COMP318: Ontologies and Semantic Web

Describing Web Resources in RDF

UNIVERSITY OF LIVERPOOL

Dr Valentina Tamma
Room: Ashton 2.12
Dept of computer science
University of Liverpool
V.Tamma@liverpool.ac.uk

Recap

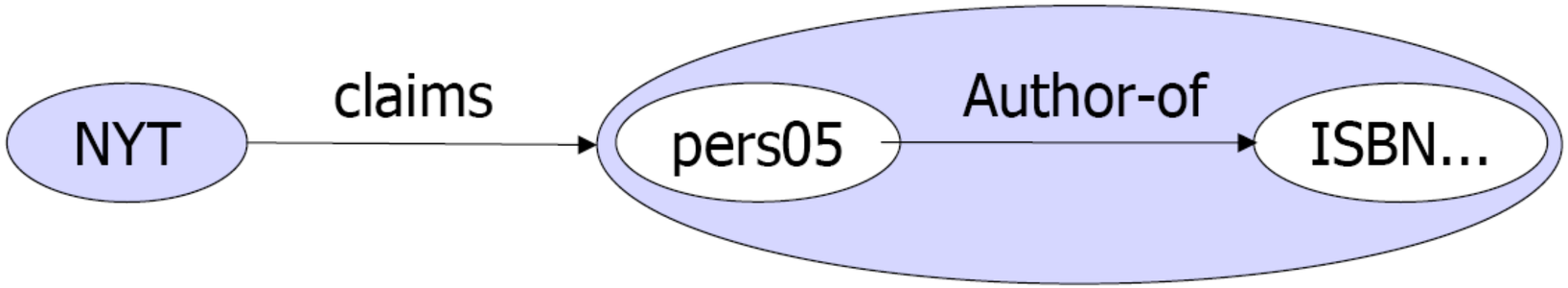
- Limitation of XML
- RDF
  - Basic ideas behind RDF
  - Statements about resources
    - triples
    - graphs
    - XML vocabularies
  - Modelling primitives in RDF
  - Reification

Higher order statements

- RDF allows you to make statements about other RDF statements
  - “Ralph believes that the web contains one billion documents”
- Higher-order statements
  - allow us to express beliefs (and other modalities)
  - are important for trust models, digital signatures,etc.
  - also: metadata about metadata
  - are represented by modelling RDF in RDF itself
- Reification

Reification

- Any RDF statement can be an object
- We must be able to refer to a statement using an identifier
  - allows users to point to a particular statement (and part of a graph)
- RDF allows such reference through a reification mechanism which turns a statement into a resource
  - newer versions of RDF introduce named graphs where an identifier is assigned to a set of statements



Reification Example

```
<rdf:Description rdf:about="#949352">
          <uni:name>John Smith</uni:name>
</rdf:Description>
```

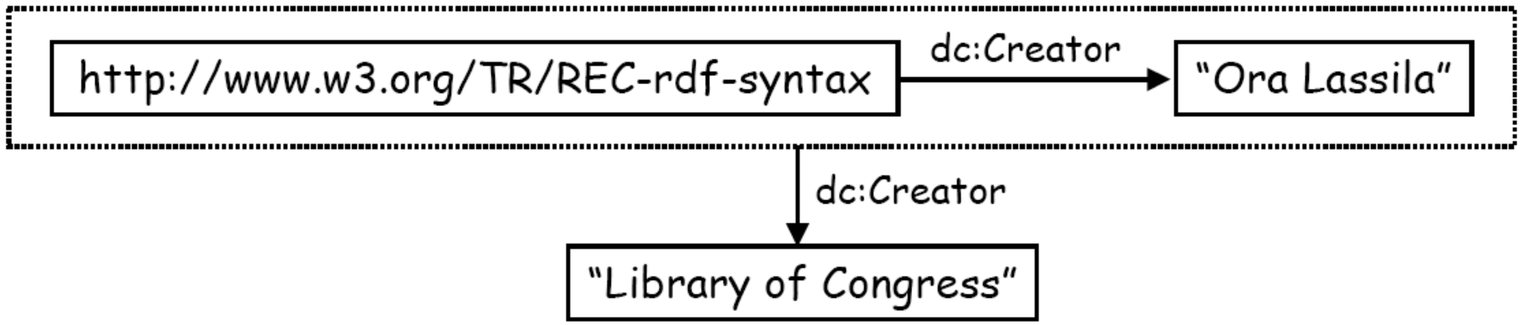
```
<rdf:Statement rdf:ID="StatementAbout949352">
          <rdf:subject rdf:resource="#949352"/>
          <rdf:predicate rdf:resource=
               "http://www.jsmydomain.org/uni-ns#name"/>
          <rdf:object>John Smith</rdf:object>
</rdf:Statement>
```

Reification (2)

- `rdf:subject`, `rdf:predicate` and `rdf:object` allow us to access the parts of a statement
- The ID of the statement can be used to refer to it, as can be done for any description
- We write an `rdf:Description` if we don't want to talk about a statement further
- We write `rdf:Statement` if we wish to further refer to a statement

Reification (3)

- Thus, RDF provides a built-in vocabulary for reification



{ x, rdf:predicate, dc:creator }
{ x, rdf:subject, “http://www.w3.org/TR/RED-rdf-syntax }
{ x, rdf:object, “Ora Lassila” }
{ x, rdf:type, “rdf:statement” }
{ x, dc:Creator, “Library of Congress” }

Representing the RDF language

- RDF document is a collection of triples

```
subject, predicate, object
subject, predicate, object
subject, predicate, object
           ...
```

- subject: URI or bnode
- predicate: URI
- object: URI or bnode or literal

How do we represent these - and share them between applications/users/etc...

RDF Serialisation formats

- RDF has been given a syntax in XML
  - This syntax inherits the benefits of XML
  - Other serialisations of RDF possible:
    - Notation 3 (N3)
      - Syntax for RDF
      - Logical language for RDF
    - N-Quads
      - Superset of N-triples for serialising multiple RDF graphs
    - Turtle
      - Refinement of N3
      - Just RDF representation
    - JSON-LD
      - JSON based serialisation

Terse RDF Triple Language

- Turtle
  - Refinement of N3
  - Just RDF representation
- Plain text syntax for RDF
  - Based on Unicode
  - RDF 1.1 turtle recommendation in 2014
- Mechanisms for namespace abbreviation
  - Allows grouping of triples according to subject
- Shortcuts for collections
- In short:
  - Takes good things of RDF/XML
  - and leaves out angle brackets

Prefixes

- Mechanism for namespace abbreviation
- Syntax:

```
@prefix abbr: <URI>
```

- Example:

```
@prefix rdf:
   <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
```

- Default:

```
@prefix : <URI>
```

- Example:

```
@prefix : <http://example.org/myOntology#>
```

Identifiers in Turtle

- URIs: <URI>

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

- Qnames (Qualified names): namespace-abbr?:localname

```
rdf:type dc:title :hasName
```

- Literals: `"string"(@lang)?(^^type)?`

```
"John" "Hello"@en-GB "1.4"^^xs:decimal
```

- Typed literal shortcuts
  - `integer: 2 45`
  - `decimal: 2.4 5.67`
  - `boolean: true false`

Triples in Turtle

- Simple triple: subject predicate object .
- :john rdf:label “John".
- Grouping triples: subject predicate object ; predicate object ...

```
:john
      rdf:label "John" ;
      rdf:type ex:Person ;
      ex:homePage <http://example.org/johnspage/>.
```

Blank Nodes in Turtle

- **Simple blank node: [ ] or _:x**

```
:john ex:hasFather [] .

:john ex:hasFather _:x .
```

- **Blank node as subject: [predicate object; predicate object ...]**

```
[ ex:hasName "John"] .

[ ex:authorOf :lotr ;

  ex:hasName "Tolkien"] .
```

- **Collections: (object1 ... objectn)**

```
:doc1 ex:hasAuthor
(:john :mary) .
```

Short for

```
:doc1 ex:hasAuthor

[ rdf:first :john;

rdf:rest
[ rdf:first :mary;

rdf:rest rdf:nil ]

] .
```

More on URIs

- URI = Uniform Resource Identifier
  - allow for denoting resources in a general, unambiguous way
- Resource: any object that possesses a clear identity (within the context of a given application)
  - books, cities, humans, publishers, but also relations between those, abstract concepts, etc.
- already realised in some domains:
  - ISBN for books
- URIs do not need to correspond to an actual location
  - but it is good practice if they do
    - a picture, a FOAF description, a map…

Syntax of URIs

- Builds on concept of URLs but...
  - not every URI refers to a Web document
  - however, often the URL of a document is used as its URI
- URI starts with so-called URI schema separated from the following part by ":”
  - e.g, http, ftp, mailto
    - but starting with http does not necessarily mean http-accessible...
- mostly hierarchically organised

Self-defined URIs?

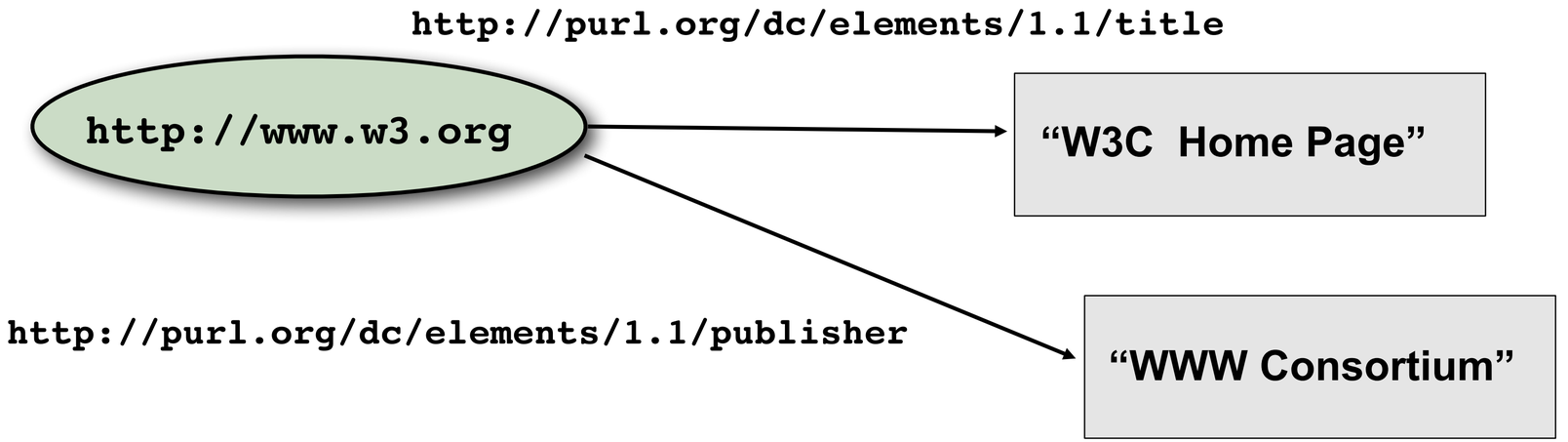
- Necessary if no URI exists (yet) for a resource
  - or it is not known
- strategy for avoiding unwanted clashes:
  - use http URIs of webspace you control
  - AND provide some documentation about the URI
- Need to distinguish the URI of a resource from URI of the associated documents describing it:
  - Example: URI for “Lord of the Ring”

```
http://www.wikipedia.org/wiki/LordOfTheRing#URI
```

```
http://www.wikipedia.org/wiki/LordOfTheRing
```

Literals

- Literals represent data values
  - denoted as string
  - interpreted via assigned datatype
    - literals without explicitly associated datatype are treated like strings

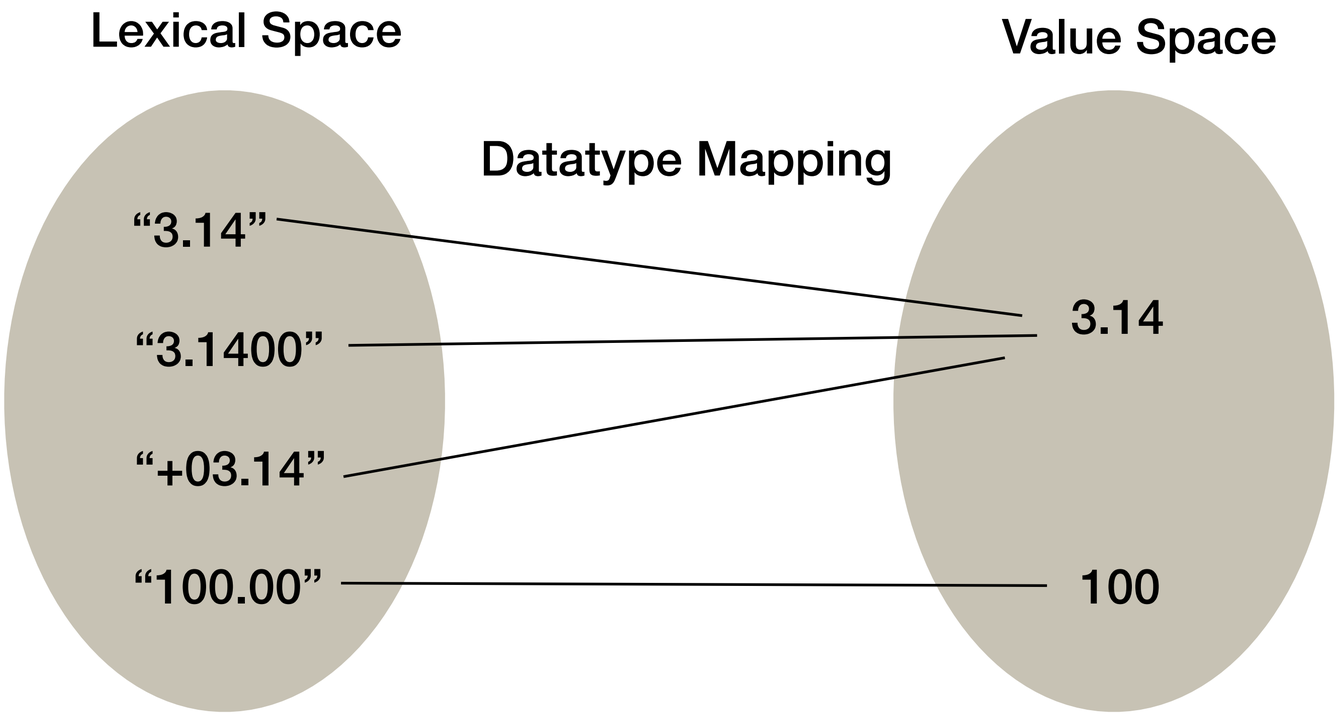


Datatypes in RDF

- Without datatypes literals are untyped, interpreted as strings
  - e.g. "`02`”, ”`2`”, ”`2.0`” all different
- typing literals with datatypes allows for more adequate treatment of values
  - semantic is clearer
- datatypes denoted by URIs and can be freely chosen
  - frequently: xsd datatypes from XML
    - syntax of typed literal: `"datavalue"^^datatype-URI`
- `rdf:XMLLiteral` is the only datatype that is part of the RDF standard
  - denotes arbitrary balanced XML “snippets”

Datatypes in RDF

- Example: `xsd:decimal`



"3.14"="+03.14" holds for `xsd:decimal` but not for `xsd:string`

RDF Vocabulary

- RDF defines a number of resources and properties
  - We have already seen: rdf:XMLLiteral, rdf:type, ...
  - RDF vocabulary is defined in the namespace:
  - http://www.w3.org/1999/02/22-rdf-syntax-ns#

- Classes:
  - `rdf:Property rdf:Statement rdf:XMLLiteral rdf:Seq rdf:Bag`
  - `rdf:Alt rdf:List`

- Properties:
  - `rdf:type rdf:subject rdf:predicate rdf:object rdf:first rdf:rest`
  - `rdf: n rdf:value`

- Resources:
  - `rdf:nil`

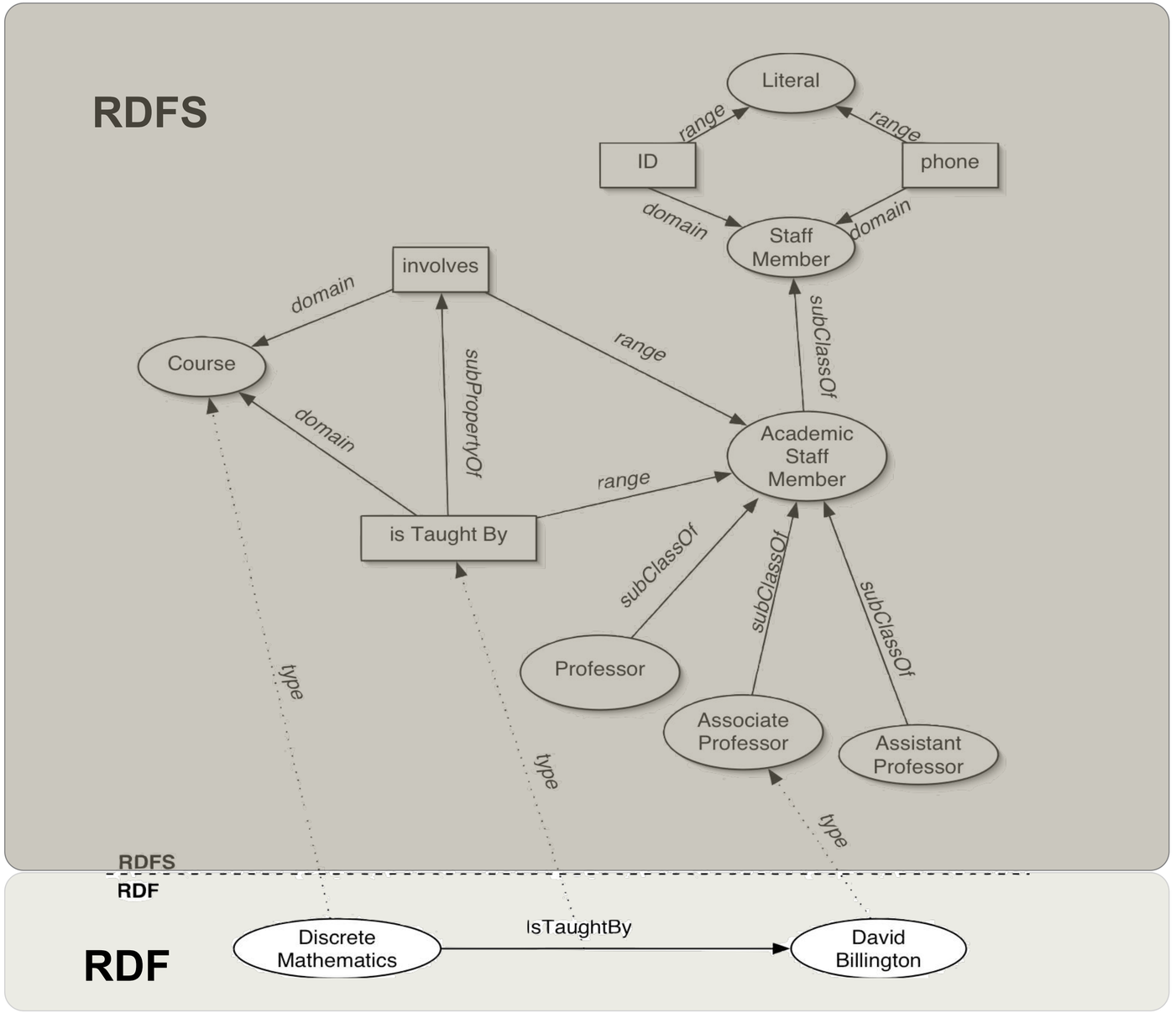
RDF Vocabulary Description Language

- Types in RDF:
  - ⟨#john,rdf:type,#Student⟩
- Definition of what is “#Student” ⇒ A language for defining types in RDF:
  - Define classes:
    - “#Student is a class”
  - Relationships between classes:
    - “#Student is a sub-class of #Person”
  - Properties of classes:
    - “#Person has a property hasName”
- RDF Schema is such a language

RDF vs RDFS

- RDF language for describing structured information
  - individuals:
    - the book entitled “Lord of the rings”, the author “J.R.R Tolkien”...
  - relations between individuals:
    - The book “Lord of the rings” is authored by “J.R.R Tolkien”
  - types of literals and resources:
    - They belong to class of elements sharing the same characteristics
      - natural numbers, dates, ...

- How do we model classes of individuals?


RDFS
Literal
range
range
ID
phone
domain
domain
Staff Member
involves
domain
subClassOf
range
Course
subPropertyOf
domain
Academic Staff Member
range
is Taught By
subClassOf
subClassOf
subClassOf
Professor
Associate Professor
Assistant Professor
type
type
type
RDFS
RDF
RDF
Discrete Mathematics
IsTaughtBy
David Billington

Recap

- RDF language
  - Turtle syntax
  - URIs
  - Datatypes
- Schema modelling in RDF: RDF Schema