignoring the ones you don't understand.

Problems with basing you understanding on the structure include

- Without having gone to the trouble of getting the schema, or having an application hand-programmed to recognise a particular document type, you can't pick up any semantic information from a document;
- When an XML schema changes, it could typically introduce new intermediate elements (like "details" in the tree above or "div" is HTML). These may or may or may not invalidate any query which has been based on the structure of the document.
- If you haven't gone to the trouble of making a semantic model, then you may not have a well defined one.

I'll end this with some examples of the last problem. Clearly they can be avoided by good design even in an XML system which does not use RDF. Using RDF makes things easier.

Get it right

If you haven't gone to the trouble of making a semantic model, then you may not have a well defined one. What does that mean? I can give some general examples of ambiguities which crop up in practice. In RDF, you need a good idea about what is being said about what, and they would tend not to arise.

Look at a label on the jam jar which says: "Expires 1999". What expires: the label, or the jam? Here the ambiguity is between a statement about a statement about a document, and a statement about a document.

Another example is an element which qualifies another apparently element. When information is assembled in a set of independently thrown in records often ambiguities can arise because of the lack of logic. HTTP headers (or email headers) are a good example. These things can work when one program handles all the records, but when you start mixing records you get trouble. In XML it is all too easy to fall into the trap of having two elements, one describing the author, and a separate one as a flag that the "author" element in fact means not the direct author but that of a work translated to make the book in question. Suddenly, the "author" tag, which used to allow you to conclude that the author of a finnish document must speak finnish, now can be invalidated by an element somewhere else on the record.

Another symptom of a specification where the actual semantics may not be as obvious as as first sight is ordering. When we hear that the order of a set of records is important, but the records seem to be defined independently, how can that be? Independent assertions are always valid taken individually or in any order. In a server configuration file, for example,

he statement which looks like "any member has access to the page" might really mean "any member has access to the page unless there is no other rule in this file which has matched the page". That isn't what the spec said, but it did mention that the rules were processed in order until one applied. Represented logically, in fact there is a large nested conditional. There is implicit ordering when mail headers say, "this message is encrypted", "this message is compressed", "this message is ASCII encoded", "this message is in HTML". In fact the message is an ASCII encoded version of an encrypted version of a compressed version of a message in HTML. In email headers the logic of this has to be written into the fine print of the specification.

Order in documents

There is something fundamentally different between giving a machine a knowledge tree, and giving a person a document. A document for a person is generally serialized so that, when read serially by a human being, the result will be to build up a graph of associations in that person's head. The order is important.

For a graph of knowledge, order is not important, so long as the nodes in common between different statements are identified consistently. (There are concepts of ordered lists which are important although in RDF they break down at the fine level of detail to an unordered set of statements like "The first element of L is x", the "third element of L is z", etc so order disappears at the lowest level.). In machine-readable documents a list of ostensibly independent statements where order is important often turn out to be statements which are by no means independent.

Some people have been reluctant to consider using an RDF tree because they do not wish to give up the order, but my assumption is that this is from constraints on processing human readable documents. These documents are typically not ripe for RDF conversion anyway.

Conclusion:

Sometimes it seems there is a set of people for whom the semantic web is the only graph which they would consider, and another for whom the document tree (or graph if you include links) is all they would consider. But it is important to recognise the difference.

In this series:

- *What the Semantic Web is not* - answering some FAQs of the unconvinced.
- Evolvability: properties of the language for evolution of the technology
- Web Architecture from 50,000 feet

Not put in yet:

.@@@ RDF does not have to be serialized in XML but ...

Up to Design Issues