INF3580 – Semantic Technology – Spring 2010

Lecture 2: RDF, The Resource Description Framework

Audun Stolpe

2nd February 2010

Department of Informatics

University of Oslo

Today's Plan

1. The RDF data model
2. Semantic Web architecture
3. Nodes and edges closer up
4. Merging graphs
5. The Turtle syntax

My lectures

Date	Topic	Keywords
02.02.2010	The RDF data model	Graphs, vocabularies
23.02.2010	Semantics	Models, entailment
02.03.2010	The RDFS language	Taxonomies, design patterns
09.03.2010	OWL Basics	Open worlds, more patterns
23.03.2010	Rules	Expressive limitations of OWL

Outline

The conceptual components of RDF

The RDF datamodel in a nutshell:

The conceptual components of RDF

The RDF datamodel in a nutshell:

- Information is encoded in *triples*.

The conceptual components of RDF

The RDF datamodel in a nutshell:

- Information is encoded in *triples*.
  - Triples = subject-predicate-object patterns

The conceptual components of RDF

The RDF datamodel in a nutshell:

- Information is encoded in *triples*.
  - Triples = subject-predicate-object patterns
- Things (in a broad sense) are labelled with *URIs*

The conceptual components of RDF

The RDF datamodel in a nutshell:

- Information is encoded in *triples*.
  - Triples = subject-predicate-object patterns
- Things (in a broad sense) are labelled with *URIs*
  - URIs act as globally valid names

The conceptual components of RDF

The RDF datamodel in a nutshell:

- Information is encoded in *triples*.
  - Triples = subject-predicate-object patterns
- Things (in a broad sense) are labelled with *URIs*
  - URIs act as globally valid names
- Sets of names are organized in *vocabularies*

The conceptual components of RDF

The RDF datamodel in a nutshell:

- Information is encoded in *triples*.
  - Triples = subject-predicate-object patterns
- Things (in a broad sense) are labelled with *URIs*
  - URIs act as globally valid names
- Sets of names are organized in *vocabularies*
  - Vocabularies are demarcated by *namespaces*

The conceptual components of RDF

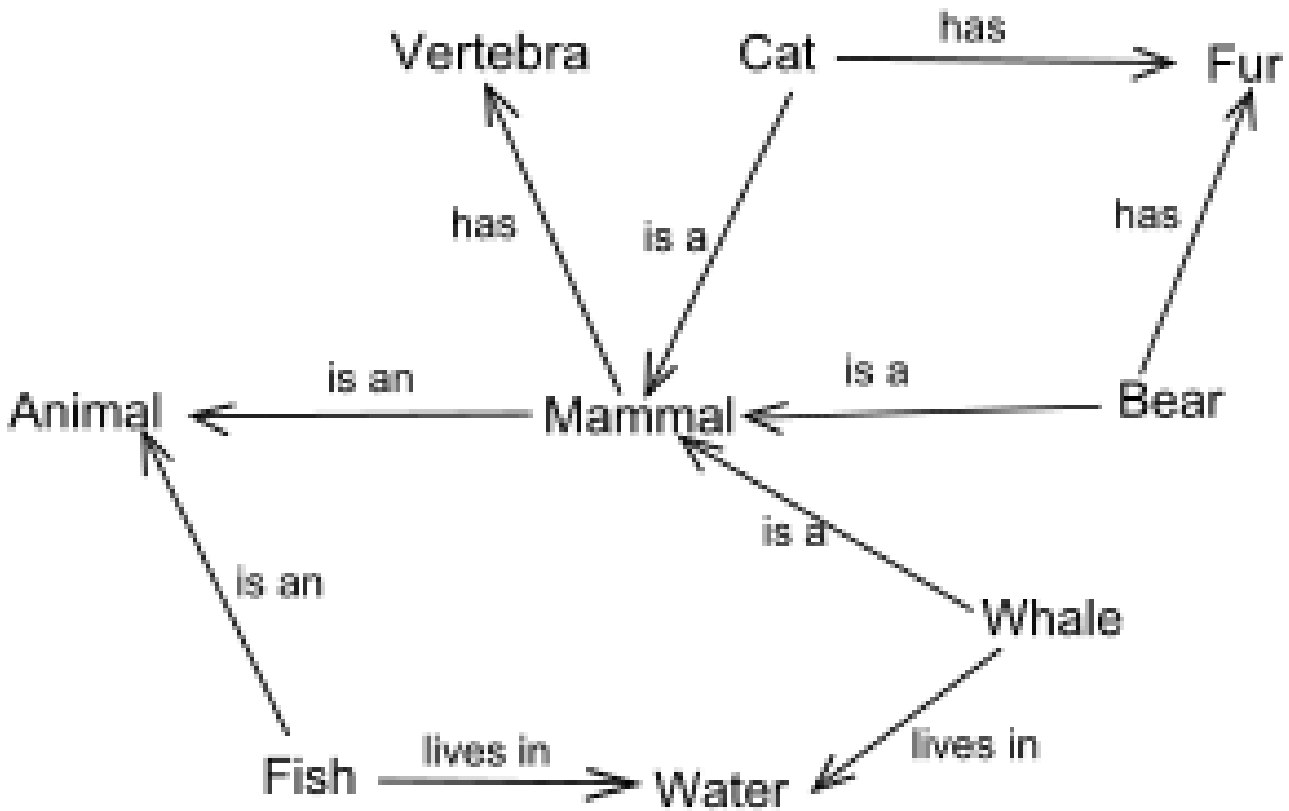
The RDF datamodel in a nutshell:

- Information is encoded in *triples*.
  - Triples = subject-predicate-object patterns
- Things (in a broad sense) are labelled with *URIs*
  - URIs act as globally valid names
- Sets of names are organized in *vocabularies*
  - Vocabularies are demarcated by *namespaces*

We shall look at each in turn.

The triple as the least common denominator of relational data

Graphs are suitable for encoding meaning:



Yonder days

Encoding meaning in graphs has a long history. Examples include:

- Charles S. Peirce's system of existential graphs (see Roberts 1973)
  - Same expressive power as first order logic.
- The psycholingustic semantic networks of Collins and Quillian (1969)
  - Modelled human information storage and management
- Networks for machine translation (Masterman 1961)
  - One of the first computer implementations of networks.
- The conceptual graphs of John Sowa (1984)
  - Used to represent the conceptual schemas used in database systems.

All examples of *associationist* theories of meaning.

RDF, essential 'abouts':

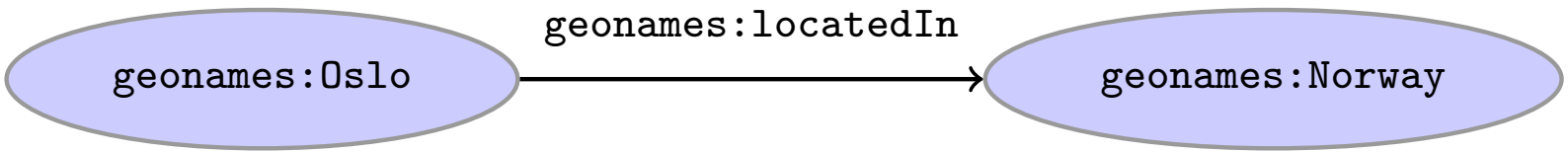
- The *Resource Description Framework* was initially intended for annotation of web-accessible resources (1999).
- It has since developed into a general purpose language for describing structured information—on the web or elsewhere.
- The goal of RDF is to enable applications to exchange data on the Web in a meaning-preserving way.
- It is considered the basic representation format underlying the Semantic Web.

RDF graphs

Conceptually, RDF graphs are nothing new—they are just descriptions (often in the form of a document) of directed graphs.

- Due to the origin of RDF the nodes are usually called *resources*.
- Edges are called *predicates*, *relations* or *properties*.
- Both nodes and edges are labelled with identifiers.

Example:



Here, `geonames` denotes a *namespace*, whereas `locatedIn` is the name of the relation. More about namespaces shortly.

Features

Typing relations makes semantics explicit:

Features

Typing relations makes semantics explicit:

- Typed relations constitute the fabric of the Semantic Web.

Features

Typing relations makes semantics explicit:

- Typed relations constitute the fabric of the Semantic Web.
- The types expose the semantics of the nodes in the graph.

Features

Typing relations makes semantics explicit:

- Typed relations constitute the fabric of the Semantic Web.
- The types expose the semantics of the nodes in the graph.
- This has the effect of decoupling data from applications.

Features

Typing relations makes semantics explicit:

- Typed relations constitute the fabric of the Semantic Web.
- The types expose the semantics of the nodes in the graph.
- This has the effect of decoupling data from applications.
- In a sense, therefore, the data describes itself.

Features

Typing relations makes semantics explicit:

- Typed relations constitute the fabric of the Semantic Web.
- The types expose the semantics of the nodes in the graph.
- This has the effect of decoupling data from applications.
- In a sense, therefore, the data describes itself.
- This lightens the programming burden.

Features

Typing relations makes semantics explicit:

- Typed relations constitute the fabric of the Semantic Web.
- The types expose the semantics of the nodes in the graph.
- This has the effect of decoupling data from applications.
- In a sense, therefore, the data describes itself.
- This lightens the programming burden.
- Typically therefore, Semantic Web applications are generic and general purpose, whilst data sets are rich and knowledge intensive.

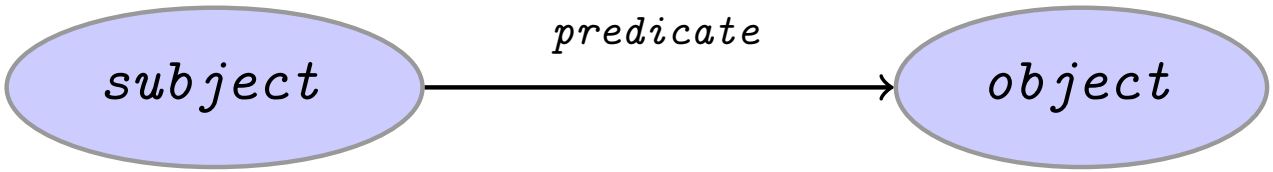
Subjects, predicates and objects

The directedness of RDF graphs gives triples a grammatical form:

We call

- the node from which the relation exits the *subject*,
- the relation itself the *predicate*, and
- the node at the distal end the *object*

of the triple. Example:



Such triples are the *morphemes* of RDF.

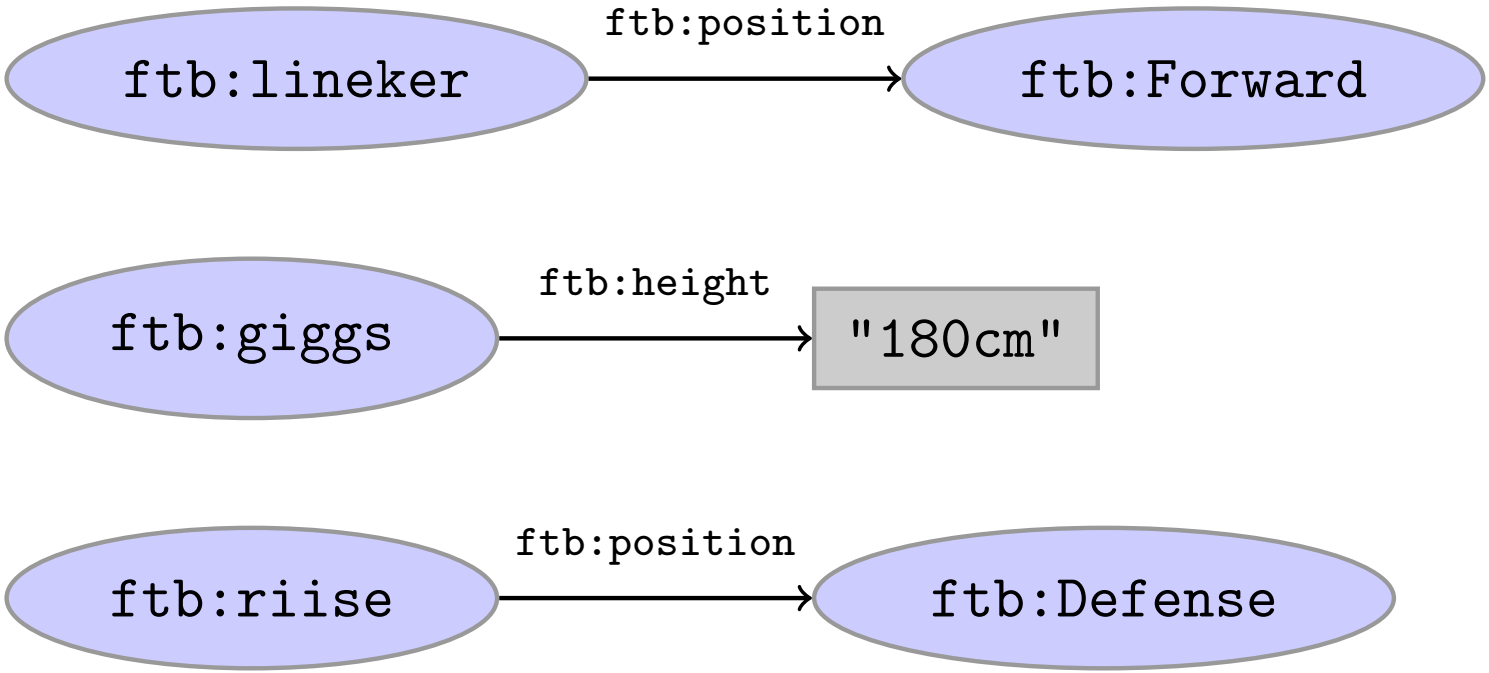
Comparison with tabular data

ID	Name	Nationality	Height	Weight	Position
1	Gary Lineker	ENG	180 cm	70–77 kg	Forward
2	Ryan Giggs	Wales	181 cm	11 stone	Midfielder
3	Alan Shearer	ENG	182 cm	78 kg	Forward
4	Riise	NOR	177 cm	75 kg	Defense

Table: Premier League players.

Each cell can be described with three items of information; ID, column name and cell value.

Table excerpts:



Simple algorithm for porting tabular data to RDF

1. Make each row in the table a resources,
2. Make each column name a predicate
3. Make each cell value a literal,
4. Link each row to each cell value by means of the RDFized column names.

Many so-called RDFizers are based upon this simple procedure:

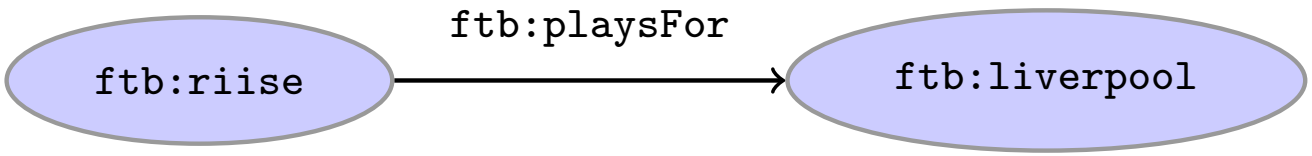
- D2RQ–treats relational databases as virtual RDF graphs
- D2RMAP–database to RDF mapping language and processor.

Live example:

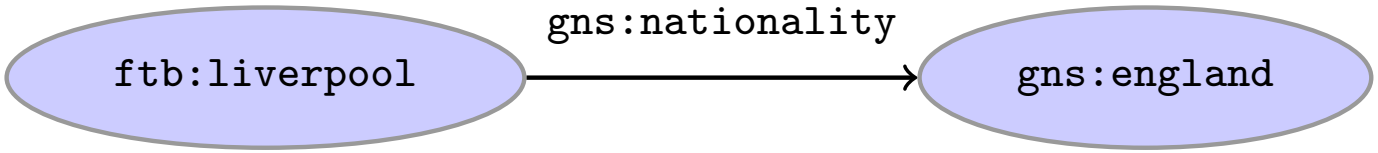
- D2R server publishing the DBLP Bibliography Database

Triples can be chained

Given



and



form:

Traversing such chains constitutes the basis of *inference* and *querying*.

Taking stock so far

Triples ^{w}/typed relations, a powerful way of encoding semantic connections:

- All tabular data can be expressed in the form of triples.
- Triples can be chained together to form larger graphs.
- Chaining corresponds to database joins.
- Relations are *directed*.
- Triples therefore conform to a simple subject-predicate-object grammar.
- Query answering and inferencing becomes graph traversal.

Referring to things—URIs as names

Nameclashes

Two tables from different sources.

ID	Name	Nationality	Height	Weight	Position
1	Gary Lineker	ENG	180 cm	70–77 kg	Forward
2	Ryan Giggs	Wales	181 cm	11 stone	Midfielder
3	Alan Shearer	ENG	182 cm	78 kg	Forward
4	Riise	NOR	177 cm	75 kg	Defense

Table: Premier League players.

ID	Name	Date of birth	Caps	Goals	Position
4	Gary Winston Lineker	30.11.1960	80	48	Centre Forward
5	Alan Shearer	13.09.1970	63	30	Centre Forward
6	Ryan Giggs	29.11.1973	64	12	Left Midfielder
7	Paul John Gascoigne	27.05.1967	57	10	Midfielder

Table: National team football players.

The necessity of qualifying names

Who does `ID 4` name? Riise or Lineker?

- We simply do not know, if the names are not further qualified.
- We need to say 'the player ID 4 *in the database so-and-so*'.
- To be sure that the *so-and-so* suffices, we would want:
  - Absolute names that would never need further qualification
  - That is, the name would refer to the same entity in all contexts.
  - Unclear references would never occur.

Unfortunately that is not possible (contrary to popular belief).

The URI: A good approximation

URIs have attractive properties that make them suitable as names:

- URIs belong to domains that are controlled by its owners.
- "Don't name with your neighbours domain" is easy to remember.
- A URI can resolve to a web document that indicates its meaning.
- Convention tends to fix prominent sets of URIs, e. g.
  - FOAF
  - Dublin Core
  - DBpedia
  - GeoNames
- Naming practices tend to converge by use.

The structure of URIs

The general construction scheme of URIs is

`scheme:[//authority]path[?query][#fragment]`

scheme Classifies the type of URI. Examples are `http`, `mailto`, `file` and `irc`.

authority Typically a domain name.

path Paths are organised hierarchically using / as a separator.

query Optional part typically used for providing parameters to, say, a Web Service.

fragment For parts of documents or resources

URIs are a generalisation of URLs—that is, of web adresses. A URI does not necessarily identify a Web resource.

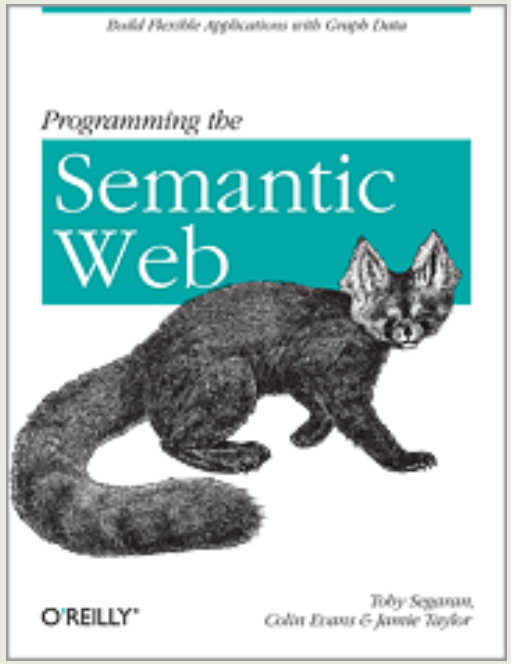
On the negative side

- URIs do not prevent synonymous uses of different names
- URIs do not prevent homonymous uses of the same name
- Any URI can in principle be misused

One sees the following quite frequently:

A common mistake

"Because URIs uniquely identify resources (things in the world), we consider them *strong identifiers*. There is no ambiguity about what they represent, and they always represent the same thing, regardless of the context we find them in."



A good book

This is simply wrong. URIs are not foolproof, but they are sufficiently clear to support a sustainable and stable practice.

Namespaces and vocabularies

Vocabularies

How do we appreciate the situation wrt URIs as names?

- On the one hand, URIs do not eliminate the possibility of ambiguity.
- But consistent naming conventions might.
  - At least for all *practical* purposes,
  - if everyone respects other people's domain names
- A widely adopted solution is therefore to
  - collect the names you need,
  - and keep them safe 'behind' your domain name
- Such a collection of names is usually called an *RDF-vocabulary*.

Vocabularies contd.

A typical example of an RDF-vocabulary is RDF itself:

- It contains local names like `Description`, `type` and `resource`
- which are qualified by prepending the w3c domain (and more) to each:
  - `http://www.w3.org/1999/02/22-rdf-syntax-ns#Description`
  - `http://www.w3.org/1999/02/22-rdf-syntax-ns#type`
  - `http://www.w3.org/1999/02/22-rdf-syntax-ns#resource`

Names in a vocabulary should be given a clearly defined meaning,

- either informally, in the form of a specification document,
- or, as in the case of RDF, formally, by way of a model theoretic semantics.

Excerpt from the RDF vocabulary

Logic

`rdf:Description` `rdf:resource` `rdf:Property` `rdf:type`

Identification/reference

`rdf:about` `rdf:ID` `rdf:nodeID`

Reification

`rdf:Statement` `rdf:subject` `rdf:predicate` `rdf:object`

Collections

`rdf:Seq` `rdf:Alt` `rdf:Bag` `rdf:List`

RDF-vocabulary contd.

Things to note:

- The list on the previous slide is not complete.
- The list may vary from one serialization to the next.
- Some names are used in the XML/RDF format only, e.g. `rdf:Description`
- Only names that belong to all serializations are *conceptually* essential.

Example—XML syntax (optional)

Germany is a country

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:geonames="http://www.geonames.org/ontology/#"
>

  <rdf:Description rdf:about="http://www.geonames.org/ontology/#germany">
    <rdf:type rdf:resource="http://www.geonames.org/ontology/#Country"/>
  </rdf:Description>
</rdf:RDF>
```

Namespaces

A namespace is the common part of the URIs that make up a particular vocabulary:

Namespace/common prefix	Local names
`http://xmlns.com/foaf/0.1/`	`homepage` `skypeId` `.....`

Kinds of namespaces

Most vocabularies you will come across, use either a

- *hash namespace*, or a
- *slash namespace*

depending on whether the character that separates the namespace from the local name is a '#' or a '/':

- FOAF is a slash namespace:
  - `http://xmlns.com/foaf/0.1/Person`
  - `http://xmlns.com/foaf/0.1/maker`
- SKOS (Simple Knowledge Organisation System) is a slash namespace:
  - `http://www.w3.org/2004/02/skos/core#Concept`
  - `http://www.w3.org/2004/02/skos/core#prefLabel`

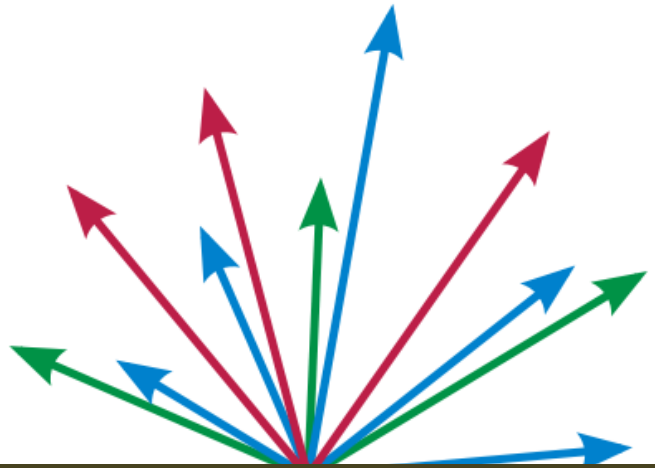
Abbreviating namespaces

Depending on the serialization format, one may declare abbreviations for namespaces. Vocabulary elements may be identified accordingly:

- `foaf:homepage`
- `foaf:skypeId`

Denote respectively:

- `http://xmlns.com/foaf/0.1/homepage`
- `http://xmlns.com/foaf/0.1/skypeId`



Naming is non-trivial

The question of how to design an easy to use, robust namespace is non-trivial, and should not be taken lightly:

Here is some good advice:

- Don't invent new vocabularies if there is already one out there that covers your needs.
- If you do need a new one, adher to a policy described in a 'best practice' document.

For prototyping and documentation, w3c gives you these to play with;

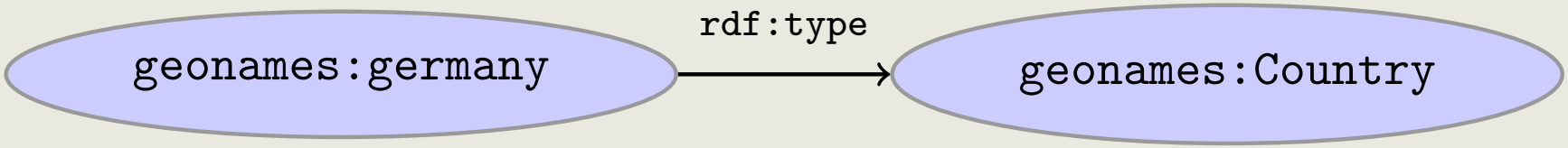
- `http://www.example.com`
- `http://www.example.net`
- `http://www.example.org`

Example in Turtle syntax

Germany is a country

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

geonames:germany rdf:type geonames:Country .
```



Supplementary reading

- Best Practice Recipes for Publishing RDF Vocabularies:
  - `http://www.w3.org/TR/2008/NOTE-swbp-vocab-pub-20080828/`
- Cool URIs for the Semantic Web:
  - `http://www.w3.org/TR/cooluris/`

The RDF data model: Summary

- Semantics is encoded in directed graphs, with typed edges.
- Edges and nodes are identified using URIs.
- Although any arbitrary collection of URIs forms a *vocabulary*, it is more common to reserve the term for URIs that share a common prefix, that is, for URIs that belong to the same namespace.
- Depending on the serialization, namespaces may be abbreviated.
- A domain names are usually the core of a namspace, but namespaces may in turn be divuded into different subspaces

Outline

Locating RDF in the Semantic Web Stack architecture

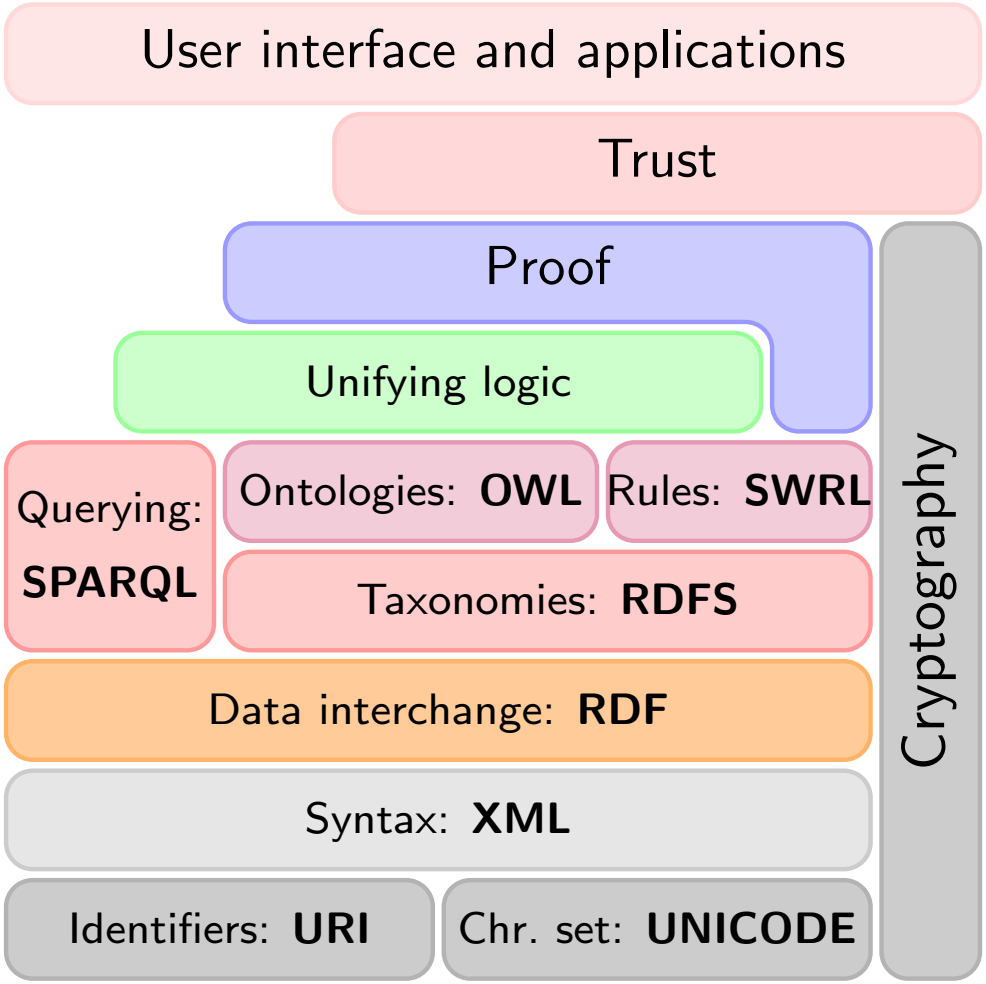


Figure: Semantic Web Stack

Outline

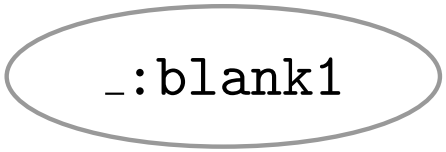
The variety of nodes

In the principal case nodes and edges in RDF are all URIs. However other forms are allowed, notably:

- Literal values for 'raw' data such as numbers and strings
- Blank nodes when the identity of a node is not an issue

They are usually drawn as follows:

`Blank nodes` `Literals`



Blank nodes—what are they for?

We use blank nodes whenever:

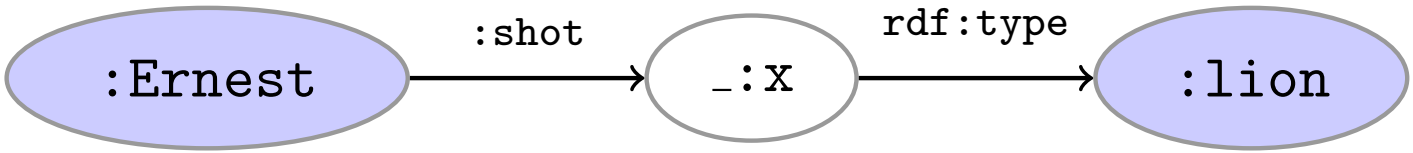
- We wish to group together related objects (instead of using e.g. `rdf:Bag`).
- We need to assert the existence of an object, but do not care about its name.
- We need to represent a many-valued relationships such as e.g. *'x buys y from z'*.

We defer many valued relationships until later

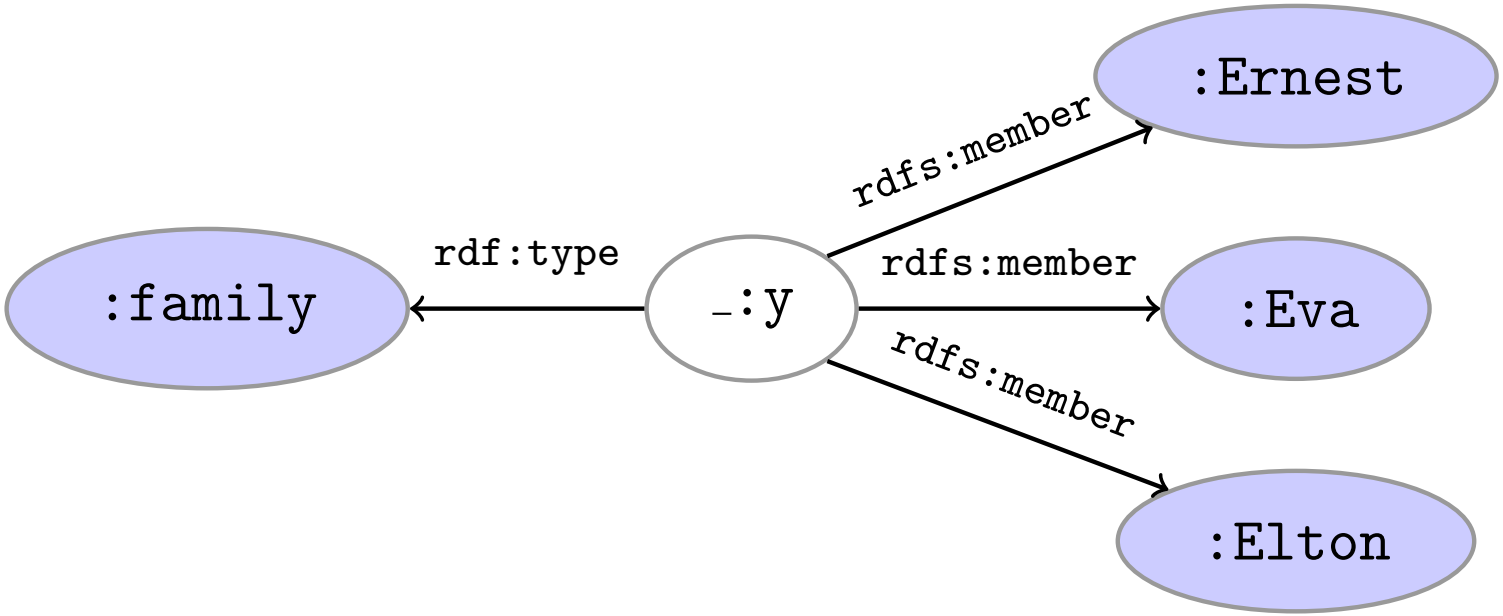
- or consult `http://www.w3.org/TR/swbp-n-aryRelations`.

Blank nodes contd.

Ernest shot a lion:



Ernest, Eva and Elton are members of the same family:



Triple grammar

The rules of triple grammar are simple:

- Only URIs may occur in predicate position
- Literals may only occur in object position
- Blank nodes may occur in subject and object position, but not in predicate position

Capice?

Literal values

Literals in RDF represent data values.

- Untyped literals are always interpreted as strings.
- In general though, a literal value may have either
  - An associated *datatype*, or
  - A *language tag* that specifices the language of the string.

  but not both.

Knowing the datatype of a literal is knowing its meaning; e.g.

- 42 as a date, vs.
- "042" as a string

Literal values contd.

Double carets are often, again depending on the serialization, appended to strings to associate them with datatypes.

```
"2010-01-09"^^xsd:date
```

Whereas language tags are appended in the following manner:

```
"Mothers of invention"@en
```

Literal values in Turtle

It is common to use the XML Schema datatypes, and the ISO 639 language tags:

In turtle syntax:

```
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix dcterms:  <http://purl.org/dc/terms/> .

<http://www.w3.org/TR/rdf-primer>
      dcterms:title    "RDF primer"@en;
      dcterms:issued   "2004-02-10"^^xsd:date .
```

The words `title` and `issued` are taken from the *dublin core vocabulary* http://dublincore.org/.

Literal values in RDF/XML (optional)

In RDF/XML syntax:

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:dcterms="http://purl.org/dc/terms/">

   <rdf:Description rdf:about="http://www.w3.org/TR/rdf-primer">
     <dcterms:title xml:lang="en">
       RDF Primer
     </dcterms:title>
     <dcterms:issued rdf:datatype="http://www.w3.org/2001/XMLSchema#date">
       2004-02-10
     </dcterms:issued>
   </rdf:Description>
</rdf:RDF>
```

Note the use of `xml:lang` and `rdf:datatype` instead of `@` and `^^`.

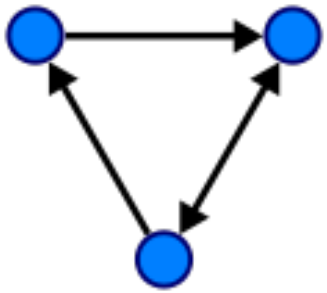
Outline

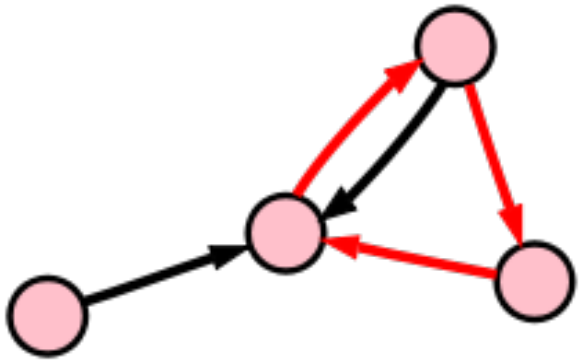
Joining graphs

RDF models are directed graphs (digraphs):

Joining graphs

RDF models are directed graphs (digraphs):

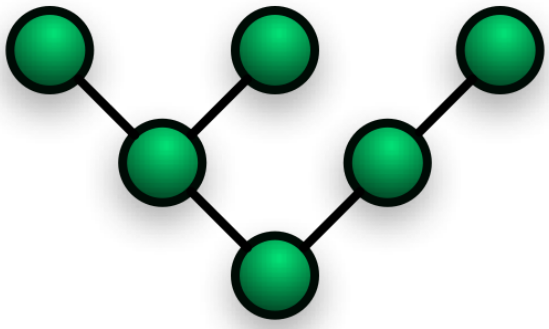


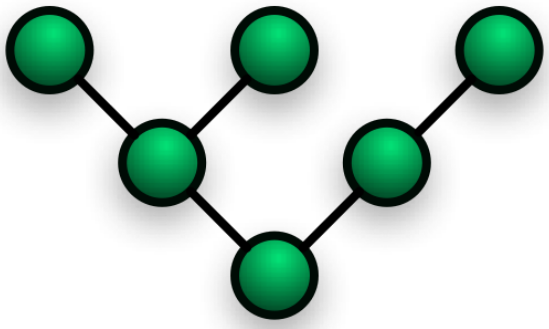


A nice thing about digraphs is that if you add one digraph to another, then you get another digraph.

Merging contd.

This contrasts with trees





Which would lack a common root, and therefore not be a tree (note that this requires special steps to be taken whenever an RDF graph is represented in RDF/XML).

Merging contd.

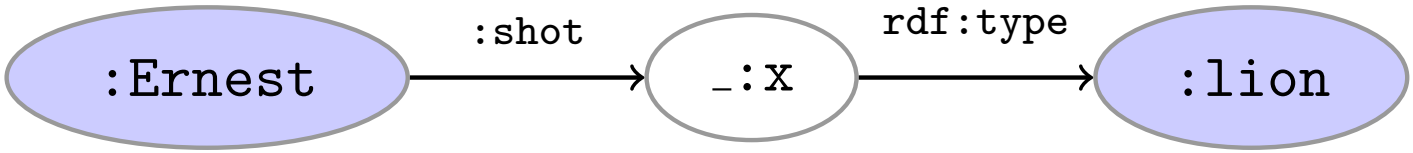
Thus RDF is a data model optimized for sharing and interchange:

- A triple is a digraph,
- A set of triples is a digraph,
- The union of a set of sets of triples is a digraph, and
- URIs ensure that namespaces will usually not overlap.

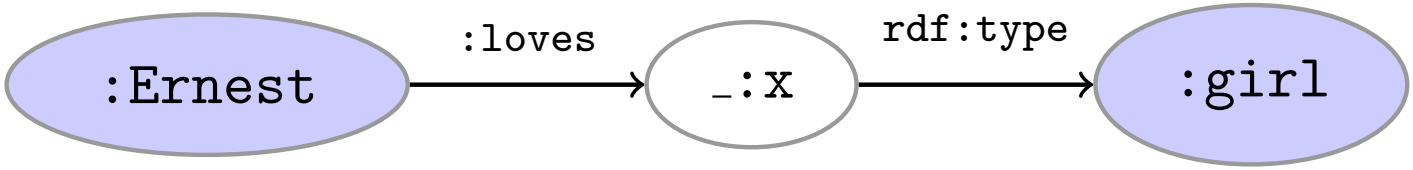
Hence, any number of triples (that is, any graph) can be added to any graph without ever violating the RDF data model.

Blank nodes must be renamed

Ernest shot a lion,

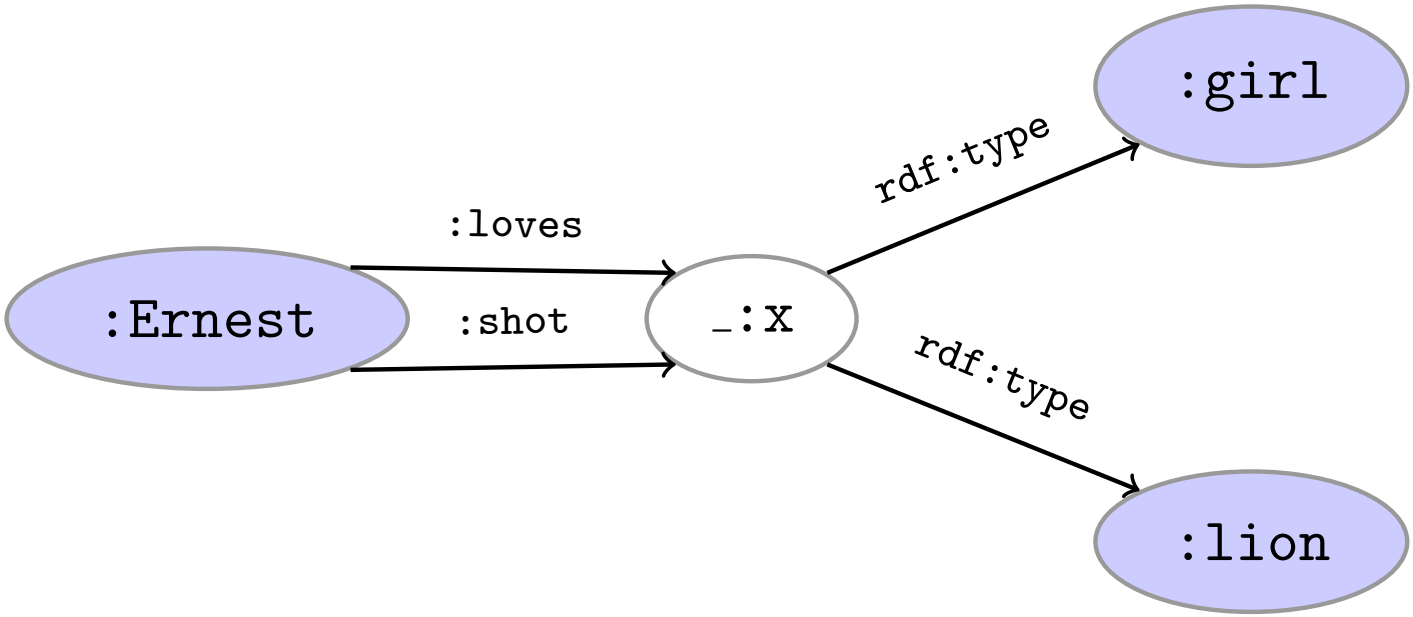


and Ernest loves a girl,



Renaming contd.

But he probably did not shoot a female lion that he loves:



The procedure

Thus, merging becomes a two-step procedure:

1. First rename blank nodes, so that no two blanks have the same id,
2. next, collapse all other nodes with the same id.

The renaming step stems from the semantics of blank nodes, which behave as existentially quantified variables.

Outline

The Turtle syntax

Statements/assertions/triples

Statements are triples terminated by a period:

```
<http://folk.uio.no/>
  <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    <http://xmlns.com/foaf/1.0/Person/> .
```

rdf:type may be abbreviated with 'a':

```
<http://folk.uio.no/audus> a
    <http://purl.org/dc/terms/foaf/1.0/Person/> .
```

Namespaces

Namespace prefixes are declared with @:

```
@prefix foaf: <http://purl.org/dc/terms/foaf/1.0/Person/> .

<http://folk.uio.no/audus> a foaf:Person .
```

A base namespace may be declared:

```
@prefix foaf: <http://xmlns.com/foaf/1.0/Person/> .
@prefix : <http://folk.uio.no/> .

:audus a foaf:Person .
```

Literals

Literal values are enclosed in double quotes:

```
@prefic dcterms: <http://purl.org/dc/terms/> .
@prefix : <http://folk.uio.no/> .

:audus dcterms:created "1974-04-15" .
```

Possibly with type and language information:

```
:audus dcterms:created "1974-04-15"^^xsd:date .
```

Literals

Literal values are enclosed in double quotes:

```
@prefic dcterms: <http://purl.org/dc/terms/> .
@prefix : <http://folk.uio.no/> .

:audus dcterms:created "1974-04-15" .
```

Possibly with type and language information:

```
:audus dcterms:created "1974-04-15"^^xsd:date .
```

Statements with shared subjects

Statements may share a subject with ';'

```
:audus dcterms:created "1974-04-15"^^xsd:date ;
       foaf:firstName "Audun" ;
       foaf:lastName  "Stolpe" .
```

Objects that share subject and predicate may be listed

```
:audus foaf:knows :martingi, :martige, :elian .
```

Blank nodes

Blank nodes are designated with underscores:

Audun knows a person:

```
:audus foaf:knows _:someperson .
_:someperson a foaf:Person.
```

Or, when cross reference is not needed, with []:

Audun knows someone who uses Skype:

```
:audus foaf:knows _:someperson .
_:someperson foaf:skypeId [] .
```

Supplementary reading—W3C specs:

- Concepts and Abstract Syntax:
  - `http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/`
- RDF/XML Syntax Specification:
  - `http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/`
- RDF Semantics:
  - `http://www.w3.org/TR/2004/REC-rdf-mt-20040210/`
- RDF Primer:
  - `http://www.w3.org/TR/2004/REC-rdf-primer-20040210/`