Università
della
Svizzera
italiana

Facoltà
di scienze della
comunicazione

MSc in Communication Sciences 2011-12
Program in Technologies for Human Communication

Davide Eynard

Internet Technology

13 – RDFS and SPARQL

RDF - Summary

- Main characteristics of RDF:
  - Abstract syntax based on *triples* (subj-pred-obj)
  - The data model is a *graph*, instead of a *tree*
  - *Resources* (identified by URIs) vs *literals* (xsd datatypes)
- Building patterns:
  - Blank nodes
  - Reification
- Serializations
  - RDF/XML, N3 family, RDFa
  - *NOTE:* RDF→ser→RDF returns the original RDF, while ser→RDF→ser does not necessarily return the same serialization!

Explicit reification

- Shakespeare wrote Hamlet

```
lit:Shakespeare lit:wrote    lit:Hamlet .

lit:Hamlet      lit:author   lit:Shakespeare .
```

- Shakespeare wrote Hamlet in 1601

```
bio:n1   bio:author          lit:Shakespeare ;
         bio:title           "Hamlet" ;
         bio:publicationDate 1601 .
```

- Wikipedia says that Shakespeare wrote Hamlet

```
q:n1     rdf:subject         lit:Shakespeare ;
         rdf:predicate       lit:wrote ;
         rdf:object          lit:Hamlet .

web:Wikipedia   m:says       q:n1 .
```

RDF Schema

- RDF Schema is a semantic extension of RDF which defines *classes* and *properties* that may be used to describe classes, properties and other resources
- RDFS provides mechanisms for describing groups of related resources and the relationships between these resources
- RDFS vocabulary descriptions are written in RDF
  - Similar to XML Schema wrt XML
- Similarities with OOP
  - No real inheritance, but subclassing (an instance of a subclass is also an instance of the parent class)
  - Nothing similar to method override
- RDF Schema defines *meaning* in terms of possible *inferences*

RDF Schema

- Just as Semantic Web modeling in RDF is about graphs, Semantic Web modeling in the RDF Schema Language (RDFS) is about sets.
- What do we gain by specifying explicitly that something is a set?
  - We gain a description of the meaning of membership in a set
- How can we specify what we mean by set membership?
  - In RDFS, we express meaning through the mechanism of *inference*

RDF Schema at a glance

- Classes
  - rdfs:Resource
  - rdfs:Class
  - rdfs:Literal
  - rdfs:Datatype
  - rdf:XMLLiteral
  - rdf:Property
- Reification vocabulary
  - rdf:Statement
  - rdf:subject
  - rdf:predicate
  - rdf:object
- Properties
  - rdfs:domain
  - rdfs:range
  - rdf:type
  - rdfs:subClassOf
  - rdfs:subPropertyOf
  - rdfs:label
  - rdfs:comment
- Utility properties
  - rdfs:SeeAlso
  - rdfs:isDefinedBy
  - rdf:value

Type and Relationship propagation

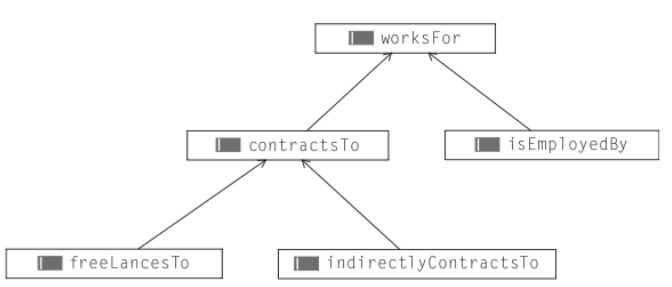
- rdfs:subClassOf
  - If we have triples of the form

```
A rdfs:subClassOf B.
r rdf:type A.
```

then we can infer

```
r rdf:type B.
```

- rdfs:subPropertyOf
  - If we have triples of the form



```
P rdfs:subPropertyOf R.
```

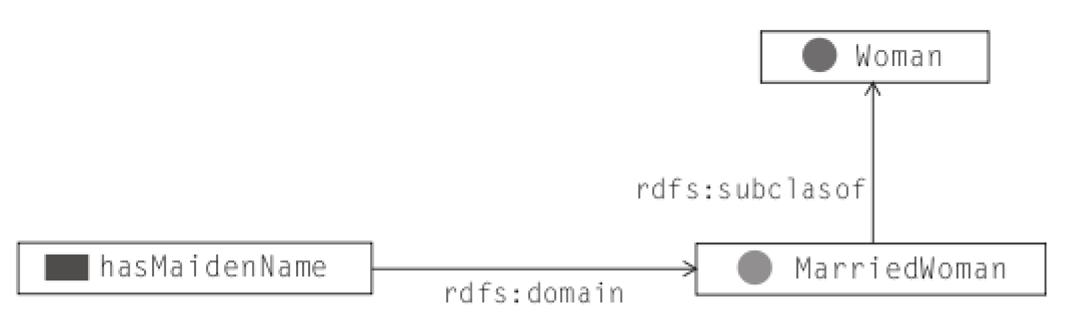
then, if we have the triple “a P b”, we can infer

```
a R b.
```

Typing data

- rdfs:domain

```
IF
P rdfs:domain D .
and
x P y .
THEN
x rdf:type D .
```



- rdfs:range

```
IF
P rdfs:range R .
and
x P y .
THEN
y rdf:type R .
```

RDF validation

- In RDFS, there is no way to assert that a particular individual is not a member of a particular class
  - no notion of an incorrect or inconsistent inference
- Unlike the case of XML Schema, an RDF Schema will never proclaim an input as invalid; it will simply infer appropriate type information

RDFS modeling: intersection

- Set intersection

```
C rdfs:subClassOf A.
C rdfs:subClassOf B.
x rdf:type C.

Then:

x rdf:type B.
x rdf:type A.
```

- Property intersection

```
:lodgedIn rdfs:subPropertyOf :billedFor.
:logdedIn rdfs:subPropertyOf :assignedTo.
:Marcus :lodgedIn :Room101.

Then:

:Marcus :billedFor :Room101.
:Marcus :assignedTo :Room101.
```

RDFS modeling: union

- ## Set union

```
A rdfs:subClassOf C .
B rdfs:subClassOf C .
```

Writing “x rdf:type A .” *or* “x rdf:type B .” implies:

```
x rdf:type C .
```

- ## Property union

```
P rdfs:subPropertyOf R.
Q rdfs:subPropertyOf R.
```

Writing “x P y .” *or* “x Q y .” implies:

```
x R y .
```

Non-modeling properties

- rdfs:label
  - provides a readable/printable name for any resource
- rdfs:seeAlso
  - used for cross-referencing
- rdfs:isDefinedBy
  - provides a link to the primary source of information about a resource. This allows modelers to specify where the definitional description of a resource can be found
  - rdfs:subPropertyOf of rdfs:seeAlso.
- rdfs:comment
  - model documentation

SPARQL by example

- SPARQL is a *recursive acronym* that stands for SPARQL Protocol And RDF Query Language
- Features:
  - Graph patterns
  - Optional values
  - Matching alternatives
  - Multiple RDF graphs as data sources
  - ORDER BY, DISTINCT, OFFSET, LIMIT
  - Filters on returned values
- Let's try it! Go to http://sparql.org

Our first SPARQL query

- Go to http://www.sparql.org/query.html and type the following:

```
PREFIX books:   <http://example.org/book/>
PREFIX dc:      <http://purl.org/dc/elements/1.1/>
PREFIX vcard:   <http://www.w3.org/2001/vcard-rdf/3.0#>
SELECT ?s ?p ?o
WHERE
  { ?s ?p ?o }
```

- The meaning is (after a sequence of namespace declarations):
  - for all the triples "subject – predicate – object"
  - show me the subject, the predicate, and the object
- ?s, ?p, and ?o are *variables* that are filled with the results that satisfy your query (in this case, all the triples in the graph)
- Use the *construct* option to create an RDF you can validate and display using the W3C validation service!

SPARQL queries

- Another basic query:

```
PREFIX books:   <http://example.org/book/>
PREFIX dc:      <http://purl.org/dc/elements/1.1/>
SELECT ?book ?title
WHERE
  { ?book dc:title ?title .
    ?book dc:creator "J.K. Rowling"
}
```

- It reads like follows:
  - there is a *?book* whose title is *?title*
  - the same *?book* has a creator which is *"J.K. Rowling"*
  - show me the list of book-title pairs you have (that is, give me all the books by J.K. Rowling)
- NOTE: SPARQL builds a *graph* out of your query and tries to match it with the data inside the KB

SPARQL queries with anonymous nodes

- A more complex query on the books dataset:

```
PREFIX books:   <http://example.org/book/>
PREFIX dc:      <http://purl.org/dc/elements/1.1/>
PREFIX vcard:   <http://www.w3.org/2001/vcard-rdf/3.0#>
SELECT DISTINCT ?name
WHERE
  { ?s dc:creator ?o .
    ?o vcard:N ?n .
    ?n vcard:Given ?name
}
```

- It reads like follows:
  - there is a *?s* whose creator is *?o*
  - this *?o* has a full name which is *?n*
  - this *?n* has a first/given name which is *?name*
  - show me all the distinct *?names* you have

SPARQL queries with *OPTIONAL* clause

```
PREFIX books:   <http://example.org/book/>
PREFIX dc:      <http://purl.org/dc/elements/1.1/>
SELECT ?book ?title
WHERE
  { ?book dc:title ?title .
    ?book dc:creator "J.K. Rowling"
}
```

- What if there are books for which the creator has not been specified? Check the difference:

```
PREFIX books:   <http://example.org/book/>
PREFIX dc:      <http://purl.org/dc/elements/1.1/>
SELECT ?book ?title
WHERE
  { ?book dc:title ?title .
    optional {?book dc:creator "J.K. Rowling"}
}
```

SPARQL queries with anonymous nodes

- A more complex query on the books dataset:

```
PREFIX books:   <http://example.org/book/>
PREFIX dc:      <http://purl.org/dc/elements/1.1/>
PREFIX vcard:   <http://www.w3.org/2001/vcard-rdf/3.0#>
SELECT ?book ?title ?creator
WHERE
  {
    {?book dc:title ?title .
     ?book dc:creator "J.K. Rowling"}
    UNION
    {?book dc:title ?title .
     ?book dc:creator ?creator .
     ?creator vcard:FN "J.K. Rowling"}
}
```

- NOTE: this technique is very useful to merge information coming from different schemas

Changing datasets on SPARQLer

- Three steps:
  - choose "general purpose SPARQL processor"
  - specify the target graph URI (for instance, try with http://rdf.freebase.com/rdf/en.arnold_schwarzenegger)
  - write a query, for instance:

```
PREFIX fb:   <http://rdf.freebase.com/ns/>
select ?film
where{
    ?s fb:film.performance.film ?film
}
```

- NOTE: to get the name of the properties, you can always send an SPO (?s ?p ?o) query!

More advanced queries on Freebase data

- Get the list of movies and the characters:

```
PREFIX fb:   <http://rdf.freebase.com/ns/>
PREFIX rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

select ?filmtitle ?character
where{
    ?film a fb:film.performance .
    ?film fb:film.performance.film ?filmtitle .
    ?film fb:film.performance.character ?character
}
```

- Get the list of related webpages:

```
PREFIX fb:   <http://rdf.freebase.com/ns/>
PREFIX rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

select ?o
where{
    ?s fb:common.webpage.uri ?o
}
```

References

- Some Web references:
  - RDF Primer: http://www.w3.org/TR/REC-rdf-syntax
  - RDF Schema: http://www.w3.org/TR/rdf-schema
  - Dean Allemang, Jim Hendler: “Semantic Web for the Working Ontologist”. http://workingontologist.org
- Tools:
  - W3C RDF Validator: http://www.w3.org/RDF/Validator
  - Morla RDF editor: http://www.morlardf.net