RDF Triples in XML

Jeremy J. Carroll, Patrick Stickler[1]
Digital Media Systems Laboratory
HP Laboratories Bristol
HPL-2003-268
February 11th , 2004*

E-mail: jjc@hpl.hp.com, Patrick.Stickler@nokia.com

semantic web, RDF, XML

Many approaches to writing RDF in XML have been proposed. The revised standard RDF/XML still has many known problems. It is not intrinsically difficult to have a clear serialization of RDF in XML, and we present a simple solution. We add the ability to name graphs, noting that in practice this is already widely used. We use XSLT as a general syntactic extensibility mechanism to provide human friendly macros for our syntax.

* Internal Accession Date Only
Approved for External Publication
[1] Nokia Hatanpäänkatu 1, 33900 Tampere, Finland

RDF Triples in XML

Jeremy J. Carroll
Hewlett-Packard Labs
Bristol, BS34 12QZ
UK
jjc@hpl.hp.com

Patrick Stickler
Nokia
Hatanpäänkatu 1
33900 Tampere
Finland
Patrick.Stickler@nokia.com

ABSTRACT

Many approaches to writing RDF in XML have been proposed. The revised standard RDF/XML still has many known problems. It is not intrinsically difficult to have a clear serialization of RDF in XML, and we present a simple solution. We add the ability to name graphs, noting that in practice this is already widely used. We use XSLT as a general syntactic extensibility mechanism to provide human friendly macros for our syntax.

Categories and Subject Descriptors

I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods—*Representation languages*

General Terms

Languages

Keywords

Semantic Web, RDF, XML

1. INTRODUCTION

It is well known that RDF/XML presents problems.

A cursory search with Google reveals half-a-dozen suggestions for alternative XML syntaxes for RDF.

This paper presents another. Distinctively we select the simplicity of N-triples [22] as our guide, and have an explicitly minimalist set of requirements.

For cases where this set of requirements is insufficient we indicate the use of the stylesheet processing instruction to provide general purpose syntactic extensibility using XSLT [17].

A further distinctive feature of our syntax is explicit support for naming of graphs.

1.1 Examples

Example 1: Here is a TriX document:

```
<graphset
   xmlns="http://jena.sourceforge.net/TriX/">

<graph>
   <triple>
      <uri>http://example.org/Bob</uri>
      <uri>http://example.org/wife</uri>
      <uri>http://example.org/Mary</uri>
   </triple>
   <triple>
      <uri>http://example.org/Bob</uri>
      <uri>http://example.org/name</uri>
      <plainLiteral>Bob</plainLiteral>
   </triple>
   <triple>
      <uri>http://example.org/Mary</uri>
      <uri>http://example.org/age</uri>
      <typedLiteral datatype=
    "http://www.w3.org/2001/XMLSchema#integer"
      >32</typedLiteral>
   </triple>
</graph>

</graphset>
```

Syntactic extensions to the minimalist core, require a processing instruction. **Example 2** is the same graph expressed using qnames and XSD type support:

```
<?xml-stylesheet type="text/xml" href=
 "http://jena.sourceforge.net/TriX/all.xsl"
  ?>

<graphset
   xmlns="http://jena.sourceforge.net/TriX/"
   xmlns:eg="http://example.org/" >

<graph>
   <triple>
      <qname> eg:Bob </qname>
      <qname> eg:wife </qname>
      <qname> eg:Mary </qname>
   </triple>
   <triple>
      <qname> eg:Bob </qname>
      <qname> eg:name </qname>
      <plainLiteral>Bob</plainLiteral>
   </triple>
   <triple>
      <qname> eg:Mary </qname>
      <qname> eg:age </qname>
      <integer> 32 </integer>
   </triple>
</graph>

</graphset>
```

2. THE REQUIREMENTS

The requirements we address are the following:

1. The format serializes the RDF graph.
2. The format is compatible with XML tools, such as XML Schema [26], DTDs [11], XPath [18], XSLT [17]. In particular, it is straight forward to access the graph structure using such tools.
3. As few other features are included as possible.

The last requirement is the most important. We will see that one of the problems with RDF syntax is an excess of requirements

from different communities creating a political problem that may get solved with a technical hack.

We argue later that the two additional features we add, naming of graphs and syntactic extensibility, are well-chosen and appropriate. Moreover they do not reflect the needs of any specific community, but meet general requirements of many RDF users.

3. WHAT'S WRONG WITH RDF/XML?

3.1 A Brief History of RDF Syntax

The original RDF Syntax working group took input from Guha's MCF [23], Microsoft's Web Collections [25], and Lassila's Lisp oriented PICS-NG format [38].

Mixing these together, taking something from everything, resulted in RDF/XML in 1999 [30]. Since its publication, there have been a steady stream of alternatives.

Berners-Lee started the process, by proposing an unstriped syntax [5]. Melnik followed up with an attribute based proposal [35] which could be used to bridge [34] between XML and RDF.

The next year (2000), Berners-Lee gave up on a usable XML syntax for RDF, and proposed N3 [7].

In 2001, the RDF Core Working Group started, partly to fix the RDF/XML syntax. Adobe launched XMP [1], which uses a proper subset of RDF/XML. Robie [42] showed that a normalized subset of RDF/XML could be used effectively with XQuery [9].

Seeing that RDF/XML was being revised rather than replaced, Bray proposed another XML syntax RPV [10] in 2002.

In 2003, while completing the revision of RDF/XML [4], Beckett proposed a simple XML form [3] inspired by N-triples [22], a simple subset of N3 [7]. Both N-triples, and Beckett's proposals stick very closely to the abstract syntax [29], which is a great strength. Meanwhile, Dubinko proposed another syntax [20], more suited for embedding within HTML. The problem of embedding RDF inside HTML is itself non-trivial [39], and is the topic of a recent W3C taskforce [41].

Our history closes by returning to Berners-Lee, who in a recent keynote presentation [6] referred to the 'RDF syntax shock'.

3.2 RDF/XML Revised, but not Fixed

The W3C has just completed a major clean up of the syntax [4], along with a clarification of the underlying data model [29], and its intended interpretation [24].

While many syntactic problems have been fixed, and it is at least plausible to have interoperability between RDF/XML implementations, some of the 'postponed issues' [33] indicate the extent of the original mess.

- 'RDF embedded in XHTML and other XML documents is hard [i.e. impossible] to validate.'
- 'it is not possible to define [...] a subset [of RDF/XML] that [...] can represent all [...] RDF graphs [and] can be described by an DTD or an XML Schema'

In brief, RDF/XML does not layer RDF on top of XML in a useful way.

Meanwhile, there are other unresolved syntactic issues, involving qnames, collections, literals as subjects, blank nodes as predicates, reification and quoting. Hence, a further round of work on RDF/XML is likely to be a continuation of legacy hell, with additional requirements pulling in different directions, and old requirements not getting dropped.

3.3 Our Requirements and Prior Work

The requirement that the graph be simply reflected in the XML, rules out most of the previous proposals. Many are based too closely on RDF/XML to be salvagable, for example: XMP [1], Dubinko [20] and Robie's normalized RDF/XML [42].

The two early proposals from Berners-Lee [5] and Melnik [35] both use attributes that can be added to an arbitrary XML document, in a way that breaks DTDs and XML Schemata.

Bray's RPV [10] does not address blank nodes. This leaves Beckett's proposals [3], which, while incompletely worked out, do show that it is simple and straightforward to represent an RDF graph as a set of elements each with three children.

4. WHAT'S RIGHT WITH RDF/XML?

Given the number of suggestions for change and RDF/XML's lack of popularity with the practioners, why does it continue?

Once you get used to it, it is surprisingly concise. The RDF data model, in which everything is triples, is inevitable verbose - but writing these triples in RDF/XML tends to ameliorate things.

The use of qnames to abbreviate URI references is concise, and sufficiently liked that this convention is widely used, also in non-XML contexts, e.g. in N3 [7], and the OWL Semantics [40] document. The use of typed nodes, to avoid making a common triple explicit, adds to the efficiency with which RDF/XML encodes the RDF graph, and permits syntaxes which, to some extent, hide the underlying triple structure.

This hiding of the triple structure makes it easy for users to get into an RDF application such as OWL with only a partial understanding of its representation in RDF.

However, RDF/XML neithers permits complete hiding of the underlying RDF, nor does it make it clear what that underlying RDF is. We suggest that it is better to have clarity in the basic syntax, with hiding achieved by using alternative syntactic forms that are transformed into the basic syntax.

RDF/XML also provides a number of syntactic features which are useful for certain sorts of construct:

- `rdf:parseType="Literal"` is the only sensible way of embedding XML into the RDF graph. (The alternative requires knowledge of Exclusive XML Canonicalization [27]).
- `rdf:parseType="Collection"` is useful when writing OWL Ontologies [19].
- `rdf:parseType="Resource"` is used extensively in XMP [1].
- The use of property attributes is useful when embedding RDF in HTML.

Thus many communities find that while RDF/XML has many features they do not like, certain key features are highly attractive and keep them enagaged.

5. TRIX SYNTAX

The core of TRIX is the `triple` element, which contains three children, the subject, predicate and object of the triple.

Each of these children is either a `uri` element, an `id` element, a `plainLiteral` or a `typedLiteral` element depending on whether the corresponding node in the graph is an RDF URI reference, a blank node or a literal (plain or typed).

The element content contains the label of the node (or the blank

node identifier). Whitespace normalization is applied to `uri`[1] and `id` element content.

We strongly prefer the use of absolute URI references in `uri`. This ensures that XML based tools can easily compare two `uri` nodes for equality. Relative URIs, if used, are resolved against the base URL used to retrieve the document (as in RDF/XML without `xml:base`).

`plainLiteral` elements can be modified by an `xml:lang` attribute. `xml:lang` is prohibited elsewhere in the document (for example, it is not permitted on the root element). This avoids any confusion as to whether it applies to typed literals. It does not.

`typedLiteral` elements require a `datatype` attribute. As in RDF/XML. no whitespace processing is performed. We note it is difficult to write the legal lexical forms for `rdf:XMLLiteral` which have to be exclusive canonical XML [27], which is escaped either with a CDATA block, or using XML character escaping conventions.

A `graph` element has any number of `triple` elements as children. Optionally, the first child of a `graph` is a `uri` or `id` element, that names the graph (see below). The `graph` element has a boolean valued attribute `asserted`, which takes the default of `true`.

The root element of the document is a `graphset` element, which has zero or more graphs as its child elements.

The ability to have more than one graph in a document, the ability to name graphs, and the ability to mark some graphs as unasserted, are all motivated by the extension of associating names with graphs.

TRIX is described by a DTD, shown in table 1 and by an XML Schema, shown in table 2. This format is very close to the RDF abstract syntax [29], the only deviation being the ability to name graphs.

```
<!-- TriX: RDF Triples in XML -->
<!ELEMENT graphset     (graph*)>
<!ATTLIST graphset     xmlns    CDATA
          #FIXED "http://example.org/TriX/">
<!ELEMENT graph        ((id|uri)?, triple*)>
<!ATTLIST graph        asserted (true|false) "true">
<!ELEMENT triple       ((id|uri), uri,
        (id|uri|plainLiteral|typedLiteral))>
<!ELEMENT id           (#PCDATA)>
<!ELEMENT uri          (#PCDATA)>
<!ELEMENT plainLiteral (#PCDATA)>
<!ATTLIST plainLiteral xml:lang CDATA #IMPLIED>
<!ELEMENT typedLiteral (#PCDATA)>
<!ATTLIST typedLiteral datatype CDATA #REQUIRED>
```

Table 1: TRIX DTD

6. NAMING GRAPHS

TRIX provides for graph naming either with global names by the use of an optional `uri` element before the triples of a graph, or with file scoped names, by the use of an optional `id` element.

[1]The XML Schema in table 2, uses the `xsd:anyURI` simple type for these elements. The whitespace facet with value `collapse` converts two successive spaces to a single space. This limits the ability to represent all RDF URI references, which may include multiple successive spaces. These problems will be resolved when the Internationalized Resource Identifier proposal[21], which prohibits spaces, works its way through to the definition of both `anyURI` and RDF URI references.

```
<schema
  xmlns           = "http://www.w3.org/2001/XMLSchema"
  xmlns:xsd       = "http://www.w3.org/2001/XMLSchema"
  xmlns:xml       = "http://www.w3.org/XML/1998/namespace"
  xmlns:trix      = "http://jena.sourceforge.net/TriX/"
  targetNamespace = "http://jena.sourceforge.net/TriX/">

  <import namespace="http://www.w3.org/XML/1998/namespace"
              schemaLocation="xml.xsd"/>

  <element name="graphset">
    <complexType>
      <sequence>
        <element maxOccurs="unbounded" ref="trix:graph"/>
      </sequence>
    </complexType>
  </element>

  <element name="graph">
    <complexType>
      <sequence>
        <choice minOccurs="0">
          <element ref="trix:id"/>
          <element ref="trix:uri"/>
        </choice>
        <element maxOccurs="unbounded" ref="trix:triple"/>
      </sequence>
      <attribute name="asserted" type="boolean" de-
fault="true"/>
    </complexType>
  </element>

  <element name="triple">
    <complexType>
      <sequence>
        <choice>
          <element ref="trix:id"/>
          <element ref="trix:uri"/>
        </choice>
        <element ref="trix:uri"/>
        <choice>
          <element ref="trix:id"/>
          <element ref="trix:uri"/>
          <element ref="trix:plainLiteral"/>
          <element ref="trix:typedLiteral"/>
        </choice>
      </sequence>
    </complexType>
  </element>

  <element name="id" type="NCName"/>

  <element name="uri" type="anyURI"/>

  <element name="plainLiteral">
    <complexType>
      <simpleContent>
        <extension base="xsd:string">
          <attribute ref="xml:lang"/>
        </extension>
      </simpleContent>
    </complexType>
  </element>

  <element name="typedLiteral">
    <complexType>
      <simpleContent>
        <extension base="xsd:string">
          <attribute name="datatype"
            type="anyURI" use="required"/>
        </extension>
      </simpleContent>
    </complexType>
  </element>

</schema>
```

Table 2: An XML Schema for TRIX

Example 3 shows a named graph including its own provenance information:

```
<graphset
   xmlns="http://jena.sourceforge.net/TriX/">

<graph>
   <id>binfo</id>
   <triple>
      <uri>http://example.org/aBook</uri>
      <uri>http://purl.org/dc/elements/1.1/title</uri>
      <typedLiteral datatype=
"http://www.w3.org/1999/02/22-rdf-syntax-ns#XMLLiteral"
      >&lt;ex:title xmlns:ex="http://example.org/"&gt;
          A Good Book
       &lt;/ex:title&gt;</typedLiteral>
   </triple>
   <triple>
      <uri>http://example.org/aBook</uri>
      <uri>
    http://www.w3.org/2000/01/rdf-schema#comment
      </uri>
      <plainLiteral xml:lang="en"
      >This is a really good book!</plainLiteral>
   </triple>
   <triple>
      <id>binfo</id>
      <uri>http://example.org/source</uri>
      <uri>http://example.org/book-description.rdf</uri>
   </triple>
</graph>

</graphset>
```

Since we take an explicitly minimalist stance, we have to make a strong case for this feature in TRIX .

We first give examples of naming of graphs in the field, showing how the current technology is used for this. We find the current solutions muddled and *ad hoc*, and believe a standardized approach will be highly beneficial.

Moreover, the requirement for graph naming, is not from one community within the Semantic Web, but a requirement that goes across the board. It is needed for metadata repositories, and for ontological systems. Graph naming occurs in Semantic Web programming environments and query languages. Nearly all users of the Semantic Web name their graphs, the base syntax should provide explicit support.

6.1 Do Graphs need Naming?

6.1.1 Syndication

An obvious use for naming graphs is when many different sources need to be aggregated, and it is desired to retain clarity about which information came from which source. This is straightforward if there are distinct graphs, and also a union graph. If the graphs have names, then the provenance information can be attached to the names. Example 3 shows a graph including its provenance information.

6.1.2 Semantic Web Languages and Frameworks

One approach to graphs as first class objects occurs in N3 [7], which provides contexts: these are sets of triples which are treated as anonymous resources. They can then be named using `owl:sameAs`. Alternatively they can participate in other graphs simply like a blank node.

Query languages such as RQL [28] and RDQL [37] obviously require the ability to refer to graphs. Often the document URL is used as the name of the graph it contains.

Systems with views, such as TRIPLE [36], RVL [31] and Jena2 [14], not only use the naming of graphs of actual triples, but permit the naming of views of virtual triples (in some systems the views may potentially be infinite). In RVL, the views are named using XML Namespaces names; in TRIPLE the views are named using resources.

6.1.3 Within the Standards

One place in which graphs are named and referred to extensively is in the RDF Test Cases [22] and OWL Test Cases [15]. In order to be able to name many graphs, and describe the relationships between them, each of these depends on a repository of hundreds of files. The relationships described in the test manifest files, such as entailment or equivalence, are described as relationships between documents. What is intended is in fact a relationship between the graphs contained within the documents.

The RDF recommendations provide for reification of statements as a mechanism for using RDF to talk about RDF. However, it is known not to work well. In typical use cases, such as adding provenance information, their is a large triple bloat. Adding a reification quad for every triple causes a five fold increase. Doing anything with these then requires minimally one extra triple to link the reified triple in with say a 'reified graph'. More frequently, the same provenance information, perhaps four or five triples, are duplicated and added to every reified triple. Thus the use of reification results in maybe a tenfold blow up. What is worse, is that having done this, the triples do not mean what one might hope. As is clarified in the RDF Semantics [24], reification is *not* a quoting mechanism.

The OWL Ontology element and the OWL imports mechanisms both try to refer to named graphs. They use the document URL as the name. This creates somewhat unclear semantics, stated in operational terms. The subject of `owl:imports` triples gets almost entirely ignored. The OWL recommendations fail to adequately account for the intended relationship between the ontology name and the ontology content (whether thought of as abstract syntax trees or RDF triples [2]). This is particularly clear when trying to convert the imports closure of a document, which is a large graph, into a set of abstract syntax trees, one corresponding to each ontology element. There is no method for determining which triple is mapped into which tree. Explicit graph naming would help to make the intensions clearer.

6.1.4 Signing Graphs

Carroll [13] presents an algorithm for generating a canonical names for the blank nodes and hence a canonical ordering of the triples of a (possibly slightly modified) RDF graph.

This could become a core part of the Semantic Web infrastructure by permitting verification of provenance information.

However, it requires the ability to separate out separate subgraphs of whatever data a system is using, so that the various pieces from different sources can have their signatures verified.

6.2 A Minimalist Graph Naming Mechanism

The name associated with a graph is a way of referring to the syntactic object. In RDF terms, it is the equivalence class of RDF graphs. Blank node labels, and the order of the triples, do not matter. The choice of which URI we use to refer to each resource in the graph does matter. Contrast with the semantics of reification which concerns the interpretation of, for example, the predicate URI, rather than the URI itself.

To say anything about the graph, e.g. provenance information, some triples are needed that involve this node. These triples can be included within the graph, which then includes assertions about itself, or they can be in a separate graph in the same document, or they can be in a separate document (which requires the use of a uri node naming the graph). Example 3 shows the first of these possi-

bilities. In the second case, we may wish to state the provenance information, without committing ourselves to the original graph. This is shown in **example 4**, modified from example 3:

```
<graphset
   xmlns="http://jena.sourceforge.net/TriX/">

<graph asserted="false">
   <id>binfo</id>
   <triple>
      <uri>http://example.org/aBook</uri>
      <uri>http://purl.org/dc/elements/1.1/title</uri>
      <typedLiteral datatype=
  "http://www.w3.org/1999/02/22-rdf-syntax-ns#XMLLiteral"
      >&lt;ex:title xmlns:ex="http://example.org/"&gt;
          A Good Book
       &lt;/ex:title&gt;</typedLiteral>
   </triple>
   <triple>
      <uri>http://example.org/aBook</uri>
      <uri>
   http://www.w3.org/2000/01/rdf-schema#comment
      </uri>
      <plainLiteral xml:lang="en"
      >This is a really good book!</plainLiteral>
   </triple>
</graph>
<graph>
   <triple>
      <id>binfo</id>
      <uri>http://example.org/source</uri>
      <uri>http://example.org/book-description.rdf</uri>
   </triple>
</graph>

</graphset>
```

The first graph, is merely quoted, indicated by the `asserted = "false"` attribute. The assertional content of the example is given in the second graph.

Other possible additional requirements are dealt with in the next section as syntactic extensions. Graph naming might have been provided in a similar style by mapping a syntactic extension to the RDF reification vocabulary. However, this would be limited by the meaning of the reification vocabulary, as described in RDF semantics [24]. Since the intent is to provide a mechanism that can be used for quoting, which is explicitly excluded by the RDF semantics, providing core syntax is necessary.

6.3 The Semantics of Graph Naming

The formal semantics of this construct is beyond the scope of this paper.

The intended informal semantics is that the node used for naming a graph is interpreted as the RDF graph specified within the `<graph>` element. Thus, statements about the node are statements about the graph. More strictly such a node denotes an equivalence class of RDF graphs. RDF graph equivalence, as defined by RDF Concepts permits reordering of the triples, and relabelling of the blank nodes.

This differs from merely extending RDF triples to RDF quads, in that the full extent of the graph is known, and is not treated with the open world assumption. Unlike a subject resource, which may have additional properties not mentioned in a document, the assertion of a named graph asserts that this graph is exactly the triples given, and there are not any others that have been omitted. Significantly, this intended semantics *is* a quoting mechanism and does not suffer the 'two-stage interpretation process' discussed for RDF reification in RDF Semantics. A naive extension of the RDF model theory to cover quads rather than triples would replicate this defect in the reification semantics.

The meaning of the `graphset` is intended to be the logical AND of the meaning of the asserted graphs. Thus a graph for which `asserted` is `false` is quoted, and can be referred to in other graphs, but does not contribute to the meaning of the `graphset`.

It is likely that details of the informal semantics will need to change as work proceeds on the formal semantics of naming graphs. The graph may include triples involving itself, which may create semantic difficulties. Some semantic theories may exclude such graphs – much as OWL DL semantics excludes RDF graphs in which `rdf:type` is given a subproperty. Similar difficulties may occur when two graphs within a `graphset` share a blank node. Such a case is neither explicitly covered, nor explicitly excluded by the RDF semantics.

The problem of different RDF graphs having different assertional status is already present in the RDF and OWL recommendations, in the RDF and OWL Test Cases [15, 22]. An OWL consistency test consists of two files: a Manifest file that is intended as an assertion about a second file, which contains a consistent OWL document. The second file is not intended to be asserted in the same way.

6.4 A Further Example

As well as provenance information, named graphs can be used to encode rules (such as using the `log:implies` connective in N3), and test cases.

Example 5 shows how an RDF test case might be formulated in TRIX . The vocabulary is closely based on the vocabulary used in the RDF Test Cases [22].

```
<graphset>

<graph>
   <triple>
      <uri>
    http://example.org/tests/language-tag-case
      </uri>
      <uri>http://example.org/entailmentRules</uri>
      <uri>http://www.w3.org/1999/02/22-rdf-syntax-ns#</uri>
   </triple>
   <triple>
      <uri>
    http://example.org/tests/language-tag-case
      </uri>
      <uri>http://example.org/premise</uri>
      <uri>http://example.org/tests/graph1</uri>
   </triple>
   <triple>
      <uri>
    http://example.org/tests/language-tag-case
      </uri>
      <uri>http://example.org/conclusion</uri>
      <uri>http://example.org/tests/graph2</uri>
   </triple>
</graph>

<graph asserted="false">
   <uri>http://example.org/tests/graph1</uri>
   <triple>
      <id>x</id>
      <uri>http://example.org/property</uri>
      <plainLiteral xml:lang="en-us">a</plainLiteral>
   </triple>
</graph>

<graph asserted="false">
   <uri>http://example.org/tests/graph2</uri>
   <triple>
      <id>x</id>
      <uri>http://example.org/property</uri>
      <plainLiteral xml:lang="en-US">a</plainLiteral>
   </triple>
</graph>
</graphset>
```

6.5 The Liar's Paradox

Unfortunately, named graphs combined with 'logical' vocabu-

lary (concerning logical metaproperties such as entailment) can be used to encode the liar's paradox.

For example, in N3, we can say:

```
@prefix log: <http://www.w3.org/2000/10/swap/log#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix eg: <http://example.org/> .
{
  eg:liar
  log:implies {
    eg:noone a owl:Nothing .
  } .
} owl:sameAs eg:liar .
eg:liar a log:Truth .
```

The same example could be encoded in TRIX , with the N3 formula construct using { and } corresponding to the bnode naming a graph with the given triples. We could also make a similar example using vocabulary like the RDF Test Cases [22] vocabulary (replacing the `test:premiseDocument` and `test:conclusionDocument` with `eg:premise` and `eg:conclusion`, as in example 5).

This would appear a fatal error with our proposal for graph naming. However, we have only clarified pre-existing problems. Using the actual RDF test vocabulary, we can use a manifest file's own URI as the URI of the premise document, and create an analogous paradoxical RDF test case.

Our clarification shows that some solution is required to this potential for paradox, since in the Semantic Web we (already) name graphs (if only with document URLs), and already have (implicit) quoting.

We take Wittgenstein's [43] optimisitic view of such paradoxes: 'All we have to do is to make a new stipulation to cover the case in which the rules conflict, and the matter's resolved.' The paradox does not arise from named graphs *per se*, but from the combination of named graphs and logical connectives (that depend on a further reference to the semantic theory). Thus, a simple resolution would be to rule such properties as out of scope for RDF properties. This would invalidate the logical connectives in N3, and the Manifest file format of the RDF and OWL test suites, but would leave the provenance and signing examples intact. It is probably not too difficult to find some appropriate well-foundnesses criteria, such as not permitting 'logical' properties to be used in triples whose subject or object contains (directly or indirectly) the triple itself. It would be particularly attractive to be able to classify such paradoxes as merely inconsistent. This paper is not the place for such explorations: by clarifying the syntactic self-reference that is already in use within the Semantic Web, we have articulated the need for such work to be done.

7. EXTENSIBILITY

We have seen in section 3, that there are many different communities with an interest in XML syntaxes for RDF. Each community brings their own requirements.

Moreover requirements related to ease of writing and reading an XML syntax for RDF tend, in general, to conflict with the core requirements of giving a transparent representation of the graph in a way that can easily be processed with XML tools. This is because the RDF graph tends to be too fine-grained and detailed for direct human consumption, and user-friendly syntaxes need to use 'macros' of some sort. In RDF/XML macros are provided for typed nodes, property attribtues, three parseTypes, striping, reification and container membership. These macros then create problems for XML tools.

The answer we suggest is to have a general purpose and interoperable extensibility mechanism. Each community can then define and use whatever syntactic extensions they wish, declaring the extensions they are using at the top of the data files. As long as the extensions are described in a standard way and are identified with URLs, any processor can apply them.

To be more specific we use XSLT as the syntactic extensibility mechanism, and the stylesheet processing instruction [16] as the declaration.

We start by showing in detail how the TRIX syntax can be made more user-friendly using qnames, using this mechanism. We then sketch other useful extensions, for `xml:base`, XMLLiterals, collections, and typed literals.

7.1 QNames

Using qnames to abbreviate URI references is popular, appearing most noticeably in many e-mail messages discussing RDF triples.

This convention is not strictly necessary, similar effect can be achieved in TRIX using XML entities. If the size of documents using full URIrefs is an issue then standard compression techniques can be used.

However, human readers and writers of RDF documents would like to see and use qnames. We hence, extend the TRIX syntax to include a `qname` element. Its content is a qname which abbreviates a URI reference, in the normal way. This can be transformed into a `uri` element using an XSLT program with the following rule:

```
<xsl:template match="trix:qname">
  <uri>
  <xsl:value-of
        select="namespace::*[
      local-name()=substring-before(text(),':')
                  ]"/>
  <xsl:value-of select="substring-after(text(),':')"/>
  </uri>
</xsl:template>
```

Example 2, in the introduction, shows this being used.

7.2 xml:base

The use of relative URIs is often convenient when writing documents. They also may make a document easier to read, by eliminating redundant information.

A further transformation resolves any relative URIs inside `uri` elements, using the inscope `xml:base` value [32].

Hence, the first triple of example 1 can be written using this extension:

```
<?xml-stylesheet type="text/xml" href=
 "http://jena.sourceforge.net/TriX/xmlbase.xsl"
  ?>
<graphset
   xml:base="http://example.org/"
   xmlns="http://jena.sourceforge.net/TriX/">

<graph>
   <triple>
      <uri>Bob</uri>
      <uri>wife</uri>
      <uri>Mary</uri>
   </triple>
   .
   .
</graph>

</graphset>
```

7.3 Typed literals

Always using `datatype` with a URI for typed literals is repetitive. A solution for the XML Schema builtin simple types [8], is to provide a transform that permits each such simple type as an element name, and converts it into an appropriate literal. This transform can perform the appropriate whitespace processing, as given by the whitespace facet of the datatype.

A sample XSLT template is as follows:

```
<xsl:template match="trix:decimal">
  <typedLiteral
    datatype="http://www.w3.org/2001/XMLSchema#decimal">
  <xsl:value-of
        select="normalize-space(text())"/>
  </typedLiteral>
</xsl:template>
```

which transforms, for example, `<decimal> 4.0 </decimal>` into `<typedLiteral datatype="http://www.w3.org/2001/XMLSchema#-decimal"> 4.0 </literal>`. Again, this is illustrated in example 2.

7.4 XMLLiterals

Since the lexical form of an XMLLiteral has to be in exclusive Canonical XML, it is virtually impossible to create these except with machine support.

Since the definition of these in RDF concepts specifies that the *InclusiveNamespaces PrefixList* is empty, all the information needed to perform the canonicalization is in the XPath nodeset, and so, the transformation can be performed with XSLT (with some difficulty)[2].

So, the extensibility mechanism is powerful enough to support a transform that transforms say:

```
<xmlliteral><foo b="B" a="A"/></xmlliteral>
```

into

```
<typedliteral datatype=
"http://www.w3.org/1999/02/22-rdf-syntax-ns#XMLLiteral"
>&lt;foo a="A" b="B">&lt;/foo></typedliteral>
```

7.5 Collections

The `rdf:parseType="Collection"` construct of RDF/XML introduces many triples and blank nodes to represent list structures in RDF.

A similar TRIX extension can be defined using an XSL transfrom. One slightly tricky detail concerns the names of blank nodes. Since the transform needs to introduce new nodes, it must be sure not to use names being used elsewhere. One way is to rename all preexisting blank nodes using a rule such as:

```
<xsl:template match="trix:id">
  <id>
  <xsl:text>u.</xsl:text>
  <xsl:value-of
        select="normalize-space(text())"/>
  </id>
</xsl:template>
```

Using this, and a more complex set of rules for the collections themselves, a transfrom can be defined that converts:

```
 <triple>
  <id> aDescription </id>
  <uri> &owl;intersectionOf </uri>
  <collection>
     <id> one </id>
  </collection>
 </triple>
```

into

```
 <triple>
  <id> u.aDescription </id>
  <uri> &owl;intersectionOf </uri>
  <id> t.23 </id>
 </triple>
 <triple>
   <id> t.23 </id>
   <uri> &rdf;first </uri>
   <id> u.one </id>
 </triple>
 <triple>
   <id> t.23 </id>
   <uri> &rdf;rest </uri>
   <id> &rdf;nil </id>
 </triple>
```

Such a transform is indifferent to the nature of the collection content, and so can also be used with a collection of literals (or a mixed collection). This addresses the problem seen with the datarange construct in OWL DL exhibited in test `oneof-004` of the OWL Test Cases [15].

7.6 RDF/XML as a TRIX Extension

In fact, it is possible to write an RDF/XML parser using XSLT. An example is Snail [12], which while unusably slow[3], does show that it can be done.

Hence it would be possible to view RDF/XML as a syntactic extension to TRIX . Prepending an appropriate stylesheet processing instructions provides backward compatibility.

7.7 An Evolving Set of Syntactic Extensions

With such a web based approach to syntactic extensibility anyone can define their own extensions. Those that are useful will be used; those that are not, will not.

This will form an evolutionary system for designing useful XML serializations for RDF.

Since XSLT is not always the most efficient processing environment some TRIX processors may be coded with prior knowledge of well-known extensions. For these, the stylesheets would not be invoked, but instead some equivalent code would be used.

8. CANONICAL TRIX

Canonical TRIX documents can be defined by:

- Requiring each graph in the graphset to have a name (potentially introducing a new blank node).
- Canonical assigning identifiers for the blank nodes.
- Lexicographically ordering the triples in each graph.
- Sorting the graphs into lexicographic order by their names
- Following a set of rules concerning the optional whitespace.

Blank node labels can be assigned using the techniques described for signing RDF graphs in [13].

The simplest rule for optional whitespace would be that there is none. It may be preferred to have a newline before each start element (except the document root), possibly indented by one space for children of the root, two spaces for grandchildren of the root, and three spaces for great grandchildren.

This suffers from the same limitations as for signing RDF graphs, and some graphs need to be modifed to semantically equivalent ones, before canonicalization. Details are in [13].

[2]The sort in XSLT 1.0 leaves too much as implementation defined. It is possible in XSLT 2.0 to specify precisely the sort needed for attribute ordering in XML Canonicalization.

[3]Snail's purpose was to illustrate an approach to defining RDF/XML rather than to be a serious implementation.

9. EVALUATION

9.1 Comparison with RDF/XML

TRiX achieves the goal of being generically processable by XML tools. XPath [18] expressions to pick out triples and/or resources, are straightforward. Queries can be reformulated from RDF query languages, such as RDQL [37] into XML languages such as XQuery [9].

RDF/XML is more user friendly and more concise.

TRiX with syntactic extensions achieves both sets of goals, in that, by applying the transfroms, the advantages of TRiX can be realized, or by not applying the transforms, the advantages of RDF/XML can be realized.

The simplicity of the TRiX serialization reflects the underlying simplicity of the RDF conceptual model, rather than the misleading impression left by the baroqueness of RDF/XML.

9.2 Comparison with Beckett's Proposals

In our survey in section 3.1, we identified Beckett's proposals [3] as the most promising.

He identifies choices such as:

- whether to use named elements for subject, predicate and object or to rely on position within a triple.
- whether to permit the use of qnames to abbreivate urirefs.
- whether to use attributes or element content.

We have used position to identify the rôle in the triple, the proposed `subject` element gives redundant information that might be useful to a human reader, but we do not really expect TRiX to be very human readable.

For similar reasons, we avoid allowing qnames as abbreviations, except as a syntactic extension. The uniformity makes it easier to process the RDF graph with XML tools, since there is no need to consider the case where a node is represented by a `qname` element in one triple, and by a `uri` element in another. It also avoids the difficulties caused by the differences in treatment of qnames between RDF and XML. In RDF, a qname is merely an abbreviation, whereas in XML a qname is a pair: a namespace name and a local name.

We determined that using attributes for literal content creates unnecessary problems, concerning XML attribute value normalization [11]. Hence, literal values, as in the examples in [3], must be expressed as element content. For uniformity, we hence also express urirefs and blank node identifiers using element content.

The naming of graphs and syntactic extensibility are not discussed by Beckett in [3].

10. CONCLUSIONS

The problem of how to serialize RDF in XML has produced many proposals. Most, particularly RDF/XML, obscure the nature of the RDF graph, hence making the problem seem difficult. Despite the revision of RDF/XML, discussions continue.

With little difficulty, we have produced a thought-out and simple proposal. We suggest that it is time that the Semantic Web community choose a simple serialization such as ours, and stopped wasting time with this problem.

The use of XSLT as an extensibility mechanism permits the inevitably rather unreadable machine-friendly syntax to be represented in a more human-friendly fashion. It also permits backward compatibility with RDF/XML.

Naming graphs is a necessary part of the Semantic Web, and should be included in the core syntax. More work on the semantics of graph naming is needed, particularly to address the difficulties of logical predicates.

11. REFERENCES

[1] Adobe. XMP – Extensible Metadata Platform. `http://partners.adobe.com/asn/developer/xmp/pdf/MetadataFramework.pdf`, 2001.

[2] S. Bechhofer and J. Carroll. OWL DL: Tress or Triples? Submitted to WWW2004, 2003.

[3] D. Beckett. A retrospective on the development of the RDF/XML Revised Syntax . `http://ilrt.org/people/cmdjb/2003/05/iswc/`, 2003.

[4] D. Beckett. RDF/XML Syntax Specification (Revised). `http://www.w3.org/TR/rdf-syntax-grammar/`, 2003.

[5] T. Berners-Lee. A strawman Unstriped syntax for RDF in XML. `http://www.w3.org/DesignIssues/Syntax`, 1999.

[6] T. Berners-Lee. SW status and direction. `http://www.w3.org/2003/Talks/1023-iswc-tbl/all.htm`, 2003. Keynote address at ISWC 2003.

[7] T. Berners-Lee, R. R. Swick, J. Reagle, S. Hawke, and D. Connolly. Primer: Getting into RDF & Semantic Web using N3. `http://www.w3.org/2000/10/swap/Primer`.

[8] P. Biron and A.Malhotra. XML Schema Part 2: Datatypes. `http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/`, 2001.

[9] S. Boag, D. Chamberlin, M.F.Fernández, D. Florescu, J. Robie, and J. Siméon. XQuery 1.0: An XML Query Language. `http://www.w3.org/TR/2003/WD-xquery-20030822/`, 2003.

[10] T. Bray. The RPV (Resource/Property/Value) Syntax for RDF. `http://http://www.textuality.com/xml/RPV.html`.

[11] T. Bray, J. Paoli, C. Sperberg-McQueen, and E. Maler. Extensible Markup Language (XML) 1.0 (Second Edition). `http://www.w3.org/TR/2000/REC-xml-20001006`, 2000.

[12] J. Carroll. Snail: Excruciatingly Slow RDF Parsing. `http://www-uk.hpl.hp.com/people/jjc/snail/`, 2001.

[13] J. Carroll. Signing RDF Graphs. In *The Semantic Web - ISWC 2003*, number 2870 in LNCS, pages 369–384. Springer, 2003.

[14] J. Carroll, I.Dickinson, C. Dollin, D. Reynolds, A. Seaborne, and K. Wilkinson. The jena semantic web platform: Architecture and design. Submitted to WWW2004, 2003.

[15] J. J. Carroll and J. D. Roo. Web Ontology Language (OWL) Test Cases. `http://www.w3.org/TR/owl-test/`, 2003.

[16] J. Clark. Associating Style Sheets with XML documents Version 1.0. `http://www.w3.org/1999/06/REC-xml-stylesheet-19990629/`, 1999.

[17] J. Clark. XSL Transformations (XSLT) Version 1.0. `http://www.w3.org/TR/1999/REC-xslt-19991116`, 1999.

[18] J. Clark and S. DeRose. XML Path Language (XPath) Version 1.0. `http://www.w3.org/TR/1999/REC-xpath-19991116`, 1999.

[19] M. Dean and G. Schreiber. OWL Web Ontology Language Reference. `http://www.w3.org/TR/owl-ref/`, 2003.

[20] M. Dubinko. Metadata for Grandma. `http://www.dubinko.info/writing/meta/`, 2002.

[21] M. Dürst and M. Suignard. Internationalized Resource Identifiers (IRIs) draft-duerst-iri-04. `http://www.w3.org/`

International/iri-edit/draft-duerst-iri-04, 2003.

[22] J. Grant and D. Beckett. RDF Test Cases . http://www.w3.org/TR/rdf-testcases/, 2003.

[23] R. Guha and T. Bray. Meta Content Framework Using XML . http://www.w3.org/TR/NOTE-MCF-XML-970624/, 1997.

[24] P. Hayes. RDF Semantics . http://www.w3.org/TR/rdf-mt/, 2003.

[25] A. Hopmann, S. Berkun, and G. Hatoun. Web Collections using XML . http://www.w3.org/TR/NOTE-XMLsubmit, 1997.

[26] H.S.Thompson, D. Beech, M. Maloney, and N. Mendelsohn. XML Schema Part 1: Structures. http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/, 2001.

[27] J. R. J. Boyer, D.E.Eastlake 3rd. Exclusive XML Canonicalization Version 1.0. http://www.w3.org/TR/2002/REC-xml-exc-c14n-20020718/, 2002.

[28] G. Karvounarkis, S. Alexaki, V. Christophides, D. Plexousakis, and M. Scholl. Rql: A declarative query language for rdf. In *Proceedings of the Eleventh International World Wide Web Conference*, pages 592–603, 2002.

[29] G. Klyne and J. Carroll. Resource Description Framework (RDF): Concepts and Abstract Syntax. http://www.w3.org/TR/rdf-concepts/, 2003.

[30] O. Lassila and R.R.Swick. Resource Description Framework (RDF) Model and Syntax Specification . http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/, 1999.

[31] A. Magkanarki, V. Tannen, V. Christophides, and D. Plexousakis. Viewing the semantic web through rvl lenses. In *The Semantic Web - ISWC 2003*, number 2870 in LNCS, pages 96–112. Springer, 2003.

[32] J. Marsh. XML Base. http://www.w3.org/TR/2001/REC-xmlbase-20010627/, 2001.

[33] B. McBride. RDF Issue Tracking. http://www.w3.org/2000/03/rdf-tracking/, 2003.

[34] S. Melnik. Bridging the Gap between RDF and XML. http://www-db.stanford.edu/~melnik/rdf/fusion.html, 1999.

[35] S. Melnik. Simplified Syntax for RDF. http://www-db.stanford.edu/~melnik/rdf/syntax.html, 1999.

[36] Z. Miklós, G. Neumann, U. Zdun, and M. Sintek. Querying semantic web resources using triple views. In *The Semantic Web - ISWC 2003*, number 2870 in LNCS, pages 517–532. Springer, 2003.

[37] L. Miller, A. Seaborne, and A. Reggiori. Three implementations of squishql, a simple rdf query language. In *The Semantic Web — ISWC 2002*, page 423ff., 2002.

[38] O.Lassila. Web Collections using XML . http://www.w3.org/TR/NOTE-pics-ng-metadata-970514.html, 1997.

[39] S. Palmer. RDF in HTML: approaches. http://infomesh.net/2002/rdfinhtml/, 2002.

[40] P. F. Patel-Schneider, P. Hayes, and I. Horrocks. OWL Web Ontology Language Semantics and Abstract Syntax. http://www.w3.org/TR/owl-semantics/, 2003.

[41] J. Reagle and D. Hazaël-Massieux. RDF in XHTML. http://www.w3.org/2003/03/rdf-in-xml.html, 2003.

[42] J. Robie. The syntactic web. In *XML 2001*, 2001.

[43] L. Wittgenstein. *Philosophical Remarks*. Blackwell, 1975. Appendix 2, F. Waismann's shorthand notes, Wed. 17 December, 1930, Neuwaldegg.