Representing RDF and RDF Schema in the HDM

Dean Williams
School of Computer Science and Information Systems
Birkbeck College, University of London
dean@dcs.bbk.ac.uk

Abstract. The AutoMed project uses a nested hypergraph common data model (the HDM) to integrate heterogeneous datasources. Previous work has shown how data models including the relational, ER and XML models, can be mapped on to HDM to facilitate schema integration. RDF is a general-purpose language for representing information on the World Wide Web. It provides a data model for describing properties of resources in the web and their interrelationships. RDF Schema provides a basic type system for RDF models which is implemented in terms of classes and properties. These technologies are increasingly used in the semantic web effort in particular for defining ontologies. This technical report shows how RDF and RDF Schema can be modeled in the HDM thereby allowing data in these formats to be treated as an AutoMed datasource.

1 RDF

The Resource Description Framework (RDF) [1] is a language for representing information on the World Wide Web. RDF allows properties of Web resources to be stated in the form of Subject, Predicate and Object triples.

A statement such as “Dean Williams is the author of the webpage http://www.dcs.bbk.ac.uk/~dean/index.html” can be represent by a triple where the subject is the webpage URL, the predicate is the word “author” and the object is the words “Dean Williams”.

As in the example above, resources are identified using the Uniform Resource Identifier (URI) [2] web standard. URI’s are a more general identifier than the well-known Uniform Resource Locator (URL) used in the web. While URI’s cover defined, centralised schemes (such as the http: of URL’s) they also allow for anyone to create their own URI naming schemes.

In the example statement the concept of ‘author’ could be used in different ways by different systems. A web site maintenance system might use the author concept in a different way from a book publishing house system. Referring to the concept using a URI with an explicit namespace would allow the concept to be identified uniquely e.g. http://www.dcs.bbk.ac.uk/rdf/1.0/author.
Triples are often written using the N-Triples [3] notation where the subject, predicate and object are written beneath each other in that order with a period marking the end of the triple e.g.:

http://www.dcs.bbk.ac.uk/~dean/index.html
http://www.dcs.bbk.ac.uk/rdf/1.0/author
“Dean Williams” .

Values in RDF can be URI’s, literals or unlabelled nodes (which are known as ‘blank’ nodes). These blank nodes can be used to structure property values e.g. by linking each line of an address. More generally they can be thought to represent concepts to which properties apply. When implemented, arbitrary identifiers are assigned to these blank nodes and so they are analogous to object ID’s in Object Oriented systems.

There are also restrictions in the RDF data model [4] concerning the type of value each part of the triple can have, namely:

- The subject S can be a URI or a blank node.
- The predicate P has to be a URI.
- The object O can be a URI, blank or literal.

Nodes and edges in a graph can be used to represent RDF statements. By convention nodes are drawn as ovals and will be either labeled or blank, literals are rectangles, edges are single headed arrows linking nodes or literals.

Figure 1 shows the webpage author example extended to allow for various properties, such as employee id, of the persons identified as the author to be related to that person.
Fig. 1. Example RDF statement

The graph in Figure 1 represents the following four triples with arbitrary id’s of the form _n being used to denote blank nodes:

http://www.dcs.bbk.ac.uk/~dean/index.html
http://www.dcs.bbk.ac.uk/rdf/1.0/author
_1.

_1
http://www.dcs.bbk.ac.uk/rdf/1.0/name
"Dean Williams".

_1
http://www.dcs.bbk.ac.uk/rdf/1.0/employeeID
"22945".

_1
http://www.dcs.bbk.ac.uk/rdf/1.0/employmentDate
"1/4/02".
2. Representing RDF in the HDM.

The HDM data model, described in [5,6,7] models data in terms of nodes, edges and constraints. Using the HDM model, the graph based RDF data model described above this structure can be represented as shown in figure 2.

Fig. 2. HDM representation of RDF

In the HDM notation this will be represented by the following:

Nodes:

<<rdf:blank>>
<<rdf:URI>>
<<rdf:literal>>
<<rdf:triple>>
<<rdf:subject>>
<<rdf:predicate>>
<<rdf:object>>

Edges:

<<s,triple,subject>>
<<p,triple,predicate>>
<<o,triple,object>>

Constraints:

subject ⊆ URI ∪ blank
predicate ⊆ URI
object ⊆ URI ∪ blank ∪ literal

For the example above, the data in Figure 1 can be modeled by the following HDM:
blank    = { _1 }
URI      = { http://www.dcs.bbk.ac.uk/~dean/index.html ,
             http://www.dcs.bbk.ac.uk/rdf/1.0/author ,
             http://www.dcs.bbk.ac.uk/rdf/1.0/employmentDate ,
             http://www.dcs.bbk.ac.uk/rdf/1.0/name }

literal  = { "Dean Williams", "22945", "1/4/02" }

triple   = { _2, _3, _4, _5 }
subject  = { http://www.dcs.bbk.ac.uk/~dean/index.html , _1 }
predicate= { http://www.dcs.bbk.ac.uk/rdf/1.0/author ,
             http://www.dcs.bbk.ac.uk/rdf/1.0/name ,
             http://www.dcs.bbk.ac.uk/rdf/1.0/employeeID ,
             http://www.dcs.bbk.ac.uk/rdf/1.0/employmentDate }
object   = { _1, "Dean Williams", "22945", "1/4/02" }

edges = {

<<s, _2, http://www.dcs.bbk.ac.uk/~dean/index.html >> ,
<<p, _2, http://www.dcs.bbk.ac.uk/rdf/1.0/author >> ,
<<o, _2, _1>> ,
<<s, _3, _1 >> ,
<<p, _3, http://www.dcs.bbk.ac.uk/rdf/1.0/name >> ,
<<o, _3, "Dean Williams">>,
<<s, _4, _1 >> ,
<<p, _4, http://www.dcs.bbk.ac.uk/rdf/1.0/employeeID >> ,
<<o, _4, "22945">>,
<<s, _5, _1 >> ,
<<p, _5, http://www.dcs.bbk.ac.uk/rdf/1.0/employeeID >> ,
<<o, _5, "22945">> }

The constraints specified hold for these nodes and edges.

3 RDF Containers and Reification
The use of blank nodes to group together related properties in RDF was shown previously. RDF also provides for containers to refer to collections of resources. There are three types of container in RDF:

- **Bag.** Unordered list with duplicates allowed.
- **Sequence.** Ordered list, duplicates allowed.
- **Alternative.** List of resources that represent alternatives for the single value of a property.

The `rdf:type` property is used to specify the container and its type. Each member of the collection is then assigned a unique membership property.

Extending the employee example above a bag might be used to represent any relevant work related qualifications e.g. “first aider”, “minibus driver”. If a record of the job titles of posts the employee had held was required, the order in which they had been held is significant and so a sequence container would be more appropriate. If there were more than one telephone in an employee’s office it might be appropriate to use the Alternative container to show the numbers which can be used.

Figure 3 shows these three containers appended to the original example. It is important to note that apart from the type names assigned the schemas of these three container types is identical – the application program making use of the data will need to be able to interpret their semantic differences.
Fig. 3. Extended employee details example showing collections.

It is also important in RDF to be able to make statements about statements e.g. “Personnel says that Dean Williams’ employee number is 22945”. This is a fact about something the personnel department has said, not about Dean’s employee number. In order to make a statement about this statement it is necessary to first remodel the original statement - this process of remodeling this the statement is known as reification. RDF has a method of modeling such statements that makes use of a specific value of the property ‘type’. The statement has four properties:

- type of the statement is “rdf:statement”
- subject is the resource described by the modelled statement i.e. Dean in the example.
- predicate is the original property i.e. employee number.
- object is the object of the original property i.e. “22945”.

It is now possible to attach a property to the reified statement.
To extend the HDM representation of RDF to include support for containers it is necessary to make two additions:

1) To represent types a new node "<<RDFtype>> will exist with extent \{bag, sequence, alternative, statement\}. If a node has a type an edge "<<triple,type>> will exist and the constraint type \subseteq RDFtype must hold.

2) For the sequence container the order of the elements must be preserved. The requirements of this feature are the same as for the list semantics already considered for modeling XML in the HDM [4]. As RDF requires that a unique membership number be assigned to each of the membership edges this will be enforced in the HDM model with an extra edge going from each of the membership edges to a node order with a cardinality constraint that each instance of these extra edges is related to one and only one instance of order.

A Java program to create a Model for RDF and a schema for storing RDF can be seen at www.dcs.bbk.ac.uk/~dean/eg/RDF.java

This is slightly unusual code for AutoMed as the schema will be the same for all RDF data sources, unlike say the relational model where many tables will be defined there is only one triple store.

4. RDF Schema

RDF Schema provides a type system for RDF, which is implemented in terms of classes and properties – these are very similar to the classes and properties of OO properties with one exception. Everything in RDF is described as a resource. A resource has a type, which can define it as either a class or a property. Properties are therefore defined independently of classes (this is not the case in OO languages).
A summary of the components of RDF Schema is below:

Resources:

- **rdfs:Resource**: This is the base class for RDF schema, everything that is being described by RDF is a resource and is an instance of the class rdfs:Resource.
- **rdfs:Class**: A predefined resource for defining classes, this is used by the rdf:type property to specify that a resource is a class.
- **rdfs:Property**: A predefined resource for defining properties, this is used by the rdf:type property to specify that a resource is a property.

Properties:

- **rdf:type**: A property of a resource which, if given the value “rdfs:Class” defines a class definition or if the value is “rdfs:Property” then the resource defines a property.
- **rdfs:subClassOf**: Specifies that the class is a subclass of another class. Each class can have many parent classes.
- **rdfs:range**: This property specifies a class whose instances contain the possible values that the property may have e.g. the range of that property might be the Job class.
- **rdfs:domain**: The classes, which the property belongs to. Zero, one or many of these definitions may be given. If no definitions are given then the property is assumed to apply to all classes.
- **rdfs:subPropertyOf**: Specifies that the property is a subproperty of another property. Each property can have many parent properties.
- **rdfs:seeAlso**: Additional information about the resource.
- **rdfs:isDefinedBy**: Sub-property of rdfs:seeAlso which is analogous to the idea of namespaces.

5. Representing RDF Schema in the HDM.

The relationships between these components can be represented by the graph shown in Figure 3 below.
Fig. 5. Components of RDF Schema

In the HDM notation this can be represented by:

Nodes:
<<resource>>
<<property>>
<<class>>

Edges:
<<type,resource,class>>
<<type,resource,property>>
<<domain,property,class>>
<<range,property,class>>
<<subClass,class,class>>
<<subProperty,class,class>>
<<seeAlso,resource,resource>>
<<isDefinedBy,resource,resource>>

An example of an RDF Schema and an instance of data complying to this schema is now given together with the graphs the XML documents represent and the HDM instances required to store the data.
This example is based on the Wordnet [8] English language database. An RDF version of the database is available as is an RDF Schema definition of the database. In Wordnet concepts are defined and word forms are linked to the concept e.g. the words “bike” and “bicycle” can both be used to refer to the same concept. Relationships between concepts

A cut down version of the schema is contained in Appendix A, which includes the class lexical concept and its subclass noun. The following properties are defined: word form maps a string to a concept; hyponym of can be thought of as an ‘isa’ hierarchy; glossary entry is a text description of a concept. Figure 4 shows a graph representation of this RDF Schema model.

![Graph representation of RDF Schema contained in Appendix A](image)

**Fig. 6.** Graph representation of RDF Schema contained in Appendix A

In HDM notation this schema would be represented by:

```plaintext
class = {}
property = {}
resource = { class, property, lexicalConcept, noun, 
hyponymOf, glossaryEntry, literal, wordForm }

edges = {
  <<type, lexicalConcept, class>> ,
```
Appendix B shows a fragment of the Wordnet database for two concepts “locomotive” which is a hyponym of “wheeled vehicle”. Several alternative word forms for the concept “locomotive” are given e.g. “engine”. Glossary entries for the two concepts are also listed. Figure 5 shows this data in graph form.

Fig. 7. Graph Based Representation of Appendix B

This RDF can be modeled in the HDM as:

```plaintext
blank = {}
URI = { http://www.bbk.ac.uk/concept/103610313,
        http://www.bbk.ac.uk/concept/102937872,
        http://www.bbk.ac.uk/schema/wordForm }
```
literal = { "Wheeled Vehicle", "engine", "railway locomotive", "locomotive", "locomotive engine", "self-propelled engine used to draw trains along railway tracks", "a vehicle that moves on wheels and usually has a container for transporting things or people" }

triple = { _1, _2, _3, _4, _5, _6, _7 }

subject = { http://www.bbk.ac.uk/concept/103610313 , http://www.bbk.ac.uk/concept/102937872 }


object = { http://www.bbk.ac.uk/concept/103610313 , "Wheeled Vehicle", "engine", "railway locomotive", "locomotive", "locomotive engine", "self-propelled engine....tracks", "a vehicle...or people" }

edges = {
  <<s, _1, http://www.bbk.ac.uk/concept/103610313 >> ,
  <<p, _1, http://www.bbk.ac.uk/schema/wordForm >> ,
  <<o, _1, "Wheeled Vehicle">>,
  <<s, _2, http://www.bbk.ac.uk/concept/102937872 >> ,
  <<p, _2, http://www.bbk.ac.uk/schema/wordForm >> ,
  <<o, _2, "engine">>,
  <<s, _3, http://www.bbk.ac.uk/concept/102937872 >> ,
  <<p, _3, http://www.bbk.ac.uk/schema/wordForm >> ,
  <<o, _3, "railway locomotive">>,
  <<s, _4, http://www.bbk.ac.uk/concept/102937872 >> ,
  <<p, _4, http://www.bbk.ac.uk/schema/wordForm >> ,
  <<o, _4, "locomotive">>,
  <<s, _5, http://www.bbk.ac.uk/concept/102937872 >> ,
  <<p, _5, http://www.bbk.ac.uk/schema/wordForm >> ,
  <<o, _5, "locomotive engine">>,
  <<s, _6, http://www.bbk.ac.uk/concept/102937872 >> ,
  <<p, _6, http://www.bbk.ac.uk/schema/ glossaryEntry >> ,
}
A Java program to create a model for RDFS and build the schema above can be seen at www.dcs.bbk.ac.uk/~dean/eg/RDF.java

6 Conclusion and Future Work

This technical report shows that RDF and RDF Schema data can be mapped on to the HDM data model and can therefore be exploited as a data source in the AutoMed system.

A question being examined by ongoing work in the AutoMed project is to see how text data combined with structured data can be better used [9]. The ability to treat ontologies, both natural language such as WordNet and domain specific, as data sources able to be integrated with database schema information using the tools provided by AutoMed will be beneficial to this effort.

RDF data model is a natural candidate for being stored in a triple based database and an HDM store is being developed which will allow data to be stored in HDM format rather than just accessed using HDM as the common data model. Together with the functional query language IQL being developed for AutoMed this may form a useful RDF data store in its own right.

The proposal above stores the RDF Schema data structure and the instance RDF data separately – a parser would be required to ensure the instance data conforms to the schema. Depending on the constraints language ultimately implemented in AutoMed, it may be possible to consider the possibility of implementing a set of constraints that ensures only compliant RDF data is stored.

RDF Schema provides a basic typing schema but richer facilities are often required by designers of Ontologies e.g. cardinality constraints etc. A number of languages are proposed in the Semantic Web effort; in particular DAML+OIL is becoming widely used. Treating DAML + OIL as an AutoMed data source would be a useful addition and is part of ongoing work.

References

1. http://www.w3.org/TR/REC-rdf-syntax/
http://www.w3.org/TR/rdf-mt/


D. Williams, *Database Driven Discovery of Structure from Partially Structured Data*, BNCOD PhD Summer School 2002
Appendix A RDFS Schema for Wordnet

<!--
This is a cutdown version of the unofficial RDF Schema for WordNet data. The original was written by Sergey Melnik and available at
http://www.semanticweb.org/library/wordnet/wordnet-20000620.rdf
Dean Williams. dean@dcs.bbk.ac.uk
-->

<!DOCTYPE rdf:RDF [
<!ENTITY rdf 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
<!ENTITY s 'http://www.w3.org/TR/1999/PR-rdf-schema-19990303#'>
<!ENTITY wn 'http://www.cogsci.princeton.edu/~wn/schema/'>
]

<rdf:RDF

xmlns="&rdf;"
xmlns:rdf="#rdf;"
xmlns:s="#s;"
xmlns:wn="#wn;"
>

<s:Class rdf:about="#wn;LexicalConcept"

s:comment='A lexical concept identifies a sense or a meaning, captured by a set of synonyms that serves as an unambiguous designator. The synonym set does not explain what the concept is; it merely signifies that the concept exists.' />

<s:Class rdf:about="#wn;Noun"

s:comment='A noun.'>

<s:subClassOf rdf:resource="#wn;LexicalConcept"/>
</s:Class>

</rdf:RDF>

<rdf:Property rdf:about="#wn;wordForm"
A word form is used to refer to the physical utterance or inscription and "word meaning" to refer to the lexicalized concept that a form can be used to express.'

This is a lexical relation that specifies that the first concept is a hyponym of the second concept. This relation holds for nouns and verbs. The reflexive operator, hypernym, implies that the second concept is a hypernym of the first one.'

The glossary entry (a gloss) helps to resolve the polysemy. The gloss is not intended for use in constructing a new lexical concept by someone not already familiar with it, and it differs from a synonym in that it is not used to gain access to information stored in the mental lexicon. It fulfills its purpose if it enables the user of WordNet, who is assumed to know English, to differentiate this sense from others with which it could be confused.'
Appendix B Sample Wordnet RDFS Word Descriptions

<!-- This is a cut down sample of the data -->
<rdf:RDF xmlns:rdf="&rdf;"
            xmlns:a="&a;"
            xmlns:b="&b;">

  <b:Noun rdf:about="&a;103610313"
            b:wordForm="wheeled vehicle"/>
  <b:Noun rdf:about="&a;102937872"
            b:wordForm="engine"
            b:wordForm="locomotive"
            b:wordForm="locomotive engine"
            b:wordForm="railway locomotive"/>
  <rdf:Description rdf:about="&a;102937872" rdf:resource="&a;103610313"/>
  <b:glossaryEntry>a vehicle that moves on wheels and usually has a container for transporting things or people</b:glossaryEntry>
  <b:glossaryEntry>self-propelled engine used to draw trains along railway tracks</b:glossaryEntry>
</rdf:RDF>