3.1 Basics: RDF/S data model and semantics

Contents

3.1.1 The Concepts of RDF

RDF is a formalism for knowledge representation reminiscent of the Entity-Relationship model developped in the 70es of the 20th century for databases. In contrast to thÃs model, developped at a time where computers were not networked, RDF has been conceived for distributed knowledge representation over the Web. RDF is the primary language of the W3C for its Semantic Web activities.

3.1.1.1 Triple, Resources, and Literals

RDF is based on triples, also called statements, like the following:

```
ThomasMann isAuthorOf TheMagicMountain
ThomasMann hasAward   NobelPrize
```

Triples have the form "subject property object". Properties are also called "predicates".

Triples are statements about "resources". A resource is either something uniquely characterized by a URI, or a blank node, that is an anonymous resource. A blank node can be understood as a resource the existence of which is know, its exact nature oder identity is not.

In an RDF specification, a (non-anonymous) resource can be associated with an arbitrary URI: RDF does not require that a file that can be retrieved at the given URI is related in some way to the resource, even thouth this is often the case in RDF specifications. Therefore, authors and consumers of an RDF specification must agree on the semantics of URIs.

URI are global resource identifiers that denote the same resource in different RDF specifications. In contrast blank nodes are local to the RDF specification they occur in. In other words, the same blank node identifier in two distinct RDF specifications does not necessarily denote the same (anonymous) resource.

Subject, property and objects of an RDF triple are as follows:

The subject of a triple is a resource, that is, a URI or a blank node. The property of a triple is a non-anonymous resource, that is, a URI. The object of a triple is a URI, a blank node or a literal.

A "literal" is a value like "23" or "word". RDF distingusihes between "plain" and "typed" literals:

- A plain literal is a string possibly assigned an optional language tag (using the lang attribute and values specified in Section 2.12 "Language Identification" of the W3C recommendation XML 1.0, for example xml:lang="en", xml:lang="de" oder xml:lang="" if the absence of a language

specification is to be made explicit, assuming that "xml" is the namespace prefix for the URI denoting XML, that is, http://www.w3.org/XML/1998/namespace).

- A typed literal is a string necessarily assigned a datatype URI (specifying a datatype of XML Schema. If xsd is a namespace prefix bound to the URI of XML Schema, then xsd:boolean specifiies for example the boolean datatype of XML Schema which has values "0" or "false" and "1" or "true").

Subjects and objects denote "things" that either can be further specified in RDF (the resources) or that are self explanatory and cannot be further specified in RSF (the literals). A literal being no resource can be neither a subject nor a property. The RDF query langage SPARQL and many recent RDF applications drop the restriction that literals cannot be subjects of triples.

Properties are binary relations between "things". An salient characteritic of RDF is that properties can be "things" further specified in RDF:

```
ThomasMann isAuthorOf   TheMagicMountain
isAuthorOf rdfs:type    rdfs:Property
isAuthorOf rdfs:domain  Persons
isAuthorOf rdfs:range   Artefacts
isAuthorOf refersTo     Creation
```

The RDF triple "isAuthorOf rdfs:type rdfs:Property" expresses that "isAuthorOf" is an RDF property. "type", "range" and "domains" are properties in a pre-defined RDF vocabulary called RDFS, "Property" is a class in the RDFS vocabulary. This is indicated by usig the namespace prefix "rdfs" (as a shorthand notation for the URI http://www.w3.org/2000/01/rdf-schema# used to charaterize RDFS).

3.1.1.2 Reification

RDF offers a mean, called "reification", to make RDF statements about RDF statements. The reification of a triple, also called statement, t= s p o consists in generating:

- a blank node BN aiming at representing the triple t.
- a first triple stating that BN has subject s.
- a second triple stating that BN has property p.
- a third triple stating that BN has object o.

To this aim, the following vacabulary is provided by RDF:

```
rdf:Statement rdf:subject rdf:predicate rdf:object
```

It is used as follows (in Turtle syntax), assuming that s p and o denote URIs:

```
_:xxx rdf:type rdf:Statement .
_:xxx rdf:subject <s> .
_:xxx rdf:predicate <p> .
_:xxx rdf:object <o> .
```

Reification makes it possible to express for example "Anna said Bob knows Clara"

3.1.1.3 Containers and Collections for Aggregation

RDF has "containers" to aggregate values in bags, sequences, alternatives, and collections.

- An RDF bag describes an unordered list of values which may contain duplicate values.
- An RDF sequence describes an ordered list of values which may contain duplicate values.
- An RDF alternative describes (unodered) alternative values.

The specification of a bag, sequence or alternative does not preclude that further values, not mentioned in the given specification, might also belong to the bag, sequence or alternative.

In addition to containers, RDF has collections. SCollections differ from containers as follows: The specification of an RDF collection precludes membership of further values in the collection that are not mentioned in the collection specification.

3.1.1.4 Classes and inheritance

RDF Schema, short RDFS, though, gives rise to specify classes, membership into clases and inheritance between classes, like for example:

```
Father and Person are classes
Fathers are Persons
Al hasFather Ben
```

Thus, RDF can be used for specifying classes of RDF resources and inheritance between such clases: RDF can be used for specifying ontologies, also called schemas or vocabularies.

Two approaches have been considered for enhancing RDF and RDFS with richer specifications: OWL, using classical logic negation, and rules, using non-monotonic negation.

3.1.1.5 RDF Graph and Context

A triple "sub prop obj" is represented as a (directed) graph with nodes labelled by "sub" and "obj2 and with a (directed) arc from node sub to node obj which is labelled by "prop".

A set of RDF triples is a RDF graph the nodes of which are the triple subjects and objetcs related by property-labelled arcs. In an RDF graph, a URI uniquely identifies a single node. In contrast, each instance of a literal yields a node labelled by this literal. Thus, in an RDF graph, there is a single node labelled by a given URI, while there might be several distinct nodes labelled by a same literal.

EXAMPLE:

The following uses the Turtle syntax:

- _:i denotes a blank nodes
- lib is a namespace prefixe denoting http://worldtexts.org
- geo is a namespace prefic denoting http://worldplaces.org
- rdf a namespace prefix denoting http://www.w3.org/1999/02/22-rdf-syntax-ns#
- expressions between quotes denote literals

```
_:1 isAuthotOf      lib:MagicMountain
```

```
                _:1 hasFirstName    "Thomas"
                _:1 hasLatName      "Mann"
                _:1 citizenOf       geo:USA
                _:1 citizenOf       geo:Germany
                _:1 awarded         _BN:2
                _:2 awardName       "NobelPrize"
                _:2 awardYear       "1929"
      MagicMountain rdf:type         lib:Book
      MagicMountain titled           "Magic Mountain"
               Book rdf:SubClassOf   lib:Publication
```

In an RDF graph, there is a unique node labelled by a given blank node.

URI are global resource characerization, that is, a same URI denote the same resoiurce in distinxt RDF graphs. Blank node, in contrast, a local resource characterizations: Two occurrences of a same blank node in distinct RDF graphs do not necessarily characterize the same resource.

EXAMPLE (from the RDF primer or Wikipedia-en)

A RDF graph can be named.

The "context" of an RDF triple is the (name of the) graph it belongs to.

3.1.1.6 Serializations

RDF has a couple commonly used serializations, that is, syntaxes. RDF is not ony an XML application, that is, there is an XML serialization of XML, but there also are other non-XML serializations of RDF.

An example (Turtle):

```
@prefix : <http://www.example.org/> .
:Anna    a           :Person .
:Anna    :hasMother  :Bella .
:Anna    :hasFather  :Charlie .
:Charlie :hasBrother :Dave .
```

The same example in RDF/XML:

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:ns="http://www.example.org/#">
  <ns:Person rdf:about="http://www.example.org/#Anna">
    <ns:hasMother rdf:resource="http://www.example.org/#Bella" />
    <ns:hasFather>
      <rdf:Description rdf:about="http://www.example.org/#Charlie">
        <ns:hasBrother rdf:resource="http://www.example.org/#Dave" />
      </rdf:Description>
    </ns:hasFather>
  </ns:Person>
</rdf:RDF>
```

Currently commonly used serializations of RDF are:

- an XML application, called RDF/XML, which is verbose and difficult to read for humans

- Turtle, appropriate for humans,
- RDFa, for extending any markup language such as HTML with RDF annotations in the form of attribute-vaue pairs, primarily for machine processing.

RDFa is a language for expressing RDF specifications through attributes of a markup language, considered for search by Google and Yahoo, for example HTML:

"Using a few simple XHTML attributes, authors can mark up human-readable data with machine-readable indicators for browsers and other programs to interpret. A web page can include markup for items as simple as the title of an article, or as complex as a user's complete social network."

(from the "RDFa Primer")

A technicality is worth mentioning: Because XML constains the syntax of qualified names (QNames) used in RDF/XML for representing subjects, properties and objects constrains more than RDF the syntax of subject, properties and objects, some RDF graphs are not representable with RDF/XML.

3.1.1.7 RDF Storage and Querying

RDF triples, or RDF graphs, are commonly stored

- in relational databases
- in triple stores, that is database or storage systems specifically designed for RDF.
- in quad stores, that is, like triple stores database or storage systems specifically designed for RDF that, in contrast with triple stores associate to a triple the name of the RDF graph they belong to.

There is a query language for RDF, caled SPARQL, for retrieving data from triple or quad stores. SPARQL is SQL-like and a recommendation of the W3C of January 15, 2008. SPARQL has four types of queries: `SELECT`, `ASK`, `DESCRIBE`, and `CONSTRUCT`.

`SELECT` queries serve for retrieving resources and yield tables of variable bindings as answers. Like mSQL, SPARQL has a `WHERE` clause for expressing RDF relationships.

`ASK` is for yes/no queries, `DESCRIBE` for meta-data on an RDF graph, and `CONSTRUCT` for RDF answers.

3.1.1.8 RDF vs. Relational Databases

RDF triples are rather similar to relational database tuples. RDF differs from relational databases, however, in a few significant aspects:

- Schema: Relational databaees require predifned and fixed schema. RDF in contrast can accomodate new triples, new resources, new literal datatypes at any time. In this respect, RDF vcan be seen as similar to NoSQL databases. In other words, relational databases are conceptually closed, while RDF graphs is conceptually open
- Value vs. identity-oriented: rRelatiional databases are value-oriented: They have no means for expressing reaource or object identity. RDF in contrast has resource object identies both, global with URIs and local with blank nodes
- Meta-Level: In a relational database a relation (or table) is no object or resource accessible in the data model. A RDF property in contrast is an RDF resource.
- Classes and inheritace: Relational databases have no classes, no inheritance, RDFS has clasex and inheritace. (Note that object-relational databases have classes anf inheritance.)

3.1.1.9 RDF Applications

Popular applications are (see also "Applications" at Wikipedia-en "RDF"):

- RSS (RDF Site Summary)
- FOAF
- SKOS
- SIOC (Semantically-Interlinked Online Communities, http://sioc-project.org/)
- Linked Data (http://linkeddata.org/, http://esw.w3.org/topic/SweoIG/TaskForces/CommunityProjects/LinkingOpenData)
- DBpedia as a central part of linked data, a conversion of Wikipedia's content into RDF triples.

3.1.1.9 Further Remarks

RDF has ... three semantics of increasing complexity defined as model theories that depart from the standard Tarski-style model theories.

The RDF data model and syntax has been published firsrt as a W3C Recommendation in 1999. The currwnrt specification, published in 2004, is a revision of that initial specification.

3.1.1.10 References

The specification of RDF consists of the following W3C recommendations:

- RDF Primer, W3C Recommendation 10 February 2004, http://www.w3.org/TR/rdf-primer/
- Resource Description Framework (RDF): Concepts and Abstract Syntax, W3C Recommendation 10 February 2004, http://www.w3.org/TR/rdf-concepts/
- RDF Semantics, W3C Recommendation 10 February 2004, http://www.w3.org/TR/rdf-mt/
- RDF/XML Syntax Specification (Revised), W3C Recommendation 10 February 2004, http://www.w3.org/TR/rdf-syntax-grammar/
- RDF Vocabulary Description Language 1.0: RDF Schema, W3C Recommendation 10 February 2004, http://www.w3.org/TR/rdf-schema/
- RDF Test Cases, W3C Recommendation 10 February 2004, http://www.w3.org/TR/rdf-testcases/

Note, in addition, the following RDF-related W3C recommendations:

- RDFa Primer, Bridging the Human and Data Webs, W3C Working Group Note 14 October 2008, http://www.w3.org/TR/xhtml-rdfa-primer/
- RDFa in XHTML: Syntax and Processing, A collection of attributes and processing rules for extending XHTML to support RDF, W3C Recommendation 14 October 2008, http://www.w3.org/TR/rdfa-syntax

Good for learning:

- Joshua Tauberer. Quick Intro to RDF, http://rdfabout.com/quickintro.xpd
- Joshua Tauberer, RDF in Depth, http://rdfabout.com/intro/
- Joshua Tauberer, What is RDF and what is it good for? 2008, http://www.rdfabout.com/intro/?section=contents

3.1.2 RDF Serializations

Overview of commonly used RDF serilizations and their usage:

- RDF/XML is the serialization originally developed by the RDF Core Working Group. It has been published as a W3C recommendation and revised in 2004. RDF/XML is verbose, offers many alternatives which contribute to make it complicated.

The character encoding of a RDF/XML serialization can be set by the XML encoding attribut. The MIME Type of a RDF/XML serialization is: application/rdf+xml The usual file extension of an RDF/XML serialization is: .rdf

- N3 is an alternative compact textual syntax with various additions such as rules introduced by Tim BL in 1998 so as to overcome the verbosity and questionable choices of RDDF/XML. N3, like Turtle, offers abbreviations, @prefix, and grouping of multiple subject-property pairs and objects. N3 is strictly more expressive than RDF/XML, that is, not every N3 specification can be translated in RDF/XML.
- A fragment of N3 with the same expressivity as RDF/XML has been defined by Dave Backett as Turtle.

The character encoding of an N3 RDF serialization is: UTF-8
The MIME Type of an N3 RDF serialization is: text/rdf+n3
The usual file extension is: .n3

- N Triples has been defined for the RDF Test Cases and other RDF recommendations based on N3 because RDF/XML was too complicated for such documents. In contrast to N3, N Triples does not extend the expressive power of RDF/XML (with for example rules). In contrast to N3 and Turtle, N Triples does not offer abbreviations, @prefix, and grouping of multiple subject-property pairs and objects (also called stripes).

The character encoding of a N Triples RDF serialization is 7-bit US-ASCII.
The MIME Type of N Triples is: text/plain
The usual file extension of a N Triples RDF serialization is: .nt

- Turtle is a restriction of N3: In contrat to N3 which extends XML/RDF (among others with rules), Turtle is exactly as expressive as RDF/XML. Like N3, Turtle extends N Triples with abbreviations, @prefix, and grouping of multiple subject-property pairs and objects.

Turtle does not have the "path syntax" of N3.
Turtle has been developped by Dave Beckett.

The character encoding of a Turtle RDF serialization is: UTF-8
The MIME Typeof a Turtel serialization is one of: application/turtle and application/x-turtle
The usual file extension of a Turtle RDF serialization is: .ttl

- The so-called "graph patterns" of SPARQL are similar to a RDF serialization. They are similar to N3 and Turtle but offer in addition variables and literals as subjects.

- RDFa is not an RDF serialization proper. Instead, it is an attribute-value syntax for adding RDF meta-data to the components of any structured document (like an HTML or XML document).

Reference:

Andreas Langegger: For everybody who is confused about the various RDF serializations/syntaxes, Blog "Home of AndyL", 2008, http://www.langegger.at/content/everybody-who-confused-about-various-rdf-serializationssyntaxes

In the remaining of this section, RDF/XML, Turtle and RDFa are introduced. The graph patterns of SPARQL are introduced later with SPARQL.

3.1.2.1 RDF/XML

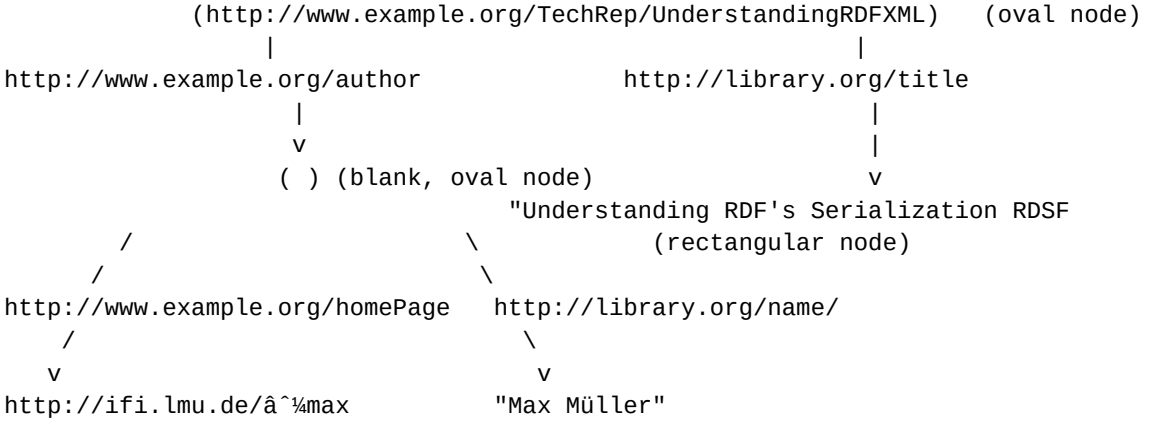
An RDF/XML specification is an XML document the root of which is an rdf:RDF element. (In case the RDF graph to be specified has a root, that is, a single node from which all nodes of the RDF graph can be reached, then the rdf:RDF element can be dispensed with.)

```
<?xml version="1.0" encoding="utf-8" ?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:...="..."
         ...
         xmlns:...="...">
  ...
</rdf:RDF>
```

Note the namespace prefix declarations, the first of which "rdf" refers to the RDF/XML namespace, the following depend on the RDF grahp to be specified.

With RDF/XML,

- RDF Nodes and RDF properties (or predicates) are represented in XML as element names, attribute names, element contents and attribute values.
- URI references are represented as XML "qualified names" or QNames, that is a "namespace name" consisting of a URI reference and a local name. Possibly, URI references can in addition have a namespace prefix. If they have no namespace prefix, then they are declared with the default namespace.
- RDF literals, which can only be object nodes, are represented either as XML element text content or as XML attribute values.
- RDF graphs are represented as collections of paths, themselves represented as sequences of elements inside elements which alternate between elements for nodes and predicate arcs. Such sequences are called "(node/arc) stripes".



An RDF/XML (striped) representation of the leftmost path:

```
<?xml version="1.0" encoding="utf-8" ?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:ex="http://www.example.org/"
         xmlns:lib="library.org/name">

   <rdf:Description
     rdf:about="http://www.example.org/TechRep/UnderstandingRDFXML">
     <ex:author>
       <rdf:Description>
         <ex:homePage>
           <rdf:Description rdf:about="http://ifi.lmu.de/â¼max">
           </rdf:Description>
         </ex:homePage>
         <lib:name>
            Max MÃ¼ller
         </lib:name>
       </rdf:Description>
     </ex:author>
   </rdf:Description>

</rdf:RDF>
```

Note that no identifier is assigned to the author blank node in the RDF/XML specification.

This RDF/XML specificaton of a single path f the RDF graph given above can be extended into an RDFG/XML specification of the full RDF graph as follows:

```
<?xml version="1.0" encoding="utf-8" ?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:ex="http://www.example.org/"
         xmlns:lib="http://library.org/title">

   <rdf:Description
     rdf:about="http://www.example.org/TechRep/UnderstandingRDFXML">
     <ex:author>
       <rdf:Description>
         <ex:homePage>
           <rdf:Description rdf:about="http://ifi.lmu.de/â¼max">
           <lib:name>
              Max MÃ¼ller
           </lib:name>
           </rdf:Description>
         </ex:homePage>
       </rdf:Description>
     </ex:author>
     <lib:title>
        Understanding RDF's Serialization RDSF/XML
     </lib:title>
   </rdf:Description>
```

```
</rdf:RDF>
```

Note that

- the "author" blank node still is assigned no identifier
- the plain literal value "Understanding RDF's Serialization RDSF/XML" is specified as the (string or textual) content of the "title" property element.

In some cases, blank nodes identifiers are necessary (when they occur too often in the RDF graph preventing a tree-shaped striped representation of that graph with a single occurrence of each blank node.) A blank node can be given identifiers (local the the RDF/XML specification) as follows, where there is a rdf:Description element for each arc, and therefore several "rdf:Description" elements with the same value for the attribute "rdf:about".

```
<?xml version="1.0" encoding="utf-8" ?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:ex="http://www.example.org/"
         xmlns:lib="http://library.org/title">

   <rdf:Description
     rdf:about="http://www.example.org/TechRep/UnderstandingRDFXML">
     <ex:author>
       <rdf:Description rdf:nodeID="maxM">
         <ex:homePage>
           <rdf:Description rdf:about="http://ifi.lmu.de/â¼max">
           </rdf:Description>
         </ex:homePage>
         <lib:name>
            Max MÃ¼ller
         </lib:name>
       </rdf:Description>
     </ex:author>
     <lib:title>
        Understanding RDF's Serialization RDSF/XML
     </lib:title>
   </rdf:Description>

</rdf:RDF>
```

Note that URIs specifying nodes *cannot* be abbreviated by using namespace, as in the following *incorrect* specification:

```
   <rdf:Description rdf:about="ifi:â¼max">
   </rdf:Description>
```

Indeed, suh a usage introduces markup, the namespace prefix, into the content, here the attribbut value, of an XML document. The recognition of markup in content is not part of XML. It is possible but must be agreed upon by produceras and users of an XML application.

Even thouth the above-mentioned introductioin of namespace prefixes in attribute values of an RDF/XML specification is not part of the RSF/XML formalims, it is common (with RDF/XML and, among others, with XML Schema too).

Instead of the striped RDF/XML specification given above, one can instead specify the following one which makes a blank node identifier for the "author" blank node necessary:

```
<?xml version="1.0" encoding="utf-8" ?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:ex="http://www.example.org/"
         xmlns:lib="http://library.org/title">

   <rdf:Description
     rdf:about="http://www.example.org/TechRep/UnderstandingRDFXML">
     <ex:author>
       <rdf:Description rdf:nodeID="maxM">
         <ex:homePage>
           <rdf:Description rdf:about="http://ifi.lmu.de/â¸max">
           </rdf:Description>
         </ex:homePage>
       </rdf:Description>
       <rdf:Description rdf:nodeID="maxM">
         <lib:name>
            Max MÃ¼ller
         </lib:name>
       </rdf:Description>
     </ex:author>
   </rdf:Description>
   <rdf:Description
     rdf:about="http://www.example.org/TechRep/UnderstandingRDFXML">
     <lib:title>
        Understanding RDF's Serialization RDSF/XML
     </lib:title>
   </rdf:Description>
</rdf:RDF>
```

RDF typed literals (after the datatypes of XML Schema) are specified as follows:

```
<?xml version="1.0" encoding="utf-8" ?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:ex="http://www.example.org/"
         xmlns:lib="http://library.org/title">

   </rdf:Description>
   <rdf:Description
     rdf:about="http://www.example.org/TechRep/UnderstandingRDFXML">
     <lib:title>
        Understanding RDF's Serialization RDSF/XML
     </lib:title>
     <lib:size rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
        789
     </lib:size>
   </rdf:Description>
</rdf:RDF>
```

In this example, the object 789 of the triple

```
   http://www.example.org/TechRep/UnderstandingRDFXML   lib:size   789
```

is given the XML Schema datatype integer.

Containers are expressed in RDF/XML in two different ways (using rdf:_n or rdf:li elements) as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <rdf:Seq rdf:about="http://example.org/ThomasMannsBooks">
    <rdf:_1 rdf:resource="http://example.org/TheClown"/>
    <rdf:_2 rdf:resource="http://example.org/Buddenbrooks"/>
    <rdf:_3 rdf:resource="http://example.org/MagicMountain"/>
  </rdf:Seq>
</rdf:RDF>

<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <rdf:Seq rdf:about="http://example.org/ThomasMannsBooks">
    <rdf:li rdf:resource="http://example.org/TheClown"/>
    <rdf:li rdf:resource="http://example.org/Buddenbrooks"/>
    <rdf:li rdf:resource="http://example.org/MagiocMountain"/>
  </rdf:Seq>
</rdf:RDF>
```

Instead of rdf:Seq, which denote a (ordered) sequence, one can use rdf:Bag which denote an unordered bag (or multiset), or rdf:Alt which denote alternative value for a same concept.

Collections are expressed in RDF/XML as follows (wrongly assuming that Thomas Mann had only written the three books mentioned):

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:ex="http://example.org/stuff/1.0/">
  <rdf:Description rdf:about="http://example.org/AllThomasMannsBook">
    <ex:hasBook rdf:parseType="Collection">
      <rdf:Description rdf:about="http://example.org/TheClown"/>
      <rdf:Description rdf:about="http://example.org/Buddenbrooks"/>
      <rdf:Description rdf:about="http://example.org/MagicMountain"/>
    </ex:hasBook>
  </rdf:Description>
</rdf:RDF>
```

The reification of the triple

```
   http://example.org    ex:name   "Example"
```

is expressed in RDF/XML as follows:

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
```

```
        xmlns:ex="http://example.org/"
        xml:base="http://example.org/triples/">
  <rdf:Description rdf:about="http://example.org/">
    <ex:name rdf:ID="t234">Example</ex:prop>
  </rdf:Description>
</rdf:RDF>
```

The reified triple is assigned the URI http://example.org/tropleS#t234

RDF/XML alloows many shorthands notations like the following:

- If the object R of a property is a URI which not a subject (in another triple), then R can be specified as value of the rdf:resource attribute of the property instead of with a Description element:

```
<rdf:Description rdf:about="http://www.w3.org/TR/rdf-syntax-
grammar">
  <ex:author>
    <rdf:Description>
      <ex:homePage rdf:resource="http://ifi.lmu.de/â¼max"/>
      <ex:name>Max MÃ¼ller</ex:name>
    </rdf:Description>
  </ex:author>
  <lib:title>Understanding RDF's Serialization RDSF/XML</lib:title>
</rdf:Description>
```

- If the object of a property is a plain literal L, then L can be specified as value of an attribute of the Description element (expressing the property) the attribute name being the property's name:

```
<rdf:Description rdf:about="http://www.w3.org/TR/rdf-syntax-
grammar">
  <ex:author>
    <rdf:Description ex:name="Mx MÃ¼ller">
      <ex:homePage rdf:resource="http://ifi.lmu.de/â¼max"/>
    </rdf:Description>
  </ex:author>
  <lib:title>Understanding RDF's Serialization RDSF/XML</lib:title>
</rdf:Description>
```

References:

RDF/XML Syntax Specification (Revised), W3C Recommendation 10 February 2004, http://www.w3.org/TR/rdf-syntax-grammar/

http://www.w3schools.com/rdf/default.asp

3.1.2.2 Turtle

Turtle has been derived from N3 and N Triples, that had been developped because RDF/XML is complicated, hard to read and hard to use.

Turtle is a simplification of N3: in contrat to N3, Turtle does not extend RDF/XML with additional featuresd (such as rules). Turtle does not have the "path syntax" of N3.

Turtle extends N Triples with usefull features: abbreviations, @prefix, and grouping of multiple subject-property pairs and objects.

Triples are represented as follows (note the fullstop after each triple):

```
<http://www.example.org/TechRep/UnderstandingTurtle>
                      <http://library.org/author> "Max Müller" .
```

Namespace prefix can be declared an used as follows:

```
@prefix ex  <http://www.example.org/>
@prefix lib <http://library.org/>
ex:TechRep/UnderstandingTurtle lib:author "MaxMüller" .
```

A default, or base, prefix which does not have to be explicitly mentioned can be specified:

```
@base ex  <http://www.example.org/>
@prefix lib <http://library.org/>
:TechRep/UnderstandingTurtle lib:author "Max Müller" .
```

Typed literals are expressed as follows, using the namespace prefix xsd for http://www.w3.org/2001/XMLSchema

```
"Max Müller"^^xsd:string
"23"^^xsd:integer
```

like for example in the following specification:

```
@base ex  <http://www.example.org/>
@prefix lib <http://library.org/>
@prefix xsd <http://www.w3.org/2001/XMLSchema>
:TechRep/UnderstandingTurtle lib:author "Max Müller"^^xsd:string .
:person/MaxMueller           :age       "23"^^xsd:int .
```

Turtle has a shorthand notation for typing resources. Instead of

```
@base   ex   <http://www.example.org/>
@prefix rdf  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
:person/MaxMueller rdf:type :male .
```

one can also write:

```
@base   ex   <http://www.example.org/>
:person/MaxMueller a :male .
```

Several triples with the same subject can be striped, or factorized, as
follows (note the several ";" and the final "."):

```
@base   ex   <http://www.example.org/>
```

```
@prefix lib <http://library.org/>
@prefix xsd <http://www.w3.org/2001/XMLSchema>
:person/MaxMueller a               :male ;
                   :age            "23"^^xsd:int ;
                   lib:hasAuthored  :TechRep/UnderstandingTurtle .
```

Blanknodes are expressed

- as [], when no blank node identifier is needed,
- as as _:nodeID, where nodeID is an indentifier (enforcing a certain syntax), when a blank node identifier is needed.

The following two examples illustrate both cases:

```
@base   ex   <http://www.example.org/>
@prefix lib <http://library.org/>
[]   a               :male ;
     :age            "23"^^xsd:int ;
     lib:hasAuthored  :TechRep/UnderstandingTurtle .

@base   ex   <http://www.example.org/>
@prefix lib <http://library.org/>
_:1   a               :female ;
     :age            "23"^^xsd:int ;
     :knows          _:2 .
_2:   a              :male ;
     :knows          _1 .
```

The first syntax for blank nodes can be used as follows, when the blank node is both an object in a triple, and subject of in other triples:

```
@base   ex   <http://www.example.org/>
@prefix lib <http://library.org/>
:Mary   a               :female ;
         :age            "23"^^xsd:int ;
         :knows          [
                            a      :male;
                            :knows :Mary
                         ] .
```

This last notation is striped. It might be seen as harmful, for it requires significant changes if a triple is added the object of which is the blank node.

Reference:

Turtle - Terse RDF Triple Language, W3C Team Submission 14 January 2008, http://www.w3.org/TeamSubmission/turtle/
(Sections 8, 9 and 10 compare Turtle respectively with N Troiple, N3 and SPARQL.)

Haystack Blog: A Quick Tutorial on the Turtle RDF Serialization, http://groups.csail.mit.edu/haystack/blog/2008/11/06/a-quick-tutorial-on-the-tutrle-rdf-serialization/

Tim Berners-Lee. Notation 3, 1999, revised 2009, http://www.w3.org/DesignIssues/Notation3.html
(The Appendix "N3 Subsets" comparises N3 with various RDFserializations.)

3.1.2.3 RDFa

RDFa is an extension to XHTML specifyimng attributes for adding RDF meta-data to an XHTML document.

History

RDFa has been proposed in 2004 by Mark Birbeck in a W3C note entitled "XHTML and RDF". Imn spite of this explixit reference to XHTML, RDFa has been thought od as a means to add a metadata to any XML application. In April 2007 the XHTML 2 Working Group produced a module to support RDF annotation within the XHTML 1 family including an extension of XHTML 1.1 clled XHTML+RDFa 1.0. In October 2007 the public Working Draft entitled "RDFa in XHTML: Syntax and Processing" was published which became a W3C recommendation in October 2008. A "RDF Primer" was publihed in June 2008.

RDFa Attributes that can be assigned to XML elements

- about: a URI or CURIE specifying the resource the metadata is about (subject of RDF triples)
- rel and rev: specify a relationship and reverse-relationship with another resource
- href, src and resource: specify a partner resource
- property: specify a property (for the content of the XML element)
- content (optional): overrides the content of the element when using the property attribute
- datatype (optional, for use with the "property" attribute): specifies a datatype of text
- typeof (optional): specifies RDF type(s) of the subject (the resource that the metadata is about).

RDFa meta-data added to XHTML:

The following is a div element specifying Dublin Core metadata (meta-data about publications). It can be included in an XHTML document (at places whwre div elements are allowed):

```
<div xmlns:dc="http://purl.org/dc/elements/1.1/"
  about="http://www.example.com/books/MagicMountain">
  <span property="dc:title">Magic Mountain</span>
  <span property="dc:creator">Thomas Mann</span>
  <span property="dc:date">1924-11</span>
</div>
```

The following shows how portions of text of an XHTML document can be
assigned RDFa meta-data:

```
<p xmlns:dc="http://purl.org/dc/elements/1.1/"
   about="http://www.example.com/books/MagicMountain">
  In the novel
  <cite property="dc:title">Magic Mountain</cite>,
  <span property="dc:creator">Thomas Mann</span>
  reflects his impressions when, suffering from a lung complaint, he
```

```
  was confined to a sanatorium in Davos, Switzerland.
  The novel, which is considered to be one of the most influential
  works of 20th century German literature has benn published in
  <span property="dc:date" content="1924-11">November 1924</span>.
</p>
```

Related to RDFa are:

- other microformats such as hCalendar nd XNF (XHTML Friends Network, for social relationships)
- eRDF (Embedded RDF), an alternative to RDFa
- GRDDL, an apporach to extract annotated data from XML documents and transform it into an RDF graph

References:

RDFa Primer - Bridging the Human and Data Webs, W3C Working Group Note 14 October 2008, http://www.w3.org/TR/xhtml-rdfa-primer/

RDFa in XHTML: Syntax and Processing -A collection of attributes and processing rules for extending XHTML to support RDF, W3C Recommendation 14 October 2008, http://www.w3.org/TR/rdfa-syntax

Wiki of the RDFa community with RDFa tools and examples http://rdfa.info/wiki/RDFa_Wiki

3.1.3 RDFS or RDF Schema

RDF Schema, short RDFS (or RDF(S), RDF-S, and RDF/S) gives rise to specify classes and inheritance over resources. An RDFS specification is called an "RDF ontology" or an "RDF vocabulary". Classes and inheritance after RDFS remind of object-orientation in programming. Significant differences, however, are as follows:

- Multiple inheritance is not precluded by RDFS.
- A RDFS vocabulary can be modified, for example extended with further (sub-)classes.

Note that RDFS itself is defined as a RDF vocabulary, that is, using the very concepts that RDFS introduces.

A first version of RDFS has been published by the W3C in 1998. RDFS is a W3C recommendation since 2004. Many RDFS components are part of the more expressive language Web Ontology Language (OWL).

In the following,

- the namespace prefix rdf is assumed to refer to the URI http://www.w3.org/1999/02/22-rdf-syntax-ns#,
- the namespace prefix rdfs is assumed to refer to the URI httpBreit, Franziska und Braunweiler, Monika
- the namespace prefix ex is assumed to refer to the URI http://www.example.org/

- rdfs:Class serves to declare a resource as a class.

  Example:

```
ex:person rdf:type rdfs:Class
ex:Mary   rdf:type ex:person
ex:Mary   rdf:type rdfs:Resource
```

- rdfs:subClassOf allows to declare a sub-class.

  Example:
  ```
  ex:student rdf:subClassOf ex:person
  ```

- The rdf:Property class serves to specify properties:

  Example:
  ```
  ex:name   rdf:type  rdf:Property
  ex:age  rdf:type  rdf:Property
  ```

  Note that the class Property is defined in RDF, not RDFS.

- rdfs:domain and rdfs:range are (predifined) properties in RDFS allowing to specify classes the subject, respectively the object of a predicate must belong to.

  Example:
  ```
  ex:age     rdfs:domain ex:person
  ex:age     rdfs:range  rdfs:Literal
  ```

- rdfs:Literal is the class of all literal values

- The property rdfs:subPropertyOf serves to express that all resources related by one property are also related by another.

  Example:
  ```
  ex:p1  rdf:type rdf:Property
  ex:p2  ref:type rdf:Property
  ex p1 rdfs:subPropertyOf p2
  ```

- rdfs:Datatype is the class of datatypes.

RDFS has the following further properties:

- rdfs:seeAlso (a property) that expresses a resource that provide additional information about the subject resource.
- rdfs:isDefinedBy (a property) that specifies a resource defining the subject resource. This property may be used to indicate an RDF vocabulary in which a resource is described.
- rdfs:label (a property) that provides a human-readable version of a resource's name
- rdfs:comment (a property) that provides a human-readable description of a resource.

References:

Section 5 "Defining RDF Vocabularies: RDF Schema" of: RDF Primer, W3C Recommendation 10 February 2004, http://www.w3.org/TR/rdf-primer/

RDF Vocabulary Description Language 1.0: RDF Schema, W3C Recommendation 10 February 2004, http://www.w3.org/TR/rdf-schema

3.1.4 RDF/S Semantics

RDF/S semantics is fourfold and defined in terms of the following:

- simple RDF interpretations
- Denotations (or interpretations) of literals
- RDF interpretations
- RDFS interpretations

3.1.4.1 An Introduction to Model Theory

RDF semantics is defined as a so-called model theory. This concept is introduced in the following referring to RDF triples and graphs. A model theory defines interpretations (or structures), models and logical entailment (or logical implication, or logical consequence).

An interpretation I of an RDF graph, that is a set of RDF triples, consists in a set U called universe (or universe of discourse, or domain) and of assigments as follows:

- Each resource, that is each URI and each blank node, as well as each literal is assigned an element in the universe U called its interpretation. One says that resources and literals are interpreted in U. The interpretation of a resource or literal x according to I is denoted I(x).

- Each property is assigned a binary relation over U, that is, a subset of U X U called its interpretation. The interpretation of a property p according to I is denoted IEXT(p).

In an interpretation, two distinct URI, blank node, or literal might, but do not have to, be interpreted by a same element of U; two distinct properties might, but do not have to be assigned the same binary relation over U.

An interpretation I with universe U of a set S of RDF triples is called a model of S if for every triple s p o in S, the pair (I(s), I(o)) of elements of U is an element of IEXT(p). Thus, an model I of a set S of triples interpretes subjects, properties and objects in such a manner that all relationships expressed by the triples in S are satisfied in the interpretation I.

Consider the following set S of triples (in the Turtle syntax):

```
@base ex <http://example.org/>
:Anna   :sisterOf :Bella .
:Bella  :sisterOf :Claire .
```

Let `U = {Alex, Bill}` and assume that `ex:Anna`, `ex:Bella`, and `ex:Claire` are interpreted in U as folows:

```
I(ex:Anna) = Alex
I(ex:Bella) = Bill
I(ex:Claire) = Alex
```

Assume further that Alex and Bill are brothers. Then interpreting the property ex:sisterOf by the brother relationshp over U, that is, by `{(Alex, Bill), (Bill Alex)}` yields a model of S.

This model is surprizing for three reasons: the genders suggested by the RDF specification S are not respected and two distinct resources, `ex:Anna` and `ex:Claire`, are identically interpreted, and the interpretationof `ex:sisterOf` is not reflexive. None the less, it is a model of S.

Another model of S consists in the set of integers Z with `ex:Anna` interpreted as 1, `ex:Bella` as 2, ex:Claire as 3 and `ex:sisterOf` as the successor relationship (n+1 is the sucessor of n). This model is counterintruitive because its universe has more elements than S has subjects and objects. None the less, it is a model of S.

3.1.4.2 Logical Entailement

A set S1 of RDF triples is said to logically entail (or logically imply) a set of RDF triples S2 if, whatever model I of S1 one considers, I is also a model of S2. In other words, S1 logically implies S2 if it is impossible to satisfy S1 without satisfying S2 too.

The set of RDF triple S considered above does not logically imply `ex:Anna = ex:Claire` because there are models of S in which this equality does not hold. Thus, even though there are models of S that fullfil this equality, it is not logically implied by S (one says also is not a logcal consequence of S).

The set of RDF triple S defined above neither logically imply `ex:Bella ex:sisterOf ex:Anna`, nor `ex:Anna ex:sisterOf ex:Claire`. Indeed, even though the "sister-of" relationship is usually reflexive and transitive, this is not specified in S ad therefore S has models in which this does not hold.

Thus, the common umndertamnding of identifiers (such as `ex:sisterOf`) occuring in a set of RDF triples do not impact on the logicall meaning, or semantics, of this set of triples. This is is highly desirable and like in programming: A method might be called, say, "addition" but implement the multiplication.

3.1.4.3 Particularities of RDF Interpretations and Models

RDF interpretations differ from the interpretations of classical logic in two essential aspects:

- typed literals cannot be freely interpreted but instead only as what they mean.
- predicates are mapped both to elements of the universe and (binary) relation over the universe.

Consider the following set S of triples (in the Turtle syntax):

```
@base ex <http://example.org>
@prefix xsd <http://www.w3.org/2001/XMLSchema>
:Anna :sisterOf :Bella ;
      :age      "23"^^xsd:int .
```

The following interpretation with universe `U = { AnnaMüller, BellaMüller }`, the element of which denote two sisters, is a perfect model of S even though it is counter-intuitive to interprete the integer 23 as a human being.

```
I(ex:Anna)  = AnnaMüller
I(ex:Bella) = BellaMüller
IEXT(ex:age) = {( AnnaMüller, BellaMüller) }
```

RDF interpretations first depart from the interpretations of classical logic by forbidding such counter-intuitive interpretations of literals: RDF interpretations require literals to be interpreted as "themselves". Thus, the interpretation of S mentioned above is no RDF interpretation of S.

RDF interpretations furthermore depart from the interpretations of classical logic by assigning to a property not only a (binary) relation (over the universe), but also an element of the universe.

Consider again the set S of triples:

```
@base ex <http://example.org/>
:Anna   :sisterOf :Bella .
:Bella  :sisterOf :Claire .
```

Consider the universe U = {AnnaMüller, BellaMüller, ClaireMüller}. The following specifes an RDF interpretation of S:

```
I(ex:Anna)     = AnnaMüller
I(ex:Bella)    = BellaMüller
I(ex:Claire)   = ClaireMüller
I(ex:sisterOf) = sister

IEXT(ex:sisterOf) =
             {(AnnaMüller, BellaMüller), (BellaMüller, ClaireMüller)}
```

Thus, an RDF interpretations requires not only a mapping I of resources (URI and blank nodes, including perperties) into elements of the universe, but also a mapping IEXT assigning (binary) relations, called extensions, to preperties.

One might think of assigning an element of the universe to a property as giving it a "name". This makes it possible to distinguish two properties with the same extension. For example, the sisterOf and a bestFriends properties might have the same extensions. Giving them "names" in an RDF extension aims at distinguishing them in this interpretation.

Note that in an RDF interpretaton, the interpretation of an RDF property might occur in the extension of that very property. We shall see below (in Section 3.1.4.9) that this is highly questionable.

Reference:

P. Hayes and C. Menzel. A Semantics for the Knowledge Interchange Format, , Proceedings of the 2001 Workshop on the IEEE Standard Upper Ontology, August 2001. http://reliant.teknowledge.com/IJCAI01/HayesMenzel-SKIF-IJCAI2001.pdf

3.1.4.4 Simple RDF Interpretations

Definition 1 (RDF triple and RDF graph).

Let U be a set of URIs, L a set of literals and B a set of blank node identifiers. Assume these three sets to be disjoints.

An RDF triple over U, B, L is a triple (s, p, o) of subject, predicate and object, where s â^^ U â^ª B, p â^^ U and o â^^ U â^ª B â^ª L.

An RDF graph over U, B, L is a set of triples over U, B, L.

An RDF graph is called ground, if it does not contain any blank nodes.

Definition 2. (RDF Vocabulary)

An RDF vocabulary is a set V = U âˆª L where U is a set of URIs and L is a set of literals. An element of an RDF vocabulary is called a name.

The vocabulary of an RDF graph are all the URIs and literals that occur in the triples of this RDF graph.

Definition 3 (Simple RDF interpretation).

A Simple interpretation of an RDF vocabulary V = U âˆª L is a six-tuple (IR, IP, IEXT , IS, IL, LV) where

- IR â‰ âˆ… is the domain or universe of I.
- IP is the set of properties of I.
- IEXT : IP â†’ P(IR Ã— IR).
- IS is a mapping (total function) U â†’ IR âˆª IP.
- IL is a mapping (total function) from typed Literals in V into IR.
- LV âŠ† IR: the set of literal values.

Typed literals are treated differently from plain literals so as to give a (special) semantics to typed literals that are not well-typed.

Note that the definition of simple RDF interpretation neither requires IR and IP be disjunct, nor IP âŠ† IR. If, however, a URI u appears within an RDF graph G both as predicate and subject , then in a simple RDF interpretation I which is a model of G, IS(u) must be in both IR and IP. Note that non-simple RDF interpretations (see Definition 7) require IP âŠ† IR.

3.1.4.5 Denotations and Ground RDF Graph Entailement

In order to interprete literals as "what they mean", the notion of "denotation" is used.

Recall that so called language tags are assigned to string literals using the @ character like in "Buch"@de, "livre"@fr and "book"@en.

Definition 4 (Denotation of ground RDF graphs)

Given an RDF interpretation I = (IR, IP, IEXT , IS, IL, LV) over a vocabulary V , the denotation of a ground RDF graph is defined as follows:

- if E is a plain literal "aaa" in V then I(E) = aaa
- if E is a plain literal "aaa"@ttt in V then I(E) = <aaa,ttt>
- if E is a typed literal in V then I(E)=IL(E)
- if E is a URI reference in V then I(E)=IS(E)
- if E is a ground triple (s, p, o) then I(E) = true iff s, p, o are in V , I(p) is in IP and (I(s), I(o)) is in IEXT (I(p)).

- if E is a ground RDF graph then I(E) = false if I(E') = false for some triple Eâ€² in E, otherwise I(E) = true.

Note that the denotation of an empty RDF graph is true. Note also that if some names in an RDF graph are not in the vocabulary of the interpretation, then the denotation of the graph is false in I.

Definition 5 (Model and Entailement - Ground RDF Graphs)

An RDF interpretation I is a model of a ground RDF graph G, if the denotation of G is true under I.

A ground RDF graph G1 entails a ground RDF graph G2, if all models of G1 are also models of G2 .

Ground entailment between ground graphs G1 and G2 reduces to the subset relationship between G1 and G2. Thus, it is decidable in linear time.

Entailment between RDF graphs becomes more involved when blank nodes come into play.

3.1.4.6 Denotation of general RDF Graph ad RDF Graph Entailement

Definition 6 (Denotation of general RDF graphs).

Let G be an RDF graph, blank(G) the set of blank nodes in G, I an interpretation and A a mapping (total function) from blank(G) to IR. [I + A] is an interpretation extending I by mapping a blank node B in blank(G) to A(B).

I(G) = true if [I + A](G) = true for some mapping A from blank(G) to IR.

Definition 7 (Model, simple entailment).

Let G1 and G2 be two (not necessarily ground) RDF graphs, I an RDF intepretation.

I is a model of G1 if the denotation of G1 is true under I.

G1 simply entails G2 if every model of G1 is also a model of G2.

3.1.4.7 RDF and RDFS Interpretations

RDF interpretation restrict simple RDF interpretations by requring properties being typed "properties" and ill-typed XML literals to be intereted differently from well-typed literals.

An XML Literal is well-typed, if it the serialization of some well-formed fragment of XML. Otherwise it is called ill-typed.

Definition 8 (RDF interpretation (Adapted from [Hay04]))

An RDF interpretation I = (IR, IP, IEXT , IS, IL, LV) is a simple RDF interpretations such that:

- x âˆˆ IP iff (x, I(rdf:Property)) âˆˆ IEXT (rdf:type)
- For all literals l which are typed as XML literals and which are well-typed, IL(l) must denote the XML value of l (i.e. an XML fragment).

- For a well-typed XML literal l, IL(l) must be in LV.
- For a well-typed XML literal l, (IL(l), I(rdf : XMLLiteral)) is in IEXT (rdf : type).
- For an ill-typed XML literal l, IL(l) is not in LV , and (IL(l), I(rdf:XMLLiteral)) is not in IEXT (rdf:type).

RDFS interpretations restrict RDF interpretations by requiring the object-oriented concepts of RDFS be interpretaed "as what they mean":

Definition 9 (RDFS interpretation [Hay04]). An RDFS interpretation

I = (IR, IP, IEXT , IS, IL, LV) is an RDF interpretation such that :

- x âˆˆ ICEXT(y) iff (x, y) âˆˆ IEXT(I(type))
- IC = ICEXT(I(Class))
- IR = ICEXT(I(Resource))
- LV = ICEXT(I(Literal))
- If (x, y) is in IEXT(I(domain)) and (u, v) is in IEXT(x) then u is in ICEXT(y)
- If (x, y) is in IEXT (I(range)) and (u, v) is in IEXT (x) then v is in ICEXT (y).
- IEXT(I(subPropertyOf)) is transitive and reï¬‚exive on IP.
- If (x, y) is in IEXT(I(subPropertyOf)) then x and y are in IP and IEXT(x) is a subset of IEXT(y).
- If x is in IC then (x, I(Resource)) is in IEXT(I(subClassOf)).
- If (x, y) is in IEXT(I(subClassOf)) then x and y are in IC and ICEXT(x) is a subset of ICEXT(y).
- IEXT(I(subClassOf)) is transitive and reï¬‚exive on IC.
- If x is in ICEXT(I(ContainerMembershipProperty)) then (x, I(member)) is in IEXT(I(subPropertyOf)).
- If x is in ICEXT(I(Datatype)) then (x, I(Literal)) is in IEXT(I(subClassOf)).

Entailement can be refined as follows: sinple entailement refers to models that are simple interpretations. RDF entailement refers to models that are RDF interpretationsa. RDFS entailement refers to models that are RDFS interpretations:

Definition 10 (Model, entailment)

Let G1 and G2 be two RDF graphs, I an intepretation.

I is an RDF model (RDFS model, reps.) of G1 if it is an RDF interpretation (RDFS interpretation, resp.) which satisfies G1 (that is, in which G1 is true).

G1 simply entails (RDF entails, RDFS entails, resp.) G2 if every model (RDF model, RDFS model, resp.) of G1 is also a model (RDF model, RDFS model, resp.) of G2.

3.1.4.8 Properties of RDF Entailement

Definition 10 (Valid and invalid transformations of RDF graphs)

A transformation of an RDF graph G1 into an RDF graph G2 is "simply valid" if G1 entails G2. Otherwise it is invalid.

Lemma 1 (Empty RDF graph lemma).
The empty set of RDF triples is entailed by every RDF graph.

Lemma 2 (Subgraph lemma).
A RDF graph entails all its subgraphs.

An instance of an RDF graph is obtained by consistently replacing some of its blank nodes by URIS.

Lemma 3 (Instance lemma).
An RDF graph is entailed by every of its instances.

Definition 11 (Merge of RDF graphs)
The merge of a set of two RDF graphs G1 and G2 is the union of G'1 and G'2 where Gâ€²i is obtained from Gi by a consistent renaming of blank nodes such that G'1 and G'2 have no blank node in common. (Such a blank node renaming is called a "standardization apart of G1 and G2).

Lemma 4 (Merging lemma).
The merge of a set S of RDF graphs is entailed by the union of S and entails every graph in S.

Lemma 5 (Interpolation lemma).
G1 simply entails a graph G2 if and only if a subgraph of G1 is an instance of G2.

Definition 12 (Lean RDF Graphs)
An RDF graph is lean if it does not contain any redundant information, that is, if it is not logically equivalent to any of its strict subgraph.

Lemma 6 (Leanness lemma).
An RDF graph G is lean with respect to simple entailment if it has no instance, which is a proper subgraph of G.

In [Hay04], Lemma 6 is the definition of "lean RDF graphs". The intuitive meaning of leanness, however, is the one of non-redundancy, as definied in Definiton 12. After Definitoin 12, leanness expresses redundancy not only due to the presence of blank nodes, but also due to the presence of triples in the RDF graph which are already contained under the RDFS semantics, or some other semantic extension of RDF. For RDF graphs both notions collapse. However, if formulas, like for example rules, are considered, Definition 12 is more general, hence more convenient, than the defintion of leaness of [Hay04].

Note that leanness under the RDFS semantics has been considered in [GHM04] before [Hay04] .

Lemma 7 (Anonymity Lemma).
Let E be a lean graph and Eâ€² a proper instance of E. Then E does not entail Eâ€².

Lemma 8 (Compactness Lemma).
If G1 entails G2 and if G2 is infinite, then some ï¬nite subset Gâ€²1 of G1 entails G2.

The proofs of these lemmas are rather simple and left as exercises. They are given in [Hay04] and in the recommendation [RDFSem04].

References:

[Hay04] Patrick Hayes. RDF semantics. Technical report, W3C, February 2004

[GHM04] Claudio GutiÃ©rrez, Carlos A. Hurtado, and Alberto O. Mendelzon. Foundations of semantic web databases. In Proceedings of the 23rd ACM SIGMOD-SIGACT-SIGART symposium on Principles of Database Systems (PODS), pages 95â€“106, 2004
[RDFSem04] RDF Semantics, W3C Recommendation 10 February 2004 http://www.w3.org/TR/rdf-mt/

3.1.4.9 Discussion

Besides properties being interpreted both as elements in the universe and as binary relations over that universe, RDF/S interpretations do not significantly depart from classical logic interpretations: The requirement literals being interpreted "as themselves" amonts to so-called standard interpretations (for example of natural numbers) in classical logic; giving an interpretation to ill-typed literals has no counterpart in classical logic, yet is in the spirit of logical interpretations of progeamming languages (like for example completing partial functions into total ones in the denotational semantics of programming languages.

RDF/S uncommon interpretation of relation symbols is meaningful because it makes it possible to refer to those relations only that are referred to in the syntax. In classical logic, in contrast, such a restriction is impossible: A relation is only characterized by its properties. If the universe is denumerable, what is common due to numbers, then the set of binary (an n-ary with n >= 2) relations is not denumerable, that is not computable, making many properties like unifications uncomputable. RDF/S uncommon interpretation of properties nicely avoids it.

Extending RDF/S with negation and extensional set (or class) definition, as a query language and/or rule language provide, however makes RDF/S uncommon interpretation of properties undesirable. Classes like a class C of all classes not containing themselves can be defined that do not exist:
Would C exist, then either C is in C, ort C is not in C. If C is in C, then be definiton C is not in C, a contradiction. If C is not in C, then by definiton, C is in C, again a contradiction.

This paradox is not new. It is the barber paradox that Russel (XXX) oberserved. It was precisely for avoidng it, that is for avoiding definitions of sets that cannot exist that set theory and the model theory of classical logic strictly differently relatiion (or predicate) symbols and constants and functional terms, the former being interpreted as relations over the universe, the later as elementd of the universe. This yields a stratification of a syntax' interpretation.

3.1.5 Linked Data

Linked Data is the name given to the vision of annotating web content with RDF triples so as (1) to provide a machine processable description of these contents and (2) to help and enhance the search for specific contewnt on the Web.

The main principle is (1) to use http:-type URIs for RDF resources and RDF documents at these locations for describing the resources and (2) to refer to other RDF resources in RDF descriptions so as to make it possible for crawlers to follows such references.

References:

http://linkeddata.org/
Tim Bernes-Lee: Linked Data, 2006 http://www.w3.org/DesignIssues/LinkedData.html
Joshua Tauberer: Linked Data for the Web, 2008 http://www.rdfabout.com/intro/?section=8