OZONE: A Zoomable Interface for Navigating Ontology Information

Bongwon Suh and Benjamin B. Bederson

Human Computer Interaction Laboratory

Computer Science Department

University of Maryland at College Park

College Park, Maryland 20742

{sbw, bederson}@cs.umd.edu

+1-301-864-7345

Abstract

We present OZONE (Zoomable Ontology Navigator), for searching and browsing ontological information. OZONE visualizes query conditions and provides interactive, guided browsing for DAML (DARPA Agent Markup Language) ontologies. To visually represent objects in DAML, we define a visual model for its classes, properties and relationships between them. Properties can be expanded into classes for query refinement. The visual query can be formulated incrementally as users explore class and property structures interactively. Zoomable interface techniques are employed for effective navigation and usability.

Keywords: Ontology, DAML, Browsing, Zoomable User Interface (ZUI), Jazz, WWW.

Introduction

Information on the World Wide Web (WWW) has expanded enormously during the last several years. Searching the Web by string matching and link analysis has been one of the most popular ways of finding information in the vast quantity of data. But since these approaches rely on syntactic information, the search result of such schemes is often limited and ineffective. The Web has been designed for direct human processing. It is humans that write most of the web pages. Therefore, for accurate knowledge extraction, it is crucial to identify embedded semantic knowledge in them.

RDF (Resource Description Framework) and RDF schema are W3C recommendations to add metadata in order to turn the WWW into a machine-readable knowledgebase [3][11][17]. RDF offers a distinguished vocabulary to model classes, properties, and other basic schema primitives that can be referred to from its model. This model also can be extended to address sophisticated ontology representation techniques. An ontology is defined as shared formal conceptualization of a particular domain [6]. Practically, it can be regarded as a vocabulary and its definitions as well as relationships between concepts in it. Ontologies specify what concepts to represent and how they are related. They can be used to convey semantic information through machine-based communication and their description can be reused between similar domains. Using semantic information, the WWW will enable intelligent services such as information brokers, search agents and information filters which offer greater functionality and interoperability than current stand-alone services [6].

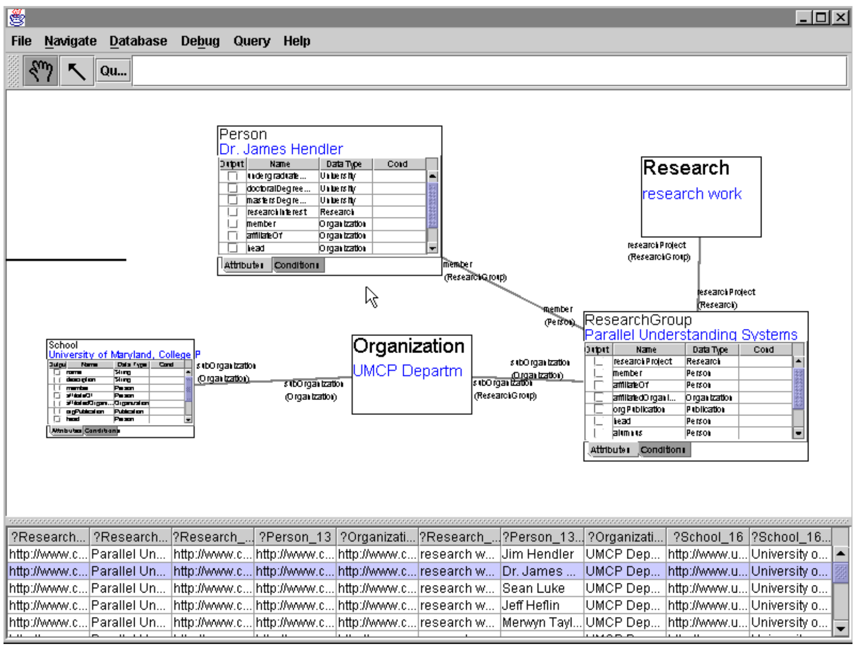

Figure 1 OZONE Overview

Nodes and links represent query conditions visually. Queries can be formulated interactively and incrementally by manipulating objects on the screen. During query formulation, a user can check the intermediate results, which are displayed at the bottom of the screen. When a result row is selected, each entry in the table is remapped into a corresponding visual node and shown under its title as a blue label.

DAML (DARPA Agent Markup Language) is one of the newly emerging standards for the metadata framework [5]. DAML is based on RDF and extends it to facilitate agent-based computing. DAML allows communities to extend simple ontologies for their own use and also provides mechanisms for the explicit representation of semantic knowledge to augment web pages. Based on the RDF framework, every DAML individual is represented as a simple triple of subject, predicate and object. It might be easy for machines to interpret but, for humans, a more abstract model is needed.

An Example Query and Motivation

A typical query involves relations between multiple classes where some details are known and some aren't. As an example, assume that a user wants to find people working on a specific project. Further the user knows that the people who he/she is looking for work in a particular research group at the University of Maryland. But the user does not know about the organization of research groups.

In the case that a user does not know much about the ontology structure, it is very difficult to form a valid query. The user has to know the names and semantic meanings of classes, properties and their relationships precisely. Furthermore, ontology structures are not well-formed, which confuses the user. For example, a member of a research group can be a faculty or a student and each of them has a different set of applicable operations. This aspect hinders efficient query formulation.

To clearly demonstrate the issues surrounding this example, we define a specific ontology that we will use throughout this paper. The example ontology includes five classes and their properties as shown in Table 1.

Class Node	Property	Type of Property
Person	member (is a member of)	Organization
ResearchGroup	researchProject (has a research project of)	Research
	member (has a member of)	Person
	subOrganization (have a sub-organization of)	Organization
Organization	subOrganization (have a sub-organization of)	Organization
	subOrganization (is a sub-organization of)	Organization
	member (has a member of)	Person
Research	researchProject (is a research project of)	ResearchGroup
School	subOrganization (has a sub organization of)	Organization

Table 1 Example Ontology

Now we can formally state the query first introduced using this ontology. The text-based RDF query languages such as Squish [14], RDFDB QL [8] are proposed to form complex queries efficiently. The following is an example Squish statement to get an answer for the previous question.

SELECT ?rg, ?r, ?p, ?o, ?s

WHERE (rdf::type ?rg daml::ResearchGroup), (rdf::type ?s daml::School),

(daml::researchProject ?rg ?r), (daml::member ?rg ?p),

(daml::subOrganization ?rg ?o), (daml::subOrganization ?o ?s),

(daml::name ?s "University of Maryland")

USING rdf FOR http://www.w3.org/1999/02/22-rdf-syntax-ns#

daml FOR http:/foo.com/abc.daml#

This approach is effective when query composers know every detail of the ontology, which is not typical for casual users. Users have to precisely specify classes and properties. The data types of variables in the logical expressions can confuse users. As seen on the previous expression, the types of variables are implicitly set. As an instance, the type of *?o* in *(daml::subOrganization ?rg ?o)* is given as the "Organization" implicitly because the "subOrganization" property is defined to have "Organization" type. But in the following condition *(daml::subOrganization ?o ?s)* of the above example, *?s* is a "School" type variable as defined explicitly in the query statement.

Our goal is to make a more intuitive interface that allows users to access complex ontology information without knowledge of their structure. First, an easy-to-use and intuitive interface, especially for casual users, should be provided. Since semantic information is supposed to be added into web pages and to be used by users who are not acquainted with them, a query interface should not require much time to learn it. Second, logical expressions should be supported naturally. Instances of classes are linked with logical relationships as shown above. Logical relationships as well as inheritance hierarchies in ontology structures can confuse users. Third, context information should be provided for users. For a class, context information is composed of properties that can be applied to the class, its super classes, and its sub classes. Showing this information to users allow them to compose queries interactively and to avoid errors since it is well known that people can recognize information faster and more accurately than when they recall it [16]. Fourth, fast feedback should be supported in order to form queries incrementally. Query composers can be sure that they are on the right track by checking intermediate results. Furthermore, the intermediate results also can be used as input conditions. For example, literal values in an intermediate result can be fed back into the query condition to restrict the result set. Knowing data characteristics is beneficial for getting the right information.

In this paper, we present OZONE (Zoomable Ontology Navigator), a visual interface for the exploration of semantic information that is defined in DAML. OZONE reads ontology information and rearranges it visually with context information so that ontology information can be queried and browsed easily and effectively. OZONE is implemented using the Parka [18] knowledgebase, the Jazz zoomable interface toolkit [2], the RDF parser from Pro Solutions [15], and the API for XML Processing (JAXP) XML parser [10] from Sun Microsystems. Figure 1 shows a snapshot of the query formulation of the previous example.

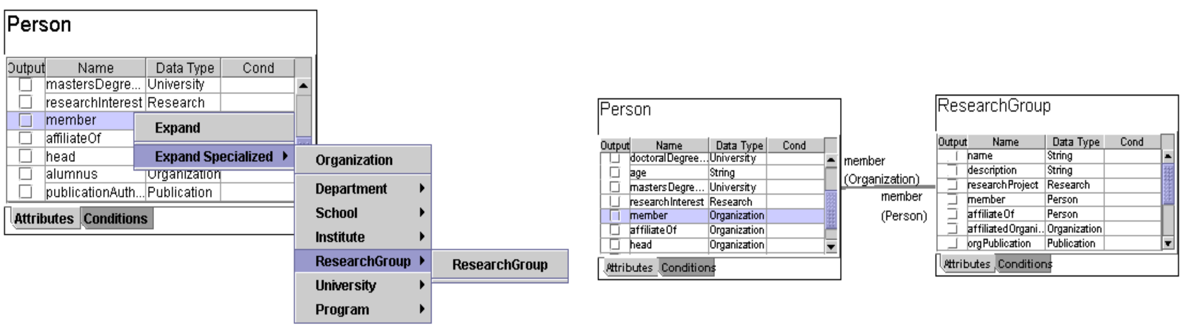

Figure 2 Property expansion: Selecting a property from a visual class node

In the left figure, clicking the "member" property shows the menu of data types that the "member" property can have. As shown, a person can be a member of "Organization" or more specifically a member of "Department" and so on. In this example, the "ResearchGroup" is selected. The right figure shows the result of this selection. The "ReseachGroup" node is newly created and linked with the "Person" node by the "member" relationship.

We will now walk through the query we started this paper with. Since the user wants information about people, he/she begins to form a query by selecting the "Person" class from a class list that contains all classes of the ontology. This action puts the "Person" class on the display. Since the goal of the query is to find information about people in a particular research group, the user scans the properties of the "Person" to find a property that relates a person with an organization. The user clicks the "member" property of the visual node because he/she finds that it is the most appropriate property to specify "is a member of" relationship. When the user clicks, a pop-up menu appears as shown in Figure 2. Properties are typed in the ontology definition. And when the user selects a property, he/she can choose its type. A class node of the chosen type is generated on the screen and linked with the current class as result of this selection, which is called as the "property expansion".

As a data type of the “member” property, the user can select the “Organization” for the expansion. But, as assumed, the user knows that a person who he/she is looking for is in a particular research group. So he/she decides to choose a more specific class. By following the guided hierarchy, he/she can find that the “ResearchGroup” menu is a subclass of the “Organization” and the user chooses the “ResearchGroup” as an organization that people that he/she wants to find work at. As a result of it, the new node “ResearchGroup” and a link between them are drawn on the screen. The labels of the link show that these two nodes have the “member” relationship as depicted in Figure 2.

Since it is obvious that a research group has research topics, the user can find “researchProject” among its properties. As a result of selecting the “researchProject” property in the “ResearchGroup” node, the “Research” node is generated on the screen and linked with the “ResearchGroup” node. After a simple typing and menu selection, the user will get objects on the screen as Figure 3.

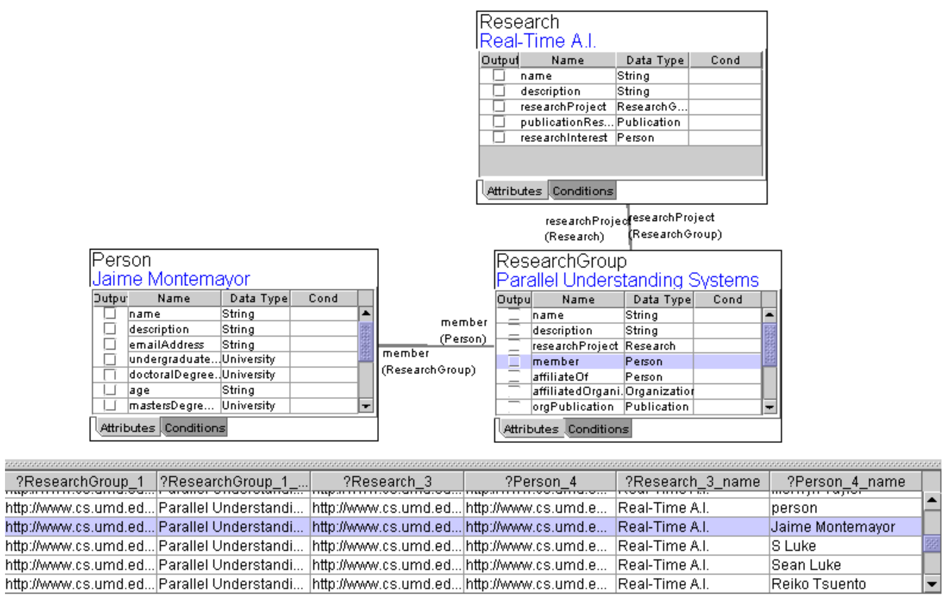

Figure 3 Intermediate results

A user can execute an intermediate query with objects on the screen. This allows users to form queries interactively and incrementally. Double clicking on the table entries launches a web browser and shows the web page of the URL

The intermediate result of the query formed so far can be obtained any time by pressing the “Query” button in the tool bar. As shown in Figure 3, the intermediate result is shown at the bottom of the screen. But the intermediate query is so premature that the result of it usually includes more information than the user wanted. This intermediate query will return all information about any person who is in a research group that has any project.

To further restrict the intermediate query, the user chooses to specify the school that the research group is in. This procedure begins with expanding the “subOrganization” of the “ResearchGroup”. As a result of it, an “Organization” node is drawn on the screen as shown in Figure 4. This expansion adds a new condition to the query so that the research group should be under a specific organization. But there is no guarantee that this organization represents schools. The user can execute the query again at this moment to figure out the characteristics of the organization values. By checking the intermediate result, the user realizes that the organization node represents departments rather than schools. To specify the school, the user expands the “subOrganization” property of this “Organization”. In this case, “subOrganization” is expanded into “School” as depicted in Figure 4.

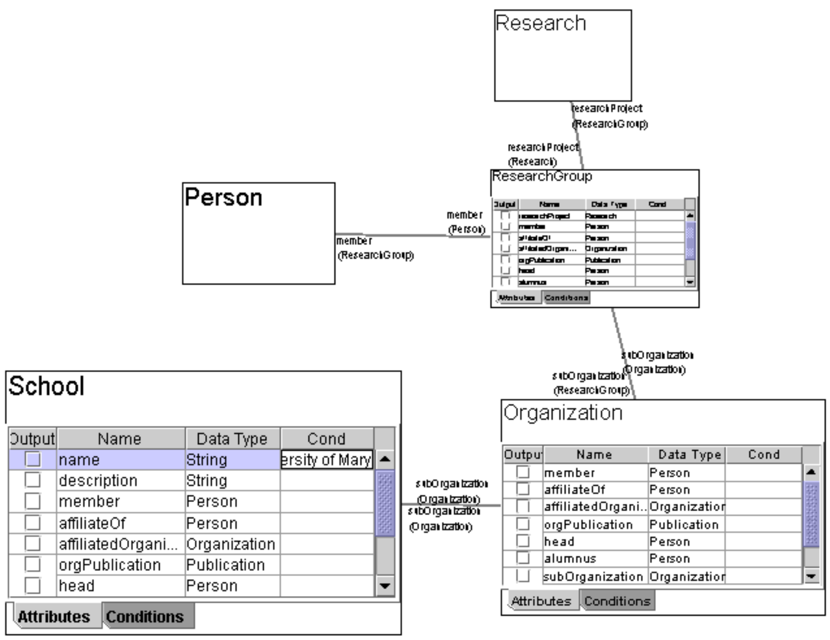

Figure 4 Specifying the text value for a property

A user can specify values of properties by typing text strings inside visual nodes. For example, a school name is entered as a property value of the “name” property of the “School” class. The Person and Research node have been zoomed out to hide unnecessary detil.

Finally the school name that the user wants to specify can be typed into the “name” property of the “School” node. In this example, the user wanted to know people at the University of Maryland. Therefore the corresponding string is entered as the value of the name property as shown in Figure 4. Whenever users know detail information of classes, they can type it into its property directly. As a result of the above steps, a visual query, which is equivalent with Figure 1, is formulated. This query answers the first question in a table format.

As shown in the above examples, the advantages of querying ontology information by OZONE are as follows:

First, OZONE enables novices and casual users who are not acquainted with ontologies to access and search information. During the query formulation, menus of candidate properties guide users to specify relationships while browsing ontologies rather than look up or search them. This feature supports interactive and incremental query formulation, which is essential for casual users.

Second, even for experts, it is not easy to remember all the details of ontology definitions. Users are provided with context information such as property lists, class hierarchies, and intermediate results so that the users can avoid errors and understand ontologies better. OZONE could reduce time to learn ontology structures and save time to get information from a knowledgebase.

Third, OZONE can handle complex queries in which multiple classes are interrelated. Not only does OZONE support complex queries, it also can abstract a complicated query, which is denoted by a graph with multiple nodes, into a single node. Those queries can be saved, loaded, and shared. This feature allows users to specify complex conditions easily and precisely.

Fourth, OZONE uses a zoomable interface which allows for effective screen usage and information hiding. As described in a later section, each node can be moved or scaled, and smaller nodes are depicted with appropriately less-detailed information.

Related Work

Our work is inspired by the SHOE (Simple HTML Ontology Extension) project [12]. In the SHOE framework, web pages are marked up with semantic information, which are gathered into a knowledgebase by a web-crawler. Users can query the semantic information by specifying search conditions [9]. But the SHOE search tool limits its search within one class and it is hard to specify query conditions on multiple nodes at the same time. For example, a query such as “find people whose research group is in a specific department” cannot be specified.

As exemplified before, text based query languages such as Squish [14] and RDFDB QL [8] have disadvantages for novices and casual users. It is hard to receive fast feedback in those systems. There are some graphical approaches. OntoBroker [7] includes a query interface and a hyperbolic viewer for exploring ontologies but searching with complicated conditions is limited. Chimaera [13] is an ontology environment that helps users to edit, merge and diagnose ontologies but the query specification is limited.

Parka [18] is a knowledgebase system that is scalable, efficient and can be used on parallel machine. Parka provides a good tradeoff between query efficiency and common types of inferences. The information stored in Parka originated from the SHOE project. The data set of the WebKB project [4], which consists of 8,282 real web pages found at computer science department websites, has been transformed into SHOE formats and stored in Parka as well as some manually generated classifications. Since a SHOE ontology can be transformed easily into a DAML ontology, we use this data set for testing OZONE. All figures in this paper are created from that data set.

Data Model and Browsing

In this section, we define the visual model of classes and properties of the DAML ontology, followed by the description of the OZONE user interface.

Classes and Properties

In the DAML framework, an ontology consists of collections of RDF triples, *<subject, predicate, object>*. Classes and properties are defined as subsets of the RDF objects as follows:

C is a class if the RDF triple collection has a syntactic structure,

<rdf:type, ?C, rdfs:Class>

or <rdf:type, ?C, rdfs:Restriction>

P is a property if the collection has a syntactic structure

<rdf:type, ?R, rdfs:Property>

Practically, classes may be thought of as a vocabulary that represents semantic meanings for a set of objects, or object groups that share common attributes. For example, the “Person” is a class in our running example. Properties may be regarded as attributes of classes and in this sense correspond to traditional attribute-value pairs. For instance, the “member” is the property of the “Person” class.

The actual DAML definitions of classes and properties are shown in Table 2 with the explanation of their semantic meanings.

```
<Class ID=”Organization”>
        <label>organization</label>
        <subClassOf resource = “#SocialGroup”>
</Class>

<Property ID=”member”>
        <domain resource = “#Organization”>
        <range resource = “#Person”>
</Property>

<Property ID=”researchProject”>
        <domain resource = “#ReseachGroup”>
        <range resource = “#Research”>
</Property>

<Property ID=”subOrganization”>
        <label>has a member of</label>
        <domain resource = “#Organization”>
        <range resource = “#Organization”>
</Property>

<Property ID=”researchProject”>
        <domain resource = “#ReseachGroup”>
        <range resource = “#Research”>
</Property>

<Class ID=”ReseachGroup”>
        <subClassOf resource = “#Organization”>
</Class>

<Property ID=”emailAddress”>
        <label>receives e-mail at </label>
        <domain resource = “#Person”>
</Property>
```

defined in http://foo.com/abc.daml

Organization	Class Defined as a subclass of SocialGroup Key: http://foo.com/abc.daml#Organization or Organization
subOrganization	Property Instances must be <*O1, O2*>, where $O1, O2 \in Organization$ Expandable property and Inverse-expandable property of “Organization”
member	Property Instances must be <*O, P*>, where $O \in Organization, P \in Person$ Expandable property of “Organization” Inverse-expandable property of “Person”
ResearchGroup	Class: Defined as a subclass of Organization Key: http://foo.com/abc.daml#ReseachGroup or ReseachGroup
researchProject	Property Instances must be <*RG, R*>, where RG $\in ReseachGroup, R \in Reseach$ Expandable property of “ReseachGroup” Inverse-expandable property of “Research”
emailAddress	Property Instances must be <*P, Literal*>, where $P \in Person$, *Literal* is a string Expandable property of “Person”

Table 2. Example classes and properties

Class hierarchy is an important structure in ontologies. Classes in DAML can be defined as subclasses of existing classes. This allows the DAML framework to extend simple ontologies into the explicit representation. In Table 2, the “ResearchGroup” is defined as a subclass of the “Organization” and has more specific semantics. Every instance of “ReseachGroup” is also an instance of its super-class “Organization” because of its inheritance hierarchy. This aspect allows properties to have multiple types. Suppose that the “Student” class was defined as a sub-class of the “Person” class, the “member” property of the “Organization” could be applied to “Student” objects because the “Student” objects also can be regarded as “Person” objects.

In the DAML framework, the context information in an ontology is composed of a vocabulary, relationships and hierarchies of the classes. The vocabulary definition includes the names of clsasses and properties in the ontology. Those relationships between classes are represented by properties. Such information plays an important role in query formulation. With this information, users can specify query conditions interactively and incrementally by navigating the ontology structure. Having rapid and consistent feedback is also essential for effective browsing. To satisfy these requirements, we developed a visual model for DAML classes and properties, which will be explained in the next section.

Visual Model

In OZONE, a class node is defined as an aggregation of its key and properties. The key of a class node is obtained from the DAML ontology directly. But the properties of a class should be collected from DAML definitions that are scattered over ontologies. The properties in the DAML ontology are defined as sets of object pairs as shown in Table 2. For an example, every instance of the “member” property must be <*organization, person*> pair by the definition of the DAML property. If either of its values is known, the other value can be found because of its duality. In other words, for “Organization” objects, “Person” objects can be obtained from the property and vice versa. We define the type of properties in a class node using this characteristic. In the “Organization” class, the “member” is defined as a “Person” type property while it is an “Organization” type property in the “Person” class.

As stated above, properties can have class types and, therefore, classes can be induced from those typed properties. For example, the “Person” class can be induced from the “member” property of the “Organization” class. This procedure is called “*property expansion*” because a class is expanded from a property. Figure 2 shows the expansion of the “ResearchGroup” from the “member” property of the “Person” class. In Figure 5, the “Research” is used to illustrate multiple property expansions.

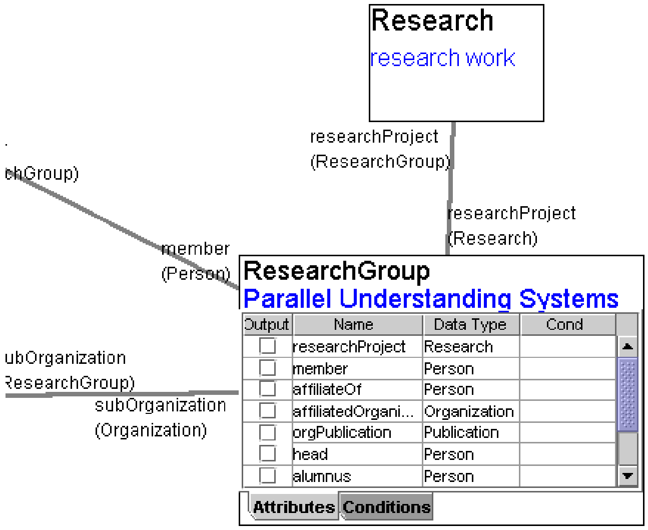

Figure 5 ResearchGroup Node

The properties of the “ResearchGroup” class are listed in a table. Three classes have been expanded from the properties of the “ResearchGroup” node. The “Research” node is expanded from the “researchProject” property and the link has been made with the labels to show the relationship. Since the “Research” node has been zoomed out, only its title is shown.

OZONE provides a list of properties for each class. It helps users to understand the meaning of classes and also provides intuitive and effective feedbacks. The class visual node is displayed in zoomable space so that it can be moved, zoomed in, zoomed out freely.

The properties for a class can be collected from RDF triples by the following rules.

Property p becomes a property of class c if one of the followings is satisfied

i) $c \in domain(p), \exists c' \in C$ such that $c' \in range(p)$ → *Expandable property*

ii) $c \in range(p), \exists c' \in C$ such that $c' \in domain(p)$ → *Inverse-expandable property*

iii) $c \in domain(p), \forall l \in range(p), l \in L$ → *Literal property (non-expandable)*

where C is a set of DAML classes, P is a set of DAML properties and L is a set of literals.

The properties are categorized into three types as stated above. The first property type is expandable so that user can expand this type of property into another class *c’* as defined above. The inverse-expandable property behaves the same way except it expands the domain class of the selected property rather than the range class as with the regular expandable properties. The last type is a non-expandable property and the objects of this type are not allowed to have further manipulation except having literal values for string matching. For example, “emailAdress” property in Table 2 has no range definition and accordingly only string input is accepted as a valid query operation.

OZONE parses DAML ontologies and retains the information inside until the end of its session. To be scalable, internal ontology representations require more efficient management, which is not implemented yet.

Links and Browsing

Relationships in DAML are modeled as links in OZONE and are used to facilitate query refinement. A link is created as a result of the ‘property expansion’ as mentioned before. A link can be specified between two classes when a property is an expandable property of one class and inverse-expandable property of the other at the same time. For example, the “member” property in Table 2 suggests that a link can exist between “Organization” and “Person” class. Links are labeled with appropriate property and class information as shown in Figure 5. The property expansion of the “researchProject” property in the “ResearchGroup” draws a “Research” node on the screen and makes a link between the current “ResearchGroup” node and the newly drawn “Research” node.

Expanding a property that originated from property *p* in class *c* implies adding query conditions to class *c* as follows:

$$E(c, p) = \{ \langle o \rangle \mid o \text{ is an instance of } c,\ \exists\, o' \text{ is an instance of } c' \text{ such that } \langle o, o' \rangle \in p \}$$

The expansion of the inverse-expandable property is basically identical except that the range of the property is drawn instead of its domain.

Suppose that there is a visual node drawn on the screen without any conditions. Since there is no condition, every instance matching the class will be returned as a result, yielding a large result set. For example, the “ResearchGroup” node without any link would return all research groups in a knowledgebase as its result. Property expansion can be used to narrow the query result by adding conditions to a query. For example, expanding the ‘researchProject’ property filters out instances of the “ResearchGroup” that do not have any ‘Research’. Instances of ‘ResearchGroup’ that have relationships with ‘Research’ instances are included in the result set. Formally, the following set is the result of this property expansion.

$$Result\ Set1 = \{< r, rg> \mid r \in Research \wedge rg \in ResearchGroup \wedge <rg, r> \in researchProject\ \}$$

The property expansion can be applied multiple times as a query becomes complicated. By adding more conditions on the 'ResearchGroup' class, the result becomes more specific. For example, expanding the "member" property of that node adds another condition as follows.

$$Result\ Set2 = \{<r,rg,p> \mid r \in Research \wedge rg \in ResearchGroup \wedge p \in Person \wedge <rg, r> \in researchProject \wedge <rg, p> \in member\ \}$$

In addition to property expansion, specifying a string value for a literal property is another way of adding a query condition. Literal value *l* for property *p* on class *c* is interpreted by adding a query condition to class *c* as follows.

$$L(c, p, l) = \{ <o> \mid o \text{ is an instance of } c, < o, l> \in p\ \}$$
$$\text{, where } p \text{ is a literal property and } l \in L \text{ (a set of Literals)}$$

As an instance, specifying the "name" of the "School" class as "University of Maryland" restricts instances of the "School" as follows:

$$Result\ Set3 = \{<s> \mid s \in School \wedge <s, \text{“University of Maryland”}> \in name\ \}$$

Currently, string matching is the only way to check equality. However OZONE will be improved to handle various types such as number, date, currency and so on. Comparison operator (<, >, !=) and partial string matching will be added as well.

Multiple query conditions are connected conjunctively to produce the result. For example, the above *Result Set2* is obtained by connecting conditions in *Result Set1* conjunctively with new conditions such as $p \in Person \wedge <rg, p> \in member$. Because of this strategy, OZONE is not able to specify disjunctive conditions. But browsing is a task that concentrates more on finding information than on recalling the whole information. We believe that this model is appropriate for discovering new knowledge.

Browsing begins with searching a class from the class list or the class hierarchy tree. Once a class is selected, it is put on the screen as a visual node. Now query conditions can be added to that node either by inputting a string or by property expansion. During this procedure, a user can execute the query whenever he/she wants to see the intermediate results. This procedure is shown in Figures 2, 3, and 4 and explained in previous sections. After the intermediate query is executed, its results are shown at the bottom of the screen in a table. Retrieved class information in the table is remapped into class nodes when a row in the table is selected. Key values of classes are shown as blue labels under their title. Double clicking each entry will launch a web browser with its key value that is a URL. Figure 1 shows an example of the complete query.

Specialization/Generalization

DAML ontologies have the inheritance hierarchy of classes in it. So we can take advantage of the hierarchy structure when querying. If a searching result is too big, OZONE can narrow the result by specialization. For example, searching for ‘Organization’ can be specialized into ‘School’. It will exclude instances such as ‘Company’, ‘Government’, and so on. Generalization is the opposite concept to specialization. When general results are needed, the query condition is loosened up by generalization. For example, ‘Student’ can be generalized into ‘Person’. The query result would include instance such as “Faculty”, “Staff”, and so on as well as “Student”.

These operations are accessed with two mechanisms in OZONE. First, they can be applied by property expansion. The subclass of an expandable property type can be expanded instead of the original property type. As in Figure 2, the “ReseachGroup” class is expanded instead of its superclass “Organization” from the “Person” class. Secondly, generalization and specialization can be applied to a visual node itself. By adjusting the level of coverage, users can narrow or widen the result. Clicking the header of a class node will show popup menu that contains the list of super classes and subclasses of the class.

The hierarchy structure in ontologies is pre-defined by the ontology designers who are experts in their domain and ontologies. Since the generalization and specialization are guided by ontologies, more accurate querying is possible than with non-guided searching.

Query Abstraction by Grouping

In OZONE, any sub-graph can be grouped and transformed into a single node by choosing the ‘Group’ menu in the main menu after selecting nodes on the screen. The collection of nodes is zoomed out and a simple new node replaces the collection. Users can access the detailed sub nodes at any time by zooming in.

This feature offers a general way of abstracting queries. High-level queries can be formulated in simpler forms by abstracting low-level details. A group node is also useful when importing queries. Even if users do not know the structure inside a group, they can use the group node the same way as they use a regular class node. For example, a query such “find people whose research group is in a specific department” can be abstracted into a single node “Member of HCIL at Univ. of Maryland”. And this node can be used as the same way as the “Person”. Users can add more conditions on this node either by property expansion or by string value.

The grouping also helps to reduce the number of objects on screen. In Figure 6, five individual visual nodes are abstracted into a single node.

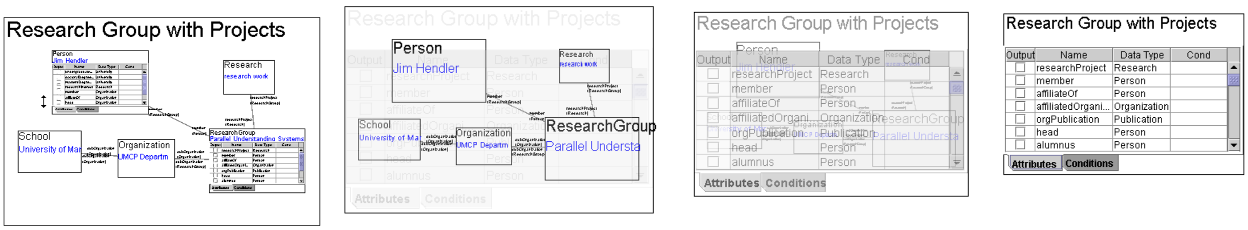

Figure 6 Grouping nodes

Five classes and links between them are grouped into a group node ‘Research Group with Projects’. This group node can be treated as the same way as a regular class node. When a group node is zoomed in, the details begin to appear while the group node is fading out. The transition is animated according to the zoom level. The above sequence demonstrates this transition.

To be able to edit query conditions on a group node, properties of group members need to appear on the group node. Since multiple nodes are subordinated by one group node, name collisions can happen between properties from different nodes. In addition, it is not the best approach to list all properties of child classes on the parent group node. This problem is aggravated when groups are cascaded. To avoid these problems, users need to choose which properties are shown on a group node.

User Interface

OZONE provides basic editing functions such as deleting, moving, centering, zooming in, zooming out, grouping, saving, and loading. Since all the screen objects are built with Java Swing widgets, users can edit and navigate query graphs using standard GUI operations. Individual nodes and group nodes can be zoomed in and out independently without restriction according to users’ preferences. This makes it possible to avoid the screen being crowded by too many objects. In addition the entire view can be zoomed in or out incrementally. All zooms and transitions are animated so users can maintain a sense of object constancy.

Detail view At any scale, double clicking on a class node zooms it into its original size and moves it to the center of the screen. Inputting searching conditions and browsing by property expansion also can be performed in this view. During the transition, zooming is animated and guidelines appear to avoid confusion as to which node is zoomed in. Clicking outside of the selected object cancels the detail view.

Semantic Zooming Another advantage of using a zoomable interface is to allow objects to have multiple representations, which is called semantic zooming [1] . When an object is zoomed out and becomes small, contents inside the object, such as text, become illegible and less useful. In OZONE, the appearances of class nodes and labels on links are adapted to the scale so that a reasonable amount of information is shown to users. When a node is zoomed out, the property information disappears and its title dominates the node. But when a node is zoomed in, all the details reappear. The same strategy is used for grouping. When a group node is small, it is drawn as a regular class node and there is no difference between the group node and regular class nodes. But when it's zoomed in, all subordinating classes and relationships reappear so that the user can manipulate the details. Figure 5 and 6 shows examples of the semantic zooming.

Future Work

Since OZONE is in its early stage, there is much room for improvement. The major improvements will be to merge searching and browsing with authoring. Adding new ontology information is like solving a jigsaw puzzle because semantic data should coexist with its context information that is scattered over the Web. It is crucial to identify relationships between ontologies and their classes before creating a new instance. It would save effort if users could author a new piece of information by typing new information while the context information is shown on the same display. Currently, the visual nodes of OZONE are limited to specifying query condition. Changing them to accept new data would be a crucial improvement for OZONE.

Another desirable improvement would be automatic suggestion when searching a class. Since ontologies are defined by ontology designers, keywords specified by casual users do not necessarily match the class names defined in ontology vocabularies. For example, 'faculty' may be included in the ontology instead of 'teacher'. In this case, users might fail to get the right result when they try to find information using the keyword 'teacher'. By comparing users' searching keywords with words in WordNet [19] and suggesting the closest classes in ontology, this kind of confusion can be avoided. This approach is likely to be successful because it is expected that ontology vocabularies will be defined as common English words so that their structure will be similar to WordNet hierarchy. We will explore this as an alternative way of searching information in ontologies. While the above approach focuses on semantic aspect of words, automatic filling-out of ontologies will provide for syntactical solution for searching. According to the user's initial input string, the rest of the class name will be suggested and filled out.

Regarding implementation, scalability can be an issue for OZONE. Since it retains all ontology information, dealing with huge ontologies will cause a significant slow down. Supporting general relational databases other than Parka will also be advantageous for practical reasons.

Finally, we have only started our work in using OZONE with real users. We plan on performing usability studies with representative users doing realistic tasks and refining the interface based on their feedback.

Conclusion

In summary, OZONE is an interface for navigating ontology-based knowledgebases. It visualizes classes and properties of DAML to facilitate efficient query formulation. The DAML ontology framework provides semantic layers on web pages so that search agents can find information accurately and effectively. But searching information in an ontology data set is not easy because it does not have well formed structure and logical expressions should be used to represent relationships between individuals in ontologies.

OZONE illustrates an ontology query that is unlike the query interfaces that requires users to remember detailed ontology information. Classes are provided with property information so that users can specify relationships with other classes by clicking on a property in a visual class. Once a class is selected, users can browse classes to refine query conditions by expanding properties. During a query formulation, users can check their intermediate results. Consequently, this strategy allows users to find what they want interactively and incrementally. Casual users can search ontology information without learning the ontology structure deeply. For an expert, more efficient query formulation is possible by avoiding errors.

Specialization and generalization is another way of query refinements and is also supported by menus. Class hierarchies in ontologies are provided to users during query formulation to narrow down or broaden query results. In OZONE, query conditions can be abstracted. Multiple class nodes are shrunk into a single node for efficient screen usage. It also provides high-level views for complex queries.

Our experience with designing and developing OZONE for navigating ontology information has highlighted a number of new issues. First, authoring ontology information is a main bottleneck for facilitating semantic services. Second, browsing can conflict with searching. It is hard to form complex queries by browsing while it is not easy to provide users with available options when searching. Allowing both functionalities without sacrificing usability needs to be researched further. Finally, an effective user interface dealing with ontology information can impact the development of intelligent services on the WWW.

Acknowledgement

We would like to thank Dr. James Hendler and Jeff Heflin for their support, suggestions, ideas, and technical help. And thanks to Lance Good who have commented on implementation issues.

This work has been supported in part by DARPA's Command Post of the Future project.

References

[1] Bederson, B.B., Hollan, J.D., Perlin, K., Meyer, J., Bacon, D., Furnas, G. W., "Pad++: A Zoomable Graphical Sketchpad for Exploring Alternate Interface Physics," *Journal of Visual Languages and Computing*, *7*, 3 – 31, 1996

[2] Bederson, B.B., Meyer, J., and Good, L., "Jazz: An Extensible Zoomable User Interface Graphics ToolKit in Java," *In Proceedings of User Interface and Software Technology (UIST 2000)*, ACM Press, 2000

[3] Brickley, D. and Guha, R., "Resource Description Framework (RDF) Schema Specification," W3c http://www.w3c.org/TR/2000/CR-rdf-schema-20000327/, 2000

[4] Craven, M., DiPasquo, D., Freitag, D., McCallum, A., Mitchell, T., Nigam K., and Slattery, S. "Learning to Extract Symbolic Knowledge from the World Wide Web," *In Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98)*, 1998

[5] DAML, http://www.daml.org

[6] Decker, S., Melnik, S., Van Harmelen, F., Fensel, D., Klein, M., Broekstra, J., Erdmann, M. and Horrocks, I., "The semantic web: The roles of XML and RDF," *IEEE Internet Computing* Sept.-Oct. 2000.

[7] Fensel, D., Angele, J., Decker, S., Erdmann, M., Schnurr, H.P., Studer, R. and Witt, A., "On2broker: Improving Access to Information Sources at the WWW," http://www.aifb.uni-karlsruhe.de/WBS/broker/inhalt-paper.html

[8] Guha, R. V. RDFDB QL, http://web1.guha.com/rdfdb/query.html

[9] Heflin, J.and Hendler, J. "Searching the Web with SHOE," *AAAI2000 Workshop on AI for Web Search*, 2000.

[10] JAXP XML parser, Sun Microsystems http://java.sun.com/xml

[11] Lassila, O. and Swick, R. "Resource Description Framework (RDF) Model and Syntax," W3C, http://www.w3c.org/TR/1999/REC-rdf-syntax-19990222, 1999.

[12] Luke, S. and Hefflin, J., "SHOE 1.0," Proposed Specification, http://www.cs.umd.edu/projects/plus/SHOE/spec.html, 1998.

[13] McGuinness, D. L., Fikes, R., Rice, J., and Wilder, S., "An Environment for Merging and Testing Large Ontologies," *In Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR2000)*, 2000.

[14] Miller, L., "RDF querying using Squish," http://swordfish.rdfweb.org/rdfquery/

[15] RDF Parser, Pro Solutions http://www.pro-solutions.com

[16] Schneiderman, B., Designing the User Interface, 3rd ed. Addison-Wesley Publishing Co., MA, 1998.

[17] Staab, S., Erdmann, M., Mädche, A., and Decker, S., "An Extensible Approach for Modeling Ontologies in RDF(S)," *In Proceedings of ECDL 2000 Workshop on the Semantic Web*, 11-22, 2000.

[18] Stoffel, K., Taylor, M., and Hendler, J., "Efficient Management of Very Large Ontologies," *In Proceedings of American Association for Artificial Intelligence Conference*, 1997.

[19] WordNet, http://www.cogsci.princeton.edu/~wn/