3

The Resource Description Framework (RDF) and its Vocabulary Description Language RDFS

Brian McBride

Hewlett Packard Laboratories, Bristol, UK.

Summary. An informal introduction to the W3C's updated Resource Description Framework (RDF) and its vocabulary description language is given. RDF's role in the semantic web and its relationship to other semantic web languages is described. The basic concepts of RDF and RDF Schema are explained and an example RDF schema is given. Limitations of RDF are described.

3.1 Introduction

The Resource Description Framework (RDF) [1,2] is a W3C recommendation that defines a language for describing resources. It was designed for describing Web resources such as Web pages. However, RDF does not require that resources be retrievable on the Web. RDF resources may be physical objects, abstract concepts, in fact anything that has identity. Thus, RDF defines a language for describing just about anything.

RDF describes resources in terms of named properties and their values. The RDF Vocabulary Description language, RDF Schema (RDFS) [3] describes vocabularies used in RDF descriptions. RDF vocabularies describe properties, classes of resources and relationships between them.

RDF is the foundation of the Semantic Web [4]. Just as the Web is a global infrastructure representing information in documents, the Semantic Web is a global infrastructure representing information in a form that can be processed by computer. Like the Web, the semantic Web is decentralized, which imposes a severe constraint on the mechanism it uses to represent information.

Traditional frame based and object oriented systems are resource-centric. They define classes and the properties instances of those classes must or may have. This is a centralized approach. Whoever defines a class defines what properties its instances have. To add more properties requires either the cooperation of the owner of the class or the definition of a new subclass.

S. Staab et al. (eds.), *Handbook on Ontologies*

Both RDF and RDFS[1] are Web languages. They are designed to allow information and vocabularies to be developed in a decentralized fashion. Just as the Web permits anyone with access to a server to create a Web page and link it to any other Web page, RDF(S) was designed following the principle that anyone should be able to say anything about anything[2]. To be able to say anything about anything, anyone must be able to define new properties for a class. Further, they should be able to use the same property to describe any class they choose. To support this need, RDF(S) is property centric. It enables properties to be defined and then used to describe resources.

RDF(S) has a formal semantics [5]. A formal semantics is needed for two reasons. Firstly, it brings precision to the specification of RDF. Without a formal semantics, there is too much scope for implementations to differ. Secondly, languages such as Owl [6] that extend RDF(S) have a formal semantics. Such languages require RDF(S) to have a formal semantics that they can extend, otherwise, they must define their own semantics for RDF, creating the possibility that different extension languages define different semantics.

RDF(S) are members of a family of Semantic Web languages. They builds on URI's [7] the Web's language for naming things, and on XML [8], the standard syntax for representing information in the Web. The DAML+OIL ontology language and its standardized form, the Owl [6] family of ontology languages, are extensions of RDF(S). Research on languages for querying the semantic Web [9, 10] and for expressing rules is likely to inform future efforts to standardize such languages.

The remainder of this chapter is organized as follows. Section 2 describes RDF, its abstract syntax, the use of blank nodes, support for datatypes and how it may written as XML. Section 3 describes RDFS, the notions of class and subclass, property and subproperty, domain and range constraints and the built in vocabularies for containers, collections and reification. Section 4 summarizes the key features of RDF(S), describes some of its limitations and refers to later chapters that describe more powerful languages that overcome these limitations.

The reader should note that at the time of writing (March 2003), the W3C's RDFCore Working Group are revising the original specification of RDF [11] and completing work on the RDF Schema specification [3]. Whilst this work is thought to be nearly complete, late changes made by the working group may not be represented in here.

---

[1] For the remainder of this chapter, the term RDF(S) will be used instead of "both RDF and RDFS".

[2] There are things that RDF is too weak to express; for example, it lacks negation and universal quantification. It is not possible to say everything about everything using RDF.

3.2 The Resource Description Framework

RDF's abstract syntax can be represented as a directed graph with labels on both nodes and edges:

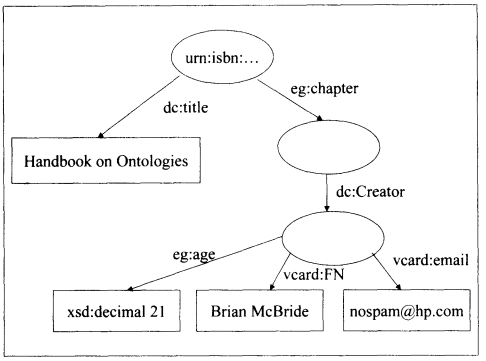

Fig. 1. An example RDF graph

RDF graphs are interpreted as follows:

- An elliptical node with a label represents a resource. If present, the label on the node is a URI[3] [7] that identifies the resource. In fig 1 the URI of the top node is not given in full, but is intended to identify this book. URI's often come in groups, called a vocabulary. Since URI's are often long and clumsy, rather than write them in full, in this chapter they are written in the form prefix:name, where the prefix identifies a vocabulary, and name is a name in that vocabulary. This is convenient shorthand for representing the full URI. The prefix "rdf" identifies names from the RDF vocabulary, "rdfs" names from the RDFS vocabulary, "dc" names from the Dublin Core [12] vocabulary, "vcard" names from the Vcard [13] vocabulary, "xsd" names from the XML Schema datatypes [14] vocabulary and "eg" names from an example vocabulary.
- An elliptical node with no label is called a blank node, or sometimes a b-node. It represents a resource for which no URI is given.
- An arc in the graph represents a property of the resource at the blunt end of the arc. The label on the arc, also a URI, identifies the property. The node at the sharp end of the arc represents the value of the property.
- Rectangular nodes represent literal values. Literals may be untyped or they may have a datatype.

Each arc in an RDF graph represents a *statement* that the resource at the blunt end of the arc, called the *subject* of the statement, has a property, called the *predicate* of the statement with a value, called the *object* of the statement. The notion

[3] Strictly, they are absolute URI's, but allowing international characters, with an optional fragment identifier. For convenience, the term 'URI' will be used throughout this chapter.

of statement corresponds to the English language notion of a simple sentence making a statement:

Sky hasColour Blue

The sky is the subject of the statement, hasColour is the predicate of the statement and Blue is the object of the statement

3.2.1 Blank Nodes

Blank nodes have a number of uses. A blank node may be used to represent a resource which either has no URI or for which no URI is known. In figure 1 for example, the top blank node represents this chapter of this book. The lower blank node represents a person. That person is uniquely identified, since, for the sake of this description, we assume that there is only one person with the email address given by the vcard:email property.

Blank nodes may be used to represent structured values. For example, the weight of a person may be represented by a blank node with two properties, one stating the value of the weight and the other stating the units of measurement.

Blank nodes may also be used to represent n-ary relationships. A single RDF statement can only represent a relationship between two nodes; so called binary relationships. N-ary relationships can be represented by introducing a blank node as in figure 2.

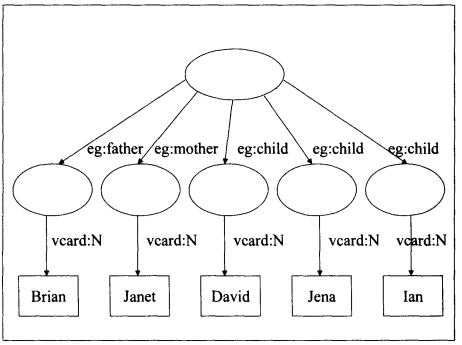

Fig. 2. Using a blank node to represent the n-ary relationship 'family'

Formally, blank nodes do not necessarily denote a specific resource. Instead, they are similar to existentially qualified variables in first order logic. Blank nodes can be read as "There is at least one resource with the given properties".

3.2.2 Datatypes and Datatype Values

RDF has adopted the XML Schema [14] model of datatypes. A datatype consists of a lexical space, a value space and a mapping from each member of the lexical space to one member of the value space.

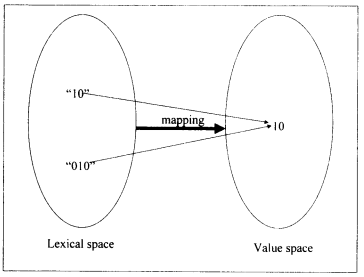

Fig. 3. The structure of a datatype

As with everything else in RDF, a datatype is identified by a URI. A datatype value is represented in RDF graphs by a rectangle containing the URI for the datatype and a string from the lexical space of the datatype. An example of this can be seen in the *eg:age* property in figure 1. Anyone can define a new datatype. Provided a URI is defined to identify it, it can be used in RDF graphs. However, for the full benefits of using datatypes to be realized, for example to process the integer 010 the same as the integer 10, software processing a graph must have specific knowledge of the datatypes referred to in the graph. The built-in datatypes defined by XML Schema have a "first amongst equals" status in RDF. They are the most likely to be interoperable amongst different software implementations.

3.2.3 RDF/XML

A graph is a convenient representation of small RDF information models for humans. For machine communication, or for large graphs, a way of representing an RDF graph as a sequence of symbols is needed. RDF/XML [15] is an XML language for representing RDF. Only a brief introduction to the syntax is possible here. For a fuller treatment the reader is referred to the language specification [15] and the RDF primer [1].

The graph in figure 1 can be written in RDF/XML as:

```
<rdf:Description rdf:about="urn:isbn:...">[4]
  <dc:Title>Handbook on Ontologies</dc:Title>
```

[4] For clarity and brevity, XML examples omit the XML header element and namespace declarations.

```
<eg:chapter>
 <rdf:Description>
  <dc:Creator>
   <rdf:Description>
    <vcard:FN>Brian McBride</vcard:FN>
    <vcard:email>nospam@hp.com</vcard:email>
    <eg:age rdf:datatype="&xsd;decimal"[5]>21</eg:age>
   </rdf:Description>
  </dc:Creator>
 </rdf:Description>
</eg:chapter>
</rdf:Description>
```

The first *rdf:Description* element represents a resource with the URI "urn:isbn:...". It has a title property represented by the *dc:Title* element. It also has a chapter property whose value is a blank node. The blank node is represented by an *rdf:Description* element without an *rdf:about* attribute. This node in turn has a single *dc:Creator* property whose value is another blank node which in its turn has two vcard properties and an *eg:age* property. The *rdf:datatype* attribute specifies the datatype of the value of the *eg:age* property.

Two other syntaxes for representing RDF, though non-standard, are in common use. Notation 3 [16], otherwise known as N3 is a more convenient notation than XML for humans to read and write. N-triples [17] is a subset of N3 used for defining test cases and for testing implementations. N-triples simply lists the subject, predicate and object of each arc in a graph, one per line.

3.3 RDF Schema

RDF enables assertion of simple statements consisting of a subject, a predicate and an object. It has no means to describe what these subjects, predicates and objects mean, nor to describe relationships between them.

RDF Schema introduces some simple ontological concepts. It defines a type system by introducing the concept of class and defines how resources may be described as belonging to one or more classes. It defines the concepts of subclass and subproperty, enabling description of hierarchies of classes and properties. It also defines the properties domain and range of a property. These are used to assert that the subject and object of a property respectively, belong to one or more classes.

---

[5] XML entities, which are assumed to be defined, are used as abbreviations in XML examples

3.3.1 Classes and Subclasses

A class represents a collection of resources, for example, the class of Web pages. Classes are themselves resources and are usually identified by URI's. A resource may be stated to be a member of a class using the *rdf:type* property as illustrated in Figure 4.

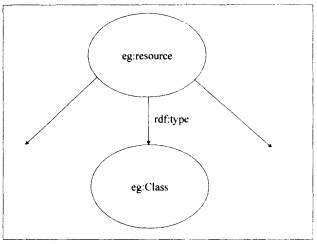

Fig. 4. eg:resource is a member of eg:Class

This graph can be written in RDF/XML as:

```
<rdf:Description rdf:about="http://example.org/resource">
  <rdf:type rdf:resource="http://example.org/Class"/>
  .... other properties
</rdf:Description>
```

The RDF/XML syntax provides a short hand notation for the rdf:type property. This graph may be written as:

```
<eg:Class rdf:about="http://example.org/resource">
  .... other properties
</eg:Class>
```

Those who expect that a class is the set of its members may be surprised to learn that in RDF Schema, the collection of all classes is itself a class.

Many modeling languages, such as UML [18] are layered. On the bottom layer is the instance data, akin to RDF statements. On the next layer there is metadata about the instance data, e.g. information about classes of data. On the layer above there is metadata about the first layer of metadata, and so on. Mixing data from the different layers is forbidden as is one layer being used for metadata about itself. RDF Schema is not layered in this sense. It allows self-reference, as in *rdfs:Class* being a member of itself.

At first sight, the class of all classes being a member of itself appears to conflict with the axiom of foundation of standard (Zermelo-Fraenkel) set theory. RDF avoids this difficulty by distinguishing between a class, which is a resource, and the *class extension* of the class, that is the set of members of the class. A class may be a member of its own class extension without violating the axiom of foundation.

The *rdfs:subClassOf* property is used to state that one class is a subclass of another, i.e. that the class extension of a class is a subset of the class extension of

another. Stating that one class is a subclass of another may be written in RDF/XML as:

```
<rdf:RDF xml:base="http://example.org/example">
  <rdfs:Class rdf:ID="Person"/>
  <rdfs:Class rdf:ID="Man">
    <rdfs:subClassOf rdf:resource="#Person"/>
  </rdfs:Class>
</rdf:RDF>
```

Not just hierarchies, but arbitrary graphs of subClassOf relationships between classes can be defined. Where two or more classes are arranged in a loop of *rdfs:subClassOf* properties, their class extensions must all have the same members. This does not mean they are the same classes. One of the classes may have a property that is not true of the others. For example, the Inland Revenue may define the class of people living in the author's house. The Post Office may define the class of people whose postcode is the same as the author's. Both these classes have the same members, but one has the property that it was defined by the post office and the other does not.

3.3.2 Properties and SubProperties

RDF properties are resources. They too form a class. RDF Schema defines *rdf:Property* to be the class of all RDF properties.

The reader familiar with other modeling languages might expect that an RDF property would be represented mathematically as a set of pairs, each pair representing two objects related by the property. This however, can lead to difficulties with standard set theory similar to those described above with the formal definition of classes.

A similar technique as used in the formal treatment of classes is used to avoid this difficulty and to maintain consistency. Formally, a property is a resource that has a *property extension* that is the set of pairs of objects related by the property.

One property may be a subproperty of another, which means that any two objects related by the subproperty are also related by the superproperty, i.e. the property extension of the subproperty is a subset of the property extension of the super property. Stating that one property is a subproperty of another may be written in RDF/XML as:

```
<rdf:RDF xml:base="http://example.org/example">
  <rdf:Property rdf:ID="partnerOf"/>
  <rdf:Property rdf:ID="husbandOf">
    <rdfs:subPropertyOf rdf:resource="#partnerOf"/>
  </rdf:Property>
</rdf:RDF>
```

As with subclass relationships arbitrary graphs of properties may be related with the subproperty relationship. Where two or more properties form a loop of subproperty relationships, then they must all have the same property extensions, though again, this does not mean that they are the same property.

RDF does not require that classes and properties are disjoint. The technique of using class and property extensions means that it is possible for a resource to be both a property and a class. However, where a concept may be used both as a property and a class, it is usually beneficial to have a different resource for each. By distinguishing between the class and the property, more accurate representation of information is possible. It is clear whether some statement applies to the property or the class. At least one style checker for RDF assumes that classes and properties are disjoint and will issue warnings where the same term is used for both.

3.3.3 Domain and Range

RDF Schema can state that all the subjects of a property belong to a given class. Similarly it can state that all the objects of a property belong to a given class.

The rdfs:domain property is used to state that all the subjects of a property belong to a class:

```
<rdf:RDF xml:base="http://example.org/example">
    <rdf:Property rdf:ID="husbandOf">
      <rdfs:domain rdf:resource="#Woman"/>[6]
      <rdfs:domain rdf:resource="#MarriedPerson"/>
    </rdf:Property>
</rdf:RDF>
```

This states that all subjects of the *eg:husbandOf* property are members of *eg:Woman* class and of the *eg:MarriedPerson* class. If more than one domain property is given, then subjects are members of all the classes.

Similarly, the *rdfs:range* property is used to state that all the objects of a property are members of a class.

```
<rdf:RDF xml:base="http://example.org/example">
    <rdf:Property rdf:ID="wifeOf">
      <rdfs:range rdf:resource="#Woman"/>
      <rdfs:range rdf:resource="#MarriedPerson"/>
    </rdf:Property>
</rdf:RDF>
```

As in the case of domain, where there is more than one range property, the object of the property is a member of all the classes.

3.3.4 Datatypes

The basic RDF model for datatypes was described in a previous section. RDF Schema defines the class *rdfs:Datatype*, the class of all datatypes. *rdfs:Datatype* is a subclass of *rdfs:Class*, i.e. all datatypes have class extensions. The class extension of a datatype is the set of values in the value space of the datatype.

---

[6] The property eg:husbandOf is used to represent heterosexual relationships.

3.3.5 Other Properties

The full name of a resource such as a class or property is a URI. URI's are often long and complex rendering them unsuitable for presentation in a user interface. *rdfs:label* may be used to provide a human readable name for a resource that can be used in a user interface.

rdfs:comment is a property that may be used to provide further information about a resource. This information is usually represented in natural language and is not amenable to machine processing.

rdfs:seeAlso may be used to link one resource to another that may provide further information about that resource. No constraints are placed on the linked resource; it may be a schema in a formal language or a description in natural language or anything else.

rdfs:isDefinedBy is a subproperty of *rdfs:seeAlso*. Whilst it still does not require that the linked resource be in any particular form, it is commonly used to indicate an RDF Schema that describes the resource.

3.3.6 Built in Vocabulary

RDF(S) defines vocabulary for representing containers, collections and occurrences of RDF statements.

3.3.6.1 Containers

An *rdfs:Container* is a resource that contains literals or resources. Three subclasses of *rdfs:Container* are defined; *rdf:Seq*, *rdf:Bag* and *rdf:Alt*. Formally, all three subclasses are ordered. However, an *rdf:Seq* is typically used where the order is significant. *rdf:Bag* is used where the order is not significant. *rdf:Alt* is used where typical processing will be to select one member of the container. These distinctions however, are not captured in the formal semantics.

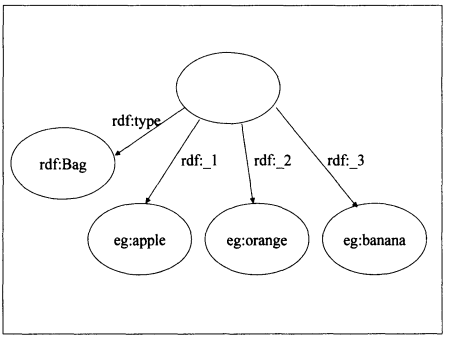

Fig. 5. A bag of fruit

Membership of a container is represented using container membership properties. The first member of the container is the value of its rdf:_1 property, the second the value of its rdf:_2 property and so on, as illustrated in figure 5. This can be written in RDF/XML as:

```
<rdf:RDF xml:base="http://example.org/">
  <rdf:Seq>
    <rdf:li rdf:resource="apple"/>
    <rdf:li rdf:resource="banana"/>
    <rdf:li rdf:resource="orange"/>
  </rdf:Seq>
</rdf:RDF>
```

where the rdf:li elements represent rdf:_1, rdf:_2 etc in sequence.

3.3.6.2 Collections

Collections were introduced into RDF(S) to overcome two disadvantages of containers. The first is that there is no way to close a container; to express the fact that there are no more members. The second is that programmers are used to using lists to represent collections. Figure 6 shows how a list structure is used to represent a collection in RDF.

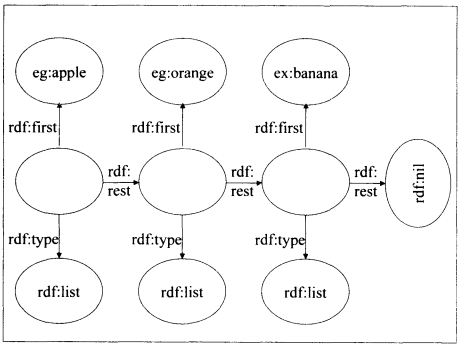

Fig. 6. An RDF Collection

Special syntax has been introduced into RDF/XML to represent collections. The collection in figure 6 can be written as

```
<rdf:RDF xml:base="http://example.org/">
  <rdf:Description rdf:parseType="Collection">
    <rdf:Description rdf:about="apple"/>
    <rdf:Description rdf:about="banana"/>
    <rdf:Description rdf:about="orange"/>
  </rdf:Description>
</rdf:RDF>
```

Note the *rdf:parseType* attribute.

3.3.6.3 Reification

RDF(S) is designed for representing information on the Web. Not all information on the Web is correct or even consistent. Consider, for example, the *eg:age* property in figure 1. It can therefore be important to track the provenance of information so that an application may decide whether to trust it, or trace back to the source of incorrect information. Reification is the tool that RDF provides to support this.

A reified statement is a resource that represents an occurrence of an RDF statement. Such a resource is a member of the class *rdf:Statement* and has three natural properties. *rdf:subject* is the subject of the statement, *rdf:predicate* is the predicate of the statement and *rdf:object* is the object of the statement. Other properties, such as where the statement occurred, when, who was responsible for it may then be attached to it. Figure 7 illustrates a reified statement.

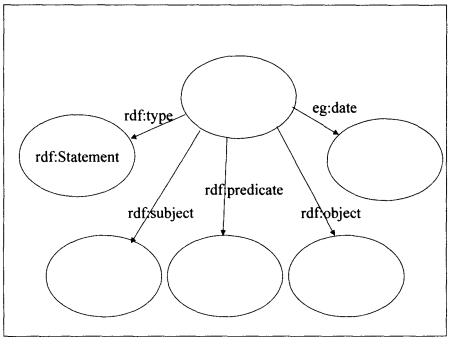

Fig. 7. A reified statement with *eg:date* property

3.3.7 Example RDF Schemas

A fuller version of the schema used in the examples above is given in table 1. Other examples of RDF schemas include:

- Platform for Privacy Preferences (P3P) [19] [20]
- Topic Maps [21] [22]
- Oil Ontology Language [23]

3.4 Conclusion

This chapter has described how RDF and RDFS were developed following the Web principle of decentralization to enable distributed integration and evolution of information models and vocabularies. The goal of decentralization led to their

property centric design, which contrasts with the resource centric approach of frame and object oriented systems.

Whilst it is useful on its own, RDFS does not support many ontological concepts such as equivalence, inverse relations and cardinality constraints found in other ontology languages. RDFS's role is to be extended by more powerful languages such as OWL [5] that are described later in this book. This ensures some level of interoperability between these languages by ensuring that each such extended language shares a common definition of the basic ontological notions, such as class and property, defined in RDFS.

RDF(S) has no notion of negation and hence no notion of contradiction. If an RDF schema states that the range of a property is both a plant and a metal, an RDF(S) processor will assume that all values of that property are members of the class of plants and the class of metals. RDF(S) cannot represent the fact that plants and metals are disjoint classes and that there can be no values for such a property.

RDF Schema can be used in one of two ways. It can be used to avoid the redundant specification of the same information. For example, by defining a range for a property, there is no need to specify for every value of that property, that it is of the type specified by the range property. That knowledge is implied by the range property.

Alternatively, it can be used to check that domain and range constraints are adhered to by checking redundant specification of the types of property values. A validator can issue an error or a warning if it cannot confirm that the subjects and objects of properties conform to range and domain constraints from other information, such as explicit type properties on the subjects and objects.

Acknowledgements

RDF and RDFS are the product of three W3C working groups, the Model and Syntax working group, the RDF Schema working group and the RDFCore working group. The members of those working groups are too numerous to acknowledge individually here. Their names are listed in the various specifications to which they contributed.

Table 1 A simple RDF schema

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xml:base="http://example.org/schema">
  <rdfs:Class rdf:ID="Person">
    <rdfs:label>person</rdfs:label>
    <rdfs:comment>The class of all people</rdfs:comment>
  </rdfs:Class>
  <rdfs:Class rdf:ID="MarriedPerson">
    <rdfs:label>married person</rdfs:label>
    <rdfs:comment>The class of all married people</rdfs:comment>
    <rdfs:subClassOf rdf:resource="#Person"/>
  </rdfs:Class>
  <rdfs:Class rdf:ID="Man">
    <rdfs:label>man</rdfs:label>
    <rdfs:comment>The class of all men</rdfs:comment>
    <rdfs:subClassOf rdf:resource="#Person"/>
  </rdfs:Class>
  <rdfs:Class rdf:ID="Woman">
    <rdfs:label>woman</rdfs:label>
    <rdfs:comment>The class of all women</rdfs:comment>
    <rdfs:subClassOf rdf:resource="#Person"/>
  </rdfs:Class>
  <rdf:Property rdf:ID="partnerOf">
    <rdfs:label>partner</rdfs:label>
    <rdfs:comment>Relates a person to their partner</rdfs:comment>
    <rdfs:domain rdf:resource="#Person"/>
    <rdfs:range   rdf:resource="#Person"/>
  </rdf:Property>
  <rdf:Property rdf:ID="wifeOf">
    <rdfs:label>wife of</rdfs:label>
    <rdfs:comment>Relates a wife to her husband</rdfs:comment>
    <rdfs:subPropertyOf rdf:resource="#partnerOf"/>
    <rdfs:domain rdf:resource="#Woman"/>
    <rdfs:domain rdf:resource="#MarriedPerson"/>
    <rdfs:range   rdf:resource="#Man"/>
    <rdfs:range   rdf:resource="#MarriedPerson"/>
  </rdf:Property>
  <rdf:Property rdf:ID="husbandOf">
    <rdfs:label>husband of</rdfs:label>
    <rdfs:comment>Relates a husband to his wife</rdfs:comment>
    <rdfs:subPropertyOf rdf:resource="#partnerOf"/>
    <rdfs:domain rdf:resource="#Man"/>
    <rdfs:domain rdf:resource="#MarriedPerson"/>
    <rdfs:range   rdf:resource="#Woman"/>
    <rdfs:range   rdf:resource="#MarriedPerson"/>
  </rdf:Property>
</rdf:RDF>
```

References

1. Manola, F., Miller, E. (eds) (2002): RDF Primer. http://www.w3.org/TR/rdf-schema/
2. Klyne, G., Carroll, J. (eds) (2002):Resource Description Framework (RDF) Concepts and Abstract Syntax. http://www.w3.org/TR/rdf-concepts/
3. Brickley, D., Guha, R.V. (eds) (2002): RDF Vocabulary Description Language 1.0: RDF Schema http://www.w3.org/TR/rdf-schema/
4. Berners-Lee, T., Hendler, J., Lassila, O. (2001): The Semantic Web. Scientific American, May 2001.
5. Hayes, P. (ed) (2002): RDF Semantics. http://www.w3.org/TR/rdf-mt/
6. Dean, M. et al (eds) (2002): OWL Web Ontology Language 1.0 Reference. http://www.w3.org/TR/owl-ref/ http://www.w3.org/TR/owl-features/
7. Berners-Lee, T. et al (1998): RFC 2396: Uniform Resource Identifiers (URI): Generic Syntax. http://www.ietf.org/rfc/rfc2396.txt
8. Bray, T., Paoli, J., C. M. Sperberg-McQueen, C. M., Maler, E. (2000): Extensible Markup Language (XML) 1.0 (Second Edition) http://www.w3.org/TR/REC-xml
9. Miller, L., Seaborne, A., Reggiori, A. (2002): Three Implementations of SquishQL, a Simple RDF Query Language. First International Semantic Web Conference (ISWC2002) 2002
10. Karvounarakis K., Alexaki, S., Christophides, V., Plexousakis, D., Scholl, M. (2002): World Wide Web Conference 2002, http://www2002.org/CDROM/refereed/329/
11. Lassila, O., Swick, R. (eds) (1999):Resource Description Framework (RDF) Model and Syntax Specification. http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/
12. Beckett, D., Miller, E., Brickley, D. (2002): Expressing Simple Dublin Core in RDF/XML. http://dublincore.org/documents/2002/07/31/dcmes-xml/
13. Iannella, R. (2001): Representing vCard Objects in RDF/XML. http://www.w3.org/TR/vcard-rdf
14. Biron, P.V., Malhotra, A. (2001): XML Schema Part 2: Datatypes. http://www.w3.org/TR/xmlschema-2/
15. Beckett, D. (ed) (2002): RDF/XML Syntax Specification (Revised). http://www.w3.org/TR/rdf-syntax-grammar/
16. Berners-Lee, T. (2002): Primer: Getting into RDF and the Semantic Web using N3. http://www.w3.org/2000/10/swap/Primer
17. Grant, J., Beckett, D. (eds) (2002): RDF Test Cases. http://www.w3.org/TR/rdf-testcases/
18. Booch, G., Jacobson, I., Rumbaugh, J. (2001): OMG Unified Modelling Language Specification. http://www.omg.org/technology/documents/formal/uml.htm
19. Crannor, L. et al (2002): The Platform for Privacy Preferences 1.0 (P3P1.0) Specification. http://www.w3.org/TR/P3P/
20. McBride, B., Wenning, R., Crannor, L. (2002): An RDF Schema for P3P. http://www.w3.org/TR/p3p-rdfschema/
21. Pepper, S., Moore, G. (eds) (2001): XML Topic Maps (XTM) 1.0. http://www.topicmaps.org/xtm/index.html
22. Garshol, L.M. (2002): An RDF Schema for topic maps. http://psi.ontopia.net/rdf/
23. Horrocks, I., Fensel, D. et al (2000) The Ontology Inference Layer OIL. http://www.cs.vu.nl/~dieter/oil/Tr/oil.pdf