Visual Displays for Browsing RDF Documents

Peter Eklund
School of Information Technology and Electrical Engineering
The University of Queensland
St. Lucia QLD 4072
peklund@itee.uq.edu.au

Introduction

Hyperbolic browsers are motivated by the "Circle Limit IV" woodcut of M.C. Escher. The hyperbolic tree view was introduced in graph drawing by Lamping and Rao [2] who observed that large structures could be compactly displayed by projecting a tree onto a hyperbolic plane. The effect of the projection is that components appear diminishing in size and radius exponentially the further they move from the centre of the diagram. The arguments for the hyperbolic display are twofold: an order of magnitude more nodes of a tree can be rendered in the same display space and the focus is maintained on the central vertex of the display and its immediate neighbourhood. The hyperbolic view is particularly useful for hierarchical diagrams with large numbers of leaves and branches and where neighbourhood relationships are meaningful. Examples of the hyperbolic view are INXIGHT's Star Tree[1] and HYPERPROF[2]. Given that the pure hyperbolic geometric projection is patented, projection onto a sphere, and diminishing radial layout views are the drawing approaches we have experimented with.

Recent interest in the Semantic Web has invigorated interest in programs that render ontological data sources (structured metadata) in the style of semantic networks well known in AI. The ONTOBROKER [1] project used a hyperbolic browser to view semantic relationships between frames in F-Logic. ONTORAMA traces its lineage to the ONTOBROKER[3] Java source code, but the code has been substantially re-written since 1999. The ONTORAMA effort involved refactoring the Java for the WEBKB-2 project [3, 4] which uses its own internal formats (conceptual graphs).

OntoRama and RDF

RDF is a widely used standard for knowledge exchange. Thus, by using RDF, RDF(S) and DAML+OIL as inputs, we broaden the application base for ONTORAMA. Further, access to a selection of RDF libraries encourages testing and generalising program operation to handle different input files. We are progressively adapting ONTORAMA to handle DAML+OIL compliant RDF(S) and reified RDF: the program is therefore still under development.

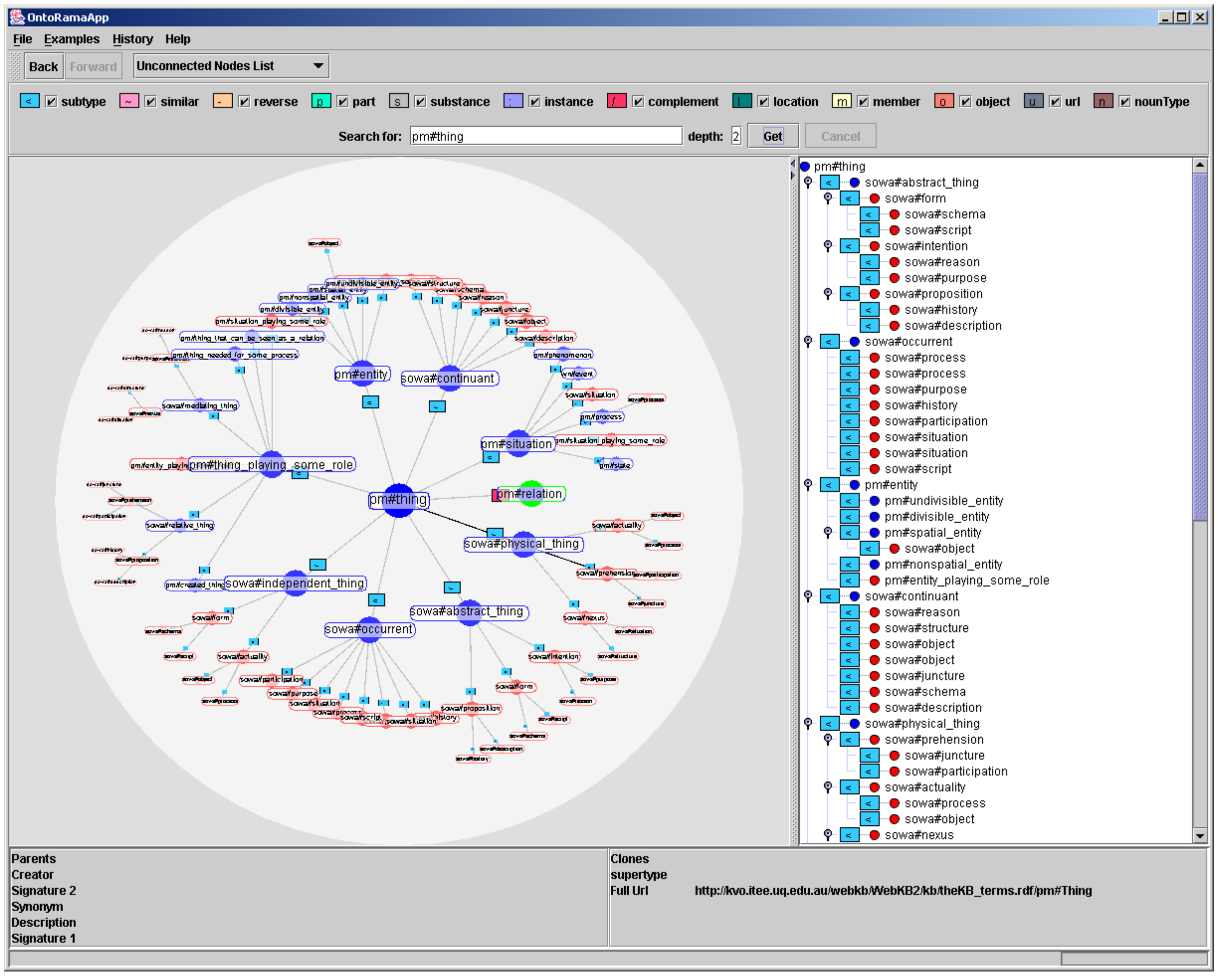
Some of the more obvious challenges for a general purpose RDF, RDF(S) and DAML+OIL browser are: (i) many RDF ontologies are not strictly hierarchical and include multiple inheritance; (ii) many properties may connect Resources and Properties so there is more than a single transitive edge type in the RDF graph; (iii) often objects are isolated and may not be reachable by following relation links from the top (root) item: not all RDF objects found by search are connected. An input file may therefore be a forest rather than a tree and the browser must be able to navigate across the individual components. (iv) Further, we need to be able to represent RDF Resources/Descriptions so that the visual representation of the documents does not loose any information. Less obviously, (v) RDF that is reified, predicates that have a situational content, must be dealt with in a visual way and (vi) there are also many syntactically equivalent forms of RDF. These create design issues for ONTORAMA since, not only is the RDF input syntax complex, but also the usual tree-view metaphor for the hyperbolic-like view breaksdown.

```
...
<rdfs:label xml:lang="en">artifact</rdfs:label>
<rdfs:label xml:lang="en">artefact</rdfs:label>
...
<dc:Creator>http://cogsci.princeton.edu/~wn/
</dc:Creator>
<rdfs:comment>a man-made object taken as a whole
</rdfs:comment>
<rdfs:subClassOf
 rdf:resource="http://wekbkb.org/pm#CreatedThing"/>
<rdfs:subClassOf
 rdf:resource="http://wekbkb.org/wn#PhysicalObject"/>
<rdfs:subClassOf
 rdf:resource="http://wekbkb.org/wn#WholeThing"/>
<daml:disjointWith
 rdf:resource="http://wekbkb.org/wn#NaturalObject"/>
...
```

RDF(S) objects may inherit properties in multiple ways, point (i) above; so some adaptation of the strict tree-structure browsing principles, particularly to the hyperbolic browsers view, needs to be addressed. Any RDF file can be considered an acyclic graph where a child can have multiple parents (as is the RDF shown above) where a single Artifact has four superclass parents, CreatedThing, PhysicalObject, WholeThing and NaturalObject.

[1]http://www.inxight.com
[2]http://www.physics.orst.edu/~bulatov/HyperProf/
[3]http://ontobroker.aifb.unikarlsruhe.de/index_ob.html

Proceedings of the 7th Australasian Document Computing Symposium,
Sydney, Australia, December 16, 2002.



Search for: pm#thing depth: 2

- pm#thing
  - sowa#abstract_thing
    - sowa#form
      - sowa#schema
      - sowa#script
    - sowa#intention
      - sowa#reason
      - sowa#purpose
    - sowa#proposition
      - sowa#history
      - sowa#description
  - sowa#occurrent
    - sowa#process
    - sowa#process
    - sowa#purpose
    - sowa#history
    - sowa#participation
    - sowa#situation
    - sowa#situation
    - sowa#script
  - pm#entity
    - pm#undivisible_entity
    - pm#divisible_entity
    - pm#spatial_entity
      - sowa#object
    - pm#nonspatial_entity
    - pm#entity_playing_some_role
  - sowa#continuant
    - sowa#reason
    - sowa#structure
    - sowa#object
    - sowa#object
    - sowa#juncture
    - sowa#schema
    - sowa#description
  - sowa#physical_thing
    - sowa#prehension
      - sowa#juncture
      - sowa#participation
    - sowa#actuality
      - sowa#process
      - sowa#object
    - sowa#nexus

Parents
Creator
Signature 2
Synonym
Description
Signature 1

Clones
supertype
Full Url http://kvo.itee.uq.edu.au/webkb/WebKB2/kb/theKB_terms.rdf/pm#Thing

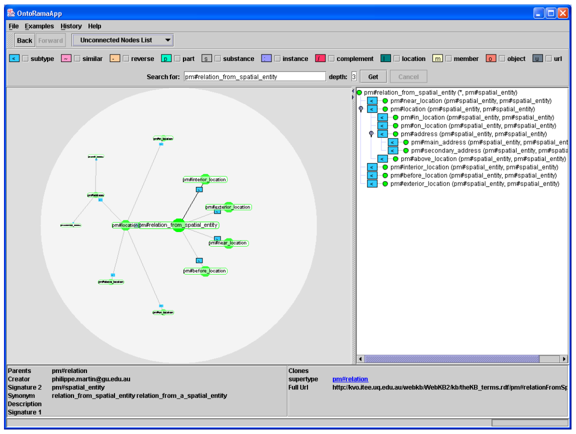
Figure 1. ONTORAMA: **The Property Hierarchy shows the subclass/superclass relations between relation types.**

The data structure must therefore conform to a tree and non-planar graphs transformed to this form. The solution is to copy entire sub-branches in the graph for vertices with multiple parents: producing a tree structure where vertices have only one incoming edge. Once a vertex (and its ancestors as a sub-tree) have been copied, each vertex can be drawn in the coordinate plane using either hyperbolic or radial layout algorithms. Copying vertices and sub-graphs in this way has a disadvantage, the size of the overall tree increases, reducing the effectiveness of the hyperbolic view and an argument for its use. In addition, the GUI needs to be designed to highlight copied vertices and sub-graphs, so that these replicated structures are easily identifiable. Copied vertices and sub-trees are rendered red to distinguish them in the interface and copies are connected to this ring to enable the user to identify and unify the copies, locating them in the view.

A disadvantage of using a tree metaphor for browsing RDF is that an RDF file may contain multiple trees (or simply objects and attributes that disconnected). Initially, ONTORAMA relied on the fact that all input items were interconnected directly (or indirectly) connected to a "top" ontology item. Overcoming this problem, to allow ONTORAMA to browse a *forest* rather than a *tree*, the various components of the tree are made accessible by a pull-down menu (top-left) called *Unconnected nodes list*, here the tree components are named after the root node name of each of the component trees. ONTORAMA must display two tree structures: one for the object or class hierarchy and the other for the property hierarchy. To illustrate, consider the following RDF at the top of page 4.

The solution to rendering property hierarchies is to introduce two graphical structures that interoperate. Color and shape are useful device to draw attention to the points where the interactions between the RDF Resource hierarchy and the Property hierarchy inter-connect. This is shown in Fig. 1, note the *green* Property Relation immediately to the right of Thing, the notation of WEBKB-2 indicates that these are exclusive types, namely that the Resource/Object hierarchy is exclusive of the Property/relation hierarchy.

We advocate multiple visual views over an ontology, and ONTORAMA includes both the hyperbolic-style view and a tree view widget. Other visual views, such as true 3D displays and


OntoRamaApp
File Examples History Help
Back Forward Unconnected Nodes List
subtype similar reverse part substance instance complement location member object url
Get Cancel
pm#interior_location
pm#before_location

Search for: pm#relation_from_spatial_entity depth: 3

```
pm#relation_from_spatial_entity (*, pm#spatial_entity)
    pm#near_location (pm#spatial_entity, pm#spatial_entity)
    pm#location (pm#spatial_entity, pm#spatial_entity)
        pm#in_location (pm#spatial_entity, pm#spatial_entity)
        pm#on_location (pm#spatial_entity, pm#spatial_entity)
        pm#address (pm#spatial_entity, pm#spatial_entity)
            pm#main_address (pm#spatial_entity, pm#spatial_ent
            pm#secondary_address (pm#spatial_entity, pm#spatia
        pm#above_location (pm#spatial_entity, pm#spatial_entity)
    pm#interior_location (pm#spatial_entity, pm#spatial_entity)
    pm#before_location (pm#spatial_entity, pm#spatial_entity)
    pm#exterior_location (pm#spatial_entity, pm#spatial_entity)
```

Parents	pm#relation
Creator	philippe.martin@gu.edu.au
Signature 2	pm#spatial_entity
Synonym	relation_from_spatial_entity relation_from_a_spatial_entity
Description	
Signature 1	

Clones	
supertype	pm#relation
Full Url	http://kvo.itee.uq.edu.au/webkb/WebKB2/kb/theKB_terms.rdf/pm#relationFromSp

Figure 2. ONTORAMA**: The Property Hierarchy shows the subclass/superclass relations between relatioon types. Note that the tree widget now displays the relational signature.**

concept lattices, may also be appropriate for browsing ontological data. Most computer users are familiar with viewing files in a directory structure using a so called "tree widget". However, a tree view does not usually represent the idea of a file belonging to more than one sub-directory since the physical (or logical) structure of most file systems insists on a strict hierarchy.

Unlike the hyperbolic view, where a fixed envelope of space displays the children and there may be hundreds of children crowded into a small radial space, the tree view allows the structure to be drawn downwards with no limiting viewing space. Therefore, the advantage of the hyperbolic view is that users can understand the ontology structure by seeing the relational links between the vertices however this needs to be supplemented by a tree view explorer when there are large numbers of children.

ONTORAMA exploits the advantages of both views in different circumstances by allowing them to complement one another. Each view should allow the user to execute similar commands, the user can click on a vertex in the hyperbolic view and gain focus in both hyperbolic and file explorer view. The vertex is highlighted in both. Vertices in the hyperbolic view can be double clicked to drill-down or roll-up in a similar manner to collapsing the directory structure in a file explorer. When a node is folded in the hyperbolic view, it is represented by a square and the rest of the sub-branch becomes invisible. Pruning branches in the hyperbolic view enables the user to adjust the focus and details of the display and enhance the cognitive effect of navigating the ontology with the GUI.

ONTORAMA can be configured to parse different RDF and XML inputs. The ONTORAMA configuration file allows the user to specify various relation links. Assigned to relation links are a symbol and colour, used in ONTORAMA to distinguish link types. These images are used to display the relation types. The user may select relations they wish to view by selecting the link check boxes (see Figure 1 (top)). Queries are then filtered to show the specified relation links.

The object type properties in ONTORAMA are fully configurable via the configuration XML file. This means if we need to display an ontology with different relation links, or different properties for each object type, we need only alter the configuration file without re-building the application. ONTORAMA reads the configuration file on start up. It consists of two sections: the `ontology` section and the `rdfMapping` section. The `ontology` section describes all details relevant to loading an ontology. The section includes a set of `relation` elements, each defining an ontology relation link. Similarly, we define concept type properties. Concept type properties are properties applicable to a concept type in the current ontology. For example, concept types have properties such as `type, description, creator, synonyms,` etc. Each relation is defined as a `relation` element is assigned an id. The id is a reference to the corresponding relation that ONTORAMA uses. We also define a name for each relation link. Some relations can have two names, which means that they inverse relations. An example of such relations are subtype/supertype. Consider the term `wn#dog`. This term is connected to the term `wn#canine` via relation link *subtype* (`wn#dog` is subtype of `wn#canine`). The inverse relation link to *subtype* is *supertype* and the following is a fragment from the configuration file describing the subtype/supertype relation:

```
<relation id="1">
 <relationType name="subtype" mappingSymbol="&lt;"/>
 <relationType name="supertype" mappingSymbol="&gt;"/>
 <display color="#33CCFF" symbol="&lt;"/>
</relation>
```

Each relation is identified by a relation id and name. Note that ONTORAMA will display only one relation link for each relation and the first declared link in the `relation` is displayed. Display properties for each relation link, such as color and symbol, are also included. Concept type properties are defined using the element `conceptProperty`. At present this element has only an id: a string describing this property. Once these elements are read by ONTORAMA, the application knows that it needs to create corresponding entries in the description panel.

```
<rdfMapping>
  <relationLinks>
    <map id="1" type="supertype" tag="subClassOf"/>
    <map id="3" type="part" tag="part"/>
    <map id="6" type="inclusive" tag="jointWith"/>
  </relationLinks>
  <conceptProperties>
    <map id="Description" tag="comment"/>
    <map id="Creator" tag="Creator"/>
    <map id="Synonym" tag="label"/>
  </conceptProperties>
</rdfMapping>
```

The `RdfMapping` section of the configuration file specifies how the application will be able to

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#"
         xmlns:dc="http://purl.org/metadata/dublin_core#"
         xmlns:daml="http://www.daml.org/2000/10/daml-ont#"
         xmlns:pm="http://wekbkb.org/pm#">

<rdf:Property rdf:about="http://www.daml.org/2000/10/daml-ont#equivalentTo">
  <rdfs:label xml:lang="en">equivalent_to</rdfs:label>
  <dc:Creator>http://www.daml.org/2001/03/daml+oil.daml</dc:Creator>
  <rdfs:subPropertyOf rdf:resource="http://wekbkb.org/pm#equal"/>
</rdf:Property>

<rdf:Property rdf:about="http://wekbkb.org/pm#equal">
 <rdfs:label xml:lang="en">equal</rdfs:label>
 <dc:Creator>philippe.martin@gu.edu.au</dc:Creator>
 <rdfs:comment>"=" in KIF</rdfs:comment>
 <rdfs:subPropertyOf rdf:resource="http://wekbkb.org/pm#equivalenceRelation"/>
 <daml:complementOf rdf:resource="http://wekbkb.org/pm#different"/>
</rdf:Property>

 <rdf:Property rdf:about= "http://wekbkb.org/pm#equivalenceRelation">
    <rdfs:label xml:lang="en">equivalence_relation</rdfs:label>
    <dc:Creator>philippe.martin@gu.edu.au</dc:Creator>
    <rdfs:comment>instance of equivalence_relation_class</rdfs:comment>
    <rdfs:subPropertyOf rdf:resource="http://wekbkb.org/pm#orderingRelation"/>
 </rdf:Property>

     <rdf:Property rdf:about="http://wekbkb.org/pm#orderingRelation">
       <rdfs:label xml:lang="en">ordering_relation</rdfs:label>
       <dc:Creator>philippe.martin@gu.edu.au</dc:Creator>
       <rdfs:comment> e.g. pm#kind, rdfs#sub_class_of, pm#part, pm#equal</rdfs:comment>
       <rdfs:subPropertyOf rdf:resource="http://wekbkb.org/pm#relation"/>
     </rdf:Property>

        <rdf:Property rdf:about="http://wekbkb.org/pm#relation">
          <rdfs:label xml:lang="en">relation</rdfs:label>
          <rdfs:label xml:lang="en">related_with</rdfs:label>
          <dc:Creator>philippe.martin@gu.edu.au</dc:Creator>
          <rdfs:comment>type for any relation (unary, binary, ...) </rdfs:comment>
          <rdf:type rdf:resource= "http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
          <daml:complementOf rdf:resource="http://wekbkb.org/pm#Thing"/>
        </rdf:Property>
</rdf:RDF>
```

find items defined in the ontology section in the source RDF file. The preceding example illustrates an `rdfMapping` element. `rdfMapping` consists of two subsubsections: `relationLinks` maps relation links defined in the ontology section to RDF tags. `conceptProperties` maps concept type properties to RDF tags. For instance, the element `<map id="1" type="supertype" tag="subClassOf"/>` is a mapping relation type *supertype* to the RDF tag *subClassOf*. This is an interesting example because (as stated previously), ONTORAMA displays the relation link *subtype*, and the application has to be told of the inverse *supertype* links to be able to display them. The following illustrates;

The last statement is translated by ONTORAMA as: `wn#TrueCat` is *supertype* of `wn#Cat`. However, ONTORAMA needs to reverse this statement as only *subtype* relation links are displayed resulting in: `wn#Cat` is the *subtype* of `wn#TrueCat`.

Conclusion

ONTORAMA has emerged as a practical tool for viewing ontologies described in RDF using Java Swing. We describe a solution to hyperbolic-style layout that is complicated by the prevalence of multiple inheritance in most ontological structures. We further describe extensions to the usual hyperbolic-style view allowing multi-link types to be rendered and filtered. Issues of application generality, configuration files for the application in XML, and the correspondence between behaviours in the application and RDF descriptions are discussed. ONTORAMA can be trialed at http://www.ontorama.org.

References

[1] Fensel, D., S. Decker, M. Erdmann, and R. Studer, "Ontobroker: Or How to Enable Intelligent Access to the WWW," *Proc. 11th Knowledge Acquisition Workshop* (KAW98), Banff, Canada, April 1998, pp. 8–23.

[2] Lamping, J. and Ramana Rao: The Hyperbolic Browser: A Focus+Context Technique for Visualizing Large Hierarchies, *Proceedings of CHI'95, ACM Conference on Human Factors in Computing Systems*, New York, pp. 401-408, 1995.

[3] Martin, P. and P. Eklund: Embedding Knowledge in Web Documents, The Eighth International World Wide Web Conference, (WWW8), pp. 324–341, Elsevier, 1999.

[4] Martin, P. and P. Eklund: Knowledge Indexation and Retrieval and the Word Wide Web, *IEEE Intelligent Systems* — Special Issue on "Knowledge Management and Knowledge Distribution over the Internet, July, pp. 18-25, 2000.