

Notes on SPARQL Query Language for RDF

<http://www.w3.org/TR/2005/WD-rdf-sparql-query-20050721/>

Enrico Franconi and Sergio Tessaris
Faculty of Computer Science
Free University of Bozen-Bolzano, Italy
lastname@inf.unibz.it

1 September 2005

This document:

<http://www.inf.unibz.it/krdb/w3c/sparql-notes-fub.pdf>

We propose the layering of SPARQL into a core language and a full language. This proposal follows the same principle which led to the definition of OWL-Lite. This core should be “semantically clean” and, ideally, its expressiveness should be close to the conjunctive query language for databases (i.e. SPJ queries) or positive queries. Our suggestion is to call this language *Core SPARQL* language (cSPARQL). In general, we believe that a distinction should be made between the semantics of the (core) query language and details which are more system protocol related. For example, in this latter category fall most of the Solution Sequence Modifiers described in <http://www.w3.org/TR/2005/WD-rdf-sparql-query-20050721/#solutionsResults>.

Query Solution

(<http://www.w3.org/TR/2005/WD-rdf-sparql-query-20050721/#BasicGraphPatternMatching>). The definition of *matching* of a solution w.r.t. a graph is given in term of *subgraph*. Solutions should be grounded on Graph Entailment as defined in <http://www.w3.org/TR/rdf-mt/>. In particular, differences can show up with queries involving axiomatic triples (see <http://www.w3.org/TR/rdf-mt/#RDFINTERP>), or containing blank nodes.

For example, given the data

```
@prefix foaf:    <http://xmlns.com/foaf/0.1/> .  
  
_:a foaf:name   "Johnny Lee Outlaw" .  
_:a foaf:mbox   <mailto:jlow@example.com> .
```

the query

```
SELECT ?x
WHERE { ?x rdf:type rdf:Property }
```

should return the following solutions:

x
foaf:name
foaf:mbox
rdf:type
rdf:subject
rdf:predicate
rdf:object
rdf:first
rdf:rest
rdf:value
rdf:_1
rdf:_2
...

Blank Nodes in Queries

(<http://www.w3.org/TR/2005/WD-rdf-sparql-query-20050721/#BlankNodes>).

A blank node in a query pattern “behaves as a variable; a blank node in a query pattern may match any RDF term”. However, in the definition of Basic Graph Pattern, it is stated that “A basic graph pattern matches on graph G with solution S if S(GP) is an RDF graph and is subgraph of G.”. A solution provides binding for the variables only, so accordingly to this definition blank nodes match only blank nodes with the same name. Again, it seems that “being a subgraph of” should be better specified (or a solution should provide binding for blank nodes as well).

Blank nodes in query results

(<http://www.w3.org/TR/2005/WD-rdf-sparql-query-20050721/#BlankNodes>).

If blank nodes are allowed as binding for distinguished variables¹ there can be a problem in defining the set of answers to a query.

For example, consider the following two RDF datasets

```

RDF1
@prefix dc:    <http://purl.org/dc/elements/1.1/> .

<http://example.org/book/book1> dc:title "SPARQL" .
```

¹*Distinguished* variables are those returned as query results

RDF2

```
@prefix dc:    <http://purl.org/dc/elements/1.1/> .

<http://example.org/book/book1> dc:title "SPARQL" .
_:b dc:title "SPARQL" .
```

The two RDF graphs are semantically equivalent. In fact, Graph RDF1 entails RDF2 because it is one of its instance (see Instance Lemma in <http://www.w3.org/TR/rdf-mt/>). Moreover, RDF1 is entailed by RDF2 because the latter contains the first one (Subgraph Lemma in <http://www.w3.org/TR/rdf-mt/>).

Given this equivalence, we would expect that querying the two graphs would yield to the same set of answers, regardless the query. However, this is not the case for the simple query

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
SELECT ?x
WHERE { ?x dc:title "SPARQL" }
```

which returns just `<http://example.org/book/book1>` in the first case and an additional blank node with the second graph.

Note that returning blank nodes as results provides an additional expressive power only in the case that a co-reference shows up in the answer. For example, with the graph

RDF2

```
@prefix dc:    <http://purl.org/dc/elements/1.1/> .

_:b dc:title "Moby-Dick" .
_:b dc:author "Melville" .
```

and the query

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
SELECT *
WHERE { ?x dc:title ?xt .
        ?y dc:author ?ya }
```

the answer

x	xt	y	ya
_:d	"Moby-Dick"	_:d	"Melville"

shows that variables `x` and `y` refer to the same blank node.

In any other situation, a non distinguished variable (or a blank node in the query) would provide the same informative content.

OPTIONAL and UNION

The two operators are equivalent from both a semantic and implementation perspective. In fact, the construct

```
{ pattern1 } UNION { pattern2 }
```

is equivalent to

```
{ { } OPTIONAL { pattern1 }  
  OPTIONAL { pattern2 } }
```

while

```
{ pattern1 } OPTIONAL { pattern2 }
```

is equivalent to

```
{ pattern1 } UNION { pattern1 pattern2 }
```

For this reason it can be convenient to choose one of the two forms to be introduced in cSPARQL (we suggest UNION for its connection with standard DB query languages), and leave the other as syntactic sugar for standard SPARQL.

Query result forms

(<http://www.w3.org/TR/2005/WD-rdf-sparql-query-20050721/#QueryForms>).

The DESCRIBE query result form has a extremely vague semantics. There are no requirements on the query answering system, which in principle is not even required to provide the same answer over the same dataset in different calls of the same query. We agree with the doubts expressed in <http://www.w3.org/2001/sw/DataAccess/ftf4.html#item14>, and we believe that the construct should be kept outside cSPARQL.

Testing values

(<http://www.w3.org/TR/2005/WD-rdf-sparql-query-20050721/#tests>).

The use of tests without any sort of safeness imposed over the way the variables can appear in the expressions could lead to unwanted results. For example, the query

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>  
SELECT ?name ?n  
WHERE { ?x foaf:name ?name .  
        FILTER ?n >= "0"^^xs:decimal }
```

according to the definition of matching, produces an infinite number of answers given the dataset

```

@prefix foaf:    <http://xmlns.com/foaf/0.1/> .

_:a foaf:name    "Johnny Lee Outlaw" .
_:a foaf:mbox    <mailto:jlow@example.com> .

```

Most likely what is really needed is to constraint variables in FILTER clauses to appear in a graph pattern as well. However, this can be not enough since a variable appearing in an OPTIONAL clause can lead to the same problem. An alternative approach is the restriction of binding to the so called *active values* in the dataset(s).

A second problem lies in the BOUND() operator. Consider, for example, the query

```

PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dc:   <http://purl.org/dc/elements/1.1/>
SELECT ?name
WHERE { ?x foaf:name ?name .
        OPTIONAL { ?x dc:date ?date } .
        FILTER (bound(?date)) }

```

and the dataset

```

@prefix foaf:    <http://xmlns.com/foaf/0.1/> .
@prefix dc:      <http://purl.org/dc/elements/1.1/> .
@prefix xs:      <http://www.w3.org/2001/XMLSchema#> .

_:a foaf:givenName "Alice".
_:b foaf:givenName "Bob" .
_:b dc:date         "2005-04-04T04:04:04Z"^^xs:dateTime .

```

According to the example the result should be just “Bob”. However, the binding $[x/_:a, name/"Alice", date/"Anything"]$ matches the query according to the definition of optional matching. Therefore “Alice” should be a solution as well. In this case, we probably need a more precise definition of solution as the minimal set of bindings.

Querying provenance

(<http://www.w3.org/TR/2005/WD-rdf-sparql-query-20050721/#queryDataset>). The construct GRAPH, combined with FROM clauses, allows to bind variables/restrict graphs the triples came from. The main problem with this feature is that the RDF Dataset definition given in <http://www.w3.org/TR/2005/WD-rdf-sparql-query-20050721/#rdfDataset> does not have a corresponding semantics in the RDF model theory. Note that

variables in queries are allowed to range over both graph names and element of the RDF universe, e.g.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dc:   <http://purl.org/dc/elements/1.1/>

SELECT ?name ?mbox
WHERE
  { ?g dc:publisher ?name .
    GRAPH ?g
      { ?person foaf:name ?name ; foaf:mbox ?mbox } }
```

These features make the development of a proper semantics for the queries quite difficult, or even impossible without extending the current RDF model theory. We reckon that the possibility of referring to graph names enables some interesting features of the query language; however, we believe that this should be excluded from cSPARQL.