

## Semantic Web Technologies

### Advanced SPARQL

Jos de Bruijn  
jos.debruijn@deri.org

Digital Enterprise Research Institute (DERI)  
University of Innsbruck, Austria

May 4, 2006

115/140

## Terms and Variables

### Definition: RDF Term

let **I** be the set of all IRIs (URIs).  
let **RDF-L** be the set of all RDF Literals  
let **RDF-B** be the set of all blank nodes in RDF graphs  
The set of **RDF Terms**, **RDF-T**, is  $I \cup \text{RDF-L} \cup \text{RDF-B}$ .

### Definition: Query Variable

A **query variable** is a member of the set  $V$  where  $V$  is infinite and disjoint from **RDF-T**.

118/140

## E-entailment regime

Allowing different entailment regimes:

### Definition: E-entailment Regime

An **E-entailment regime** is a binary relation between subsets of RDF graphs.

A graph in the range of an E-entailment is called **well-formed** for the E-entailment.

120/140

## Outline

### Graph Matching

Basic Matching  
Extending Basic Matching  
Querying Named Graphs

### Query Result Forms

116/140

## Triple Pattern, Basic Graph Pattern

### Definition: Triple Pattern

A **triple pattern** is member of the set:

$$(\text{RDF-T} \cup V) \times (I \cup V) \times (\text{RDF-T} \cup V)$$

### Definition: Basic Graph Pattern

A **Basic Graph Pattern** *BGP* is a set of Triple Patterns.

### Definition: Pattern Solution

A **pattern solution**,  $S$ , is a variable substitution whose domain includes all the variables in  $V$  and whose range is a subset of the set of RDF terms.

If  $v$  is not in the domain of  $S$  then  $S(v)$  is defined to be  $v$ .

119/140

## Graph Equivalence

Take the **simple RDF entailment** regime: Subgraph matching with blank nodes as existential variables

### Definition: Basic Graph Pattern equivalence

Two basic graph patterns  $A$  and  $B$  are **equivalent** if there is a bijection  $M$  between the terms of the triple patterns that maps

- ▶ blank nodes to blank nodes and
- ▶ variables, literals and IRIs to themselves,

such that:

$$\langle s,p,o \rangle \in A \text{ if and only if } \langle M(s),M(p),M(o) \rangle \in B$$

121/140

## Scoping Set, Scoping Graph

Restricting the variable assignments:

**Definition: Scoping Set**

A **Scoping Set**  $B$  is some set of RDF terms.

Making the match independent of blank node names:

**Definition: Scoping Graph**

The **Scoping Graph**  $G'$  for RDF graph  $G$ , is an RDF Graph that is **graph-equivalent** to  $G$

For simple RDF entailment,  $B$  is the set of all RDF Terms in  $G'$ .

122/140

## Basic Graph Pattern Matching

Given an entailment regime  $E$ , a basic graph pattern  $BGP$ , and RDF graph  $G$ , with scoping graph  $G'$ , then  $BGP$  **E-matches** with pattern solution  $S$  on graph  $G$  with respect to scoping set  $B$  if:

1.  $BGP'$  is a basic graph pattern that is graph-equivalent to  $BGP$
2.  $G'$  and  $BGP'$  do not share any blank node labels.
3.  $(G' \cup S(BGP'))$  is a well-formed RDF graph for E-entailment
4.  $G$  E-entails  $(G' \cup S(BGP'))$
5. The RDF terms introduced by  $S$  all occur in  $B$ .

124/140

## Extending Entailment in SPARQL

- ▶ Different entailment regime defines **different graph equivalence** relation (1)
- ▶ Different entailment regime may define **different well-formed RDF graphs** (3)
- ▶ Different entailment regime may define **different pattern solutions** (4)

125/140

## Matching Value Constraints

**Definition: Value Constraint**

A **value constraint** is a boolean-valued expression of variables and RDF Terms.

For value constraint  $C$ , a solution  $S$  matches  $C$  if  $S(C)$  is true, where  $S(C)$ .

127/140

## Optional Patterns

**Definition: Optional Graph Pattern**

An **optional graph pattern** is a **pair** of graph patterns.

If  $Opt(A, B)$  is an optional graph pattern, where  $A$  and  $B$  are graph patterns, then  $S$  is a solution of  $Opt(A, B)$ :

- ▶ if  $S$  is a pattern solution of  $A$  and of  $B$  **or**
- ▶ if  $S$  is a solution to  $A$ , but not to  $A$  and  $B$ .

128/140

## RDF Dataset

**Definition: RDF Dataset**

An **RDF dataset** is a set:

$$\{G, (< u_1 >, G_1), (< u_2 >, G_2), \dots, (< u_n >, G_n)\}$$

where  $G$  and each  $G_i$  are graphs, and each  $< u_i >$  is an IRI. Each  $< u_i >$  is distinct.

$G$  is called the **default graph**.  $(< u_i >, G_i)$  are called **named graphs**.

130/140

## Querying Named Graphs

### Definition: RDF Dataset Graph Pattern

If  $D$  is a dataset  $\{G, \langle u_1 \rangle, G_1, \dots\}$ , and  $P$  is a graph pattern then  $S$  is a pattern solution of RDF Dataset Graph Pattern  $\text{GRAPH}(g, P)$  if either of:

1.  $g$  is an IRI where  $g = \langle u_i \rangle$  for some  $i$ , and  $S$  is pattern solution of  $P$  on dataset  $\{G_i, \langle u_1 \rangle, G_1, \dots\}$
2.  $g$  is a variable,  $S$  maps the variable  $g$  to  $\langle u_j \rangle$ , where  $\langle u_j \rangle$  is an IRI from a named graph of  $D$ , and  $S$  is a pattern solution of  $P$  on dataset  $\{G_j, \langle u_1 \rangle, G_1, \dots\}$ .

131/140

## Query Result Forms

- ▶ SELECT: Projection of query result
- ▶ CONSTRUCT: Returning RDF Graph
- ▶ DESCRIBE: Returning descriptions of RDF resource
  - ▶ not treated here, because **underspecified**
- ▶ ASK: "yes/no" query

133/140

## Projection: SELECT

- ▶ SELECT  $x_1, \dots, x_n$
- ▶ Specify variables to "retrieve"
- ▶ **Projection** of pattern solutions
  - ▶ Consider only variables in SELECT clause
- ▶ Similar to query answers in SQL

134/140

## Returning an RDF Graph: CONSTRUCT

- ▶ CONSTRUCT { *basic triple pattern*\* }
- ▶ Query result is RDF graph
- ▶ Form of RDF Graph described using **graph template**
- ▶ Construct graph for each **pattern solution**
- ▶ Triples with unbound variables **discarded**
- ▶ Illegal RDF triples **discarded**

135/140

## CONSTRUCT Query Answers: example

### Graph

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

```
_:a foaf:name "Alice" .  
_:a foaf:mbox <mailto:alice@example.org> .
```

### Query

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>  
PREFIX vcard: <http://www.w3.org/2001/vcard-rdf/3.0#>  
CONSTRUCT { <http://example.org/person#Alice> vcard:FN ?name }  
WHERE { ?x foaf:name ?name }
```

### Result

```
@prefix vcard: <http://www.w3.org/2001/vcard-rdf/3.0#> .
```

```
<http://example.org/person#Alice> vcard:FN "Alice" .
```

136/140

## Boolean Queries: ASK

- ▶ ASK { *graph pattern* }
- ▶ "Does the query have an answer?"
- ▶ ASK **replaces** WHERE
- ▶ Queries without variables are **meaningful**

137/140

## ASK Query Answers: example

### Graph

```
@prefix foaf:      <http://xmlns.com/foaf/0.1/> .  
  
_:a foaf:name     "Alice" .  
_:a foaf:homepage <http://work.example.org/alice/> .  
  
_:b foaf:name     "Bob" .  
_:b foaf:mbox    <mailto:bob@work.example> .
```

### Query

```
PREFIX foaf:      <http://xmlns.com/foaf/0.1/>  
ASK { ?x foaf:name "Alice" }
```

### Result

yes

138/140

## Summary

### Graph Matching

- Basic Matching
- Extending Basic Matching
- Querying Named Graphs

### Query Result Forms

139/140

### Required reading

### Further reading

- ▶ SPARQL Query Language for RDF:  
<http://www.w3.org/TR/rdf-sparql-query/>

140/140