The Resource Description Framework (RDF)

Antoine Zimmermann
École des Mines de Saint-Étienne
http://www.emse.fr/~zimmermann/
antoine.zimmermann@emse.fr

RDF is…

- … to the Semantic Web what HTML is to the WWW
- … simple: everything is *just* triples
- … a *data model*: it is **not** a file format!
- … a *logical formalism*: it has a **formal semantics**
- … more than XML or JSON: XML and JSON have a *tree-based* model, RDF has a *graph-based* model
- … a Web standard: a W3C recommendation

A *lingua franca* for the Semantic Web

- Like HTML for documents, there can be many models that would achieve a Web of Data and a Semantic Web
- PDF documents can be linked and visualised in Web browsers…
- …but HTML makes it easy to read and write documents (whereas one cannot edit PDFs in a text editor)
- Design decisions govern HTML and RDF
- The decisions may not be the best for your application, but they fix a common norm

RDF basics

- The core features of RDF are:
  - **Identify** things
  - Express **relations** between things
- Additional important features are:
  - Assign **data values** to things
  - Organise things in **categories** (i.e., classes or types)
  - Add **simple knowledge** about categories and relations
- There are other, less important features (discussed later)

Identify things

- RDF is used to **describe resources**
- A **resource** may be anything (a real or imaginary entity, abstract or concrete)
- To describe a resource, it must be named or identified
- On the Web, the identification mechanism must be **uniform at Web scale**: an identifier must identify the same thing **everywhere** on the Web
- RDF uses *Internationalized Resource Identifiers* or IRIs (RFC 3987)

Internationalized Resource Identifiers

- IRIs generalise URIs (*Uniform Resource Identifiers*, RFC 3986) by allowing any UNICODE characters
- IRIs and URIs **identify** things but *may* be used as **locators** (i.e., as URLs) at the same time
- Examples:
  - `urn:ietf:rfc:3987`
  - `svn://yadiyada.foo.bar/`
  - `mailto:antoine.zimmermann@emse.fr`
  - `ftp://ftp.liris.fr/#meta`
  - `http://en.wikipedia.org/wiki/User:Wikiuser100`
  - `http://w3id.org/people/az/me`
- *Note*: to shorten notations, we use namespace **prefixes**, e.g., `rdf:` is for `http://www.w3.org/1999/02/22-rdf-syntax-ns#`

How to choose an IRI for something?

- If possible, reuse an existing IRI from an authoritative source, e.g.:
  - from a national library for books (library of congress, BNF, BNL, DNB)
  - from a government website for a ministry
- If not, make your own IRI:
  - use HTTP IRIs
  - use a namespace under your control
  - Cool URIs don't change
  - Refer to the guide on Cool URIs for the Semantic Web

Relate things

- RDF can express **binary relations** between things, such as “Laura **loves** Helmut”, “Steven **works for** Google Inc.”, “EMSE **is located in** Saint-Étienne”
- This is written as a triple:

  (*subject*, *predicate*, *object*)
- where subject and object identify the resources in the relationship, and predicate identifies the relation
- The predicate in an RDF triple is always an IRI

Example:

(`http://example.org/data/Laura`, ← subject
`http://social.relations.com/loves`, ← predicate
`http://example.org/data/Helmut`) ← object

RDF triples

- Compact syntax:
  - use namespace prefixes
  - write subject predicate object side by side, separated by spaces
- E.g.: `ex:Laura  rel:loves  ex:Helmut`
- Graph representation of a triple:

Data values

- As everything else, a data value (number, string, date) is a resource
- A specific data value can be identified with a **literal**, a character string that represents the value
- Every literal is typed such that its string representation can be interpreted as the correct value
- E.g., "42" represents the number **fourty two** if this is of type *decimal integer*, but represents **sixty six** if it is an *hexadecimal integer*

RDF literals

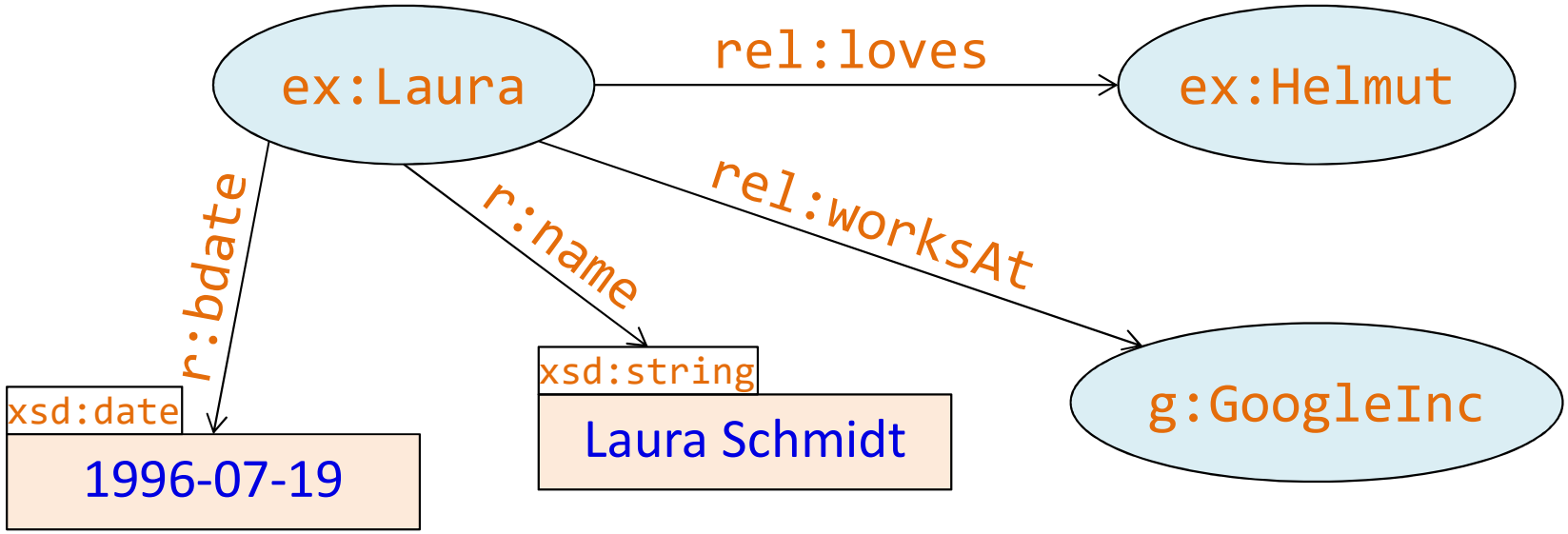
- An **RDF literal** has 2 or 3 components which are:
  - A **lexical form** which is a UNICODE string
  - A **datatype IRI** that can be any IRI
  - When the datatype IRI is `rdf:langString`, there is a **language tag** which is a BCP 47 tag
- Usually, we use standard datatype IRIs from the `xsd:` namespace (XML Schema Datatypes) and the `rdf:` namespace
- We will write literals `"lexical form"^^datatypeIRI` and when it is an `rdf:langString`, `"lexical form"@langTag`

Examples:

- `"42"^^xsd:integer`
- `"THX 1138"^^xsd:string`
- `"chat"@fr`, `"chat"@en`
- `"<p>The <em>beautiful</em> literal!</p>"^^rdf:HTML`

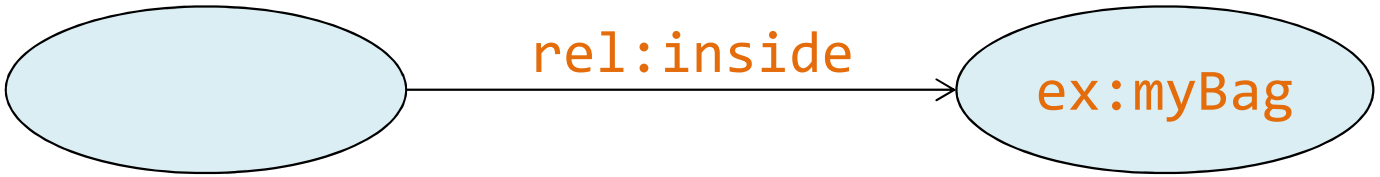
RDF graphs

- An **RDF graph** is a set of RDF triples
- RDF graphs can be drawn as directed, edge-labelled multi-graphs



Unidentified resources

- RDF can describe entities that are known to **exist** but whose identity is unknown (or is irrelevant/unimportant)
- E.g., a book has at least an author, but they may not be known
- The existence of a thing can be indicated in the subject or object position of a triple with a **blank node**
- E.g. “something is in my bag”



Putting RDF graphs into files

- An RDF graph (a set of RDF triples) can be **serialised** to multiple syntaxes (similarly to tabular data that can be written to CSV, Excel, HTML tables, SQL…)
- In this course, we will mostly use Turtle
- There is an XML-based syntax: RDF/XML
- There is a JSON-based syntax: JSON-LD
- There is an easy to parse, line-based triple syntax: N-Triples
- There is a syntax to embed RDF in HTML and XML documents: RDFa
- Let us learn to read and write Turtle in an online editor. Go to:

  https://www.emse.fr/~zimmermann/Teaching/SemWeb/

The Turtle syntax (1)

- Full IRIs:

```
<http://www.example.com/test#this>
```

- A simple triple:

```
<http://www.example.com/test#this>
      <http://relations.example.com/in>
             <http://www.example.com/test#box> .
```

- Abbreviated IRIs (declare prefixes at the beginning of the file):

```
# This is a comment
@prefix ex: <http://www.example.com/test#> . # end dot!
@prefix rel: <http://relations.example.com/> .
ex:this  rel:in  ex:box . # Another comment
```

The Turtle syntax (2)

- Literals:

```
ex:this rel:date "2019-09-13"^^xsd:date . # normal literal
ex:this rel:name "this"@en . # language-tagged literal
ex:this rel:code "TX32" . # xsd:string can be omitted
ex:this rel:number 42 . # xsd:integer (no quotes)
ex:this rel:sizeInMeters 3.75 . # xsd:decimal (use a dot)
```

- Blank nodes:

```
[]  rel:in  ex:box .
_:b1  rel:in  ex:box . # a blank node identifier…
ex:me  rel:likes  _:b1 . # …allows to reuse the same blank
node
```

The Turtle syntax (3)

- Repeat the same subject and predicate:

```
ex:box  rel:contains  ex:this .
ex:box  rel:contains  ex:that .
# can be written
ex:box  rel:contains  ex:this, ex:that . # comma
```

- Repeat subject:

```
ex:this rel:date "2019-09-13"^^xsd:date;
        rel:name "this"@en; # new lines are optional
        rel:code "TX32";
        rel:nextTo  ex:that, ex:thoot, ex:thus .
```

RDF vocabularies

- An **RDF vocabulary** is a set of IRIs
- Some vocabularies have an agreed and shared meaning
- Well known, shared vocabularies should be **reused**
- Some vocabularies are **standardised**, e.g., the RDF Schema vocabulary, PROV-O, Org, DCAT, Data Cube
- RDF vocabularies can be defined and described in RDF

The RDFS vocabulary (1)

- The RDFS voc. can be used to **organise** things in categories (**classes**), **declare** a resource as a class or as a **property** (properties are used as predicates), create **class hierarchies** and **property hierarchies**, declare the types of things used as subject or object with a given property, and more…

The RDFS vocabulary (2)

- A basic vocabulary for defining vocabularies
  - **`rdf:type`** (relates an instance to one of its classes)
    `ex:me rdf:type foaf:Person .`
  - **`rdf:Property`** (the class of all properties)
    `foaf:name rdf:type rdf:Property .`
  - **`rdfs:Class`** (the class of all classes)
    `foaf:Person rdf:type rdfs:Class .`
  - **`rdfs:Resource`** (the class of everything)
    `rdfs:Resource rdf:type rdfs:Resource .`
  - **`rdfs:Datatype`** (the class of data types)
    `xsd:integer rdf:type rdfs:Datatype .`

The RDFS vocabulary (3)

- **`rdfs:subClassOf`** (relates a class to one of its super classes)
  `foaf:Person rdfs:subClassOf foaf:Agent .`
- **`rdfs:subPropertyOf`** (relates a property to one of its super properties)
  `foaf:skypeID rdfs:subPropertyOf foaf:nick .`
- **`rdfs:domain`** (relates a property to a class of things it is about)
  `foaf:firstName rdfs:domain foaf:Person .`
- **`rdfs:range`** (relates a property to a class of things it relates to)
  `foaf:homepage rdfs:range foaf:Document .`

The RDFS vocabulary (4)

- **rdfs:label** (a human-readable name of a resource)
  ```
  ex:emse  rdfs:label  "Mines Saint-Étienne"@fr .
  ```
- **rdfs:comment** (description or commentary)
  ```
  ex:emse  rdfs:label  "Mines Saint-Étienne was
  founded in 1816, blablabla"@en .
  ```
- **rdfs:seeAlso** (pointer to any relevant resource)
  ```
  ex:emse  rdfs:seeAlso  <http://www.emse.fr/> .
  ```
- **rdfs:isDefinedBy** (usually for classes or properties of a vocabulary)
  ```
  foaf:homepage  rdfs:isDeclaredBy  foaf: .
  ```

RDF is a logic

- An RDF graph can be seen as a **logical formula**
- RDF has a **formal semantics** defining a notion of interpretation, of satisfaction, entailment, inference, deduction, consistency, etc.
- Some implicit triples may logically follow from an RDF graph
- We will talk more about this later

More info

- RDF 1.1 Primer (introductory document that you MUST read)
- RDF 1.1 Concepts and Abstract Syntax (the core specification)
- RDF Schema 1.1 (the RDFS vocabulary specification)
- RDF 1.1 Semantics (the formal semantics of RDF and of the RDFS vocabulary)