Semantic Web In Depth: Resource Description Framework

Dr Nicholas Gibbins
32/4037
nmg@ecs.soton.ac.uk
RDF syntax(es)

- RDF/XML is the standard syntax
  - Supported by almost all tools

- RDF/N3 (Notation3) is also widely used
  - Non-XML syntax
  - Not a standard
  - Patchy tool support
  - Primarily designed to be easy to write on whiteboards

- Other non-XML syntaxes exist
  - Turtle, NTriples, etc
URIs and URIrefs

- Uniform Resource Identifiers are defined by RFC2396
  - http://example.org/
  - urn:isbn:0198537379
  - mailto:nmg@ecs.soton.ac.uk

- URI references (URIrefs) are URIs with optional fragment identifiers
  - http://example.org/index.html#Introduction
  - http://www.w3.org/1999/02/22-rdf-syntax-ns#type
XML namespaces and qualified names

- RDF uses XML namespaces to refer to elements of domain vocabularies

  `xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"`

  namespace URI prefix

  namespace abbreviation

- Namespaces used to abbreviate URIrefs to qualified names (QNames)

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

  becomes

  `rdf:type`

- QNames cannot be used in attribute values
  - Use the URIref instead
The anatomy of an RDF/XML file

```xml
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:dc="http://purl.org/dc/elements/1.1/">
    <rdf:Description rdf:about="http://www.sciam.com/">
        <dc:title>Scientific American</dc:title>
    </rdf:Description>
</rdf:RDF>
```

http://www.sciam.com/ → Scientific American

http://purl.org/dc/elements/1.1/title
The anatomy of an RDF/XML file

- Resource-valued predicates use the rdf:resource attribute

```xml
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:dc="http://purl.org/dc/elements/1.1/">
  <rdf:Description rdf:about="http://www.example.org/">
    <dc:creator rdf:resource="mailto:john@example.org"/>
  </rdf:Description>
</rdf:RDF>
```

http://www.example.org/ → mailto:john@example.org

http://purl.org/dc/elements/1.1/creator
The anatomy of an RDF/XML file

- We can have multiple rdf:Description elements within an rdf:RDF element

```xml
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/dc/elements/1.1/">
  <rdf:Description rdf:about="http://www.example.org/">
    <dc:title>Example Inc. Homepage</dc:title>
  </rdf:Description>
  <rdf:Description rdf:about="http://www.example.org/">
    <dc:creator rdf:resource="mailto:john@example.org"/>
  </rdf:Description>
</rdf:RDF>
```
The anatomy of an RDF/XML file

- We can have multiple predicates within an rdf:Description element

```xml
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/dc/elements/1.1/">
  <rdf:Description rdf:about="http://www.example.org/">
    <dc:title>Example Inc. Homepage</dc:title>
    <dc:creator rdf:resource="mailto:john@example.org"/>
  </rdf:Description>
</rdf:RDF>
```
The anatomy of an NTriples file

<http://www.sciam.com/>  
<http://purl.org/dc/elements/1.1/title> “Scientific American”.
The anatomy of an Turtle/N3 file

<http://www.example.org>
  <http://purl.org/dc/elements/1.1/creator> <mailto:john@example.org> ;
  <http://purl.org/dc/elements/1.1/title> “Example Inc. Homepage”.

- Allows grouping of triples with common subject
Class membership

- An object’s membership of a class is indicated using the rdf:type property

```xml
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <rdf:Description rdf:about="http://www.example.org/">
    <rdf:type rdf:resource="http://example.org/ontology#Website"/>
  </rdf:Description>
</rdf:RDF>
```
Abbreviated forms – literal predicates

- Replace predicate element with attribute of same name on containing element

```xml
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:dc="http://purl.org/dc/elements/1.1/">
  <rdf:Description rdf:about="http://www.example.org/"
                  dc:title="Example Inc. Homepage">
  </rdf:Description>
</rdf:RDF>
```
Abbreviated forms – class membership

• Replace rdf:Description with QName of class

```xml
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:ex="http://example.org/ontology#">
  <ex:Website rdf:about="http://www.example.org/">
    <rdf:type rdf:about="http://www.example.org/ex:Website"/>
  </ex:Website>
</rdf:RDF>
```
RDF/XML striped syntax

- Consider the following graph:
RDF/XML striped syntax

• Graph could be serialised using two rdf:Description elements

```xml
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:dc="http://purl.org/dc/elements/1.1/"
         xmlns:ex="http://example.org/ontology#">
  <rdf:Description rdf:about="http://www.example.org/">
    <dc:creator rdf:resource="mailto:john@example.org"/>
  </rdf:Description>
  <rdf:Description rdf:about="mailto:john@example.org">
    <ex:name>John Smith</ex:name>
  </rdf:Description>
</rdf:RDF>
```
RDF/XML striped syntax

- Alternatively, the second statement could be inserted within the predicate element of the first

```xml
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:dc="http://purl.org/dc/elements/1.1/"
         xmlns:ex="http://example.org/ontology#">
  <rdf:Description rdf:about="http://www.example.org/">
    <dc:creator>
      <rdf:Description rdf:about="mailto:john@example.org">
        <ex:name>John Smith</ex:name>
      </rdf:Description>
    </dc:creator>
  </rdf:Description>
</rdf:RDF>
```
RDF/XML striped syntax

• The syntax is striped because property and class elements are nested alternately

```xml
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:dc="http://purl.org/dc/elements/1.1/"
         xmlns:ex="http://example.org/ontology#">
  <rdf:Description rdf:about="http://www.example.org/">
    <dc:creator>
      <rdf:Description rdf:about="mailto:john@example.org">
        <ex:name>John Smith</ex:name>
      </rdf:Description>
    </dc:creator>
  </rdf:Description>
</rdf:RDF>
```
Common RDF/XML idioms

- XML entities are defined for the XML namespace URI prefixes

```xml
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
  <!ENTITY rdf 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
  <!ENTITY dc 'http://purl.org/dc/elements/1.1/'>
  <!ENTITY ex 'http://example.org/ontology#'>
]>  
<rdf:RDF xmlns:rdf="&rdf;"
  xmlns:dc="&dc;"
  xmlns:ex="&ex;">```

- Used to abbreviate long URIrefs in attribute values (because QNames can’t be used there)
Common RDF/N3 idioms

- @prefix used to introduce QName abbreviations to N3 and Turtle documents:

  @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
  @prefix dc: <http://purl.org/dc/elements/1.1/> .
  @prefix ex: <http://example.org/ontology#> .

  <http://www.example.org> dc:creator <mailto:john@example.org> ;
  rdf:type ex:Website .
Common RDF idioms

• Assertions about the null URIref are about the RDF file itself

```xml
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:dc="http://purl.org/dc/elements/1.1/">
  <rdf:Description rdf:about=""
    <dc:creator rdf:resource="mailto:nmg@ecs.soton.ac.uk"/>
  </rdf:Description>
</rdf:RDF>
```
Blank nodes (bNodes)

- Sometimes we have resources which we do not wish to identify with a URI
- These are *blank nodes* or *anonymous resources*
Blank nodes (bNodes)

- The striped syntax simplifies the RDF/XML serialisation – remove the rdf:about attribute

```xml
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:ex="http://example.org/ontology#">
    <rdf:Description rdf:about="http://www.example.org/"
        dc:creator>
        <rdf:Description>
            <ex:name>John Smith</ex:name>
        </rdf:Description>
    </dc:creator>
</rdf:RDF>
```
Blank nodes (bNodes)

• The striped syntax is not sufficient to represent all graphs containing blank nodes unambiguously

http://www.example.org/

http://test.example.org/

dc:creator

ex:name

John Smith
Blank nodes (bNodes)

<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:ex="http://example.org/ontology#">
    <rdf:Description rdf:about="http://www.example.org/">
        <dc:creator>
            <rdf:Description>
                <ex:name>John Smith</ex:name>
            </rdf:Description>
        </dc:creator>
    </rdf:Description>
    <rdf:Description rdf:about="http://test.example.org/">
        <dc:creator>
            <rdf:Description>
                <ex:name>John Smith</ex:name>
            </rdf:Description>
        </dc:creator>
    </rdf:Description>
</rdf:RDF>
Blank nodes and node IDs

- Ambiguities resulting from blank nodes are resolved by using *node IDs*
- Node IDs are identifiers which are local to a given serialisation of an RDF graph
- Node IDs are not guaranteed to remain unchanged when an RDF file is parsed and serialised
  - The identifier strings may change
  - The graph structure will remain unchanged
Blank nodes and node IDs

```xml
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
               xmlns:dc="http://purl.org/dc/elements/1.1/"
               xmlns:ex="http://example.org/ontology#">
    <rdf:Description rdf:about="http://www.example.org/">
        <dc:creator rdf:nodeID="foo23"/>
    </rdf:Description>
    <rdf:Description rdf:about="http://test.example.org/">
        <dc:creator rdf:nodeID="foo23"/>
    </rdf:Description>
    <rdf:Description rdf:nodeID="foo23">
        <ex:name>John Smith</ex:name>
    </rdf:Description>
</rdf:RDF>
```
bNodes in N3 and Turtle

\(<\text{http://www.example.org/>}\ dc:\text{creator} [\ ex:\text{name} \text{“John Smith”} ].\)

• Or with nodeIDs:

\(<\text{http://www.example.org/>}\ dc:\text{creator} \_:\text{foo23}.\)
\(<\text{http://test.example.org/>}\ dc:\text{creator} \_:\text{foo23}.\)
\(_\text{foo23} \text{ex:name “John Smith”}.\)
rdf:about versus rdf:ID

- So far, we have used the rdf:about attribute to specify the subjects of triples
  - rdf:about takes a URIref as a value
- rdf:ID can be used to declare a new URIref within a document
  - Within the file http://www.example.org/ontology
    
    ```xml
    <rdf:Description rdf:ID="JohnSmith"> 
    ```
    declares a new URIref http://www.example.org/ontology#JohnSmith
    - Analogous to the name and id attributes in HTML
    - Relative to xml:base attribute
Datatypes

- Literal values presented so far are plain and do not have a type
  - Many applications need to be able to distinguish between different typed literals

- RDF uses XML Schema datatypes

```xml
<rdf:Description rdf:about="http://www.example.org/">
</rdf:Description>
```
Multilingual support

• In addition to typed literals, RDF also provides support for language annotations on literals

• RDF uses XML’s multilingual support

```xml
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:dc="http://purl.org/dc/elements/1.1/">
  <rdf:Description rdf:about="http://www.example.org/foreword">
    <dc:title xml:lang="en">Foreword</dc:title>
  </rdf:Description>
</rdf:RDF>
```

• Languages identified by ISO369 two letter codes
Containers

• RDF provides means for describing groups of objects
• Membership in the group is denoted by the ordinal properties rdf:_1, rdf:_2, etc
Containers

• Three types of container are available in RDF
  – rdf:Bag – an unordered group, possibly with duplicates
  – rdf:Seq – an ordered group
  – rdf:Alt – a group of alternatives (translations, media types, etc)
Containers

- Special syntax for expressing collections
  - rdf:li is a convenience element which is replaced with ordinal elements by RDF parsers

```xml
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:ex="http://example.org/ontology#">
  <rdf:Description rdf:about="http://www.example.org/">
    <ex:members rdf:about="http://www.example.org/">
      <rdf:Bag>
        <rdf:li rdf:resource="mailto:john@example.org"/>
        <rdf:li rdf:resource="mailto:bill@example.org"/>
        <rdf:li rdf:resource="mailto:sally@example.org"/>
      </rdf:Bag>
    </ex:members>
  </rdf:Description>
</rdf:RDF>
```
Collections

- Collections are a different way of expressing ordered groups in RDF
  - Containers are mutable – a third party could add new members to a container
  - Collections are immutable – cannot be altered without rendering the collection ill-formed

- Similar to cons/car/cdr lists in Lisp
Collections

RDF and RDF Schema
Collections

- As before, special syntax for expressing collections
  - `rdf:parseType` indicates special parse rules for an element

```xml
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:ex="http://example.org/ontology#">
  <rdf:Description rdf:about="http://www.example.org/">
    <ex:members rdf:parseType="Collection">
      <rdf:Description rdf:about="mailto:john@example.org"/>
      <rdf:Description rdf:about="mailto:bill@example.org"/>
      <rdf:Description rdf:about="mailto:sally@example.org"/>
    </ex:members>
  </rdf:Description>
</rdf:RDF>
```
RDF Status

- Original version published in 1999
- Working group (RDF Core) formed in April 2001
- Revised version published in early 2004
RDF references

• RDF homepage at W3C
  – http://www.w3.org/RDF/
• RDF Core homepage
  – http://www.w3.org/2001/sw/RDFCore/
• RDF/N3 Primer
  – http://www.w3.org/2000/10/swap/Primer.html
• XML Schema Part 2: Datatypes
  – http://www.w3.org/TR/xmlschema-2/
Semantic Web in Depth:
RDF Schema
Using RDF to define RDFS

- RDFS is an RDF vocabulary which contains:
  - Classes for defining classes and properties
  - Properties for defining basic characteristics of classes and properties
    - Global property domains and ranges
  - Some ancillary properties
    - Defined by, see also
RDF Schema class definitions

• We wish to define the class Person

```
<rdf:Description rdf:about="#Person">
  <rdf:type rdf:about="&rdfs;Class"/>
</rdf:Description>

<rdfs:Class rdf:about="#Person"/>
```
RDF Schema class definitions

- Employee is a subclass of Person

```
<rdfs:Class rdf:about="#Employee">
  <rdfs:subClassOf rdf:resource="#Person"/>
</rdfs:Class>
```
RDF Schema class semantics

- `rdfs:subClassOf` is transitive
  - \((A \text{ rdfs:subClassOf } B) \text{ and } (B \text{ rdfs:subClassOf } C) \text{ implies } (A \text{ rdfs:subClassOf } C)\)
RDF Schema class semantics

- rdfs:subClassOf is reflexive
  - All classes are subclasses of themselves
RDF Schema class semantics

- rdf:type distributes over rdfs:subClassOf
  - (A rdfs:subClassOf B) and (C rdf:type A) implies (C rdf:type B)
RDF Schema property definitions

- We wish to define the property worksFor

```xml
<rdf:Description rdf:about="#worksFor">
  <rdf:type rdf:resource="&rdf;Property"/>
</rdf:Description>

<rdf:Property rdf:about="#worksFor"/>
```
RDF Schema property definitions

- Important difference between RDF and object oriented programming languages
  - OO languages define classes in terms of the properties they have
  - RDF defines properties in terms of the classes whose instances they relate to each other

- The *domain* of a property is the class that the property runs *from*
- The *range* of a property is the class that a property runs *to*
RDF Schema property definitions

- The property `worksFor` relates objects of class `Employee` to objects of class `Company`

```xml
<rdf:Property rdf:about="#worksFor">
  <rdfs:domain rdf:resource="#Employee"/>
  <rdfs:range rdf:resource="#Company"/>
</rdf:Property>
```
RDF Schema property definitions

- Specialisation exists in properties as well as classes
  - worksFor is a subproperty of affiliatedTo

```
<rdf:Property rdf:about="#worksFor">
  <rdfs:subPropertyOf rdf:resource="#affiliatedTo"/>
</rdf:Property>
```
RDF Schema property semantics

- `rdfs:subPropertyOf` is transitive and reflexive
- Entailment of superproperties
RDF Schema property semantics

- Type entailments from range and domain constraints
RDF Schema predefined classes

- rdfs:Class
- rdf:Property (note different namespace)
- rdfs:Resource
- rdfs:Literal
- rdfs:Datatype
- rdf:XMLLiteral
RDF Schema predefined classes

- `rdfs:Resource`
  - `rdfs:subClassOf` to `rdfs:Class`
  - `rdfs:subClassOf` to `rdfs:Literal`
  - `rdfs:subClassOf` to `rdf:Property`
- `rdfs:Class`
- `rdfs:Literal`
- `rdf:Property`
- `rdfs:Datatype`
  - `rdfs:subClassOf` to `xsd:string`
  - `rdfs:subClassOf` to `xsd:integer`
- `xsd:string`
- `xsd:integer`
- `rdf:XMLLiteral`

RDF and RDF Schema
RDF Schema ancillary features

• rdfs:label is used to give a human-readable name for a resource

```xml
<rdfs:Class rdf:about="#Employee">
  <rdfs:comment>A person who works.</rdfs:comment>
</rdfs:Class>
```

• rdfs:comment is used to give a human-readable description for a resource

```xml
<rdfs:Class rdf:about="#Employee">
  <rdfs:comment>A person who works.</rdfs:comment>
</rdfs:Class>
```
RDF Schema ancillary features

- `rdfs:seeAlso` is used to indicate a resource which can be retrieved to give more information about something

- `rdfs:isDefinedBy` indicates a resource which is responsible for the definition of something
  - A subproperty of `rdfs:seeAlso`
RDF Schema Status

- Original version contemporary with RDF
- Revised version published in early 2004