Representing RDF and RDF Schema in the HDM

Dean Williams, Alexandra Poulovassilis
{dean,ap}@dcs.bbk.ac.uk

School of Computer Science and Information Systems, Birkbeck College, University of London

Abstract. The AutoMed project uses a hypergraph common data model (the HDM) to integrate heterogeneous data sources. 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) [6] 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.xyz.com/index.html" can be represented by a triple where the subject is the webpage URL, the predicate is 'author' and the object is 'Dean Williams'.

Resources are identified using the Uniform Resource Identifier (URI) [8] web standard. URI's are a more general identifier than the Uniform Resource Locator (URL) used in the Web. While URIs cover defined, centralised schemes (such as the http part of a URL) 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 it 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.xyz.com/rdf/1.0/author .

Triples are often written using the N-Triples [2] 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.xyz.com/\~dean/index.html
http://www.xyz.com/rdf/1.0/author
'Dean Williams' .
```

Values in RDF can be URIs, literals or unlabelled nodes, known as 'blank' nodes. Blank nodes can be used to structure property values e.g. by linking each line of an address. More generally, they represent concepts to which properties apply. Arbitrary identifiers are assigned to these blank nodes and so they are analogous to object IDs in Object Oriented systems.

There are also restrictions in the RDF data model [7] concerning the type of value each part of the triple can have, namely that the:

– The subject can be a URI or a blank node.
– The predicate must be a URI.
– The object 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 drawn as rectangles, and edges as single-headed arrows linking nodes or literals.

Figure 1 shows the webpage author example above extended to allow for various properties of the person identified as the author to be related to that person. The graph in Figure 1 represents the following four triples, where _1

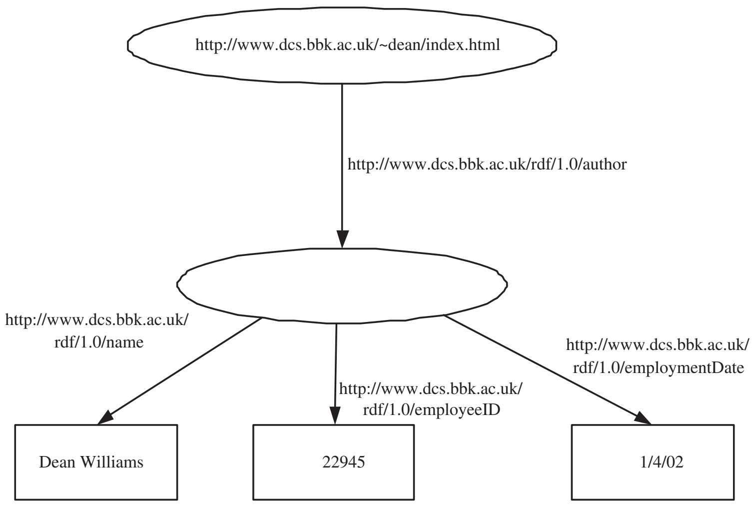

Fig. 1. Example RDF statement

denotes the one blank node.

```
http://www.xyz.com/\~dean/index.html
```

```
http://www.xyz.com/rdf/1.0/author
_1 .

_1
http://www.xyz.com/rdf/1.0/name
Dean Williams.

_1
http://www.xyz.com/rdf/1.0/employeeID
22945 .

_1
http://www.xyz.com/rdf/1.0/employmentDate
1/4/02 .
```

2 Representing RDF in the HDM

The HDM data model [3–5] models data in terms of nodes, edges and constraints. The RDF constructs URI, Literal and Blank are each represented by an HDM node and are all nodal constructs. The RDF construct Triple is represented by a combination of three HDM nodes, and edge, and three constraints; this construct is thus nodal, linking and constraint (see [3] for an explanation of the terms 'nodal', 'linking' and 'constraint'). This specification of RDF in the HDM is illustrated in Table 1. For any given RDF description, the HDM nodes ⟨⟨URI⟩⟩, ⟨⟨Literal⟩⟩ and ⟨⟨Blank⟩⟩ have as their extents the set of URIs, literals and blank nodes appearing in the description, respectively, while the HDM node ⟨⟨Triple⟩⟩ has as its extent the set of triples.

RDF Construct	HDM Representation
construct:RDFNode nodal scheme: ⟨⟨URI⟩⟩	node: ⟨⟨URI⟩⟩
construct:RDFNode scheme: ⟨⟨Literal⟩⟩	node: ⟨⟨Literal⟩⟩
construct:RDFNode scheme: ⟨⟨Blank⟩⟩	node: ⟨⟨Blank⟩⟩
construct:RDFEdge class: nodal, linking and constraint scheme: ⟨⟨Triple⟩⟩	nodes: ⟨⟨subject⟩⟩, ⟨⟨predicate⟩⟩, ⟨⟨object⟩⟩ edge: ⟨⟨Triple,subject,predicate,object⟩⟩ three constraints: ⟨⟨subject⟩⟩ $\subseteq$ (⟨⟨URI⟩⟩ $\cup$ ⟨⟨Blank⟩⟩) ⟨⟨predicate⟩⟩ $\subseteq$ ⟨⟨URI⟩⟩ ⟨⟨object⟩⟩ $\subseteq$ (⟨⟨URI⟩⟩ $\cup$ ⟨⟨Blank⟩⟩ $\cup$ ⟨⟨Literal⟩⟩)

Table 1. Definition of RDF model constructs

3 RDF Containers and Reification

The use of blank nodes to group together related properties in RDF was discussed above. 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. Extending the employee example to show the use of the different container types:

A **bag** could be used to represent any relevant work related qualifications e.g. 'first aider', 'minibus driver'.
A **sequence** could be used to record of the job titles of posts an employee had held - in this case the order in which they had been held is significant and so a sequence is more suitable than a bag.
An **alternative** might be used if there could be more than one telephone in an employee's office and several numbers could be used to contact the employee.

Figure 2 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 are identical - the application program making use of the data will need to be able to interpret their semantic differences.

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 'rdf: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 'informant' to the reified statement as can be seen in Figure 3.

The HDM specification for containers and statements is shown in Table 2. The members of containers may be any kind of resource, and resources represented by an additional HDM node $\langle\langle \mathsf{Resource} \rangle\rangle$. There is one HDM edge $\langle\langle \mathsf{bag}, \mathsf{Blank}, \mathsf{Resource} \rangle\rangle$ whose extent is all the container/member associations for bag containers, one HDM edge $\langle\langle \mathsf{sequence}, \mathsf{Blank}, \mathsf{Resource} \rangle\rangle$ whose extent is all

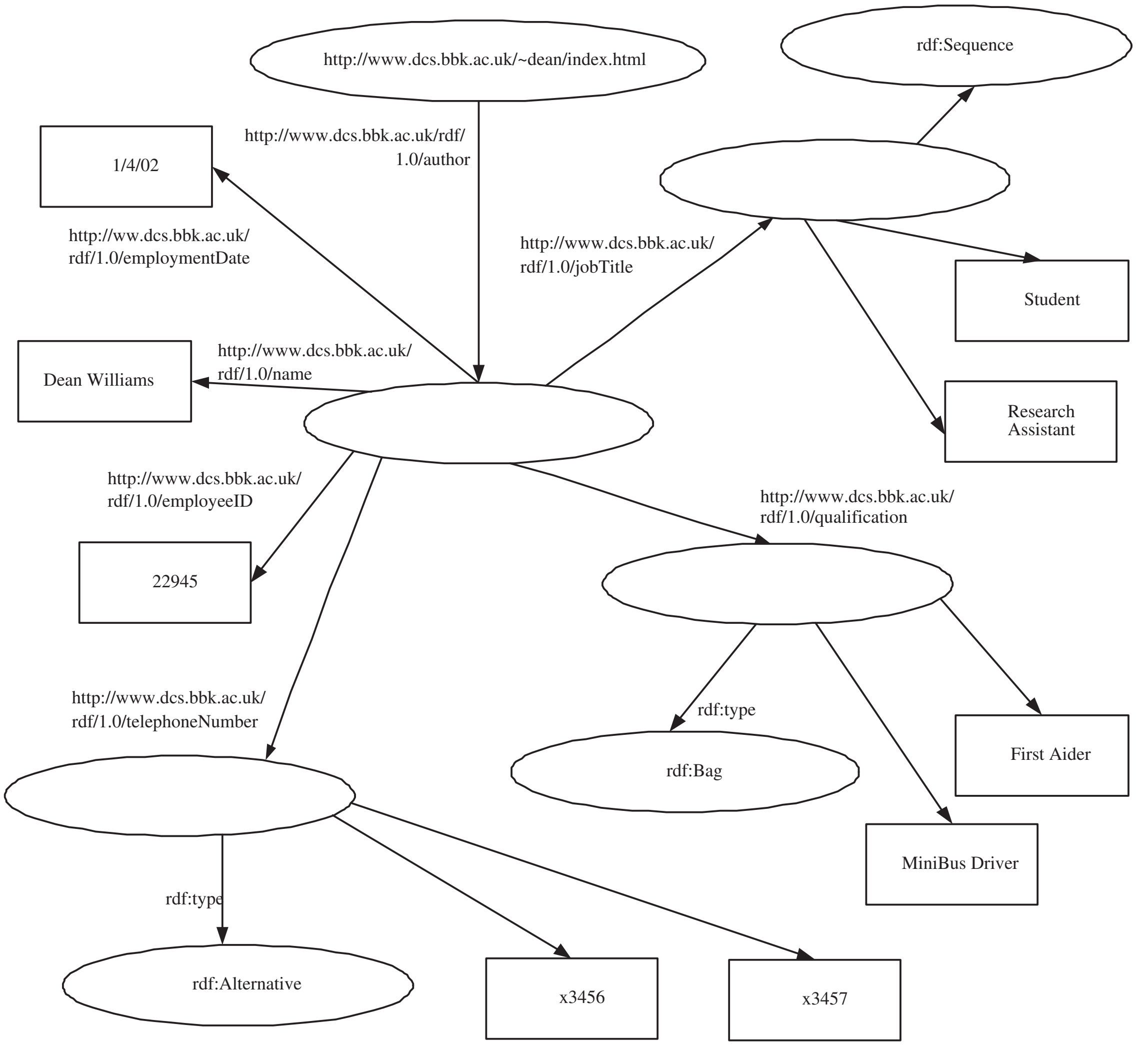

Fig. 2. Extended employee details example showing collections.

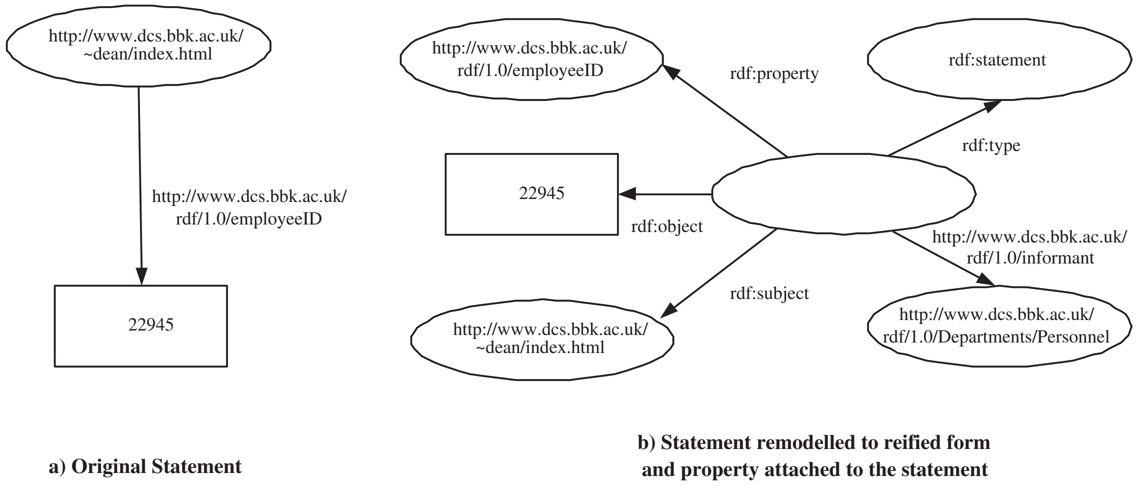

Fig. 3. Statement and its reified form with 'statement about a statement' added.

the container/member associations for sequence containers, and one HDM edge ⟨⟨alternative,Blank,Resource⟩⟩ whose extent is all the container/member associations for alternative containers. For bag and sequence containers, an extra HDM node ⟨⟨Number⟩⟩ is used to model members' cardinality and ordering, respectively (similarly to the way that order was represented for XML in [4]). In particular, we use an additional HDM edge from ⟨⟨bag,Blank,Resource⟩⟩ to ⟨⟨Number⟩⟩, and an additional HDM edge from ⟨⟨sequence,Blank,Resource⟩⟩ to ⟨⟨Number⟩⟩. For all instances of the edge ⟨⟨bag,Blank,Resource⟩⟩, there is an edge to an instance of ⟨⟨Number⟩⟩ indicating the cardinality of that member in that bag. Similarly, for all istances of the edge ⟨⟨sequence,Blank,Resource⟩⟩, there are one or more edges to an instance of ⟨⟨Number⟩⟩ indicating the position(s) of that member within that sequence. We have not shown in Table 2 the cardinality constraints implied by these semantics but they are as follows, expressed in IQL (the AutoMed query language), for ⟨⟨bag,Blank,Resource⟩⟩:

```
fold (lambda (b,m).
       makeCard ([((b',m'),n)|((b',m'),n)<-
                              <<_,<<bag,Blank,Resource>>,<<Number>>>>;
                              b'=b],(1,1),(0,N)))
     (and) True <<bag,Blank,Resource>>
```

and for ⟨⟨sequence,Blank,Resource⟩⟩:

```
fold (lambda (s,m).
      makeCard (([((s',m'),n)|((s',m'),n)<-
                              <<_,<<sequence,Blank,Resource>>,<<Number>>>>;
                              s'=s],(1,N),(0,1)))
     (and) True <<sequence,Blank,Resource>>
```

RDF Construct	HDM Representation
construct:RDFEdge class: nodal, linking and constraint scheme: ⟨⟨Statement⟩⟩	nodes: - edge: ⟨⟨Statement,Blank,subject,predicate,object⟩⟩ constraints: -
construct:RDFContainer class:nodal, linking and constraint scheme: ⟨⟨t⟩⟩	nodes: ⟨⟨Resource⟩⟩, ⟨⟨Number⟩⟩ edge: ⟨⟨t,Blank,Resource⟩⟩ constraint: ⟨⟨Resource⟩⟩ = (⟨⟨URI⟩⟩ ∪ ⟨⟨Blank⟩⟩ ∪ ⟨⟨Literal⟩⟩) if t = bag or t = sequence then edge: ⟨⟨_, ⟨⟨t,Blank,Resource⟩⟩, ⟨⟨Number⟩⟩⟩⟩

Table 2. Definition of RDF model constructs - containers and statements

www.dcs.bbk.ac.uk/ dean/eg/RDF.java contains a Java program which creates a Model for RDF and a schema for storing RDF. 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 different tables may be present in a schema.

4 Primitive Transformations on RDF

After a modelling language $\mathcal{M}$ has been specified in terms of the HDM (via the API of AutoMed's Models Definition Repository), a set of primitive transformations for $\mathcal{M}$ is then automatically available for transforming schemas of the model. One 'family' of primitive transformations is available for each different modelling construct.

In the RDF specifications of Tables 1 and 2 there are three different modelling constructs: RDFNode, RDFEdge and RDFContainer, with, respectively, 3, 2, and 3 instances. The set of primitive schema transformations generated for the RDFNode construct from the specification in Table 1 is

- addRDFNode(scheme,query)
- deleteRDFNode(scheme,query)
- extendRDFNode(scheme,query)
- contractRDFNode(scheme,query)
- renameRDFNode(scheme,new-name)

where the 'scheme' parameter is one of ⟨⟨Blank⟩⟩, ⟨⟨URI⟩⟩ or ⟨⟨Literal⟩⟩, the 'query' parameter is an IQL query, and 'new-name' can be any name currently not in use (e.g. renameRDFNode(⟨⟨URI⟩⟩,MyURI) is valid but renameRDF(⟨⟨URI⟩⟩,Literal) is not).

The set of primitive schema transformations generated for the RDFEdge construct from the specifications in Tables 1 and 2 are

- addRDFEdge(scheme,query)
- deleteRDFEdge(scheme,query)

– extendRDFEdge(scheme,query)
– contractRDFEdge(scheme,query)
– renameRDFEdge(scheme,new-name)

where the 'scheme' parameter may be ⟪Triple⟫ or ⟪Statement⟫.

Finally, the set of primitive schema transformations generated for the RDFContainer construct from the specification in Table 2 are:

– addRDFContainer(scheme,query)
– deleteRDFContainer(scheme,query)
– extendRDFContainer(scheme,query)
– contractRDFContainer(scheme,query)
– renameRDFContainer(scheme,new-name)

where the 'scheme' parameter may be ⟪Bag⟫, ⟪Sequence⟫ or ⟪Alternative⟫.

5 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 languages, with one exception. Everything in RDF is described as a resource. A resource has a type, which can define it as a either a class or a property. Properties are therefore defined independently of classes, which is not the case in OO languages.

A summary of the components of RDF Schema is as follows, and is illustrated in Figure 4:

5.1 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.

5.2 Properties

rdf:type A property of a resource which if it has value 'rdfs:Class' means that the resource defines a class, or if it has value 'rdfs:Property' means that 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.

rdfs:domain This defines the classes which the property belongs to. Zero, one or more of these definitions may be given for a property. 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 which refers to another resource.

rdfs:isDefinedBy Sub-property of rdfs:seeAlso which is analogous to the idea of namespaces.

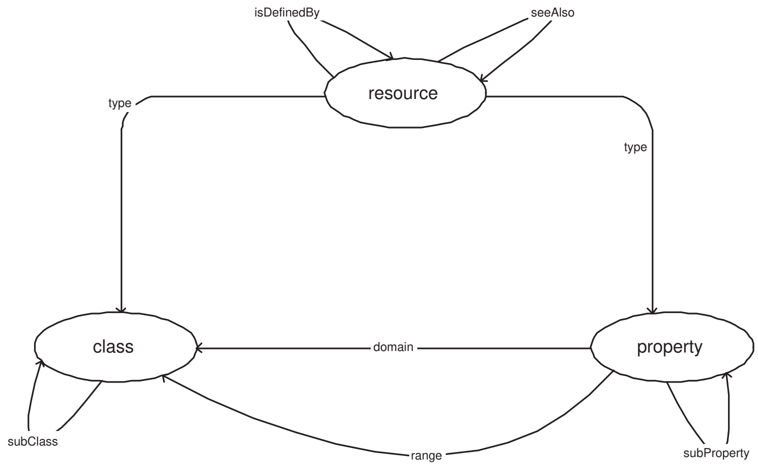

Fig. 4. Components of RDF Schema

6 Representing RDF Schema in the HDM

Table 3 specifies RDF Schema in the HDM. We see that there are two different modelling constructs: RDFSNode and RDFSEdge with, respectively, 3 and 7 instances (as with RDF, this schema will be the same for all RDFS data sources). The set of primitive schema transformations generated for these two constructs are

- addRDFSNode(scheme,query)
- deleteRDFSNode(scheme,query)
- extendRDFSNode(scheme,query)
- contractRDFSNode(scheme,query)
- renameRDFSNode(scheme,new-name)

where the ‘scheme’ parameter can be one of the 3 RDFSNode schemes listed in Table 3, and

– addRDFSEdge(scheme,query)
– deleteRDFSEdge(scheme,query)
– extendRDFSEdge(scheme,query)
– contractRDFSEdge(scheme,query)
– renameRDFSEdge(scheme,new-name)

where the 'scheme' parameter can be one of the 7 RDFSEdge schemes listed in Table 3.

RDFS Constructs	HDM Representation
construct:RDFSNode class: nodal scheme: ⟨⟨rdfs:Resource⟩⟩	node: ⟨⟨rdfs:Resource⟩⟩
construct:RDFSNode scheme: ⟨⟨rdfs:Property⟩⟩	node: ⟨⟨rdfs:Property⟩⟩
construct:RDFSNode scheme: ⟨⟨rdfs:Class⟩⟩	node: ⟨⟨rdfs:Class⟩⟩
construct:RDFSEdge class: linking and constraint scheme: ⟨⟨rdfs:type,rdfs:Resource,type⟩⟩	edge: ⟨⟨rdfs:type,rdfs:Resource,type⟩⟩ constraint: $[r \mid (r,t) \leftarrow \langle\langle \text{rdfs : type, rdfs : Resource, type} \rangle\rangle;$ $t = "rdfs : Property"]) \;=\; \langle\langle \text{rdfs : Property} \rangle\rangle$ constraint: $[r \mid (r,t) \leftarrow \langle\langle \text{rdfs : type, rdfs : Resource, type} \rangle\rangle;$ $t = "rdfs : Class"]) \;=\; \langle\langle \text{rdfs : Class} \rangle\rangle$
construct:RDFSEdge scheme: ⟨⟨rdfs:domain,rdfs:Property,rdfs:Class⟩⟩	edge: ⟨⟨rdfs:domain,rdfs:Property,rdfs:Class⟩⟩ constraint: -
construct:RDFSEdge scheme: ⟨⟨rdfs:range,rdfs:Property,rdfs:Class⟩⟩	edge: ⟨⟨rdfs:range,rdfs:Property,rdfs:Class⟩⟩ constraint: -
construct:RDFSEdge scheme: ⟨⟨rdfs:subClass,rdfs:Class,rdfs:Class⟩⟩	edge: ⟨⟨rdfs:subClass,rdfs:Class,rdfs:Class⟩⟩ constraint: -
construct:RDFSEdge scheme: ⟨⟨rdfs:subProperty,rdfs:Property,rdfs:Property⟩⟩	edge: ⟨⟨rdfs:subProperty,rdfs:Property,rdfs:Property⟩⟩ constraint: -
construct:RDFSEdge scheme: ⟨⟨rdfs:seeAlso,rdfs:Resource,rdfs:Resource⟩⟩	edge: ⟨⟨rdfs:seeAlso,rdfs:Resource,rdfs:Resource⟩⟩ constraint: -
construct:RDFSEdge scheme: ⟨⟨rdfs:isDefinedBy,rdfs:Resource,rdfs:Resource⟩⟩	edge: ⟨⟨rdfs:isDefinedBy,rdfs:Resource,rdfs:Resource⟩⟩ constraint: -

Table 3. Definition of RDFS model constructs

7 An Example

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 [1] 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 a concept e.g. the word forms ”bike” and ”bicycle” can both be used to refer to the same concept.

A cut down version of the schema is given in Appendix A, which includes the class ‘lexical concept’ and its subclass ‘noun’. The following properties are defined: ‘word form’ assigns a string to a concept; ‘hyponym of’ creates an ’isa’ hierarchy of lexical concepts; ‘glossary entry’ gives a textual description of a concept. Figure 5 shows a graph representation of this RDF Schema model.

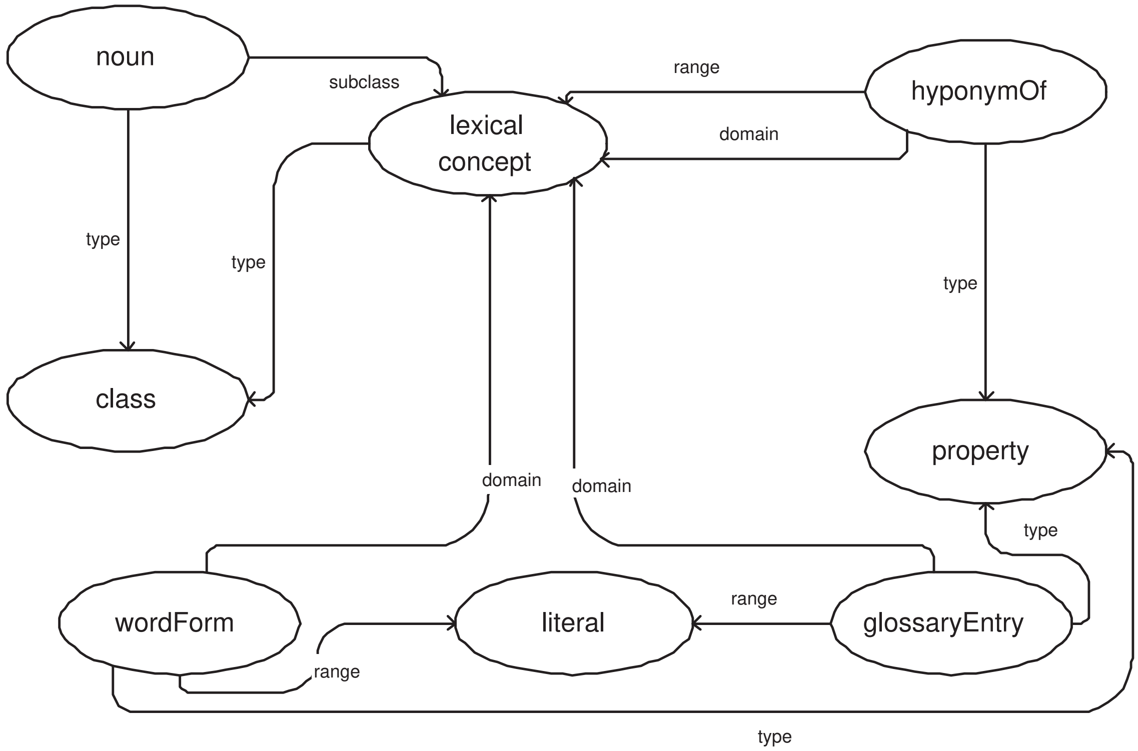

Fig. 5. Graph representation of RDF Schema contained in Appendix A

This RDF Schema model would be represented by following instances of the HDM nodes and edges specified on the right-hand column of Table 3:

```
<<rdfs:Class>>    = {lexicalConcept,noun,literal}
<<rdfs:Property>> = {hyponymOf, glossaryEntry, wordForm}
<<rdfs:Resource>> = {lexicalConcept, noun,
                     hyponymOf, glossaryEntry, wordForm}
<<rdfs:type,rdfs:Resource,type>> =
                    {(lexicalConcept,class), (noun,class),
```

```
                     (literal,class), (hyponymOf,property),
                     (glossaryEntry,property), (wordForm,property)}
<<rdfs:domain,rdfs:Property,rdfs:Class>> =
                    {(hyponymOf, lexicalConcept),
                     (glossaryEntry, lexicalConcept),
                     (wordForm, lexicalConcept)}
<<rdfs:range,rdfs:Property,rdfs:Class>> =
                    {(hyponymOf, lexicalConcept),
                     (glossaryEntry, literal) ,
                     (wordForm, literal)}
<<rdfs:subClass,rdfs:Class,rdfs:Class>> =
                    {(noun, lexicalConcept)}
<<rdfs:subProperty,rdfs:Property,rdfs:Property>> = {}
<<rdfs:seeAlso,rdfs:Resource,rdfs:Resource>> = {}
<<rdfs:isDefinedBy,rdfs:Resource,rdfs:Resource>> = {}
```

Appendix B shows a fragment of the Wordnet database for two concepts, "wheeled vehicle" and a hyponym of it, "locomotive". Several alternative word forms for the concept "locomotive" are given e.g. "engine" and "railway locomotive". Glossary entries for the two concepts are also listed. Figure 6 shows this data in graph form.

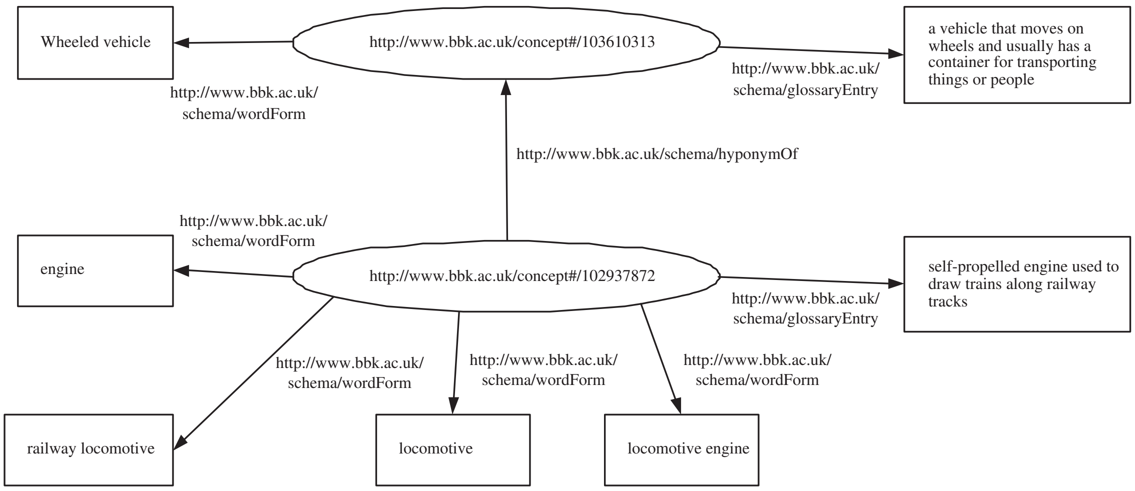

Fig. 6. Graph Based Representation of Appendix B

This RDF model would be represented by following instances of the HDM nodes and edges specified on the right-hand column of Table 1:

```
<<Blank>>   = {}
<<URI>>     = {http://www.bbk.ac.uk/concept/103610313 ,
```

```
               http://www.bbk.ac.uk/concept/102937872,
               http://www.bbk.ac.uk/schema/wordForm ,
               http://www.bbk.ac.uk/schema/glossaryEntry ,
               http://www.bbk.ac.uk/schema/hyponymOf }
<<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"}
<<subject>> = {http://www.bbk.ac.uk/concept/103610313 ,
               http://www.bbk.ac.uk/concept/102937872 }
<<predicate>>={http://www.bbk.ac.uk/schema/wordForm ,
               http://www.bbk.ac.uk/schema/glossaryEntry ,
               http://www.bbk.ac.uk/schema/hyponymOf }
<<object>> =  {http://www.bbk.ac.uk/concept/103610313,
               "Wheeled Vehicle", "engine", "railway locomotive",
               "locomotive", "locomotive engine",
               "self-propelled engine....tracks", "a vehicle...or people"}
<<Triple,subject,predicate,object>> = {
              (http://www.bbk.ac.uk/concept/103610313,
               http://www.bbk.ac.uk/schema/wordForm,
               "Wheeled Vehicle"),
              (http://www.bbk.ac.uk/concept/102937872,
               http://www.bbk.ac.uk/schema/wordForm,
               "engine"),
              (http://www.bbk.ac.uk/concept/102937872,
               http://www.bbk.ac.uk/schema/hyponymOf,
               http://www.bbk.ac.uk/concept/103610313),
              (http://www.bbk.ac.uk/concept/102937872,
               http://www.bbk.ac.uk/schema/wordForm,
               "railway locomotive"),
              (http://www.bbk.ac.uk/concept/102937872,
               http://www.bbk.ac.uk/schema/wordForm,
               "locomotive"),
              (http://www.bbk.ac.uk/concept/102937872,
               http://www.bbk.ac.uk/schema/wordForm,
               "locomotive engine"),
              (http://www.bbk.ac.uk/concept/102937872,
               http://www.bbk.ac.uk/schema/ glossaryEntry,
               "self-propelled engine....tracks"),
              (http://www.bbk.ac.uk/concept/103610313,
               http://www.bbk.ac.uk/schema/glossaryEntry,
               "a vehicle...or people")}
```

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/RDFS.java

8 Conclusions and Future Work

This technical report shows how RDF and RDF Schema can be represented in AutoMed's HDM common data model, so that such data can be exploited as AutoMed data sources.

A question being examined by ongoing work is to see how text data combined with structured data can be better used [10]. The ability to treat ontologies, both natural language such as WordNet and domain specific ones, as data sources able to be integrated with database schema information using the tools provided by AutoMed will be beneficial to this effort.

The RDF data model is a natural candidate for being stored in a graph-based database and an HDM store has been developed which will allow RDF and RDFS data to be stored, as well as 'native' HDM data. Together with AutoMed's functional query language, IQL, this may form a useful RDF data store in its own right.

The approach discussed in this paper assumes that RDF Schema and RDF descriptions are independent - a validating parser would be required to ensure that an RDF description conforms to an RDF Schema description e.g. [9]. An alternative approach that we are investigating is to extend the RDF model for a given data source with additional constraints (expressed in IQL) implied by the RDF Schema description it satisfies.

RDF Schema provides a basic typing schema but richer facilities are often required by designers of Ontologies e.g. cardinality constraints. A number of languages have been proposed in the Semantic Web effort, for example DAML+OIL. Treating DAML+OIL as an AutoMed data source would be a useful addition and is also part of planned work.

References

1. G.A.Miller, R.Beckwith, C.Fellbaum, D. Gross, and K. Miller. Introduction to wordnet: An on-line lexical database. *International Journal of Lexicography*, 3(4):235–244, 1990.
2. Dave Beckett Jan Grant. RDF test cases. *W3C Working Draft*, 2002. http://www.w3.org/TR/2002/WD-rdf-testcases-20020429/#ntriples.
3. P.J. McBrien and A. Poulovassilis. A uniform approach to inter-model transformations. In *Proc. CAiSE'99, LNCS 1626*, pages 333–348, 1999.
4. P.J. McBrien and A. Poulovassilis. A semantic approach to integrating XML and structured data sources. In *Proc. CAiSE'01, LNCS 2068*, pages 330–345, 2001.
5. P.J. McBrien and A. Poulovassilis. Schema evolution in heterogeneous database architectures, a schema transformation approach. In *Proc. CAiSE'02, LNCS 2348*, pages 484–499, 2002.
6. Ralph R.Swick Ora Lassila. Resource description framework (RDF) model and syntax specification. *W3C Recommendation*, 1999. http://www.w3.org/TR/REC-rdf-syntax/.
7. Brian McBride Patrick Hayes. RDF semantics. *W3C Working Draft*, 2002. http://www.w3.org/TR/2002/WD-rdf-mt-20021112/.

8. L. Masinter T. Berners-Lee, R. Fielding. Uniform resource identifiers (URI): Generic syntax. *The Internet Engineering Task Force*, 1998. http://www.ietf.org/rfc/rfc2396.txt.
9. Karsten Tolle. The validating RDF parser (VRP). *ICS-FORTH Institute of Computer Science - Foundation of Research Technology Hellas - Greece*, 2003. http://139.91.183.30:9090/RDF/VRP/index.html.
10. D. Williams. Database driven discovery of structure from partially structured data. Technical report, Sheffield University, 2002. BNCOD PhD Summer School.

A RDFS Schema for Wordnet

```
<?xml version="1.0"?>
<!--
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.rdfs
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:Property rdf:about="&wn;wordForm"
  s:comment='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.'>
  <s:domain rdf:resource="&wn;LexicalConcept" />
  <s:range rdf:resource="&s;Literal" />
</rdf:Property>

<rdf:Property rdf:about="&wn;hyponymOf"
  s:comment='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.'>
  <s:domain rdf:resource="&wn;LexicalConcept"/>
  <s:range rdf:resource="&wn;LexicalConcept"/>
</rdf:Property>

<rdf:Property rdf:about="&wn;glossaryEntry"
  s:comment="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.">
  <s:domain rdf:resource="&wn;LexicalConcept" />
  <s:range rdf:resource="&s;Literal" />
</rdf:Property>

</rdf:RDF>
```

B Sample Wordnet RDFS Word Descriptions

```
<!--  This is a cut down sample of the data -->
<!DOCTYPE rdf:RDF [
 <!ENTITY rdf 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
 <!ENTITY a 'http://www.cogsci.princeton.edu/~wn/concept#'>
 <!ENTITY b 'http://www.cogsci.princeton.edu/~wn/schema/'>]>

<rdf:RDF xmlns:rdf="&rdf;"
xmlns:a="&a;"
xmlns:b="&b;">

<b:Noun rdf:about="&a;103610313"
b:wordForm="wheeled vehicle"/>
</b:Noun>
<b:Noun rdf:about="&a;102937872">
<b:wordForm>engine</b:wordForm>
<b:wordForm>locomotive</b:wordForm>
<b:wordForm>locomotive engine</b:wordForm>
```

```
<b:wordForm>railway locomotive</b:wordForm>
</b:Noun>

<rdf:Description rdf:about="&a;102937872">
<b:hyponymOf rdf:resource="&a;103610313"/>
</rdf:Description>

<rdf:Description rdf:about="&a;103610313">
<b:glossaryEntry>a vehicle that moves on wheels and usually has a container
   for transporting things or people</b:glossaryEntry>
</rdf:Description>

<rdf:Description rdf:about="&a;102937872">
<b:glossaryEntry>self-propelled engine used to draw trains along railway
   tracks</b:glossaryEntry>
</rdf:Description>
```