More RDF Summary

RDF Containers

There are 3 types of containers:

- A *Bag* (`rdf:Bag`) is a group of resources or literals. It may have duplicate members, and there's no significance to the order of the members.
- A *Sequence* or *Seq* (`rdf:Seq`) is a group of resources or literals. It may have duplicate members, and the order of the members is significant.
- An *Alternative* or *Alt* (`rdf:Alt`) is a group of resources or literals that are alternatives, typically for a single value of a property.

To show that a resource is of a container type, give it an `rdf:type` property whose value is one of the predefined resources `rdf:Bag`, `rdf:Seq`, or `rdf:Alt`. The statements don't construct containers (as in a programming language data structure) but describe containers that presumably exist.

A member is described by defining a container membership property with the container resource is its subject and the member is its object. These membership properties have names of the form `rdf:_`*n*, where *n* is a decimal integer > 0, with no leading 0's.

Example

Express in N3 that Ed and Bill (as a group) created a certain document

```
[ a foaf:Document;
  dc:creator [ a rdf:Bag;
               rdf:_1 exstaff:21;
               rdf:_2 exstaff:34 ] ] .
```

In RDF/XML

```
<foaf:Document>
  <dc:creator>
    <rdf:Bag>
      <rdf:_1 rdf:resource="http://www.example.org/staffid/21"/>
      <rdf:_2 rdf:resource="http://www.example.org/staffid/34"/>
    </rdf:Bag>
  </dc:creator>
</foaf:Document>
```

Numbered properties `rdf:_1`, `rdf:_2`, … are generated from the `rdf:li` convenience elements to form the graph.

Example

The last N3 example can be rewritten as

```
[ a foaf:Document;
  dc:creator [ a rdf:Bag;
               rdf:li exstaff:21, exstaff:34 ] ] .
```

Using the container membership properties just describes a container resource as having certain things as members and needn't say that the things described as members are the only members.

RDF Collections

RDF collections support describing groups containing only the specified members. A collection is a group of things (represented as a list structure in the RDF graph) constructed using a predefined collection vocabulary consisting of

- the predefined type `rdf:List`,
- the predefined properties `rdf:first` and `rdf:rest`, and
- the predefined resource `rdf:nil`.

Example

The following encodes in N3 the statement

The Handbook's creators are Bill, Ed, and Ken.

```
<http://www.example.org/docs/handbook> dc:creator
   [ rdf:first exstaff:21;
     rdf:rest [  rdf:first exstaff:34;
                 rdf:rest [ rdf:first exstaff:46;
                            rdf:rest rdf:nil ] ] ] .
```

Each bnode is implicitly of type `rdf:List`, but this isn't explicitly shown in the graph. RDFS defines properties `rdf:first` and `rdf:rest` as having subjects of type `rdf:List`, so the info that these nodes are lists can be inferred.

One direct translation into RDF/XML represents the bnodes as nested, anonymous `rdf:Description` elements.

```
<rdf:Description rdf:about="http://www.example.org/docs/handbook">
  <dc:creator>
    <rdf:Description>
      <rdf:first rdf:resource="http://www.example.org/staffid/21" />
      <rdf:rest>
        <rdf:Description>
          <rdf:first rdf:resource="http://www.example.org/staffid/34" />
          <rdf:rest>
            <rdf:Description>
              <rdf:first rdf:resource="http://www.example.org/staffid/46" />
              <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil" />
            </rdf:Description>
          </rdf:rest>
        </rdf:Description>
      </rdf:rest>
    </rdf:Description>
  </dc:creator>
</rdf:Description>
```

Another way to translate the above N3 into RDF/XML is to use the `rdf:nodeID` attribute to give IDs to bnodes

```
<rdf:Description rdf:about="http://www.example.org/docs/handbook">
  <dc:creator rdf:nodeID="cr1" />
</rdf:Description>
<rdf:Description rdf:nodeID="cr1">
  <rdf:first rdf:resource="http://www.example.org/staffid/21" />
  <rdf:rest rdf:nodeID="cr2" />
</rdf:Description>
<rdf:Description rdf:nodeID="cr2">
  <rdf:first rdf:resource="http://www.example.org/staffid/34" />
  <rdf:rest rdf:nodeID="cr3" />
</rdf:Description>
<rdf:Description rdf:nodeID="cr3">
  <rdf:first rdf:resource="http://www.example.org/staffid/46" />
  <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
</rdf:Description>
```

N3 provides a shorthand for collections: List all the members (separated by whitespace), enclosed within parentheses.

Example
The last N3 example can be rewritten as

```
<http://www.example.org/docs/Handbook>
   dc:creator (exstaff:21 exstaff:34 exstaff:46) .
```

In RDF/XML, a collection can be described by a property element that has the attribute `rdf:parseType="Collection"` and contains a group of nested elements representing the collection's members.

Example
The last RDF/XML example can be rewritten as

```
<rdf:Description rdf:about="http://www.example.org/docs/handbook">
    <dc:creator rdf:parseType="Collection">
      <rdf:Description rdf:about="http://example.org/ staffid/21"/>
      <rdf:Description rdf:about="http://example.org/ staffid/34"/>
      <rdf:Description rdf:about="http://example.org/ staffid/46"/>
    </dc:creator>
  </rdf:Description>
```

<u>RDF Reification</u>

RDF applications sometimes need to describe other RDF statements using RDF, e.g., to record provenance information about a statement. RDF provides a vocabulary for describing RDF statements. A description of a statement using this vocabulary is a *reification* of the statement. The RDF reification vocabulary consists of the type `rdf:Statement` and the properties `rdf:subject`, `rdf:predicate`, and `rdf:object`.

Asserting the reification isn't the same as asserting the original statement: neither implies the other. And reification isn't the same as quotation. The reification describes the relationship between a particular instance of a triple and the resources the triple refers to, while quotation provides a piece of language use that we can talk about (e.g., we can refer to its grammatical structure).

A reification of a statement is given by assigning the statement a URIref that's used as the subject in triples that identify it as a `rdf:Statement` and provide values for its `rdf:subject`, `rdf:predicate`, and `rdf:object` properties. This pattern of 4 triples is the standard *reification quad*.

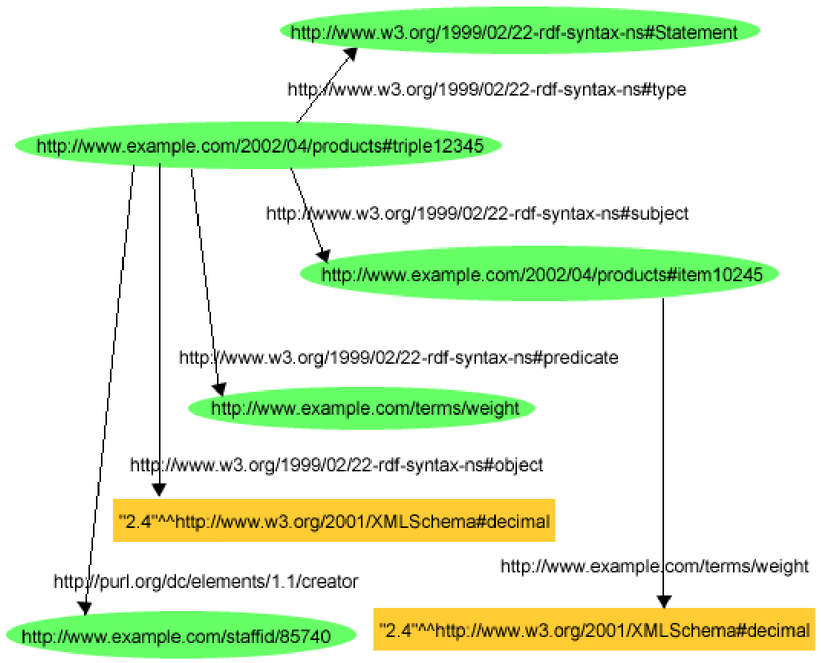
Example
Given the triple

```
  exproducts:item10245  exterms:weight  "2.4"^^xsd:decimal .
```

we produce the following reification quad along with the fact that John Smith made this statement.

```
exproducts:triple12345   rdf:type        rdf:Statement .
exproducts:triple12345   rdf:subject     exproducts:item10245 .
exproducts:triple12345   rdf:predicate   exterms:weight .
exproducts:triple12345   rdf:object      "2.4"^^xsd:decimal .
exproducts:triple12345   dc:creator      exstaff:85740 .
```

The RDF graph for the quad, the original statement, and the fact that John Smith made this statement is shown on the next page. The original triple is in the graph but is not connected with `exproducts:triple12345`.


http://www.w3.org/1999/02/22-rdf-syntax-ns#Statement
http://www.w3.org/1999/02/22-rdf-syntax-ns#type
http://www.example.com/2002/04/products#triple12345
http://www.w3.org/1999/02/22-rdf-syntax-ns#subject
http://www.example.com/2002/04/products#item10245
http://www.w3.org/1999/02/22-rdf-syntax-ns#predicate
http://www.example.com/terms/weight
http://www.w3.org/1999/02/22-rdf-syntax-ns#object
"2.4"^^http://www.w3.org/2001/XMLSchema#decimal
http://www.example.com/terms/weight
http://purl.org/dc/elements/1.1/creator
http://www.example.com/staffid/85740
"2.4"^^http://www.w3.org/2001/XMLSchema#decimal

RDF doesn't provide a built-in way to indicate how a URIref like `exproducts:triple12345` is associated with a particular statement or graph. But neither is there a built-in way of indicating how a URIref like `exproducts:item10245` is associated with an actual tent. In general, associating specific URIrefs with specific resources (here statements) must be done outside of RDF.

Since reification is intended for expressing properties such as provenance information, the subject of the reification triples is conventionally assumed to identify a particular instance of a triple in a particular RDF document. Although there could be several instances with the same triple structure in different documents, these properties need to be applied to specific instances of triples. We need some way to associate the subject of the reification triples with an individual triple in some document. RDF itself provides no way to do this, so we need a convention among users.

In RDF/XML, the last example becomes

```
<rdf:Description rdf:ID="item10245">
  <exterms:weight rdf:datatype="&xsd;decimal">2.4</exterms:weight>
</rdf:Description>
<rdf:Statement rdf:about="#triple12345">
  <rdf:subject rdf:resource="http://www.example.com/2002/04/products#item10245"/>
  <rdf:predicate rdf:resource="http://www.example.com/terms/weight"/>
  <rdf:object rdf:datatype="&xsd;decimal">2.4</rdf:object>
  <dc:creator rdf:resource="http://www.example.com/staffid/85740"/>
</rdf:Statement>
```

We can use `rdf:ID` not only in a `rdf:Description` element but also in a property element. Doing so automatically produces a reification of the triple that the property element (along with the subject) represents.

Example

In RDF/XML, the last example becomes

```
<rdf:Description rdf:ID="item10245">
  <exterms:weight rdf:ID="triple12345" rdf:datatype="&xsd;decimal">
    2.4
  </exterms:weight>
</rdf:Description>
<rdf:Description rdf:about="#triple12345">
  <dc:creator rdf:resource="http://www.example.com/staffid/85740"/>
</rdf:Description>
```

More on Structured Values: `rdf:value`

In the RDF model, a qualified property value is considered just another kind of structured value: there's a property for the typed literal representing the decimal value and one for the unit. RDF provides a predefined `rdf:value` property to describe the main value.

Example

The following RDF/XML asserts that the tent weighs 2.4 kilograms.

```
<rdf:Description
       rdf:about="http://www.example.com/2002/04/products#item10245">
  <exterms:weight rdf:parseType="Resource">
    <rdf:value rdf:datatype="&xsd;decimal">2.4</rdf:value>
    <exterms:units
           rdf:resource="http://www.example.org/units/kilograms"/>
  </exterms:weight>
</rdf:Description>
```

Here the `rdf:parseType="Resource"` attribute in the `exterms:weight` property element indicates a bnode as the value of the `exterms:weight` property and that the enclosed elements (`rdf:value` and `exterms:units`) describe properties of that bnode.

In general, use the `rdf:value` property to give the main value, and use additional properties to identify the classification scheme or other information further describing the value. RDF doesn't associate any special meaning with `rdf:value`—it's a convenience for use in these commonly-occurring situations.

XML Literals

To facilitate writing XML literals, RDF/XML provides a 3rd value of the `rdf:parseType` attribute: `rdf:parseType="Literal"` indicates that the contents of the element are to be interpreted as an XML fragment. This lets an RDF user avoid dealing directly the various transformations required for representing XML fragments in the canonical way.

Example

```
<rdf:Description rdf:ID="book12345">
  <dc:title rdf:parseType="Literal">
    <span xml:lang="en">
      The <em>&lt;br /&gt;</em> Element Considered Harmful.
    </span>
  </dc:title>
</rdf:Description>
```