The RDF Schema vocabulary

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

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

Defining vocabularies

- There are IRIs that identify **generic things**:
  - Types / Classes
  - Properties
- These are likely to be useful in **many** applications
  - Reuse existing terms (Linked Data principle #4)
  - How to find the existing terms?
  - How to define new terms that will be used by many?

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:type

- "Paul is a person"

```
ex:paul  rdf:type  ex:Person .
```

- "Product number 87876R5 is a laptop"

```
product:87876R5  rdf:type  ex:Laptop .
```

- "*X* was employed by *Y* between 2010 and 2013"

```
a:e2010-2013  rdf:type  ex:Employment .
```

Note: Abbreviated to the keyword `a` in Turtle

```
ex:paul  a  ex:Person .
```

rdf:Property

- “People know other people”

```
foaf:knows  a  rdf:Property .
ex:humphrey  foaf:knows  ex:helen .
```

- “Products are sold for a price (in an offer)”

```
prop:hasOffer  a  rdf:Property .
prop:hasPrice  a  rdf:Property .
ex:87876R5  prop:hasOffer  ex:o878676R5-2019-10-04 .
ex:o878676R5-2019-10-04  prop:hasPrice  199.90 .
```

- “People are employed by companies for a time”

```
p:hasEmployment  a  rdf:Property .
p:starts  a  rdf:Property .
p:by  a  rdf:Property .
ex:leonard  p:hasEmployment [
        p:starts "2019-09-13"^^xsd:date;
        p:by  dbr:Google .
] .
```

rdfs:Class

- People, products, employments, etc.

```
foaf:Person  a  rdfs:Class .
voc:Product  a  rdfs:Class .
voc:Employment  a  rdfs:Class .
```

- rdf:Property and rdfs:Class are classes:

```
rdf:Property  a  rdfs:Class .
rdfs:Class  a  rdfs:Class .
```

- Other special classes:

```
rdfs:Resource  a  rdfs:Class . # everything
rdfs:Literal  a  rdfs:Class . # all literal values
rdfs:Datatype  a  rdfs:Class . # all datatypes
```

rdfs:subClassOf

- “People are agents”

```
foaf:Person  rdfs:subClassOf  foaf:Agent .
```

- “PDF files are documents and digital artifacts”

```
ex:PDFFile  rdfs:subClassOf
                    foaf:Document, ex:DigitalArtifact .
```

- “Presidents are roles”

```
voc:Presidency  rdfs:subClassOf  voc:Role .
ex:obama2009-2017  a  voc:Presidency;
        voc:playedBy  dbr:Barrack_Obama;
        p:starts "2009-01-20"^^xsd:date; ;
        p:ends "2017-01-20"^^xsd:date .
```

rdfs:subPropertyOf

- “We know our friends”

```
ex:hasFriend  rdfs:subPropertyOf  foaf:knows .
```

- “Legal guardians know their wards and have legal authority on them”

```
ex:isLegalGuardianOf  rdfs:subPropertyOf
            foaf:knows, ex:hasLegalAuthorityOn .
```

rdfs:domain and rdfs:range

- "People know people"

```
foaf:knows  rdfs:domain  foaf:Person;
            rdfs:range  foaf:Person .
```

- "An event starts at a date"

```
ex:startsAt  rdfs:range  xsd:date .
```

- "A sponsor is a person or organisation"

```
ex:hasSponsor  rdfs:range  ex:PersonOrOrganisation .
```

- Note: the following means "a sponsor is a person **and** an organisation":

```
ex:hasSponsor  rdfs:range
                  ex:Person, ex:Organisation .
```

`rdf:List` and the list vocabulary

- *"Linked Data: Structured Data on the Web* is authored by D. Wood, M. Zaidman, L. Ruth, and M. Hausenblas"

```
isbn:9781617290398  ex:authorList  _:l1 .
_:l1   rdf:first  wood:david;
       rdf:rest  _:l2 .
_:l2   rdf:first  marsha:zaidman;
       rdf:rest  _:l3 .
_:l3   rdf:first  ruth:luke;
       rdf:rest  _:l4 .
_:l4   rdf:first  hausenblas:michael;
       rdf:rest  rdf:nil .
```

- In Turtle

```
isbn:9781617290398  ex:authorList
  (wood:david marsha:zaidman ruth:luke hausenblas:michael) .
```

Other things

- Datatypes: `rdf:langString`, `rdf:HTML`, `rdf:XMLLiteral`
- Container vocabulary: `rdfs:Container`, `rdf:Bag`, `rdf:Seq`, `rdf:Alt`, `rdfs:ContainerMembershipProperty`, `rdfs:member`, `rdf:_1`, `rdf:_2`, …
- Reification vocabulary: `rdf:Statement`, `rdf:subject`, `rdf:predicate`, `rdf:object`
- Other: `rdf:value`

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

Some inferences with RDFS semantics (1)

- **Given:**

```
ex:C  rdfs:subClassOf  ex:D .
ex:D  rdfs:subClassOf  ex:E .
```

It can be proved that:

```
ex:C  rdfs:subClassOf  ex:E .
```

- **Given:**

```
ex:p  rdfs:subPropertyOf  ex:q .
ex:q  rdfs:subPropertyOf  ex:r .
```

It can be proved that:

```
ex:p  rdfs:subPropertyOf  ex:r .
```

- **Given:**

```
ex:C  rdfs:subClassOf  ex:D .
ex:x  rdf:type  ex:C .
```

It can be proved that:

```
ex:x  rdf:type  ex:D .
```

Some inferences with RDFS semantics (2)

- **Given:**
```
ex:x  ex:p  ex:y .
ex:p  rdfs:subPropertyOf  ex:q .
```
It can be proved that:
```
ex:x  ex:q  ex:y .
```
- **Given:**
```
ex:p  rdfs:domain  ex:C .
ex:x  ex:p  ex:y .
```
It can be proved that:
```
ex:x  rdf:type  ex:C .
```
- **Given:**
```
ex:q  rdfs:range  ex:D .
ex:a  ex:q  ex:b .
```
It can be proved that:
```
ex:b  rdf:type  ex:D .
```

Finding existing vocabularies

- Reuse well known vocabularies (Schema.org, Dublin Core, FOAF, SIOC, Good Relations, SKOS, voiD, etc.)
- Try an ontology / vocabulary search engine or repository:
  - *Search engines*: FalconS, SWSE, Sindice, OU's Watson, Swoogle, vocab.cc (*most are dead now…*)
  - *Repositories*: Linked Open Vocabulary, ScheWeb, Schemapedia, Cupboard, Knoodl, Ontology Design Patterns, prefix.cc, DERI vocabularies, OWL Seek, SchemaCache (*some are dead…*)
- Ask mailing lists, forums (semantic-web@w3.org, public-lod@w3.org, stackoverflow.com)

Build your own vocabulary

- Editors:
  - **Protégé**, WebProtégé, NeOn TK, SWOOP, Neologism, TopBraid Composer, Vitro, Knoodl, Ontofly, Altova OWL editor, PoolParty, IBM integrated development TK, Anzo for Excel, Euler GUI
- Learn, evaluate:
  - **Protégé tutorial**, …bits and pieces here and there
  - RDF validator, OWL validator, Linked Data validator, Data Hub LOD Validator
  - Best practices for publishing RDF vocabularies
- Link to other ontologies

OWL: The Web Ontology Language

- OWL mostly focuses on descriptions of **classes** and **properties**
- Differentiates **literal values** from **other things**
- Differentiates **properties with literal values** (owl:DatatypeProperty) and **other relations between things** (owl:ObjectProperty)
- Strongly separates instances, classes, and properties
- It abstracts away from graphs, expressing descriptions of classes and properties with a logical formalism called **Description Logics**

OWL Class Features

OWL can express, among other things:

- The class of **everything** (owl:Thing) and the **empty** class (owl:Nothing)
- The **union** of classes (LivingBeing is the union of classes Prokaryote and Eukaryote) and the **intersection** of classes (Bat are in the intersection of FlyingAnimal and Mammal)
- **Disjointness** of classes (LivingBeing is disjoint from Mineral)
- The **existence** of a property for members of a class (all people have parents)
- The **cardinality** of a property (all people have 2 biological parents)
- **Typing the value** of a property for a class (parents of people are people, while parents of cats are cats, etc.)
- ...more things

OWL Property Features

OWL can express, among other things:

- The **universal property** (owl:topObjectProperty and owl:topDatatypeProperty) and the **empty property** class (owl:bottomObjectProperty and owl:bottomDatatypeProperty)
- **Inverse** properties (childOf is the inverse of parentOf)
- **Disjointness** of properties (brotherOf disjoint with partOf)
- **Functional** properties (hasBiologicalFather), **inverse functional** properties (social security number)
- **Transitive** properties (ancestorOf), **reflexive** properties (knows - assuming people know themselves), **symmetric**, **assymmetric**, **irreflexive** properties (childOf)
- **Property chains** (uncleOf is a chain of brotherOf followed by parentOf relations)

OWL Instance Features & other things

OWL can express, among other things:

- The **identity** of two names (owl:sameAs - The Morning Star is Venus)
- Difference of two things (owl:differentFrom - Donald Trump is not Joe Biden)
- **Negative** assertions (Macron is not president of the USA)
- Datatype union, intersection, restriction
- …

More info

- RDF Schema 1.1 (the RDFS vocabulary specification)
- RDF 1.1 Semantics (the formal semantics of RDF and of the RDFS vocabulary)
- OWL 2 Overview (link to all OWL 2 specifications)