Querying Non-Materialized Ontology Views

NIH 1R01HL087706-01

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

Introduction:

Application-specific views of reference ontologies, such as the Foundational Model of Anatomy (FMA), facilitate their inclusion in a more tractable semantic web. A view definition language (VDL) defines how simplified "view" or "application" ontologies are derived from larger more complex ontologies. We illustrate some initial ideas for how to execute user queries over a VDL defined ontology view, without materializing it first.

Approach:

Like SQL views in relational databases, we will define our RDF(S)/OWL views using a declarative query language. Queries in this presentation are expressed in SparQL, the W3C recommended RDF(S) query language.

Regular Paths:

SparQL lacks support for regular paths, including recursive predicates, which we view as necessary constructs of a VDL. We addressed this deficiency by extending SparQL via Jena custom functions,. One such function calculates the transitive closure from a resource *sub* over a relationship *rel* to all reachable resources *obj*:

```
sub ext:Closure (rel obj) .
```

The custom function *Closure* works as follows:

- *rel* must be a property URI
- If *sub* is a resource URI and *obj* is an unbound variable (?obj), then Closure binds ?obj to the URIs of all resources that stand in the Kleene closure *rel** from *sub*.
- If *sub* is a variable (?sub) bound to resource URI(s) and *obj* is a URI, then Closure reduces the bindings on ?sub to just those values whose Kleene closure *rel** contain *obj*.

User Query (Q1):

```
CONSTRUCT {
    ?subject ?relation ?object .}
WHERE{
    ?subject rdfs:subClassOf fma:Long_bone .
    ?subject ?relation ?object .}
```

Fig1: User Query Q1

View (V1):

```
CONSTRUCT {
    ?sub obo:has_part ?part .
    ?sub rdfs:subClassOf ?superClass . }
WHERE{
    fma:Skeletal_system_of_upper_limb
        ext:Closure (fma:regional_part ?sub) .
    ?sub fma:regional_part ?part .
    ?sub rdfs:subClassOf ?superClass . }
```

Fig2: View Query V1

Query Composed with a View (C1 = Q1 · V1):

```
CONSTRUCT {
    ?sub obo:has_part ?part .
    ?sub rdfs:subClassOf  ?superClass . }
WHERE{
    fma:Skeletal_system_of_upper_limb
        ext:Closure (fma:regional_part ?sub) .
    ?sub fma:regional_part ?part .
    ?sub rdfs:subClassOf ?superClass .
    ?sub rdfs:subClassOf fma:Long_bone .
    ?sub ?relation ?object .}
```

Fig3: Composed Query C1 = Q1 · V1

User query Q1 (Fig1) returns all triples, from the underlying ontology, whose subject *isa* Long_bone. View Query V1 (Fig2) identifies all *regional_parts* of the Skeletal_system_of_upper_limb, and for each of these returns its direct *regional_parts* (with property renamed as *obo:part*) as well as its direct *superclass*.

If we run Q1 on the entire FMA, the results include all direct relationships for Long_bones Clavicle, Humerous, Ulna, Radius, etc. The size of this result set is illustrated by Fig4. If User Query Q1 is instead posed to the view V1, the results should be restricted to the Clavicle, the only Long_bone in the view.

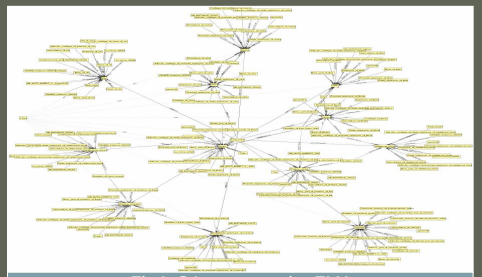

Fig4: Q1 run on entire FMA

Query Composition:

One approach to answering Q1 involves first executing V1 (Fig5) and then running Q1 against V1's materialized RDF result graph. Alternatively, we can answer Q1 without first materializing V1, by composing Q1 and V1 to form a new query C1 over the underlying ontology (FMA). Fig3 illustrates a composition of Q1 with V1 (note the substitution of ?sub for the ?subject variable in Q1), results are shown in Fig6. C1's WHERE clause imposes the combined graph matching constraints of Q1 and V1. The CONSTRUCT clause retains the triple modifications of the view query, unless overridden by Q1.

Note that the composed query contains an unnecessary graph pattern, ?sub ?relation ?object. This was left in for clarity, but this pattern does not effect the output. An optimizer could be used to remove such patterns.

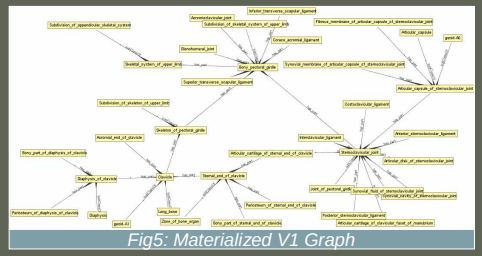

Fig5: Materialized V1 Graph

Results of Composition (C1 = Q1 · V1):

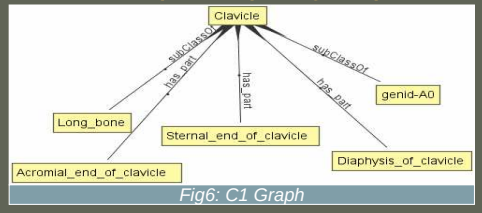

Fig6: C1 Graph

Summary:

The primary objective of this work was to investigate the complexities of composing user queries with view definition queries, in order to answer questions over non-materialized ontology views. We illustrated one possible compositional approach. The method shown here produces correct results under constrained conditions, but generated queries may be inefficient. Query optimization techniques could be used to improve efficiency. Additionally, we are investigating other compositional methodologies, such as nested CONSTRUCT queries.