Regular Paths in SparQL: Querying the NCI Thesaurus

Landon T. Detwiler[1], Dan Suciu, PhD[3], James F. Brinkley, MD, PhD[1,2,3]
Structural Informatics Group, Departments of Biological Structure[1,]
Medical Education and Biomedical Informatics[2] and Computer Science and Engineering[3]
University of Washington, Seattle, WA 98195

Abstract

OWL, the Web Ontology Language, provides syntax and semantics for representing knowledge for the semantic web. Many of the constructs of OWL have a basis in the field of description logics. While the formal underpinnings of description logics have lead to a highly computable language, it has come at a cognitive cost. OWL ontologies are often unintuitive to readers lacking a strong logic background.

In this work we describe GLEEN, a regular path expression library, which extends the RDF query language SparQL to support complex path expressions over OWL and other RDF-based ontologies. We illustrate the utility of GLEEN by showing how it can be used in a query-based approach to defining simpler, more intuitive views of OWL ontologies. In particular we show how relatively simple GLEEN-enhanced SparQL queries can create views of the OWL version of the NCI Thesaurus that match the views generated by the web-based NCI browser.

Introduction

Knowledge representation languages for the semantic web include RDF[1] (Resource Description Framework), RDFS[1] (RDF Schema), and OWL[2] (Web Ontology Language) in its various sublanguage forms. The fundamental construct in each of these languages is the *triple,* which takes the form (*Subject, Predicate, Object*). Each triple makes a statement about the subject resource describing one of its properties and associated values. The triple (nci:Heart, rdfs:subClassOf, nci:Organ), for example, states that nci:Heart is a subclass of nci:Organ. A semantic web ontology is a graph constructed of many such triples, allowing knowledge modelers to describe a network of resources (nodes) within a domain of discourse and the predicates (edges) between them.

Much as SQL queries are used to define views over relational databases, we use semantic web queries as a basis for defining simplified ontology views. There are many existing semantic web query languages including RQL, RDQL, SeRQL, Versa, N3, OWL-QL, and SparQL. To support queries over all semantic web ontologies we based our work on an RDF query language (as all RDFS and OWL documents are also valid RDF). The query language currently recommended by the W3C (World Wide Web Consortium) for querying RDF is SparQL[3].

SparQL contains constructs which allow queries to express patterns of triples, from the underlying ontology(s), to match upon. For example, the triple pattern:

nci:Heart ?property ?object .

matches all triples in the ontology whose subject is nci:Heart. The entries preceded with the '?' symbol (?property and ?object) are variables that are bound by the query engine to values from the matched triples. Some examples of triples from the National Cancer Institute (NCI) Thesaurus that would match this pattern are:

nci:Heart rdfs:label **"Heart"**
nci:Heart rdfs:subClassOf **nci:Organ**

A *path* in the graph is a list of consecutive triples s.t. the object of triple$_n$ equals the subject of triple$_{n+1}$. A *path pattern* is a description of the configuration of edges that a matching path must have (note that the path pattern does not talk about the types of intervening nodes). Triple patterns can be combined to construct some path patterns. If, for example, you wish to know the labels of the superclasses of the superclasses of nci:Heart, you could build up this path pattern from 3 triple patterns:

nci:Heart rdfs:subClassOf ?super1 .
?super1 rdfs:subClassOf ?super2 .
?super2 rdfs:label ?label .

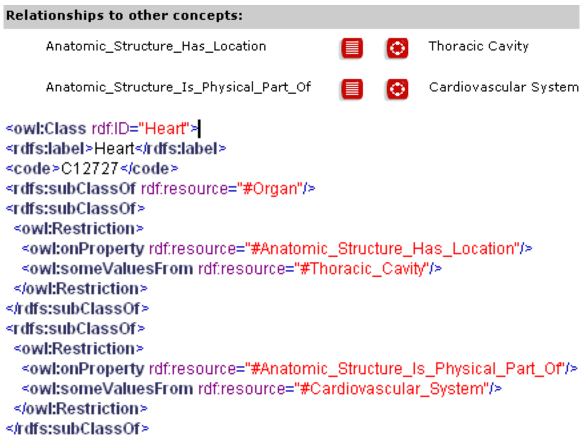
Unfortunately, there are several interesting types of path patterns that cannot be created simply by combining multiple triple patterns (i.e. if you instead wanted to know the labels of ALL superclasses, recursively, of nci:Heart).

To support more complex path expressions in SparQL we developed GLEEN, a regular path processing library. We use the term *regular path* to refer to a path expression specified using GLEEN's regular expression-like grammar. GLEEN is implemented as a property function library for ARQ[4], the SparQL query processor for the Jena RDF framework [5].

In the following sections we elaborate on the problems addressed by GLEEN. We demonstrate the utility of adding regular paths to SparQL queries using examples from the NCI Thesaurus (NCIt)[6]. While the NCIt is natively represented in the Ontylog description logic dialect, in this work we refer to the NCIt public OWL export. We follow these examples through as we describe our methods and discuss our results.

Problem: Querying OWL ontologies

In our experiments with SparQL we found that not all semantic web ontologies are equally intuitive to query. Particularly problematic are the description logic derived constructs in OWL. Such constructs are intended to maximize expressiveness while ensuring completeness and decidability of logical inference. Unfortunately, these constructs tend to obscure what many users intuit as direct relationships between resources.



Relationships to other concepts:

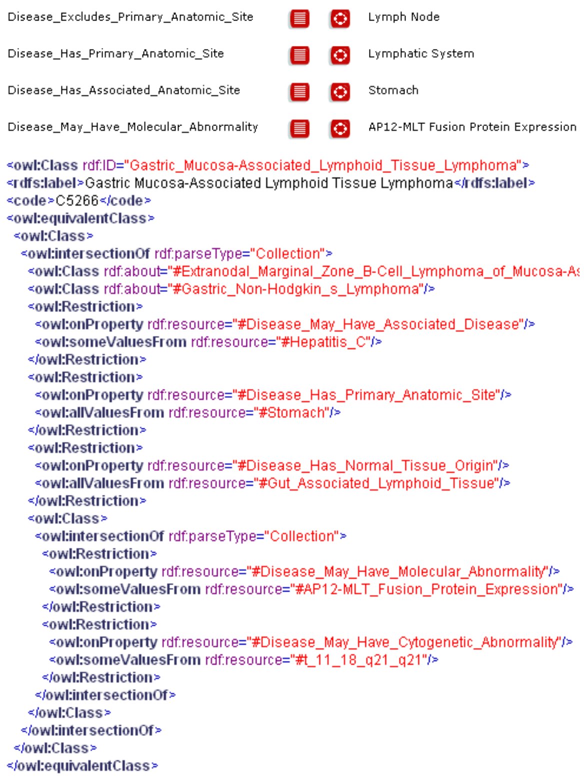
Anatomic_Structure_Has_Location	Thoracic Cavity
Anatomic_Structure_Is_Physical_Part_Of	Cardiovascular System

```
<owl:Class rdf:ID="Heart">
<rdfs:label>Heart</rdfs:label>
<code>C12727</code>
<rdfs:subClassOf rdf:resource="#Organ"/>
<rdfs:subClassOf>
 <owl:Restriction>
  <owl:onProperty rdf:resource="#Anatomic_Structure_Has_Location"/>
  <owl:someValuesFrom rdf:resource="#Thoracic_Cavity"/>
 </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
 <owl:Restriction>
  <owl:onProperty rdf:resource="#Anatomic_Structure_Is_Physical_Part_Of"/>
  <owl:someValuesFrom rdf:resource="#Cardiovascular_System"/>
 </owl:Restriction>
</rdfs:subClassOf>
```

Figure 1: Partial entries for Heart in NCIt browser (top) and in OWL (bottom)

For example Figure 1 (top) shows a portion of a screen capture, for the class *Heart*, from the NCIt terminology browser, specifically the "Relationships to other concepts" subsection which shows how the *Heart* is related to other classes in the ontology. This concept pane is in fact exposing a simplified view. In this view, there appears a direct Anatomical_Structure_Has_Location relationship (hereafter referred to as Has_Location for brevity) between *Heart* and *Thoracic Cavity*. However, examination of the partial OWL XML listing from Figure 1 (bottom) reveals that the graph does not contain a Has_Location property directly connecting the *Heart* and *Thoracic Cavity* resources. Instead the relationship is more convoluted. *Heart* is represented as a subclass of an anonymous class of type owl:Restriction. This class represents the collection of all resources that satisfy the existential condition that they have at least one value for the property Has_Location, which comes from the class *Thoracic Cavity*.

The previous example shows one way in which classes are inter-related within the NCIt. However, this is not the only manner in which such relationships are modeled. Figure 2 shows portions of a screen capture and OWL XML representation of *Gastric Mucosa-Associated Lymphoid Tissue Lymphoma* (hereafter referred to as *GM_Lymphoma* for brevity). *GM_Lymphoma* is defined as equivalent to (owl:equivalentClass) the intersection of (owl:intersectionOf) a collection of classes. This collection contains named classes, classes defined via property restrictions, and other class intersection collections.



Disease_Excludes_Primary_Anatomic_Site	Lymph Node
Disease_Has_Primary_Anatomic_Site	Lymphatic System
Disease_Has_Associated_Anatomic_Site	Stomach
Disease_May_Have_Molecular_Abnormality	AP12-MLT Fusion Protein Expression

```
<owl:Class rdf:ID="Gastric_Mucosa-Associated_Lymphoid_Tissue_Lymphoma">
<rdfs:label>Gastric Mucosa-Associated Lymphoid Tissue Lymphoma</rdfs:label>
<code>C5266</code>
<owl:equivalentClass>
 <owl:Class>
  <owl:intersectionOf rdf:parseType="Collection">
   <owl:Class rdf:about="#Extranodal_Marginal_Zone_B-Cell_Lymphoma_of_Mucosa-As
   <owl:Class rdf:about="#Gastric_Non-Hodgkin_s_Lymphoma"/>
   <owl:Restriction>
    <owl:onProperty rdf:resource="#Disease_May_Have_Associated_Disease"/>
    <owl:someValuesFrom rdf:resource="#Hepatitis_C"/>
   </owl:Restriction>
   <owl:Restriction>
    <owl:onProperty rdf:resource="#Disease_Has_Primary_Anatomic_Site"/>
    <owl:allValuesFrom rdf:resource="#Stomach"/>
   </owl:Restriction>
   <owl:Restriction>
    <owl:onProperty rdf:resource="#Disease_Has_Normal_Tissue_Origin"/>
    <owl:allValuesFrom rdf:resource="#Gut_Associated_Lymphoid_Tissue"/>
   </owl:Restriction>
   <owl:Class>
    <owl:intersectionOf rdf:parseType="Collection">
     <owl:Restriction>
      <owl:onProperty rdf:resource="#Disease_May_Have_Molecular_Abnormality"/>
      <owl:someValuesFrom rdf:resource="#AP12-MLT_Fusion_Protein_Expression"/>
     </owl:Restriction>
     <owl:Restriction>
      <owl:onProperty rdf:resource="#Disease_May_Have_Cytogenetic_Abnormality"/>
      <owl:someValuesFrom rdf:resource="#t_11_18_q21_q21"/>
     </owl:Restriction>
    </owl:intersectionOf>
   </owl:Class>
  </owl:intersectionOf>
 </owl:Class>
</owl:equivalentClass>
```

Figure 2: Partial entries for Gastric Mucosa-Associated Lymphoid Tissue Lymphoma in NCIt browser (top) and in OWL (bottom)

The NCIt screenshot in Figure 2 shows the property Disease_Excludes_Primary_Anatomic_Site with value *Lymph Node*. Where does this property come from? It is an inherited property from one of the classes in the intersection, *Extranodal Marginal Zone B-Cell Lymphoma of Mucosa-Associated Lymphoid Tissue*.

For the purpose of the following discussion we define classes as *directly related* if they are related within the NCIt browser view. Can we generate SparQL queries to retrieve the directly related classes for a given class of interest? The following query retrieves directly related classes for the *Heart* of Figure 1 [nci:Heart binds to the class with rdf:ID="Heart"]:

```
SELECT ?property ?value
WHERE {
  nci:Heart rdfs:subClassOf ?restriction .
  ?restriction owl:OnProperty ?property .
  ?restriction owl:someValuesFrom ?value .}
```

The above query illustrates that it is possible to build up some path patterns by combining (conjunctively) several triple patterns. However, this requires *a priori* knowledge of the structure of the graph.

Can we use this same sort of query to retrieve the classes directly related to *GM_lymphoma* (Figure 2)? There are no rdfs:subClassOf edges emanating from the *GM_Lymphoma* node. Therefore, the previous query pattern will not find any direct relationships. The following query would return some of the directly related classes, but not all, and it requires the query author to understand exactly how these classes are related in OWL:

```
SELECT ?property ?value
WHERE
{
  nci:GM_Lymphoma owl:equivalentClass ?equiv .
  ?equiv owl:intersectionOf ?list .
  ?list list:member ?member .
  ?member owl:intersectionOf ?nested_list .
  ?nested_list list:member ?nested_member .
  ?nested_member owl:onProperty ?property .
  ?nested_member owl:someValuesFrom ?value .
}
```

The above query is not only complicated, it also returns only 2 related classes, out of 31 shown in the browser (only 4 of which were shown in Figure 2). In fact, we note here that this query would be considerably more complicated without another built-in ARQ property function list:member.

How do we query, with SparQL, for all of the direct relationships of a class in the NCIt, without prior knowledge of the RDF graph structure? The GLEEN regular path library provides one approach.

The GLEEN path expression library
Previous work with regular path patterns in SparQL include PSPARQL[7] and SparQLeR[8]. While we were informed by these projects, they did not meet at least 2 of our principle requirements:

1. Support for persistent storage: Loading graphs entirely in memory is not really practical for something as large as the NCIt.
2. Easily incorporated into existing SparQL services (e.g. works in combination with a widely adopted SparQL engine).

We developed GLEEN as a plugin for ARQ due to ARQ's wide use. The ARQ SparQL engine provides an extension mechanism, known as property functions, which overloads the syntax of a basic triple pattern. In standard SparQL, a URI in the predicate position of a triple pattern uniquely identifies an ontology property. However, in ARQ property function triples, the property URI uniquely identifies a custom triple matching function. These functions enable custom query processing and are allowed to bind, programmatically, values to any variables in the triple pattern. Subject or object arguments of a property function triple may be lists as well as atomic values (if more than 2 parameters are required by, or set by, the function).

GLEEN is implemented as a Java library of ARQ property functions. The function we discuss here is called OnPath. The OnPath property function expects an atomic value in the subject position and a 2 element list in the object position of the triple pattern. Syntactically, a call to the GLEEN OnPath function looks like this:

```
subject gleen:OnPath ( pathExpression object )
```

The subject and object may be either the URI of an ontology resource (i.e. nci:Heart) or a variable (i.e. ?var). If both subject and object are variable, at least one must be bound (to an ontology resource(s)). The pathExpression argument is a string representation of the regular path of interest.

The path expression grammar we use in GLEEN is similar to common regular expression pattern grammars, thus the term "regular paths". It supports operators '?' (zero or one), '*' (zero or more), '+' (one or more), '|' (alternation), and '/' (concatenation). Square brackets are used as property delimiters and parentheses as grouping operators. We omit the complete specification of the grammar here; more details can be found on the GLEEN web site[9].

Let us now give a couple of examples of GLEEN path expressions:

```
[rdfs:subClassOf]*
[rdfs:subClassOf]/([owl:someValuesFrom]|[owl:allValuesFrom])
```

```
SELECT ?prop ?val
WHERE{
  nci:GM_Lymphoma gleen:OnPath (
          "([owl:equivalentClass]?/[owl:intersectionOf]/[rdf:rest]*/[rdf:first])+" ?restriction ) .
  ?restriction owl:onProperty ?prop .
  ?restriction gleen:OnPath ( "[owl:someValuesFrom] | [owl:allValuesFrom]" ?val ). }
```

Figure 3: Select query for properties and values of *GM_Lymphoma*.

The first expression matches paths with zero or more consecutive rdfs:subClassOf properties (transitive closure). The second expression matches paths beginning with a single rdfs:subClassOf edge followed by either an owl:someValuesFrom property or an owl:allValuesFrom property.

Consider again the case of *GM_Lymphoma* (Figure 2) which was defined as equivalent to the intersection of a collection of classes. One of the classes in this intersection was itself defined as an intersection of other classes. While all of the classes in this latter intersection were named classes, they could have themselves been collections, and so on. Without knowing how many levels deep this structure goes, we cannot construct a standard SparQL query to grab all of the relevant resources.

Using the OnPath function we can express this query, for the properties of *GM_Lymphoma*, as shown in Figure 3. The path pattern in the first call to OnPath looks like this:

```
([owl:equivalentClass]?/[owl:intersectionOf]/
        [rdf:rest]*/[rdf:first])+
```

This pattern matches paths containing one or more consecutive subpaths where each subpath optionally starts with a single owl:equivalentClass property followed by an owl:intersectionOf property followed by zero or more rdf:rest properties followed by an rdf:first property. While no rdf:first or rdf:rest properties are directly visible in the OWL XML of Figure 2, we note that the property owl:intersectionOf with rdf:parseType= "Collection", is a shorthand notation for a more complex RDF graph containing these properties. This pattern binds, to the ?restriction variable, classes in the owl:intersectionOf collection.

The pattern in the second call to OnPath is as follows:

```
[owl:someValuesFrom] | [owl:allValuesFrom]
```

This pattern allows us to retrieve classes related by a property restriction, whether that restriction is universal or existential.

Creating a view

Is it possible to create a view of *GM_Lymphoma* that would return the same results as the query from Figure 3, for the following simpler query?

```
SELECT ?property ?value
WHERE { nci:GM_Lymphoma ?property ?value . }
```

We create such a view, with GLEEN, by modifying the query from the previous section, changing it from a SELECT to a CONSTRUCT query (see Figure 4 top).

This query results in a simplified RDF document, a portion of which is also shown in Figure 4 (bottom). In this view, *GM_Lymphoma* has a direct property Disease_Excludes_Primary_Anatomic_Site with value *Lymph*

```
CONSTRUCT{ nci:GM_Lymphoma ?prop ?val }
WHERE{
  nci:GM_Lymphoma gleen:OnPath (
          "([owl:equivalentClass]?/[owl:intersectionOf]/[rdf:rest]*/[rdf:first])+" ?restriction ) .
  ?restriction owl:onProperty ?prop .
  ?restriction gleen:OnPath ( "[owl:someValuesFrom]|[owl:allValuesFrom]" ?val ). }

<rdf:Description rdf:about=".../Thesaurus.owl#Gastric_Mucosa-Associated_Lymphoid_Tissue_Lymphoma">
   <nci:Disease_Has_Normal_Tissue_Origin rdf:resource=".../Thesaurus.owl#Lymphoid_Tissue"/>
   <nci:Disease_Excludes_Primary_Anatomic_Site rdf:resource=".../Thesaurus.owl#Lymph_Node"/>
   <nci:Disease_Has_Normal_Cell_Origin rdf:resource=".../Thesaurus.owl#Lymphocyte"/>
   <nci:Disease_Has_Abnormal_Cell rdf:resource=".../Thesaurus.owl#Neoplastic_Lymphocyte"/>
   <nci:Disease_Has_Normal_Cell_Origin rdf:resource=".../Thesaurus.owl#Marginal_Zone_B-Lymphocyte"/>
   <nci:Disease_May_Have_Abnormal_Cell rdf:resource=".../Thesaurus.owl#Neoplastic_Monocytoid_B-Cell"/>
   <nci:Disease_Excludes_Abnormal_Cell rdf:resource=".../Thesaurus.owl#Reed-Sternberg_Cell"/>
```

Figure 4: View Query and partial results for *GM_Lymphoma*.

Node, which is consistent with the browser view shown at the top of Figure 2.

The CONSTRUCT query in Figure 4 (the most complex of our example queries) takes approximately 700mS to execute in our prototype system [Core2 PC with NCIt OWL stored in remote PostgreSQL dbms with a 100Mb network connection].

Discussion

The long-term goal of our work is to develop query-based methods for extracting simplified views of large ontologies like the NCI thesaurus, so that these views may be more easily incorporated in applications[10]. Figure 4 shows that GLEEN, while only a step towards that goal, already provides a useful service; generating simplified, materialized views of complex ontologies. A full evaluation of the usefulness of this approach, on the OWL version of the NCI thesaurus in particular, would require examination of all of the representation patterns used in NCIt (like those in Figures 1 and 2). Ideally, a limited number of queries like those in Figure 3 could be used to generate a materialized view of the entire NCI thesaurus. This view could then be embedded in an ARQ SparQL ontology web service.

While OnPath presently allows users to discover the resources reachable by a given path pattern, it does not allow users to see the exact path traversed. A path expression like

nci:Liver gleen:OnPath ("[rdfs:subClassOf]*" ?super)

would bind to ?super all super-classes, recursively, of *Liver*. It would not, however, reveal which intermediate super-class led to which subsequent result(s). Another Gleen property function *Subgraph*, not illustrated here, enables a SparQL query to discover all triples involved in any matching path. The resulting subgraph contains all intermediate results and as well as the path edges connecting them.

One limitation of the current OnPath function is that, if only the subject OR object is bound (not both), it must be bound to a resource URI, not a literal. Additionally, for tractability reasons, OnPath does not support queries where neither subject nor object are bound.

Conclusion

We developed the GLEEN regular path library as an add-on to the ARQ SparQL query processor. GLEEN supports graph path pattern matching beyond what is available in standard SparQL alone (without considerable prior knowledge of the graph). We demonstrated the utility of this sort of pattern matching by showing how it can be used to simplify a small portion of a large, complex ontology (the NCI thesaurus).

GLEEN has been released into the open-source community under the Apache 2.0 license agreement. It is free to use and extend. Further information regarding the GLEEN project as well as the latest download are available from the GLEEN homepage[9].

Acknowledgements
This work was funded by NIH grant HL087706. We also acknowledge the Jena and ARQ projects for the solid RDF/SparQL foundation they provide.

References

1. W3C RDF Primer. [cited 2008 July 7]; Available from: http://www.w3.org/TR/rdf-primer/.
2. OWL Web Ontology Language Reference. [cited 2008 July 14]; Available from: http://www.w3.org/TR/owl-ref/.
3. SPARQL Query Language for RDF. [cited 2008 March 9]; W3C SparQL Specification]. Available from: http://www.w3.org/TR/rdf-sparql-query/.
4. ARQ - A SPARQL Processor for Jena. [cited 2008 March 10]; Available from: http://jena.sourceforge.net/ARQ/.
5. Carroll JJ, Dickinson I, Dollin C, Reynolds D, Seaborne A, Wilkinson K, editors. Jena: implementing the semantic web recommendations. 13th International World Wide Web Conference; 2004; New York.
6. Golbeck J, Fragoso G, Hartel F, Hendler J, Parsia B, Oberthaler J. The national cancer institute's thesaurus and ontology. 2003.
7. Alkhateeb F, Baget J-F, Euzenat J. Extending SPARQL with regular expression patterns. Institut National de Recherche en Informatique et Automatique (INRIA), Tech Rep 6191. 2007.
8. Kochut K, Janik M. SPARQLeR: Extended Sparql for Semantic Association Discovery. European Semantic Web Conference (ESWC). 2007:145-59.
9. GLEEN: Regular Paths for ARQ SparQL. [cited 2008 March 10]; Available from: http://sig.biostr.washington.edu/projects/ontviews/gleen/.
10. Brinkley JF, Suciu D, Detwiler LT, Gennari JH, Rosse C. A framework for using reference ontologies as a foundation for the semantic web. AMIA Annual Symposium proceedings / AMIA Symposium. 2006:96-100.