INF3580 – Semantic Technologies – Spring 2010

Lecture 2: The Resource Description Framework

Audun Stolpe

1st February 2011

Department of Informatics

University of Oslo

Today's Plan

Outline

A web of data

The semantic web is a web of *data*, where . . .

A web of data

The semantic web is a web of *data*, where . . .

- websites publish their information in a machine-readable format.

A web of data

The semantic web is a web of *data*, where . . .

- websites publish their information in a machine-readable format.
- the data published by different sources is *linked*

A web of data

The semantic web is a web of *data*, where . . .

- websites publish their information in a machine-readable format.
- the data published by different sources is *linked*
- enough domain knowledge is available to machines to make use of the information

A web of data

The semantic web is a web of *data*, where . . .

- websites publish their information in a machine-readable format.
- the data published by different sources is *linked*
- enough domain knowledge is available to machines to make use of the information
- user-agents can find and combine published information in appropriate ways to answer the user's information needs.

Knowledge representation

The semantic web presupposes

Knowledge representation

The semantic web presupposes

- a simple uniform way to represent knowledge

Knowledge representation

The semantic web presupposes

- a simple uniform way to represent knowledge
- that can be interpreted and exchanged by machines

Knowledge representation

The semantic web presupposes

- a simple uniform way to represent knowledge
- that can be interpreted and exchanged by machines
- with a well enough defined notion of information content

Knowledge representation

The semantic web presupposes

- a simple uniform way to represent knowledge
- that can be interpreted and exchanged by machines
- with a well enough defined notion of information content
- to enable automatic calculation and reasoning over the data

Knowledge representation

The semantic web presupposes

- a simple uniform way to represent knowledge
- that can be interpreted and exchanged by machines
- with a well enough defined notion of information content
- to enable automatic calculation and reasoning over the data

The model of choice for the the aspiring semanticist would be

Knowledge representation

The semantic web presupposes

- a simple uniform way to represent knowledge
- that can be interpreted and exchanged by machines
- with a well enough defined notion of information content
- to enable automatic calculation and reasoning over the data

The model of choice for the the aspiring semanticist would be

- the *Resource Description Framework*

Knowledge representation

The semantic web presupposes

- a simple uniform way to represent knowledge
- that can be interpreted and exchanged by machines
- with a well enough defined notion of information content
- to enable automatic calculation and reasoning over the data

The model of choice for the the aspiring semanticist would be

- the *Resource Description Framework*
- an official W3C recommendation

Knowledge representation

The semantic web presupposes

- a simple uniform way to represent knowledge
- that can be interpreted and exchanged by machines
- with a well enough defined notion of information content
- to enable automatic calculation and reasoning over the data

The model of choice for the the aspiring semanticist would be

- the *Resource Description Framework*
- an official W3C recommendation
- and the foundation for the entire Semantic Web enterprise

Knowledge representation

The semantic web presupposes

- a simple uniform way to represent knowledge
- that can be interpreted and exchanged by machines
- with a well enough defined notion of information content
- to enable automatic calculation and reasoning over the data

The model of choice for the the aspiring semanticist would be

- the *Resource Description Framework*
- an official W3C recommendation
- and the foundation for the entire Semantic Web enterprise
- we shall have a closer look at it today ...

Outline

A brief history of RDF

- Roots in the *Meta Content Framework (MCF)*

A brief history of RDF

- Roots in the *Meta Content Framework (MCF)*
  - 1995-1997: Ramanathan V. Guha develops MCF at Apple

A brief history of RDF

- Roots in the *Meta Content Framework (MCF)*
  - 1995-1997: Ramanathan V. Guha develops MCF at Apple
  - MCF is primarily a format for structuring metadata about *web sites*

A brief history of RDF

- Roots in the *Meta Content Framework (MCF)*
  - 1995-1997: Ramanathan V. Guha develops MCF at Apple
  - MCF is primarily a format for structuring metadata about *web sites*
  - 1997: Guha moves to Netscape, submits “MCF in XML” to the W3C

A brief history of RDF

- Roots in the *Meta Content Framework (MCF)*
  - 1995-1997: Ramanathan V. Guha develops MCF at Apple
  - MCF is primarily a format for structuring metadata about *web sites*
  - 1997: Guha moves to Netscape, submits “MCF in XML” to the W3C
- 1999: W3C recommends the RDF specification and XML syntax

A brief history of RDF

- Roots in the *Meta Content Framework (MCF)*
  - 1995-1997: Ramanathan V. Guha develops MCF at Apple
  - MCF is primarily a format for structuring metadata about *web sites*
  - 1997: Guha moves to Netscape, submits “MCF in XML” to the W3C
- 1999: W3C recommends the RDF specification and XML syntax
  - RDF remains a *metadata-centric* initiative

A brief history of RDF

- Roots in the *Meta Content Framework (MCF)*
  - 1995-1997: Ramanathan V. Guha develops MCF at Apple
  - MCF is primarily a format for structuring metadata about *web sites*
  - 1997: Guha moves to Netscape, submits “MCF in XML” to the W3C
- 1999: W3C recommends the RDF specification and XML syntax
  - RDF remains a *metadata-centric* initiative
- 2004: W3C releases a new version

A brief history of RDF

- Roots in the *Meta Content Framework (MCF)*
  - 1995-1997: Ramanathan V. Guha develops MCF at Apple
  - MCF is primarily a format for structuring metadata about *web sites*
  - 1997: Guha moves to Netscape, submits “MCF in XML” to the W3C
- 1999: W3C recommends the RDF specification and XML syntax
  - RDF remains a *metadata-centric* initiative
- 2004: W3C releases a new version
  - RDF becomes a model for the description of data *in general*

A brief history of RDF

- Roots in the *Meta Content Framework (MCF)*
  - 1995-1997: Ramanathan V. Guha develops MCF at Apple
  - MCF is primarily a format for structuring metadata about *web sites*
  - 1997: Guha moves to Netscape, submits “MCF in XML” to the W3C
- 1999: W3C recommends the RDF specification and XML syntax
  - RDF remains a *metadata-centric* initiative
- 2004: W3C releases a new version
  - RDF becomes a model for the description of data *in general*
  - the idea is born that URIs can be used to stand for *anything*

RDF in the abstract

RDF is essentially

RDF in the abstract

RDF is essentially

- a model for describing relationships between data items

RDF in the abstract

RDF is essentially

- a model for describing relationships between data items
- that is based on

RDF in the abstract

RDF is essentially

- a model for describing relationships between data items
- that is based on
    - a convention for naming things

RDF in the abstract

RDF is essentially

- a model for describing relationships between data items
- that is based on
    - a convention for naming things
    - that exploits the general architecture of the web

RDF in the abstract

RDF is essentially

- a model for describing relationships between data items
- that is based on
    - a convention for naming things
    - that exploits the general architecture of the web
- more specifically,

RDF in the abstract

RDF is essentially

- a model for describing relationships between data items
- that is based on
  - a convention for naming things
  - that exploits the general architecture of the web
- more specifically,
  - pointers to/names for things are URIs (in the principal case)

RDF in the abstract

RDF is essentially

- a model for describing relationships between data items
- that is based on
  - a convention for naming things
  - that exploits the general architecture of the web
- more specifically,
  - pointers to/names for things are URIs (in the principal case)
  - all relations between things are represented by URIs

RDF on the World Wide Web

RDF on the World Wide Web

RDF exists on the internet in mainly two forms:

RDF on the World Wide Web

RDF exists on the internet in mainly two forms:

- As text files

RDF on the World Wide Web

RDF exists on the internet in mainly two forms:

- As text files
  - in one of a variety of *serialization formats* (Turtle, RDF/XML, N3 ..)

RDF on the World Wide Web

RDF exists on the internet in mainly two forms:

- As text files
    - in one of a variety of *serialization formats* (Turtle, RDF/XML, N3 ..)
    - available over standard protocols such as HTTP and FTP

RDF on the World Wide Web

RDF exists on the internet in mainly two forms:

- As text files
    - in one of a variety of *serialization formats* (Turtle, RDF/XML, N3 ..)
    - available over standard protocols such as HTTP and FTP
- or as SPARQL endpoints

RDF on the World Wide Web

RDF exists on the internet in mainly two forms:

- As text files
  - in one of a variety of *serialization formats* (Turtle, RDF/XML, N3 ..)
  - available over standard protocols such as HTTP and FTP
- or as SPARQL endpoints
  - web-oriented data servers (RESTful web services)

RDF on the World Wide Web

RDF exists on the internet in mainly two forms:

- As text files
    - in one of a variety of *serialization formats* (Turtle, RDF/XML, N3 ..)
    - available over standard protocols such as HTTP and FTP
- or as SPARQL endpoints
    - web-oriented data servers (RESTful web services)
    - that use HTTP as query interface

RDF on the World Wide Web

RDF exists on the internet in mainly two forms:

- As text files
  - in one of a variety of *serialization formats* (Turtle, RDF/XML, N3 ..)
  - available over standard protocols such as HTTP and FTP
- or as SPARQL endpoints
  - web-oriented data servers (RESTful web services)
  - that use HTTP as query interface
  - and returns data in several machine readable formats (JSON, XML, ...)

RDF on the World Wide Web

RDF exists on the internet in mainly two forms:

- As text files
    - in one of a variety of *serialization formats* (Turtle, RDF/XML, N3 ..)
    - available over standard protocols such as HTTP and FTP
- or as SPARQL endpoints
    - web-oriented data servers (RESTful web services)
    - that use HTTP as query interface
    - and returns data in several machine readable formats (JSON, XML, ...)

Other options:

RDF on the World Wide Web

RDF exists on the internet in mainly two forms:

- As text files
  - in one of a variety of *serialization formats* (Turtle, RDF/XML, N3 ..)
  - available over standard protocols such as HTTP and FTP
- or as SPARQL endpoints
  - web-oriented data servers (RESTful web services)
  - that use HTTP as query interface
  - and returns data in several machine readable formats (JSON, XML, ...)

Other options:

- RDFa: RDF embedded in (X)HTML documents

RDF on the World Wide Web

RDF exists on the internet in mainly two forms:

- As text files
  - in one of a variety of *serialization formats* (Turtle, RDF/XML, N3 ..)
  - available over standard protocols such as HTTP and FTP
- or as SPARQL endpoints
  - web-oriented data servers (RESTful web services)
  - that use HTTP as query interface
  - and returns data in several machine readable formats (JSON, XML, ...)

Other options:

- RDFa: RDF embedded in (X)HTML documents
- XMP: metadata in PDF files

RDF on the World Wide Web

RDF exists on the internet in mainly two forms:

- As text files
  - in one of a variety of *serialization formats* (Turtle, RDF/XML, N3 ..)
  - available over standard protocols such as HTTP and FTP
- or as SPARQL endpoints
  - web-oriented data servers (RESTful web services)
  - that use HTTP as query interface
  - and returns data in several machine readable formats (JSON, XML, ...)

Other options:

- RDFa: RDF embedded in (X)HTML documents
- XMP: metadata in PDF files

RDF and RESTfulness

RDF makes the web *data centric*

RDF and RESTfulness

RDF makes the web *data centric*

- the semantic web can be queried (SPARQL)

RDF and RESTfulness

RDF makes the web *data centric*

- the semantic web can be queried (SPARQL)
- data can be moved using standard web protocols (HTTP)

RDF and RESTfulness

RDF makes the web *data centric*

- the semantic web can be queried (SPARQL)
- data can be moved using standard web protocols (HTTP)
- data can be linked across servers through the use of URIs

RDF and RESTfulness

RDF makes the web *data centric*

- the semantic web can be queried (SPARQL)
- data can be moved using standard web protocols (HTTP)
- data can be linked across servers through the use of URIs
- data can document itself with dereferenceable URIs

RDF and RESTfulness

RDF makes the web *data centric*

- the semantic web can be queried (SPARQL)
- data can be moved using standard web protocols (HTTP)
- data can be linked across servers through the use of URIs
- data can document itself with dereferenceable URIs
- data can gestalt itself in different ways ...

RDF and RESTfulness

RDF makes the web *data centric*

- the semantic web can be queried (SPARQL)
- data can be moved using standard web protocols (HTTP)
- data can be linked across servers through the use of URIs
- data can document itself with dereferenceable URIs
- data can gestalt itself in different ways ...
- depending on the type of HTTP request (headers and MIME types)

... cont

The main ingredients of this style of architecture is:

… cont

The main ingredients of this style of architecture is:

- resources are referenced with a global identifier

... cont

The main ingredients of this style of architecture is:

- resources are referenced with a global identifier
- servers transfer different *representations* of resources

... cont

The main ingredients of this style of architecture is:

- resources are referenced with a global identifier
- servers transfer different *representations* of resources
- general transfer protocols carry the queries

... cont

The main ingredients of this style of architecture is:

- resources are referenced with a global identifier
- servers transfer different *representations* of resources
- general transfer protocols carry the queries

... cont

The main ingredients of this style of architecture is:

- resources are referenced with a global identifier
- servers transfer different *representations* of resources
- general transfer protocols carry the queries

It is known as a *Representational State Transfer* architecture

... cont

The main ingredients of this style of architecture is:

- resources are referenced with a global identifier
- servers transfer different *representations* of resources
- general transfer protocols carry the queries

It is known as a *Representational State Transfer* architecture

- .. the benefits of which will be revealed to you as we go ...

Try it out!

- There are plenty of large-scale SPARQL enpoints out there already

Try it out!

- There are plenty of large-scale SPARQL enpoints out there already
- for instance DBLP which contains computer science publications

Try it out!

- There are plenty of large-scale SPARQL enpoints out there already
- for instance DBLP which contains computer science publications
- watch RESTfulness in action by tinkering with HTTP headers

Try it out!

- There are plenty of large-scale SPARQL enpoints out there already
- for instance DBLP which contains computer science publications
- watch RESTfulness in action by tinkering with HTTP headers

Request data about Martin Giese's publications

```
wget -O - --header=‘‘Accept:  text/turtle’’
http://dblp.l3s.de/d2r/resource/authors/Martin_Giese
```

Try it out!

- There are plenty of large-scale SPARQL enpoints out there already
- for instance DBLP which contains computer science publications
- watch RESTfulness in action by tinkering with HTTP headers

Request data about Martin Giese's publications

```
wget -O - --header=‘‘Accept: text/turtle’’
http://dblp.l3s.de/d2r/resource/authors/Martin_Giese
```

Request a page displaying a list of Martin Giese's publications

```
wget -O - --header=‘‘Accept: text/html’’ ....
```

Outline

RDF triples

RDF is a data model, *not* a file format, it is

RDF triples

RDF is a data model, *not* a file format, it is

- an abstract *conception of* data or information

RDF triples

RDF is a data model, *not* a file format, it is

- an abstract *conception of* data or information
- to be sure RDF *is* encoded in files

RDF triples

RDF is a data model, *not* a file format, it is

- an abstract *conception of* data or information
- to be sure RDF *is* encoded in files
- yet, RDF is not identical to the format of those files

RDF triples

RDF is a data model, *not* a file format, it is

- an abstract *conception of* data or information
- to be sure RDF *is* encoded in files
- yet, RDF is not identical to the format of those files

In the RDF model a fact is essentially a triple (a, b, c)

RDF triples

RDF is a data model, *not* a file format, it is

- an abstract *conception of* data or information
- to be sure RDF *is* encoded in files
- yet, RDF is not identical to the format of those files

In the RDF model a fact is essentially a triple (a, b, c)

- triples represent subject-predicate-object patterns

RDF triples

RDF is a data model, *not* a file format, it is

- an abstract *conception of* data or information
- to be sure RDF *is* encoded in files
- yet, RDF is not identical to the format of those files

In the RDF model a fact is essentially a triple (a, b, c)

- triples represent subject-predicate-object patterns
- that is, a triple is a way of claiming that two things are related

RDF triples

RDF is a data model, *not* a file format, it is

- an abstract *conception of* data or information
- to be sure RDF *is* encoded in files
- yet, RDF is not identical to the format of those files

In the RDF model a fact is essentially a triple (a, b, c)

- triples represent subject-predicate-object patterns
- that is, a triple is a way of claiming that two things are related
- in the principal case a, b and c are all URIs, but this isn't necessary

RDF triples

RDF is a data model, *not* a file format, it is

- an abstract *conception of* data or information
- to be sure RDF *is* encoded in files
- yet, RDF is not identical to the format of those files

In the RDF model a fact is essentially a triple (a, b, c)

- triples represent subject-predicate-object patterns
- that is, a triple is a way of claiming that two things are related
- in the principal case a, b and c are all URIs, but this isn't necessary
- they are, in any case, commonly referred to as resources

Identifiers, why URIs?

URIs have attractive properties that reduce the risk of name clashes:

Identifiers, why URIs?

URIs have attractive properties that reduce the risk of name clashes:

- URIs belong to domains that are controlled by its owners

Identifiers, why URIs?

URIs have attractive properties that reduce the risk of name clashes:

- URIs belong to domains that are controlled by its owners
- "Keep off others' domains" is an easy-to-remember rule of thumb

Identifiers, why URIs?

URIs have attractive properties that reduce the risk of name clashes:

- URIs belong to domains that are controlled by its owners
- "Keep off others' domains" is an easy-to-remember rule of thumb
- A URI can resolve to a web document that indicates its meaning

Identifiers, why URIs?

URIs have attractive properties that reduce the risk of name clashes:

- URIs belong to domains that are controlled by its owners
- "Keep off others' domains" is an easy-to-remember rule of thumb
- A URI can resolve to a web document that indicates its meaning
- Convention tends to fix prominent sets of URIs, e. g.

Identifiers, why URIs?

URIs have attractive properties that reduce the risk of name clashes:

- URIs belong to domains that are controlled by its owners
- "Keep off others' domains" is an easy-to-remember rule of thumb
- A URI can resolve to a web document that indicates its meaning
- Convention tends to fix prominent sets of URIs, e. g.
    - FOAF

Identifiers, why URIs?

URIs have attractive properties that reduce the risk of name clashes:

- URIs belong to domains that are controlled by its owners
- "Keep off others' domains" is an easy-to-remember rule of thumb
- 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

Identifiers, why URIs?

URIs have attractive properties that reduce the risk of name clashes:

- URIs belong to domains that are controlled by its owners
- "Keep off others' domains" is an easy-to-remember rule of thumb
- 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

Identifiers, why URIs?

URIs have attractive properties that reduce the risk of name clashes:

- URIs belong to domains that are controlled by its owners
- "Keep off others' domains" is an easy-to-remember rule of thumb
- 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

Identifiers, why URIs?

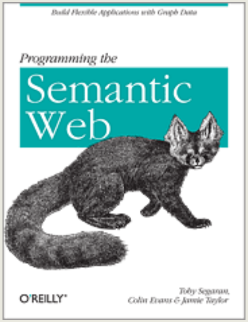
URIs have attractive properties that reduce the risk of name clashes:

- URIs belong to domains that are controlled by its owners
- "Keep off others' domains" is an easy-to-remember rule of thumb
- 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
- thus URIs tend to represent *uniquely* across the Web

A habit to suspend

Many are in the habit of thinking of it this way.....

"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."


Build Flexible Applications with Graph Data
Programming the
Semantic Web
O'REILLY
Toby Segaran, Colin Evans & Jamie Taylor

A good book

... cont

This is a habit to suspend

... cont

This is a habit to suspend

- URIs *as such* do not prevent synonymous usage

… cont

This is a habit to suspend

- URIs *as such* do not prevent synonymous usage
- nor do they prevent homonymous usage

… cont

This is a habit to suspend

- URIs *as such* do not prevent synonymous usage
- nor do they prevent homonymous usage
- it is impossible in principle to fix the meaning of a symbol

… cont

This is a habit to suspend

- URIs *as such* do not prevent synonymous usage
- nor do they prevent homonymous usage
- it is impossible in principle to fix the meaning of a symbol
- the point is rather, that

... cont

This is a habit to suspend

- URIs *as such* do not prevent synonymous usage
- nor do they prevent homonymous usage
- it is impossible in principle to fix the meaning of a symbol
- the point is rather, that
  - URIs is an established mechanism for reference

... cont

This is a habit to suspend

- URIs *as such* do not prevent synonymous usage
- nor do they prevent homonymous usage
- it is impossible in principle to fix the meaning of a symbol
- the point is rather, that
  - URIs is an established mechanism for reference
  - that is surrounded by a sufficiently stable practice

… cont

This is a habit to suspend

- URIs *as such* do not prevent synonymous usage
- nor do they prevent homonymous usage
- it is impossible in principle to fix the meaning of a symbol
- the point is rather, that
  - URIs is an established mechanism for reference
  - that is surrounded by a sufficiently stable practice
  - to keep the risk of name clashes fairly low

RDF graphs

RDF triples connect to form directed graphs

RDF graphs

RDF triples connect to form directed graphs

- the directedness captures the subject-predicate-object structure

RDF graphs

RDF triples connect to form directed graphs

- the directedness captures the subject-predicate-object structure
- ... the object of one triple, becomes the subject of the next

RDF graphs

RDF triples connect to form directed graphs

- the directedness captures the subject-predicate-object structure
- ... the object of one triple, becomes the subject of the next
- almost anything can be encoded in a graph

RDF graphs

RDF triples connect to form directed graphs

- the directedness captures the subject-predicate-object structure
- ... the object of one triple, becomes the subject of the next
- almost anything can be encoded in a graph

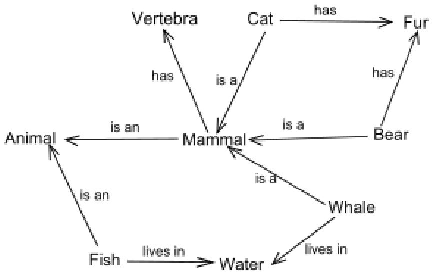


Figure: A graph describing relationships between mammals

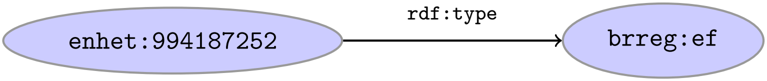
A sample RDF graph

A sample RDF graph

Connecting three facts

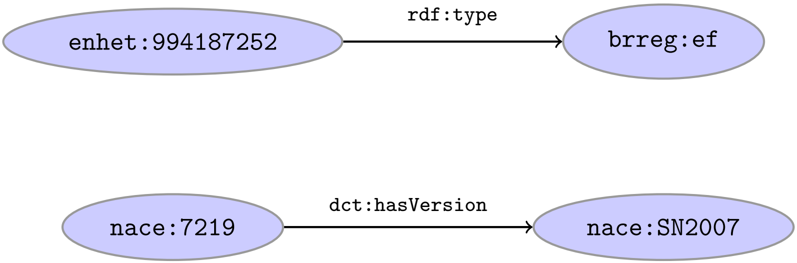
A sample RDF graph

Connecting three facts



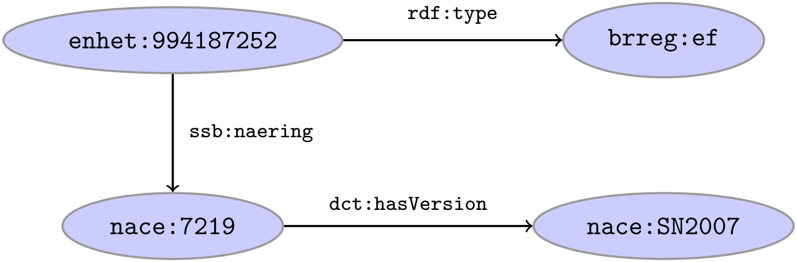
A sample RDF graph

Connecting three facts



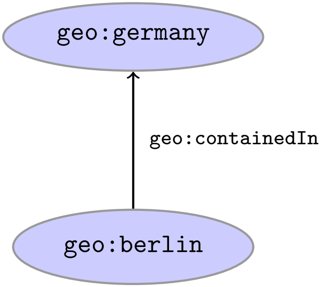
A sample RDF graph

Connecting three facts

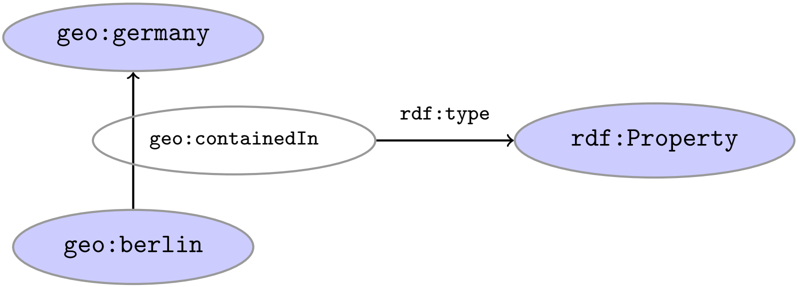


RDF graphs are not really graphs, strictly speaking ..

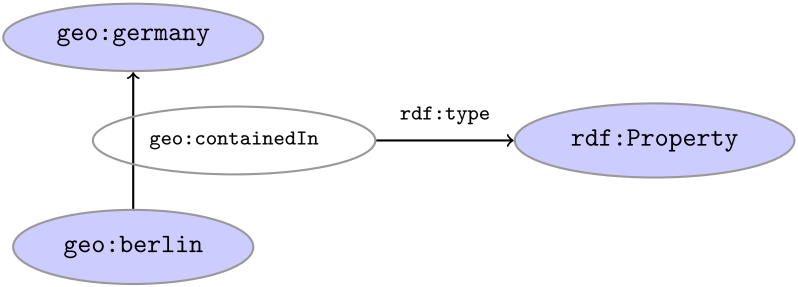
RDF graphs are not really graphs, strictly speaking ..



RDF graphs are not really graphs, strictly speaking ..



RDF graphs are not really graphs, strictly speaking ..



... since arrows are also nodes. We usually speak about *RDF graphs* anyway

RDF vocabularies

You may have noticed the discrepancy between

RDF vocabularies

You may have noticed the discrepancy between

- our emphasis of RDF's use of URIs as names

RDF vocabularies

You may have noticed the discrepancy between

- our emphasis of RDF's use of URIs as names
- and the graph-labelling on the previous slides, e.g. `geo:Germany`

RDF vocabularies

You may have noticed the discrepancy between

- our emphasis of RDF's use of URIs as names
- and the graph-labelling on the previous slides, e.g. `geo:Germany`

It is common to group related concepts under a common *base URI*, e.g.

RDF vocabularies

You may have noticed the discrepancy between

- our emphasis of RDF's use of URIs as names
- and the graph-labelling on the previous slides, e.g. `geo:Germany`

It is common to group related concepts under a common *base URI*, e.g.

- the concepts *creator* and *contributor*, under respectively

RDF vocabularies

You may have noticed the discrepancy between

- our emphasis of RDF's use of URIs as names
- and the graph-labelling on the previous slides, e.g. `geo:Germany`

It is common to group related concepts under a common *base URI*, e.g.

- the concepts *creator* and *contributor*, under respectively
    - http://purl.org/dc/terms/creator, and

RDF vocabularies

You may have noticed the discrepancy between

- our emphasis of RDF's use of URIs as names
- and the graph-labelling on the previous slides, e.g. `geo:Germany`

It is common to group related concepts under a common *base URI*, e.g.

- the concepts *creator* and *contributor*, under respectively
  - http://purl.org/dc/terms/creator, and
  - http://purl.org/dc/terms/contributor

RDF vocabularies

You may have noticed the discrepancy between

- our emphasis of RDF's use of URIs as names
- and the graph-labelling on the previous slides, e.g. `geo:Germany`

It is common to group related concepts under a common *base URI*, e.g.

- the concepts *creator* and *contributor*, under respectively
  - http://purl.org/dc/terms/creator, and
  - http://purl.org/dc/terms/contributor
- the base URI is usually abbreviated by a prefix, e.g. `dct`, yielding

RDF vocabularies

You may have noticed the discrepancy between

- our emphasis of RDF's use of URIs as names
- and the graph-labelling on the previous slides, e.g. `geo:Germany`

It is common to group related concepts under a common *base URI*, e.g.

- the concepts *creator* and *contributor*, under respectively
  - http://purl.org/dc/terms/creator, and
  - http://purl.org/dc/terms/contributor
- the base URI is usually abbreviated by a prefix, e.g. `dct`, yielding
  - `dct:creator`, and

RDF vocabularies

You may have noticed the discrepancy between

- our emphasis of RDF's use of URIs as names
- and the graph-labelling on the previous slides, e.g. `geo:Germany`

It is common to group related concepts under a common *base URI*, e.g.

- the concepts *creator* and *contributor*, under respectively
  - http://purl.org/dc/terms/creator, and
  - http://purl.org/dc/terms/contributor
- the base URI is usually abbreviated by a prefix, e.g. `dct`, yielding
  - `dct:creator`, and
  - `dct:contributor` as short forms of the URIs above

… cont

- A set uf URIs so related is a RDF vocabulary

… cont

- A set uf URIs so related is a RDF vocabulary
- vocabularies provide a way to organize and manage a set of names

… cont

- A set uf URIs so related is a RDF vocabulary
- vocabularies provide a way to organize and manage a set of names
- the most prominent example is rdf itself

... cont

- A set uf URIs so related is a RDF vocabulary
- vocabularies provide a way to organize and manage a set of names
- the most prominent example is rdf itself
- notable members of the rdf vocabulary includes

... cont

- A set uf URIs so related is a RDF vocabulary
- vocabularies provide a way to organize and manage a set of names
- the most prominent example is rdf itself
- notable members of the rdf vocabulary includes
  - `rdf:type` for typing resources

... cont

- A set uf URIs so related is a RDF vocabulary
- vocabularies provide a way to organize and manage a set of names
- the most prominent example is rdf itself
- notable members of the rdf vocabulary includes
    - `rdf:type` for typing resources
    - `rdf:Property` to distinguish predicates from objects

… cont

- A set uf URIs so related is a RDF vocabulary
- vocabularies provide a way to organize and manage a set of names
- the most prominent example is rdf itself
- notable members of the rdf vocabulary includes
    - `rdf:type` for typing resources
    - `rdf:Property` to distinguish predicates from objects
    - `rdf:List` for representing sequences

... cont

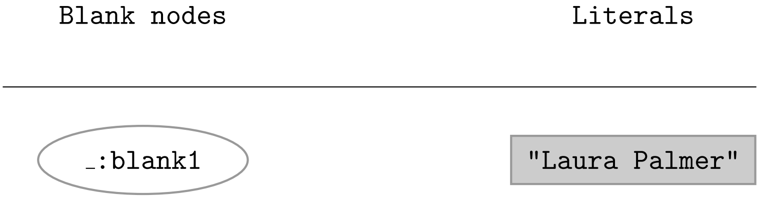
- A set uf URIs so related is a RDF vocabulary
- vocabularies provide a way to organize and manage a set of names
- the most prominent example is rdf itself
- notable members of the rdf vocabulary includes
    - `rdf:type` for typing resources
    - `rdf:Property` to distinguish predicates from objects
    - `rdf:List` for representing sequences
- FOAF, Dublin Core, VCard and the Basic Geo Vocabulary are other examples

RDF graphs: A closer look at nodes

As hinted at on slide 12, a node need not be a URI

- it can also be a literal value such as the string "Death in Venice"
- or a blank node acting as a mere placeholder for a stipulated object

Common visual representations are:



What are they for?

We use literals to

What are they for?

We use literals to

- represent datatypes such as integers, strings, XML elements, decimals

What are they for?

We use literals to

- represent datatypes such as integers, strings, XML elements, decimals
- for which it makes little sense to assign a URI (why?)

What are they for?

We use literals to

- represent datatypes such as integers, strings, XML elements, decimals
- for which it makes little sense to assign a URI (why?)

We use blank nodes whenever

What are they for?

We use literals to

- represent datatypes such as integers, strings, XML elements, decimals
- for which it makes little sense to assign a URI (why?)

We use blank nodes whenever

- we wish to assert the existence of an object,

What are they for?

We use literals to

- represent datatypes such as integers, strings, XML elements, decimals
- for which it makes little sense to assign a URI (why?)

We use blank nodes whenever

- we wish to assert the existence of an object,
    - but do not care about its identity

What are they for?

We use literals to

- represent datatypes such as integers, strings, XML elements, decimals
- for which it makes little sense to assign a URI (why?)

We use blank nodes whenever

- we wish to assert the existence of an object,
    - but do not care about its identity
- we wish to group statements together

What are they for?

We use literals to

- represent datatypes such as integers, strings, XML elements, decimals
- for which it makes little sense to assign a URI (why?)

We use blank nodes whenever

- we wish to assert the existence of an object,
  - but do not care about its identity
- we wish to group statements together
- for many-valued relationships such as e.g. ‘x buys y from z’

What are they for?

We use literals to

- represent datatypes such as integers, strings, XML elements, decimals
- for which it makes little sense to assign a URI (why?)

We use blank nodes whenever

- we wish to assert the existence of an object,
    - but do not care about its identity
- we wish to group statements together
- for many-valued relationships such as e.g. ‘x buys y from z’
- a blank node is essentially an existentially quantified variable

What are they for?

We use literals to

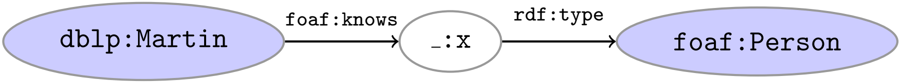
- represent datatypes such as integers, strings, XML elements, decimals
- for which it makes little sense to assign a URI (why?)

We use blank nodes whenever

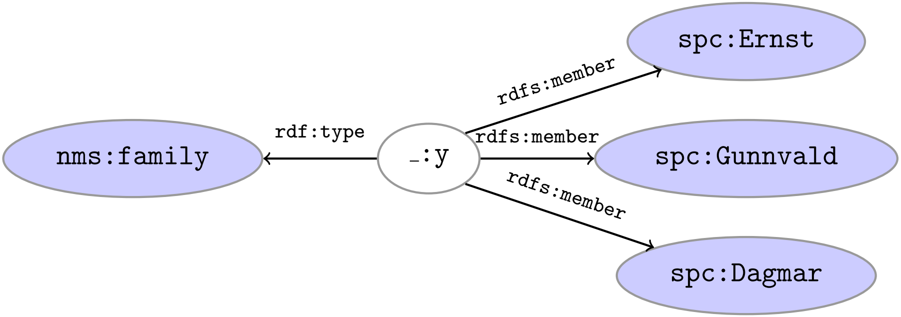
- we wish to assert the existence of an object,
  - but do not care about its identity
- we wish to group statements together
- for many-valued relationships such as e.g. ‘x buys y from z’
- a blank node is essentially an existentially quantified variable
  - $(\_ : blank1, b, c)$ means ‘*there is an object* x such that x is b-related to c’

Graphs with blanks nodes

Martin knows someone:



Ernst, Dagmar and Gunnvald are members of the same family:



Triple grammar

RDF-nodes induce a simple triple grammar:

Triple grammar

RDF-nodes induce a simple triple grammar:

- Only URIs may occur in predicate position

Triple grammar

RDF-nodes induce a simple triple grammar:

- Only URIs may occur in predicate position
- Literals may only occur in object position

Triple grammar

RDF-nodes induce a simple triple grammar:

- Only URIs may occur in predicate position
- Literals may only occur in object position
- Blank nodes may occur in subject and object position

Triple grammar

RDF-nodes induce a simple triple grammar:

- Only URIs may occur in predicate position
- Literals may only occur in object position
- Blank nodes may occur in subject and object position

Capice?

العربية

More about literal values

As mentioned, literals in RDF represent data values.

More about literal values

As mentioned, literals in RDF represent data values.

- untyped literals are always interpreted as strings.

More about literal values

As mentioned, literals in RDF represent data values.

- untyped literals are always interpreted as strings.
- in general though, a literal value may have either

More about literal values

As mentioned, 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

More about literal values

As mentioned, 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

More about literal values

As mentioned, 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

More about literal values

As mentioned, 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

The datatype of a literal determines its meaning; e.g.

More about literal values

As mentioned, 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 specifies the language of the string
  - but not both

The datatype of a literal determines its meaning; e.g.

- 42 as a date, vs.

More about literal values

As mentioned, 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

The datatype of a literal determines its meaning; e.g.

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

Outline

Serialization formats

A serialization is an encoding of a data structure in a format that can be stored

Serialization formats

A serialization is an encoding of a data structure in a format that can be stored

- an RDF serialization, specifically, is a file format

Serialization formats

A serialization is an encoding of a data structure in a format that can be stored

- an RDF serialization, specifically, is a file format
- there are many such formats

Serialization formats

A serialization is an encoding of a data structure in a format that can be stored

- an RDF serialization, specifically, is a file format
- there are many such formats
    - RDF/XML (the official W3C recommendation)

Serialization formats

A serialization is an encoding of a data structure in a format that can be stored

- an RDF serialization, specifically, is a file format
- there are many such formats
  - RDF/XML (the official W3C recommendation)
  - Turtle (very convenient format for humans)

Serialization formats

A serialization is an encoding of a data structure in a format that can be stored

- an RDF serialization, specifically, is a file format
- there are many such formats
  - RDF/XML (the official W3C recommendation)
  - Turtle (very convenient format for humans)
  - N3 (a superset of Turtle)

Serialization formats

A serialization is an encoding of a data structure in a format that can be stored

- an RDF serialization, specifically, is a file format
- there are many such formats
    - RDF/XML (the official W3C recommendation)
    - Turtle (very convenient format for humans)
    - N3 (a superset of Turtle)
    - N-Triples (very convenient format for machines)

Serialization formats

A serialization is an encoding of a data structure in a format that can be stored

- an RDF serialization, specifically, is a file format
- there are many such formats
  - RDF/XML (the official W3C recommendation)
  - Turtle (very convenient format for humans)
  - N3 (a superset of Turtle)
  - N-Triples (very convenient format for machines)
- they all express the same *abstract* data model, namely RDF

A quick look at Turtle

A Turtle file starts with a declaration of prefixes

```
@prefix place:   <http://sws.geonames.org/>.
@prefix geoont:  <http://www.geonames.org/ontology/>.
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-
@prefix foaf:    <http://xmlns.com/foaf/0.1/>.
@prefix xsd:     <http://www.w3.org/2001/XMLSchema#>.
```

- here as everywhere in Turtle, URIs are ecnlosed in angled brackets
- prefixes start with `@` and end with a dot
- you will see the ones above a lot
- of course, the abbreviations will vary since they don't matter

Statements in Turtle

Triples in Turtle (i.e. statements or facts) are

- written on the same line separated by a white space
- and terminated by a dot, e.g.

```
place:390903 rdf:type geoont:Country.
place:2945356 rdf:type geoont:Municipality.
```

- statements with the same subject admit a short form;

```
place:390903 rdf:type geoont:Country;
             geoont:Population "11262000;
             rdfs:label "Greece".
```

... cont

- so do statments with the same subject *and* predicate

```
place:390903 rdf:type geoont:Country, geoont:Region;
             geoont:Population "11262000";
             rdfs:label "Greece", "Hellas".
```

- `rdf:type` may be abbreviated `'a'`:

```
place:390903 a geoont:Country, geoont:Region;
             geoont:Population "11262000";
             rdfs:label "Greece", "Hellas".
```

Datatypes and language tags in Turtle

- Datatypes are represented wit double carets;

```
place:390903 a geoont:Country, geoont:Region;
             geoont:Population "11262000"^^xsd:int;
             rdfs:label "Greece", "Hellas".
```

- and language tags with ‘@’;

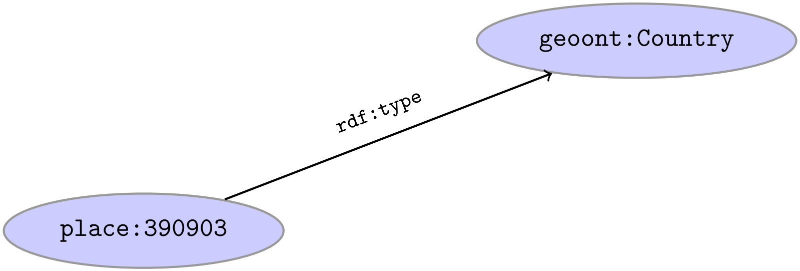
```
place:390903 a geoont:Country, geoont:Region;
             geoont:Population "11262000"^^xsd:int;
             rdfs:label "Greece"@en, "Hellas"@nb.
```

Greece is a Country with a population of 11.2 million people

```
    place:390903 a geoont:Country;
           geoont:Population "11262000"^^xsd:int .
```

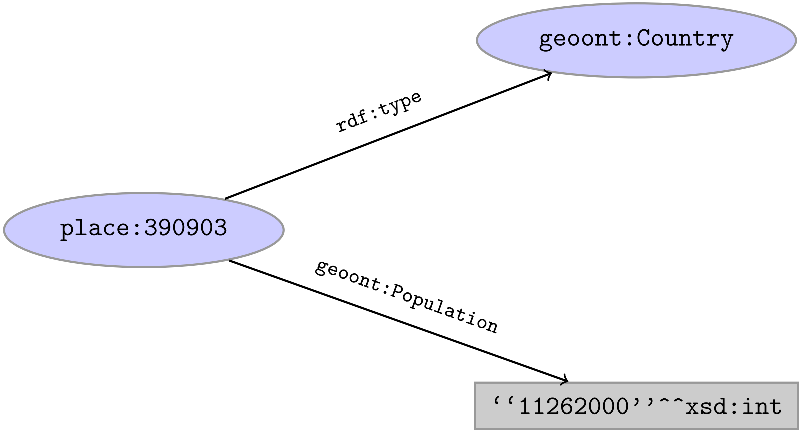
Greece is a Country with a population of 11.2 million people

```
    place:390903 a geoont:Country;
            geoont:Population "11262000"^^xsd:int .
```



Greece is a Country with a population of 11.2 million people

```
    place:390903 a geoont:Country;
            geoont:Population "11262000"^^xsd:int .
```



Outline

Quick facts

- SPARQL Protocol And RDF Query Language

Quick facts

- SPARQL Protocol And RDF Query Language
    - the SPARQL query language resembles SQL, but simpler

Quick facts

- SPARQL Protocol And RDF Query Language
  - the SPARQL query language resembles SQL, but simpler
  - based on the idea of matching graph patterns

Quick facts

- SPARQL Protocol And RDF Query Language
  - the SPARQL query language resembles SQL, but simpler
  - based on the idea of matching graph patterns
  - syntax closely resembles Turtle

Quick facts

- SPARQL Protocol And RDF Query Language
    - the SPARQL query language resembles SQL, but simpler
    - based on the idea of matching graph patterns
    - syntax closely resembles Turtle
- Try it out:

Quick facts

- SPARQL Protocol And RDF Query Language
  - the SPARQL query language resembles SQL, but simpler
  - based on the idea of matching graph patterns
  - syntax closely resembles Turtle
- Try it out:

  DBLP `http://dblp.l3s.de/d2r/snorql/`

Quick facts

- SPARQL Protocol And RDF Query Language
  - the SPARQL query language resembles SQL, but simpler
  - based on the idea of matching graph patterns
  - syntax closely resembles Turtle
- Try it out:

DBLP	`http://dblp.l3s.de/d2r/snorql/`
DBpedia	`http://dbpedia.org/sparql`

Quick facts

- SPARQL Protocol And RDF Query Language
  - the SPARQL query language resembles SQL, but simpler
  - based on the idea of matching graph patterns
  - syntax closely resembles Turtle
- Try it out:

DBLP	`http://dblp.l3s.de/d2r/snorql/`
DBpedia	`http://dbpedia.org/sparql`
DBtunes	`http://dbtune.org/musicbrainz/`

An example

Publications by people called “Martin Giese”

```
PREFIX foaf:  <http://xmlns.com/foaf/0.1/>
PREFIX dc:  <http://purl.org/dc/elements/1.1/>
SELECT ?pub WHERE {
    ?mg foaf:name "Martin Giese" .
    ?pub dc:creator ?mg .
}
```

An example

Publications by people called “Martin Giese”

```
PREFIX foaf:  <http://xmlns.com/foaf/0.1/>
PREFIX dc:  <http://purl.org/dc/elements/1.1/>
SELECT ?pub WHERE {
    ?mg foaf:name "Martin Giese" .
    ?pub dc:creator ?mg .
}
```

Answer:

?pub
<http://dblp.l3s.de/d2r/resource/publications/conf/cade/Giese01>
<http://dblp.l3s.de/d2r/resource/publications/conf/cade/BeckertGHKRSS07>
<http://dblp.l3s.de/d2r/resource/publications/conf/fase/AhrendtBBGHHMMS02>
... ...

Things to note

- Unlike Turtle, prefixes are not (well ..) prefixed by '`@`'

Things to note

- Unlike Turtle, prefixes are not (well ..) prefixed by ‘`@`’
- nor are they terminated by a period

Things to note

- Unlike Turtle, prefixes are not (well ..) prefixed by ‘`@`’
- nor are they terminated by a period
- `SELECT` is the type of query you will use the most

Things to note

- Unlike Turtle, prefixes are not (well ..) prefixed by '`@`'
- nor are they terminated by a period
- `SELECT` is the type of query you will use the most
- expressions of the form `?something` are variables

Things to note

- Unlike Turtle, prefixes are not (well ..) prefixed by ‘`@`’
- nor are they terminated by a period
- `SELECT` is the type of query you will use the most
- expressions of the form `?something` are variables
- the variables inside the `WHERE` clause are matched against the RDF graph

Things to note

- Unlike Turtle, prefixes are not (well ..) prefixed by ‘`@`’
- nor are they terminated by a period
- `SELECT` is the type of query you will use the most
- expressions of the form `?something` are variables
- the variables inside the `WHERE` clause are matched against the RDF graph
- matches for the variables outside the pattern are returned as results

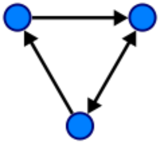
Outline

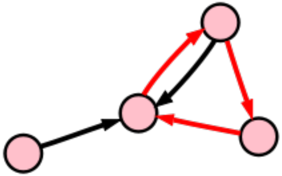
Joining graphs

As mentioned, RDF models are directed graphs (digraphs):

Joining graphs

As mentioned, RDF models are directed graphs (digraphs):

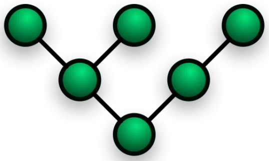


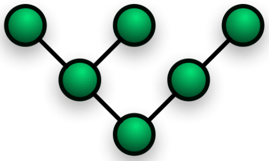


If you add one digraph to another, then you get another digraph.

Compare with trees

This contrasts with trees





- the union of the two trees lacks a common root
- hence it is not a tree
- special steps must therefore be taken to merge trees

Merging contd.

The RDF data model optimized for sharing and meshing up data:

Merging contd.

The RDF data model optimized for sharing and meshing up data:

- a triple is a digraph,

Merging contd.

The RDF data model optimized for sharing and meshing up data:

- a triple is a digraph,
- a set of triples is a digraph,

Merging contd.

The RDF data model optimized for sharing and meshing up data:

- 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

Merging contd.

The RDF data model optimized for sharing and meshing up data:

- 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 names will not clash

Merging contd.

The RDF data model optimized for sharing and meshing up data:

- 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 names will not clash
- hence, any number of triples (that is, any graph) can be added to any other

Merging contd.

The RDF data model optimized for sharing and meshing up data:

- 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 names will not clash
- hence, any number of triples (that is, any graph) can be added to any other
- without ever violating the RDF data model

Merging contd.

The RDF data model optimized for sharing and meshing up data:

- 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 names will not clash
- hence, any number of triples (that is, any graph) can be added to any other
- without ever violating the RDF data model
- whence querying several RDF-graphs as once is (almost) as simple as combining them

Qualifications

This claim is subject to a few qualifications though

Qualifications

This claim is subject to a few qualifications though

- if the serialization format is RDF/XML then the document is a tree anyway

Qualifications

This claim is subject to a few qualifications though

- if the serialization format is RDF/XML then the document is a tree anyway
    - so special steps must again be taken

Qualifications

This claim is subject to a few qualifications though

- if the serialization format is RDF/XML then the document is a tree anyway
    - so special steps must again be taken
- blank nodes must be renamed apart

Qualifications

This claim is subject to a few qualifications though

- if the serialization format is RDF/XML then the document is a tree anyway
    - so special steps must again be taken
- blank nodes must be renamed apart
    - they are not prefixed by URIs

Qualifications

This claim is subject to a few qualifications though

- if the serialization format is RDF/XML then the document is a tree anyway
    - so special steps must again be taken
- blank nodes must be renamed apart
    - they are not prefixed by URIs
    - i.e. they are not globally uniqye

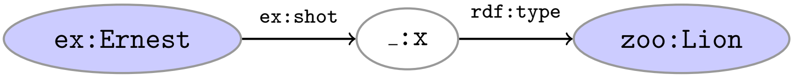
Qualifications

This claim is subject to a few qualifications though

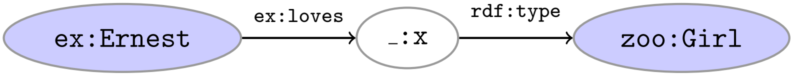
- if the serialization format is RDF/XML then the document is a tree anyway
  - so special steps must again be taken
- blank nodes must be renamed apart
  - they are not prefixed by URIs
  - i.e. they are not globally uniqye
  - so you might otherwise get unintended name clashes

Blank nodes must be renamed

Ernest shot a lion,

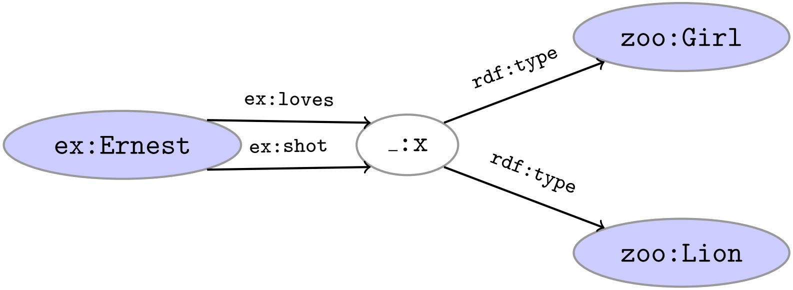


and Ernest loves a girl,



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

Locating RDF in the Semantic Web Stack architecture

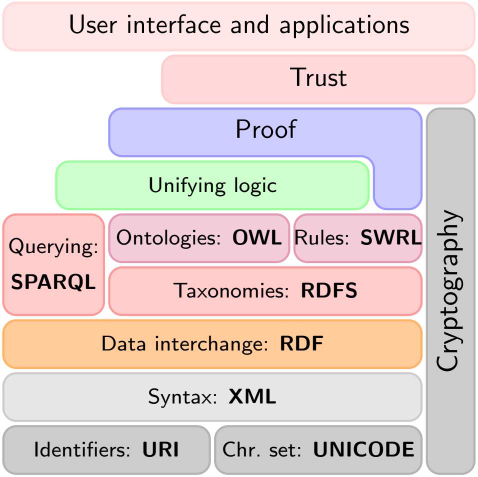


Figure: Semantic Web Stack