

## Organization of the Labs

Labs today are canceled

Labs **bi-weekly** 17:00 – 19:00 starting March 28

- ▶ Schedule
  - ▶ March 28
  - ▶ April 11
  - ▶ April 18
  - ▶ May 2
  - ▶ May 16
  - ▶ May 23
  
- ▶ May 23 will be an exam preparation section

# Semantic Web Technologies

## Advanced SPARQL

Jos de Bruijn  
`jos.debruijn@deri.org`

KRDB Research Group  
Free University of Bolzano, Italy

21 March 2007

# Outline

Entailment Regimes in RDF

Graph Matching

- Basic Matching

- Extending Basic Matching

Query Result Forms

## E-entailment regimes

- ▶ E-entailment regime = kind of entailment
  - ▶ Terminology used in SPARQL
- ▶ **RDF entailment**
  - ▶ Discussed two weeks ago
  - ▶ An interpretation  $I$  is an RDF interpretation if certain conditions hold
  - ▶ Graph  $S \models_{RDF} E$  iff every RDF interpretation which is a model of  $S$ , is a model of  $E$

## E-entailment regimes (continued)

### ► RDFS entailment

- An interpretation  $I$  is an RDFS interpretation if certain conditions hold
- Graph  $S \models_{RDFS} E$  iff every RDFS interpretation which is a model of  $S$ , is a model of  $E$
- Alternatively: Let  $S'$  be the **realization** of  $S$ , i.e.  $S'$  is obtained from  $S$  by adding the axiomatic triples and applying the entailment rules, until no rule is applicable. Then,  $S \models_{RDFS} E$  iff  $E \subseteq_{\text{modulo blank node renaming}} S'$ , or  $S' \models_S E$ .

## Blank nodes substitution

### Blank node substitution

A substitution is a mapping of the form

$\theta = \{b_1 \mapsto t_1, \dots, b_n \mapsto t_n\}$ , with  $b_1, \dots, b_n$  blank nodes (essentially, variables), and  $t_1, \dots, t_n$  blank nodes, URIs, or literals (essentially, terms).

Example:

$\theta = \{ \_ :x \mapsto \#john, \_ :z \mapsto \_ : u, \_ :y \mapsto \_ :y, \_ :w \mapsto " John Smith" \}$

### Substitution Application

$S = \{ \langle \#john, \text{hasName}, \_ :w \rangle, \langle \_ :x, \text{marriedTo}, \_ :z \rangle, \langle \_ :y, \text{marriedTo}, \_ :x \rangle \}$

$\theta(S) = \{ \langle \#john, \text{hasName}, " John Smith" \rangle, \langle \#john, \text{marriedTo}, \_ :u \rangle, \langle \_ :y, \text{marriedTo}, \#john \rangle \}$

## Simple entailment

- ▶ Simple entailment “ignores” RDF(S) vocabulary
- ▶ Simple entailment **does** take into account blank node semantics
- ▶  $S \models_s E$  iff  $\theta(E) \subseteq S$  for some blank node substitution  $\theta$ .

### Example

$$S = \{ \langle \#john, \text{hasName}, "John\ Smith" \rangle, \langle \#john, \text{marriedTo}, \_ : u \rangle, \\ \langle \_ : y, \text{marriedTo}, \#john \rangle \}$$
$$E = \{ \langle \#john, \text{hasName}, "John\ Smith" \rangle, \langle \_ : y, \text{marriedTo}, \#john \rangle \}$$
$$S \models_s E$$

## Simple entailment (continued)

### Example

$$S = \{ \langle \#john, \text{hasName}, "John\ Smith" \rangle, \langle \#john, \text{marriedTo}, \_ :u \rangle, \langle \_ :y, \text{marriedTo}, \#john \rangle \}$$
$$E = \{ \langle \#john, \text{hasName}, \_ :w \rangle, \langle \_ :x, \text{marriedTo}, \_ :z \rangle, \langle \_ :y, \text{marriedTo}, \_ :x \rangle \}$$
$$S \models_s E$$

- ▶ Observe:  $S$  is an instance of  $E$

## Simple entailment (continued)

### Example

$$S = \{ \langle \#john, \text{hasName}, \_ :w \rangle, \langle \_ :x, \text{marriedTo}, \_ :z \rangle, \langle \_ :y, \text{marriedTo}, \_ :x \rangle \}$$
$$E = \{ \langle \#john, \text{hasName}, "John Smith" \rangle, \langle \#john, \text{marriedTo}, \_ :u \rangle, \\ \langle \_ :y, \text{marriedTo}, \#john \rangle \}$$
$$S \not\vdash_s E$$

## E-entailment regime

Allowing different entailment regimes in SPARQL:

### 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.

- ▶ Simple, RDF, and RDFS entailment ( $\models_s, \models_{RDF}, \models_{RDFS}$ ) are entailment regimes
- ▶ All RDF graphs are well-formed for simple, RDF, and RDFS entailment
- ▶ Extensions (e.g. OWL) may restrict the shape of well-formed graphs

## Terms and Variables

### Definition: RDF Term

let **URI** be the set of all IRIs (URIs).

let **Lit** be the set of all RDF Literals

let **BNod** be the set of all blank nodes in RDF graphs

The set of **RDF Terms**, **RDF-T**, is  $URI \cup Lit \cup BNod$ .

### Definition: Query Variable

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

## Triple Pattern, Basic Graph Pattern

### Definition: Triple Pattern

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

$$(RDF-T \cup V) \times (URI \cup V) \times (RDF-T \cup V)$$

### 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*.

## Graph Equivalence

Assuming the **simple RDF entailment** regime.

### Definition: Basic Graph Pattern equivalence

Two basic graph patterns  $A$  and  $B$  are **graph-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:

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

In other words:  $A$  and  $B$  are graph-equivalent if  $A \models_s B$  and  $B \models_s A$ .

## Basic Graph Matching

- ▶ Graph matching is cornerstone for query answering
- ▶ Pattern solution (variable substitution) is essentially a query answer, where all variables are selected
- ▶ Given a graph pattern  $BGP$  (query) and a graph  $G$ , then pattern solution  $S$  is a query answering if  $BGP$   $E$ -matches with  $S$  on  $G$

## Basic Graph Pattern Matching

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

1.  $BGP'$  is a basic graph pattern that is graph-equivalent to  $BGP$
2.  $G'$  is an RDF graph that is graph-equivalent to  $G$
3.  $G'$  and  $BGP'$  do not share any blank node labels.
4.  $(G' \cup S(BGP'))$  is a well-formed RDF graph for E-entailment
5.  $G \models_E S(BGP')$

## Extending Entailment in SPARQL

- ▶ Different entailment regime defines **different graph equivalence** relation (1)
- ▶ Different entailment regimes may define **different well-formed RDF graphs** (4)
- ▶ Different entailment regimes may define **different pattern solutions** (5)

## 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  $S(C)$ .

## 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 of  $A$ , but not of  $A$  and  $B$ .

## 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

## 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

# SELECT 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#>
SELECT ?x
WHERE { ?x foaf:name ?name }
```

## Result

```
x
-----
_:a
```

## 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**

## 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" .
```

## Boolean Queries: ASK

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

## 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

# Summary

Entailment Regimes in RDF

Graph Matching

Basic Matching

Extending Basic Matching

Query Result Forms

## Required reading

## Further reading

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