

# Show Me What You Mean!

## Exploiting Domain Semantics in Ontology Visualization

Ravi Pavagada<sup>1</sup>, Christopher Thomas<sup>1</sup>, Amit Sheth<sup>1</sup>, William S. York<sup>2</sup>

<sup>1</sup>Large-scale Distributed Information Systems Laboratory,

Dept. of Computer Science, University of Georgia, Athens, GA, USA

<sup>2</sup>Complex Carbohydrate Research Center,

Dept. of Biochemistry and Molecular Biology, University of Georgia, Athens, GA, USA

<sup>1</sup>{pavagada, cthomas, amit}@cs.uga.edu, <sup>2</sup>will@ccrc.uga.edu

### ABSTRACT

Ontologies build the backbone for many life-sciences applications. These ontologies, however, are represented in XML-based languages that are meant for machine-consumption and hence are difficult for humans to comprehend. For a meaningful visualization of these ontologies, it is important that the display of entities and relationships captures the cognitive representation of the domain as perceived by the domain experts. In this paper we present OntoVista, an ontology visualization tool that is adaptable to the needs of different domains, especially in the life sciences. While keeping the graph structures as the predominant model, we provide a semantically enhanced graph display that gives users a more intuitive way of interpreting nodes and their relationships. Additionally, OntoVista provides comfortable interfaces for searching, semantic edge filtering and quick-browsing of ontologies. To this end, we extended the Jambalaya plugin for Protégé to allow for customization and integration of different layouts. As a use case, we demonstrate how the ontology-encoding of complex carbohydrate structures is transformed into a standard graphical representation [David Goldberg 2005] of carbohydrates familiar to biochemists

### 1 INTRODUCTION

An Exabyte of data is generated in the life sciences domains each year [5]. Structuring the data and mediating between different repositories that store this data is a huge challenge. Ontologies have been proposed as an overlying structure of knowledge that helps With this task of organization and mediation. They are defined as formal specifications of shared conceptualizations [23] and as such used to capture domain models using formal representations that are interpretable by machines. It naturally turns out, though, that the ontologies themselves are extremely large and of high complexity. While primarily meant for machine consumption, these knowledge repositories still need to be designed, evaluated and used by humans. The way we understand and process information is by putting it in a context with knowledge we already have. We draw connections, build cognitive relations. In ontologies this process is modeled by formally assigning relationships between concepts. Their underlying structure can hence be drawn as a graph. Since graphs also abstractly capture the mental representation of domains, this format seems ideal for visualizing knowledge in a comprehensible

way. However, two properties of formal knowledge bases impede this representation:

- The vastness of information at hand
- The information is presented to machines in a different format than the representation users or scientists are most familiar with.

To address the former, we need better search and filter capabilities that restrict the amount of information shown to the user at any point in time. For the latter, the formal structure of information needs to be converted into a representation that is closer to the user's cognitive representation.

The most important parts of knowledge representations are the connections or relationships between the individual pieces of information. There are many different ways things can be related. This multitude of relationships however, can be seen as a hierarchy starting with a root relationship that broadly defines a particular type of relationship. The relationship hierarchy is defined by adding more specific types of relationships, which are subsumed by more general relationships. The most common relationships in an ontology include subclass\_of (inheritance), instance\_of (instantiation) and part\_of (aggregation), all of which can be conceptually or physically viewed as different types of containment relationships.

A visualization tool that helps the user to comprehend the area of interest more quickly needs to be able to show the connections of the graph that are useful for the viewer. In this respect, a tool needs to assist the user in finding the important entities and displaying the intended relationships between those entities. Since knowledge repositories can have millions of entities and hundreds of relationship types, a user needs to be guided and assisted throughout this process. A relationship hierarchy facilitates filtering of important information only if the visualization environment allows filtering of relationships in a recursive manner based on this hierarchy. With few mouse clicks, it must be possible to change the focus from showing part\_of relationships between the entities to showing linkage, functional dependence and other relationships.

In addition to limiting the amount of information shown, the visualization tool should as much as possible present the information in a meaningful manner. The definition of a meaningful visualization, however, varies from domain to domain. For example, a symbol that makes sense to a chemist might be meaningless to a mathematician or even have a different meaning in the field of mathematics. Different groups in the same fields of science might also use different representation.

Ontologies are meant to unambiguously formalize knowledge regardless of the commonly used symbols. A visualization however, should convert this description back to the group-specific view of the domain.

Most ontology visualization environments follow some sort of graph-representation paradigm that only takes general graph-semantics into account. A domain scientist who is not a Computer Scientist or expert in formal representation needs to be presented with a display that is closer to her domain of expertise. A simple graph display contains limited visual semantics. It basically consists of nodes and edges that can be labeled, color-coded, assigned different shapes and patterns or arranged in different layouts, such as radial, fisheye, tree or spring. An extended graph display allows for nesting of subgraphs within nodes. This representation suggests some kind of containment relationship between the enclosing node and the enclosed subgraph. The presented ontology visualization aims at abstractly modeling the domain expert's cognitive representation of a domain by combining these techniques for graph representation in a way that assigns meaning to the different kinds of nodes, edges and layouts. A knowledge base that, for example, focuses on molecular structures, can show the atoms as a subgraph within the molecule entity. This subgraph is then laid out according to common representations, such as ball-and-stick. In addition to the shape of the nodes representing the atoms, the angles of the edges between those nodes are laid out according to the linkage information available in the formal representation of the molecule.

In general, concepts and relationships often have conventional symbols that are domain dependent. For example, a chemical reaction is commonly shown as an arrow between the substrate and product involved in the reaction. In an ontology, however, a chemical reaction will require a more complicated representation that is meaningful to a machine, but not to the human observer. Thus, a visualization tool should be able to translate this machine-centric representation into a human discernible format. To take a more tangible example, the visualization of a geometry ontology that defines geometrical shapes internally in terms of lines angles and radii, can show these entities in simple shapes such as squares, circles, triangles etc.

In this paper we present OntoVista, an ontology visualization tool with unique capabilities related to complex (representationally rich) biological and biochemical ontologies. OntoVista was developed by extending Jambalaya [20], an existing ontology visualization plugin for the most widely used ontology editor, Protégé [22]. The main advances compared to Jambalaya are

- A customizable layout that, based on a layout configuration file, assigns default shapes, colors, layouts and node positions.
- Recursive edge filtering based on a property hierarchy
- Improved navigation and search using a quick ontology browser and a new search interface

These capabilities enable OntoVista to be a more useful ontology visualization tool for domain scientists.

The rest of the paper is organized as follows. Section 2 gives a brief introduction to Jambalaya and GlycO [8], a complex biochemical ontology that served as our use case. In Section 3, we describe OntoVista's custom layout, Semantic Arc Filter, Ontology Browser and advanced Searches. Section 4 talks about the related work in the area of ontology visualization.

## 2 BACKGROUND

### 2.1. Jambalaya

Jambalaya is an ontology visualization and editing plugin for Protégé. It uses visualization techniques such as nested graph view, combined with pan, zoom and fisheye-view for interactive navigation. It provides various layouts such as Radial Layout, Spring Layout, horizontal Tree Layout and vertical Tree Layout, to meet different user-preferences. However, the layouts provided by Jambalaya are not customizable and cannot display views that capture the cognitive representations of the domain as perceived by the domain experts. This was the main motivation behind the development of our customizable layout. The text based search interface in Jambalaya only allows searching of classes and instances. Users are able to navigate to the selected class or instance after selecting them from the search result. While using Jambalaya extensively for the design of ontologies, we found that the searches provided in Jambalaya are not powerful enough for analysis and navigation of complex ontologies. We found a need for advanced search capabilities such as description search, relationship search, triple search, domain-range search and Semantic search while visualizing complex ontologies such as GlycO [8]. Hence our goal was to design a search interface in OntoVista that implements some of the capabilities of ontology-query languages such as SPARQL [14] without requiring the user to learn their rather complex syntax. Jambalaya provides an arc filter that can be used to filter and un-filter edges from the graph view of the ontology. This arc filter however does not take advantage of the property hierarchy that is available in well structured ontologies. Thus, we felt the need for a semantic arc filter for quick filtering and navigation.

### 2.2 GlycO

The field of glycobiology deals with the structures, chemistry, biosynthesis, and biological functions of complex carbohydrates, so-called glycans. The structures of glycans are more complicated than those of genes and proteins, which are linear chains composed of nucleotide and amino acid residues, respectively. Moreover, all of the residues in these biopolymers are connected via a single type of linkage. Conversely, glycans have a branched tree structure rather than a linear chain, and the connection between carbohydrate residues shows significant structural heterogeneity, varying, for example, in position and anomeric configuration. Thus, modeling of primary structural features is considerably more difficult for glycans than for genes and proteins.

GlycO is a highly specialized ontology for the glycobiology domain. It contains formalized descriptions of glycan structures, enzyme functions and biosynthetic pathway information. The individual glycans are modelled as collections of monosaccharide residues. The relation between connected residues is called "*is\_linked\_to*." This relation has a defined direction, which provides the structure of the glycan model as a directed graph, or more specifically, a tree. All residues which instantiate the relation "*is\_linked\_to*" must also instantiate the property "*is\_linked\_via*." When residue-A *is\_linked\_to* residue-B, the property "*is\_linked\_via*" specifies the precise atomic attachment point on residue-B where residue-A attaches, and the property "*has\_linking\_atom*" specifies the site on residue-A that is attached to residue-B. Both the local and remote binding sites are indexed using a standard chemical numbering system. Within the

configuration property file, this number is called the *LinkType* and is used by OntoVista to configure the visualization layout of residues in a way that preserves a standard graphical representation format used by glycobiologists to visualize glycans and the residues they contain. Thus, OntoVista is capable of presenting glycans in a format similar to the cartoon representation commonly found within textbooks and current research papers. The ubiquity and usefulness of this representation for glyco-biologists was a key motivator behind the creation of the custom layout in OntoVista.

### 3 ONTOVISTA

We describe OntoVista's custom layout feature that uses the information in the ontology as described by the domain experts. The information describing how to visualize contents of an ontology does not belong in the ontology itself. Rather, for this purpose we use layout settings provided by the domain experts to display a view that is meaningful to the users. The custom layout was initially designed to display complex carbohydrate molecules in a way that domain experts are used to; the so-called Cartoonist [David Goldberg 2005] representation. However, the layout is customizable and can generate domain specific views for the ontology. The view generated is very much dependent on the user's configuration settings. The user can also change the settings to generate views that are of user's interest. We have also refined searching and filtering in Jambalaya. Taking advantage of a hierarchical representation of the relationships (i.e. properties and their sub-properties) in the ontology, a semantic arc filter was developed to help users to quickly visualize the nodes connected through particular sets of relationships such as partonomy/containment, chemical interaction/ reaction etc. Thus, Semantic Arc Filter allows users to show/hide all partonomy relationships such as *part\_of* and its sub-properties or chemical interaction relationships such as *interacts\_with* and its sub-properties at once. We have enhanced the basic search capabilities that are there in Jambalaya by adding advanced searches such as relationship search, description or comment search, domain-range search, triple search and semantic search. Additionally, OntoVista provides a Quick Ontology Browser for easy access of class, sub-classes, properties and sub-properties which can be later used in queries while performing searches.

#### 3.1. Custom Layout

One of the shortcomings we found in the Jambalaya visualization tool was that the choices of node shapes and colors were limited to meta-properties of the nodes. In Jambalaya, all instances/classes in a ontology can be set to a particular shape and color. It doesn't allow users to set a specific color and shape for a given class or its instances.

Thus, Layouts in Jambalaya are designed to capture only the graph structure. In order to create domain specific views, layouts should not only be able to change the shape and color of the nodes but also be able to adjust the physical locations of nodes based on conceptual location.

The custom layout creates containers or nested views using the containment relationships such as *part\_of* or *has\_component*. The layout is then applied to all the instances of the set of classes defined as the root classes. Visualization of these instances/nodes will give a domain specific view of the ontology as perceived by domain experts. If these nodes have internal nodes the layout is applied to those nodes as well. The color and shape of a specific node is determined according to the layout settings and the

relative position of each internal node in the layout is based on information in the ontology. Each of the internal nodes can be assigned positions such as "Left", "Down", "Diagonal up" and "Diagonal down" relative to the position of the previous internal node. This is different from other layouts as it can generate domain specific views from the position information in the ontology and can set shapes and colors for each of the internal nodes based on the layout settings.

The custom layout uses a property based file that has the following layout properties.:

1. Containment relationship layout property is specifies the relationships upon which layout can cluster to create containers or nested views of nodes.
2. Root class layout property specifies the names of the classes whose instances are to be visualized. The layout can only be applied to instances of the classes specified in this property.
3. Class name layout property must be specified for each of the individual class names mentioned in the Class layout property. The desired shape and color of the instances of a particular class must be specified using this layout property. This helps to set specific shape and color for each of the internal nodes of the layout.
4. Link relationship layout property is the relationship in the ontology that specifies the position of the node. This property allows the layout to access the position of the nodes from the ontology.
5. Connection relationship layout property specifies the relationships that connects the internal nodes of the layout.
6. Link type layout property can be used to specify the possible positions of the nodes. In GlycO, the possible linkage sites (e.g. atom 2, 3, 4, or 6).
7. Positions layout property explicitly specifies the actual meaning for each of the positions in the link types property. For example in GlycO, link position value of 4 means that the node is to be placed to the "Left", link position value of 6 means that the node is to be placed "Diagonally up", link position value of 3 means that the node is to be placed "Diagonally down", link position value of 2 means that the node is to be placed directly "Below" the previous node.

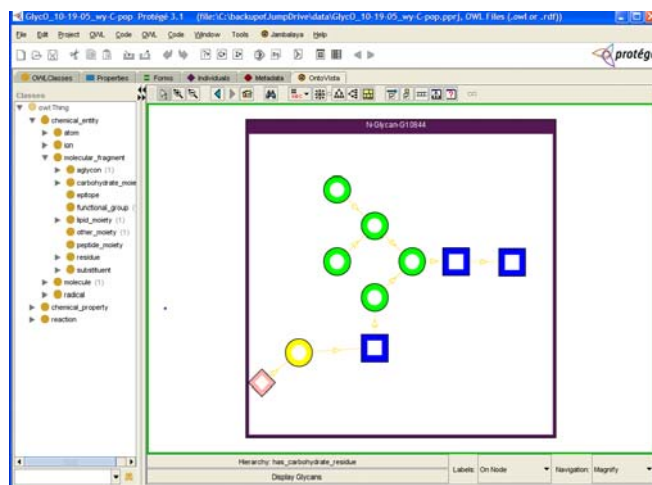


Figure 1 shows an image of a glycan generated using Custom Layout.

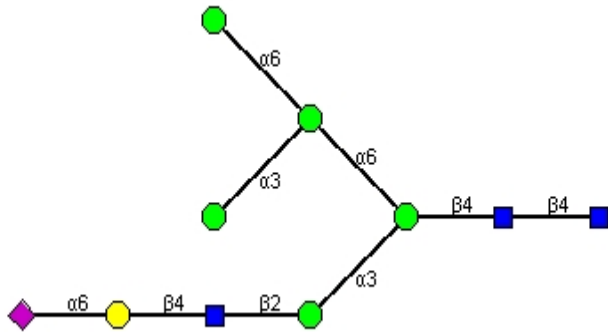


Figure 2 shows the standard cartoonist representation of the glycan shown in Figure 1.

We will demonstrate the custom layout using the Glyco ontology. Initially, we will show how we can generate the cartoonist representation of glycans using the custom layout. Cartoonist is a standard representation of glycans known to Biochemists, in which each of monosaccharide residue types is represented in different shapes and colors.

The Glyco ontology is initially loaded in OntoVista along with the layout configuration settings specific to the ontology. The custom layout then generates images of glycans using the layout settings and link information (connection that exists between the residues) in the Glyco ontology. Depending on the type of residue, the layout can display instances of *carbohydrate residues* in various shapes such as squares, diamond, circles, or triangles and colors. The key feature of Custom layout is that it can generate different images for different instances of glycans at runtime as shown in Figure 1. The corresponding cartoonist representation of the glycan is shown in Figure 2.

We will use another automobile ontology to illustrate another layout configuration. The ontology describes some internal components of car such as engine, transmission, brakes, headlight etc. We will be using engines of different cars to demonstrate our custom layout. The internal components of the engine are connected to each other by an *is\_connected\_to* relationship. Figure 3 shows an image of a Chevy cavalier car engine which is comprised of interconnected components such as engine block, spark plugs, crank and piston. The above figure shows the *champion plugs* in yellow squares and Chevy engine block in green circle based on the color and shape specified in the layout settings.

### 3.2. Semantic Filtering

The semantic arc filter was developed to help users quickly visualize the nodes connected through particular sets of relationships such as partonomy/containment, chemical interaction/ reaction etc. It is used to filter or un-filter the properties and their respective sub-properties. With a few mouse clicks the user can change the focus of the ontology view from partonomy relationships to chemical reactions. Without the hierarchical arc filter, this process would require tedious selection of every relationship involved in partonomy and chemical reactions respectively. In Glyco, for example, the property *part\_of* has sub-properties, as described in Section 2.1. By selecting *part\_of* in Semantic Arc Filter, all its sub-properties are also selected. This holds analogously for de-selection of a property. In general selections are recursively applied to the full branch of the tree that has been selected. Additionally, of course, the user can select or deselect individual sub-properties. Each selected OWL property can be of three types: schema property, instance property, or restricted property. Using the semantic Arc Filter, the user can also visualize specific sub-properties and their associated types such as property restriction, schema property and instance property.

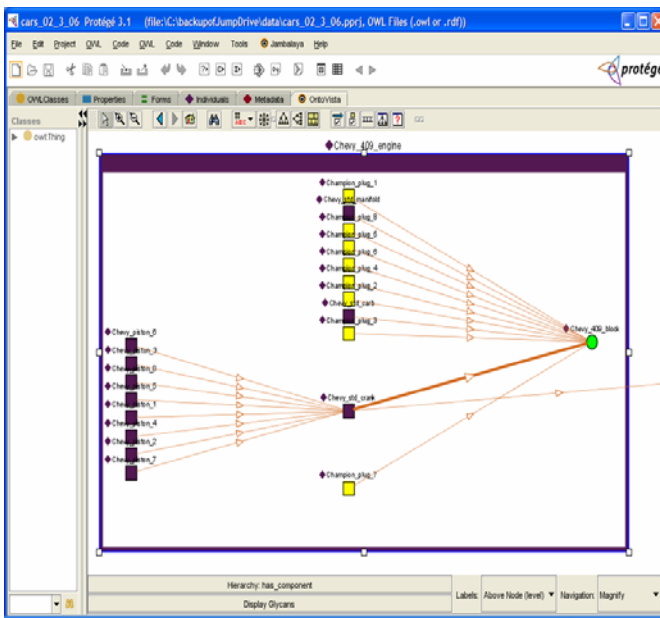


Figure 3. shows the image of an Car engine displayed according to the configuration settings.

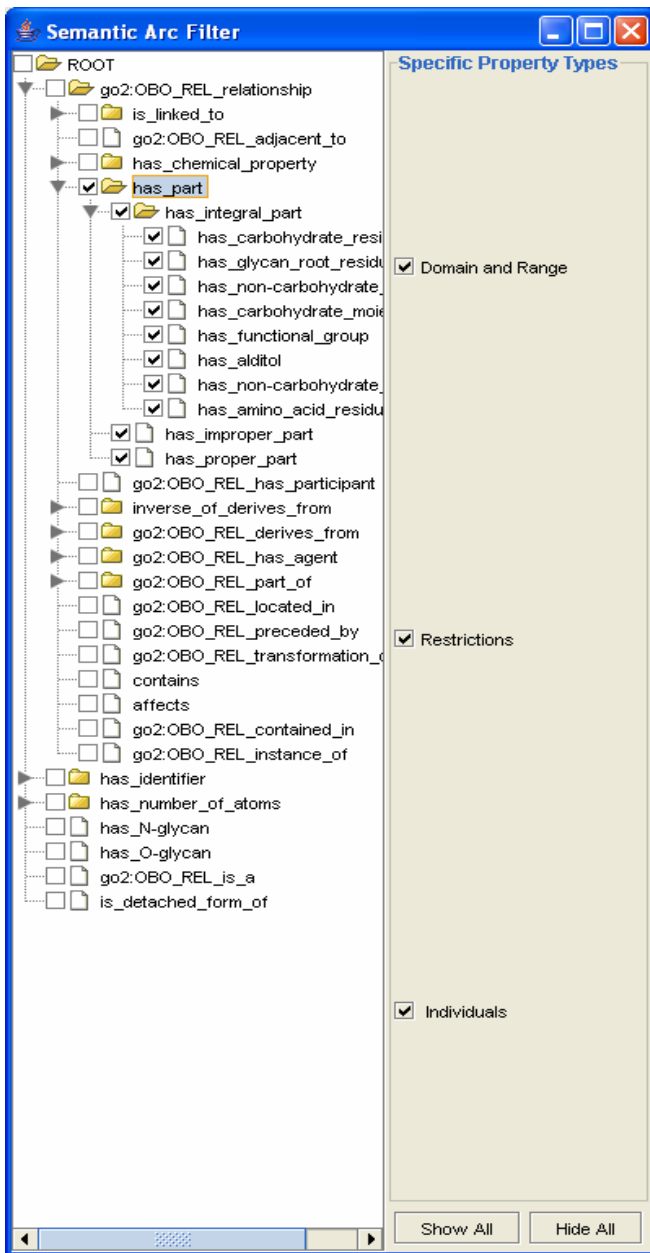


Figure 4. Semantic Arc Filter

Figure 4 shows the property hierarchy of the GlycO ontology using Semantic Arc Filter.

### 3.3. Ontology Browser

The ontology browser provides a faster and convenient way of accessing ontology information, such as classes, sub-classes, properties, sub properties, and instances. It supports dynamic navigation. Ontology Browser also helps in providing an ease of use search interface by allowing the users to select a set of classes, properties, or instance which can later be used to create queries in the search interface.

Initially a list of all classes in the ontology are displayed. Upon selection of a class, all its properties, instances, sub-classes, and super-classes are displayed. Left mouse click in the Ontology

Browser can be for dynamic navigation and the right mouse click for selection. All selections are done through a popup menu. The selected class, property, or an instance is added to the respective list fields in the search interface.

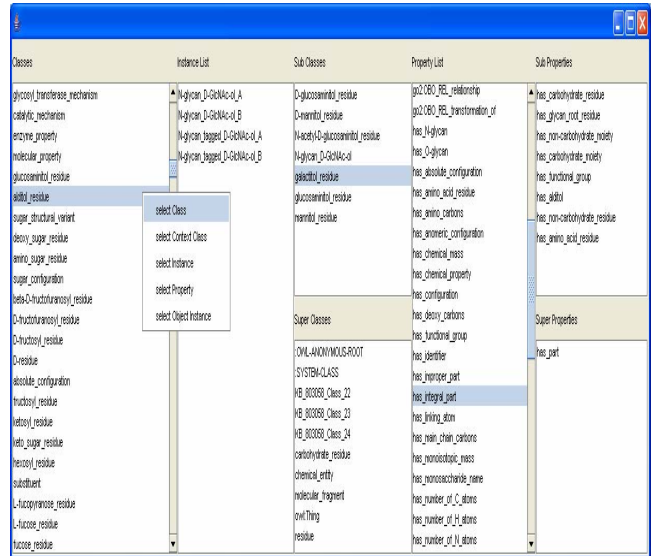


Figure 5. Ontology Browser

Figure 5, shows the ontological view of the GlycO ontology using OntoVista's Ontology Browser.

### 3.4. Searches

OntoVista provides advanced search capabilities such as class search, instance search, relationship search, RDF comment or description search, triple search, domain-range search and semantic search. By default, searches in OntoVista are pattern based searches. Search Interface also provides filter and un-filter option which can be used to add or remove the selected class/instance from the shrimp view or ontology view in Jamalaya.

OntoVista's search interface is closely linked to the Ontology Browser, which helps in providing an ease of use search interface that does not require users to type in names of classes, instances or properties.

OntoVista has a simple search interface based on a subject, predicate and object model. Figure 6 shows the view of the search interface in OntoVista. The search interface is divided into three columns. The first column allows the subject class or instance to be specified; the second column allows the predicate to be specified; the third column allows the object class or instance to be specified. The number of search fields that are filled in by the user depends on the type of search to be performed. For a class search, instance search, description search and relationship search, only one field is needed, and the classes and instances that match the pattern are returned. For a triple search, domain-range search and semantic search, two fields (one in each column) are used, and the search returns the classes, instances and properties, as appropriate, that complete RDF triples containing the two search fields. This search interface was designed for a biochemist who had no knowledge of ontology or the query language such as RDQL[17]. Thus, we gave prime importance to ease of use.

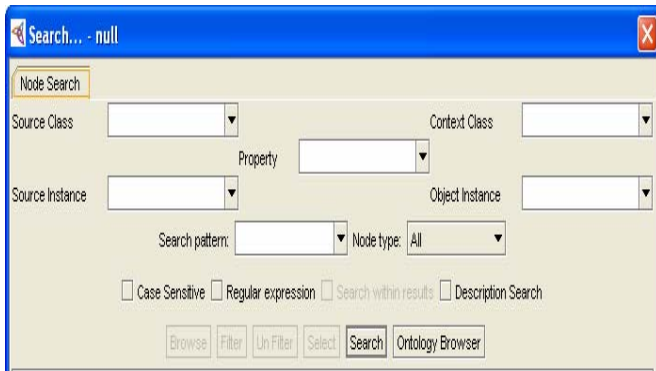


Figure 6. Search Interface

#### 3.4.1. Description search

Descriptions are comments in the RDF or OWL ontology. The user can enter the description text in the search pattern list box. Description search returns all the classes and instances that contain the entered description pattern. The user can also navigate to the specific class/instance by selecting the class/instance from the search result.

#### 3.4.2 Relationship search

Relationship searches can be used to find classes and instances that contain a given relationship. The users can either enter the relationship in the property list field of the search interface or select a property from the dropdown after adding the properties using the Ontology Browser. Relationship search returns all the classes and instance that define, restrict or instantiate the relationship. The user can then navigate to the specific class/instance by selecting the class/instance from the search result. An example in GlycO would be to find all glycans and its class types that has the relationship “*has\_carbohydrate\_residue*”.

#### 3.4.3. Triple Search

Triples are comprised of subject, predicate and object. Triple search can be used to search for either the subject or object instance, given a subject instance and a property or an object instance and a property. The user can either enter these query terms or select them from the list fields. The results of the triple search are object/subject instances and respective classes. An example of a triple search in GlycO would be to find all glycan instances that have the relationship “*has\_carbohydrate\_residue*” with residue instance “*N-glycan\_b-D-GlcpNAc\_14*”.

#### 3.4.4. Domain-Range Search

The output of a domain-range search is dependent on the input. Given a domain and a property, the search returns the range classes and their respective instances. Similarly while searching for a domain, given a range class and a property, it finds all the possible domain classes and their respective instances.

#### 3.4.5. Semantic Search

Semantic search can be used for analysis of the ontology. The user enters a subject class or instance and an object class to

determine whether some hypothesis about the knowledge in the ontology is true. For example, a user may want to know if the glycans in the GlycO ontology are comprised of residues. Semantic search on the above query would return all the glycans that have some relationship with residues

Semantic search can also be used to disambiguate instances and reduce the number of entries in the search result. For example, there could be multiple classifications of instances in the ontology. An ontology might have entries of different people with the same name, for example *Michael Jordan*. One MJ is an Entrepreneur and the other MJ is a Basketball player. Using semantic search the users can actually find the desired instance. Thus, by specifying the object class “*Entrepreneur*”, Semantic search will return *Michael Jordon* who is an Entrepreneur.

## 4. RELATED WORK

Visual BioMaze [4] is a tool to display complex biochemical networks from the data stored in relational databases. They use a notion of a *customizable representation model* that allows users to define and apply any representation model on the graph to be visualized. Their customizable representation model is similar to ours and it uses an XML based layout settings file as compared to our property based file. This tool however cannot be used to visualize ontologies. OntoVista’s searches are pattern based. OntoVista’s provides advanced search capabilities by means of triple search, domain-range search and semantic search. The search interface is easy to use compared to a tool that requires user to perform searches using RDQL query language.[17] Most visualization ontology visualization tools restrict searches to a class, property or an instance. OntoVista’s semantic arc filter is very unique feature that doesn’t exist in other tools.

Most tools visualize ontologies using the three most common techniques namely ClusterMap technique, nested view technique as in Jambalaya and graph based visualization. OntoVista uses graph based visualization combined with the clustering to generate domain specific views. OntoVista clusters on relationships to create containers or nested views of nodes. ClusterMap [1] technique visualizes ontologies through class hierarchy of relationships and clusters. Clusters are mainly formed by grouping of instances and hiding any relationships that connect instances. There are many applications developed using the ClusterMap technique. One such application is DOPE Browser [11] is a tool visualizing large document sets. It provides support for thesaurus-based search using Elsevier’s EMTREE thesaurus and makes extensive use of Cluster Maps for both visualizing and exploring query results.

Ontologies represented in RDF [9] or OWL [19] are often visualized as graphs. Most visualization tools allow users to navigate to different portions of the ontology and provide basic search capabilities such as instance or a class search. Protégé-2000 [22] developed at Stanford University, is a knowledge-modelling tool that helps users to build domain specific knowledge acquisition systems. It allows ontologies and knowledge-bases to be edited and browsed interactively. OntoViz, Jambalaya, TGVizTab, and OWLViz are some of the visualization tools developed as protégé plugins.

OntoViz [13] supports visualization of several disconnected graphs at once. The users can select a set of classes or instances to visualize. OntoViz generates graphs that are static and non-interactive which makes it less suitable for the visualization of large ontologies. Searches in OntoViz are restricted to classes.

Jambalaya [20], described in section 2.1, is another Protégé plugin, designed for visualization of large complex ontologies. OWLViz [15] is designed to be used as a Protege plugin to visualize the schema hierarchy, based only on the subclass relationship. In OWLViz, searches are restricted to classes.

TouchGraph ([www.touchgraph.com](http://www.touchgraph.com)) uses a spring-embedding algorithm to display the graph. Some users find TouchGraph difficult to use as it keeps re-adjusting the graph to create a layout that is best suitable for display. There are many applications that are built using TouchGraph. TGVizTab [10] is a protégé plugin that uses TouchGraph. It provides incremental graph navigation of ontologies. Using TGVizTab, users can search for classes or instances. OI-Modeler [12] is another tool specifically designed for creation and maintenance of ontologies. It uses TouchGraph for graphical display.

RDF Gravity [16] is a ontology visualization tool that visualizes ontologies in form of a graph. It provides a full text searches over concepts, properties and instances. RDF Gravity also provides RDQL query interface for advanced searches.

## 5. CONCLUSION

In this paper we have demonstrated how OntoVista can generate domain specific views familiar to domain experts. We have done these using two ontologies namely GlycO and Car ontology. Our goal in this research was to generate representations of domain ontologies that could be termed “semantic visualization” in the sense that the placement and containment of nodes and edges convey domain-dependent meaning. However, it was still mandatory to create a tool that is independent of particular domains. We are aware that the layouts presented here will not be powerful enough to semantically visualize every domain of interest, but we could show that for representations that are already inherently graph-based, a domain specific configuration can provide additional semantics. The second focal point in this work was improved filtering, search and navigation in ontologies. We have demonstrated how the hierarchical arc filter and the fast ontology browser together with the improved search interface contribute to this goal. Our future work will focus on the development of a visualization ontology that takes the part of the current configuration file and allows more detailed layout settings.

## 5. ACKNOWLEDGEMENTS

This work is part of the Integrated Technology Resource for Biomedical Glycomics (5 P41 RR18502-02), funded by the National Institutes of Health National Center for Research Resources. We would like to thank the CHISEL group of the University of Victoria for the Jambalaya code and Robert Lintern for providing additional assistance. Cory Hanson’s help in this work is especially acknowledged

## 6. REFERENCES

[1] Aduna Cluster Map Library version 2005.1 (Integration Guide), 2005.  
[2] David Goldberg, Mark Sutton-Smith, James Paulson, Anne Dell, *Automatic annotation of matrix-assisted laser desorption/ionization N-glycan spectra*, PROTEOMICS (2005) 5(4): 865-875  
[3] David J. Duke, Ken W. Brodli, David A. Duce, Ivan Herman. "Do You See What I Mean?," *IEEE Computer Graphics and Applications*, vol. 25, no. 3, pp. 6-9, May/June, 2005.

[4] E. Zimanyi and S. Skhiri dit Gabouje. Semantic Visualization of biochemical databases. In *Semantics for GRID Databases: Proc. of the Int. Conf. on Semantics for a Networked World*, LNCS 3226, Springer, 2004.  
[5] Eric Neumann, A Life Science Semantic Web: Are We There Yet?. *Sci. STKE* 2005, pe22 (2005).  
[6] Frank van Harmelen, Jeen Broekstra, Christiaan Fluit, Herko ter Horst, Arjohn Kampman, Jos van der Meer, Marta Sabou, *Ontology-based Information Visualisation. Workshop on Visualisation of the Semantic Web*, London, 2001.  
[7] Frank van Harmelen, Jeen Broekstra, Christiaan Fluit, Herko ter Horst, *Ontology-based Information Visualisation. In Geroimenko, V., ed.: Visualising the Semantic Web. Springer Verlag*, 2002.  
[8] Glyco-Ontology. <http://lsdis.cs.uga.edu/Projects/Glycomics>  
[9] Graham Klyne, Jeremy Carroll: *Resource Description Framework (RDF): Concepts and abstract syntax*. <http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/> (2004).  
[10] Harith Alani, TGVizTab: An Ontology Visualisation Extension for Protégé. In *Proceedings of Knowledge Capture (K-Cap'03), Workshop on Visualization Information in Knowledge Engineering*, Sanibel Island, Florida, USA.  
[11] Heiner Stuckenschmidt, [Frank van Harmelen](#), [Anita de Waard](#), [Tony Scerri](#), [Ravinder Bhogal](#), [Jan van Buel](#), [Ian Crowlesmith](#), [Christiaan Fluit](#), [Arjohn Kampman](#), [Jeen Broekstra](#), [Erik M. van Mulligen](#): Exploring Large Document Repositories with RDF Technology: The DOPE Project. *IEEE Intelligent Systems* 19, no. 3, 2004, pp. 34- 40..  
[12] [http://kaon.semanticweb.org/docus/Manual\\_KAON-OI-Modeler\\_November\\_2002.pdf](http://kaon.semanticweb.org/docus/Manual_KAON-OI-Modeler_November_2002.pdf)  
[13] <http://protege.stanford.edu/plugins/ontoviz/ontoviz.html>.  
[14] <http://web3.w3.org/TR/2005/WD-rdf-sparql-XMLres-20050801/>  
[15] <http://www.co-ode.org/downloads/owlviz/>  
[16] <http://semweb.salzburgresearch.at/apps/rdf-gravity/index.html>  
[17] <http://www.w3.org/Submission/RDQL/>  
[18] <http://doi.ieeeecomputersociety.org/10.1109/MCG.2005.55>  
[19] Ian Horrocks, Peter Patel-Schneider, Frank van Harmelen. From SHIQ and RDF to OWL: The Making of a Web Ontology Language. *Journal of Web Semantics*, 1(1), 2003.  
[20] Margaret-Anne Storey, M. A. Musen, J. Silva, C. Best, N. Ernst, R. Ferguson, and N. F. Noy, “Jambalaya: Interactive visualization to enhance ontology authoring and knowledge acquisition in protege,” in *Workshop on Interactive Tools for Knowledge Capture, K-CAP-2001*, Victoria, B.C. Canada, 2001.  
[21] Margaret-Anne Storey. "SHriMP Views: An Interactive Environment for Exploring Java Programs," *Proceedings of the International Conference on Software Engineering: Workshop on Software Visualization*, Toronto, 13-14 May 2001.  
[22] Natalya Fridman Noy, Michael Sintek, Stefan Decker, Monica Crubézy, Ray W. Ferguson, and Mark A. Musen, Creating Semantic Web contents with Protégé-2000. *IEEE Intelligent Systems*, 16(2):60–71, 2001.  
[23] Pim N. Borst. Construction of Engineering Ontologies. PhD thesis, University of Twente, Enschede, 1997.

[