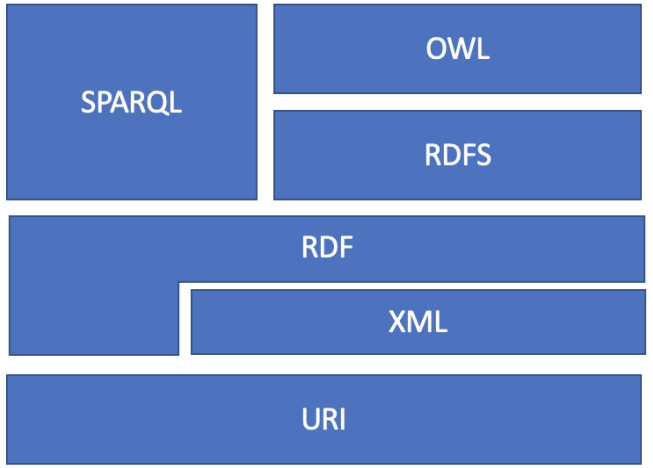
Knowledge representation

Dr. Radu Mihailescu

Associate Professor

From web of document to web of data

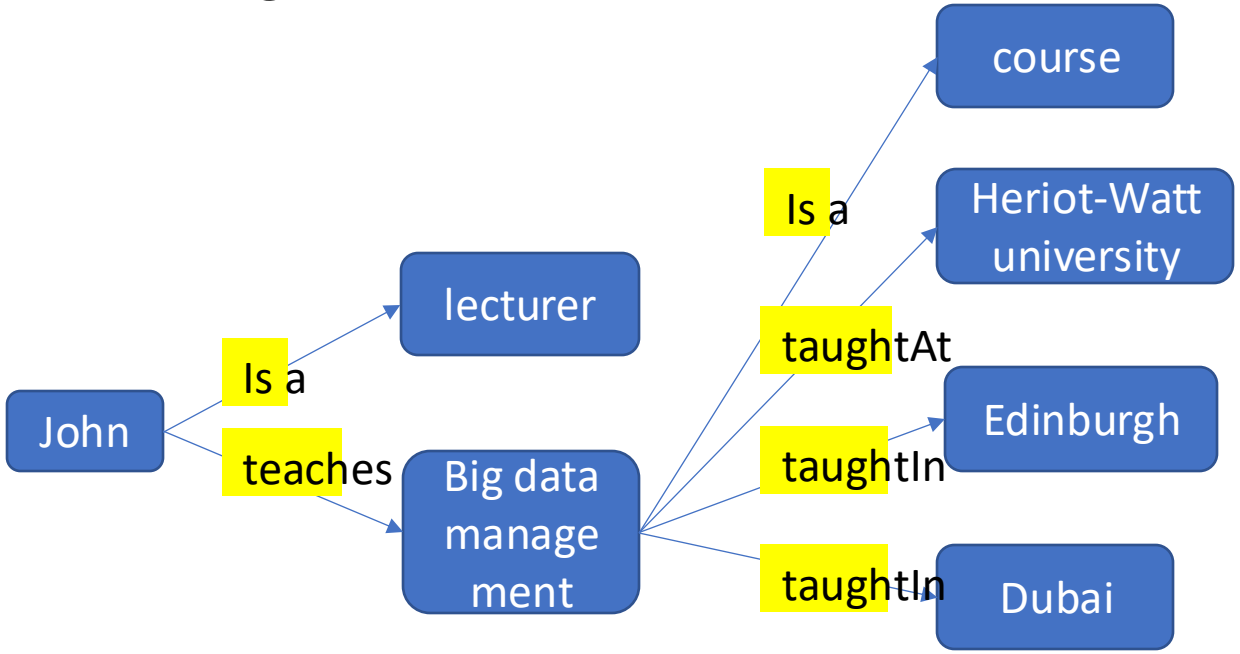
- **Web of documents** for human reading (search by words contained)
- **Web of data with** open linked data for machine understanding (search based on entities and their relationships)
- To allow for this, semantic web technologies have been created by W3C
  - Semantic annotation and retrieval: RDF, RDFS
  - Storing the Semantic Web: triple stores
  - Querying the Semantic Web: SPARQL
  - Reasoning on the Semantic Web: OWL, reasoning tools



Knowledge graphs

- Knowledge is considered as a set of relationships between things (or resources)
- A thing has properties (relationships) that links it to other things
- Any knowledge can therefore be represented as a graph, where the resources are the nodes and links are the relationships

"John is a lecturer who teaches big data management course at Heriot Wat University taught in Edinburgh and Dubai."



Characteristics of knowledge graphs

- Scaling – the same graph can represent many different knowledge simultaneously (geography, persons, occupation, universities…)
- Agreement – need to standardise the various relations in order to know what they “mean”
- Structure – need to know what nodes are related by a relations
- Plurality – the same relationship may appear several times (is_a, name…)
- Asymmetry – relationships are inherently directed, and very seldom symmetric

Reading a graph

- A knowledge graph can be seen
  - As a set of nodes and links
    - Nodes store information about things and links store relationships between things
  - A set of links, called predicates
    - A predicate has a subject (the source node) and an object (the destination node)
    - The graph can be written as a set of (subject, predicate, object)
    - These are called triples

RDF

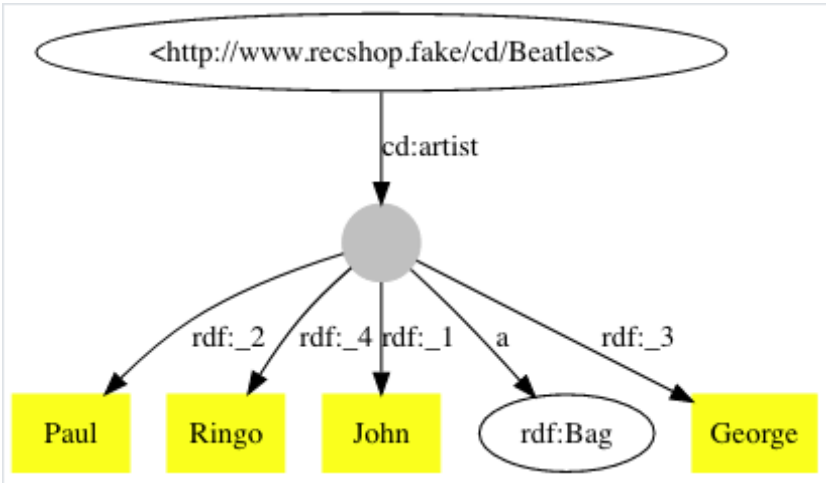
- A standard language to represent knowledge graphs
- An RDF element is a triple
  - Subject predicate object
- Subject and predicate are resources defined by IRI
- object is a resource (with IRI) or a literal (a value with an xsd type such as integer, float, string, date…)
- An RDF model is an unordered collection of statements (triples)
- Many RDF serialisation formats
  - RDF/XML – original
  - **N-triples**, N-Quads
  - **Turtle**, Trig
  - JSON-LD
  - RDF-a
- It defines the semantics of information in machine-interpretable format

```
BASE <http://www.hw.ac.uk/courses#>
prefix tr:    <http://www.hw.ac.uk/courses#>
prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dct: <http://purl.org/dc/terms/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
tr:university a rdfs:class.
tr:country a rdfs:class.
tr:professor a rdfs:class.
tr:course a rdfs:class.
tr:big_data_management rdf:type tr:course.
tr:UAE rdf:type tr:country.
tr:heriot_watt_uni rdf:type tr:university.
tr:Dubai tr:city_of tr:UAE.
tr:hadj tr:type tr:professor;
        tr:livesIn tr:Dubai;
        tr:worksAt tr:heriot_watt_uni;
        tr:teaches tr:big_data_management.
```

RDF- XML elements

Element	Description
rdf:RDF	root element of RDF documents, where a number of descriptions are defined. Defines the XML document to be an RDF document
rdf:Description	element contains the description of the resource identified by the **rdf:about** attribute.
rdf:type	instance of
rdf:Bag	an unordered **container** of resources
rdf:Seq	an ordered **container** of resources
rdf:Alt	Defines a set (**container**) of alternative resources (the user can select only one of the values)

Example: bag



RDF/XML

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:cd="http://www.recshop.fake/cd#"
>
<rdf:Description
    rdf:about="http://www.recshop.fake/cd/Beatles">
  <cd:artist>
    <rdf:Bag>
      <rdf:li>John</rdf:li>
      <rdf:li>Paul</rdf:li>
      <rdf:li>George</rdf:li>
      <rdf:li>Ringo</rdf:li>
    </rdf:Bag>
  </cd:artist>
</rdf:Description>
</rdf:RDF>
```

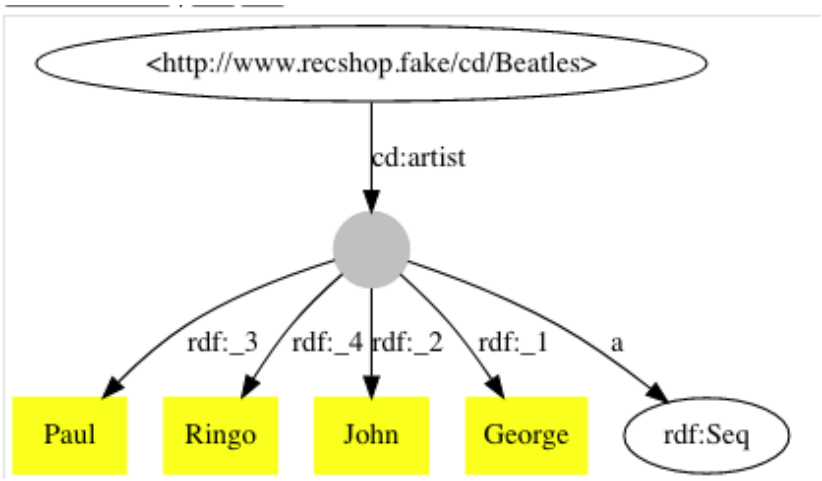
Turtle

```
@prefix cd: <http://www.recshop.fake/cd#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
<http://www.recshop.fake/cd/Beatles>
  cd:artist [
        a rdf:Bag ;
        rdf:_1 "John" ;
        rdf:_2 "Paul" ;
        rdf:_3 "George" ;
        rdf:_4 "Ringo"
  ] .
```

Example: Seq

RDF/XML

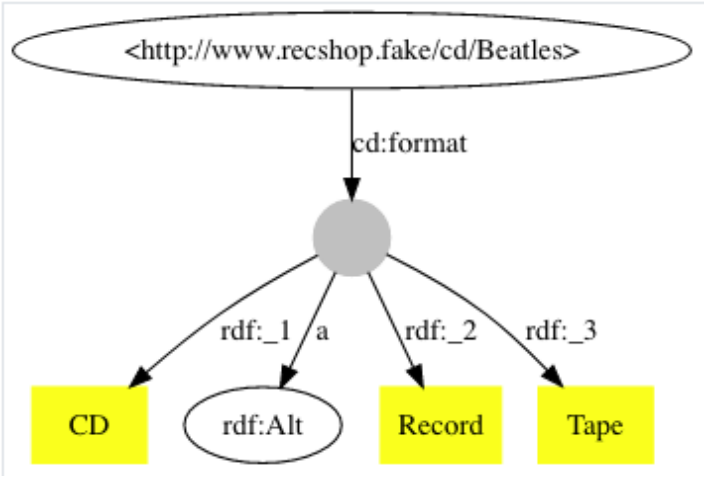
```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:cd="http://www.recshop.fake/cd#"
>
<rdf:Description
    rdf:about="http://www.recshop.fake/cd/Beatles">
   <cd:artist>
       <rdf:Seq>
            <rdf:li>John</rdf:li>
            <rdf:li>Paul</rdf:li>
            <rdf:li>George</rdf:li>
            <rdf:li>Ringo</rdf:li>
       </rdf:Seq>
   </cd:artist>
</rdf:Description>
</rdf:RDF>
```



Turtle

```
@prefix cd: <http://www.recshop.fake/cd#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
<http://www.recshop.fake/cd/Beatles>
  cd:artist [
            a rdf:Seq ;
            rdf:_1 "George" ;
            rdf:_2 "John" ;
            rdf:_3 "Paul" ;
            rdf:_4 "Ringo"
  ] .
```

Example: Alt



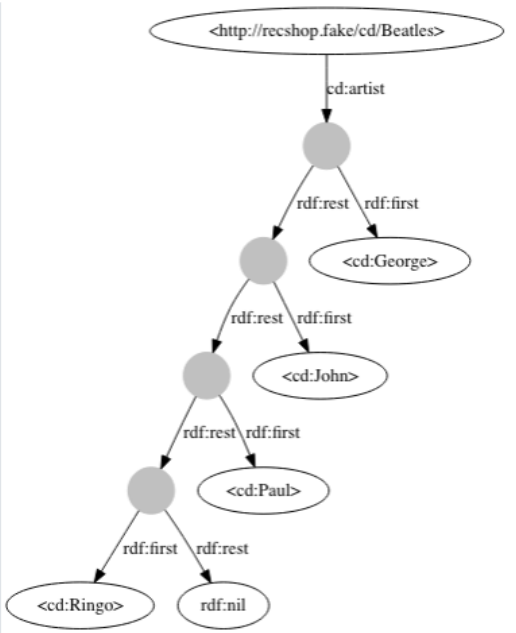
RDF/XML

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:cd="http://www.recshop.fake/cd#"
>
<rdf:Description
    rdf:about="http://www.recshop.fake/cd/Beatles">
  <cd:format>
    <rdf:Alt>
        <rdf:li>CD</rdf:li>
        <rdf:li>Record</rdf:li>
        <rdf:li>Tape</rdf:li>
    </rdf:Alt>
  </cd:format>
</rdf:Description>
</rdf:RDF>
```

Turtle

```
@prefix cd: <http://www.recshop.fake/cd#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
<http://www.recshop.fake/cd/Beatles>
  cd:format [
        a rdf:Alt ;
        rdf:_1 "CD" ;
        rdf:_2 "Record" ;
        rdf:_3 "Tape" ;
  ] .
```

Example: Collection



RDF/XML

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:cd="http://www.recshop.fake/cd#"
>
<rdf:Description
    rdf:about="http://www.recshop.fake/cd/Beatles">
   <cd:artist rdf:parseType="Collection">
       <rdf:Description rdf:about="cd:George"/>
       <rdf:Description rdf:about="cd:John"/>
       <rdf:Description rdf:about="cd:Paul"/>
       <rdf:Description rdf:about="cd:Ringo"/>
  </cd:artist>
</rdf:Description>
</rdf:RDF>
```

Turtle

```
@prefix cd: <http://www.recshop.fake/cd#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
<http://www.recshop.fake/cd/Beatles>
  cd:artist [
                <cd:George> ;
                <cd:John> ;
                <cd:Paul> ;
                <cd:Ringo> ;
  ] .
```

RDF collections are used to describe groups that can ONLY contain the specified members.

RDF – XML attributes

rdf:ID	Abbreviating a node IRI, e.g. http://example.org/here/#snack <rdf:Description rdf:ID="snack">
rdf:about	referencing an existing resource <rdf:Description rdf:about=”#snack">
rdf:resource	allows property elements to be defined as resources <rdf:Seq rdf:about="http://example.org/favourite-fruit"> <rdf:_1 rdf:resource="http://example.org/banana"/> <rdf:_2 rdf:resource="http://example.org/apple"/> <rdf:_3 rdf:resource="http://example.org/pear"/> </rdf:Seq>

RDF Classes & Properties

Classes

- rdf:XMLLiteral
- rdf:Property
- rdf:Alt
- rdf:Bag
- rdf:Seq
- rdf:List
- rdf:nil
- rdf:Statement

Properties

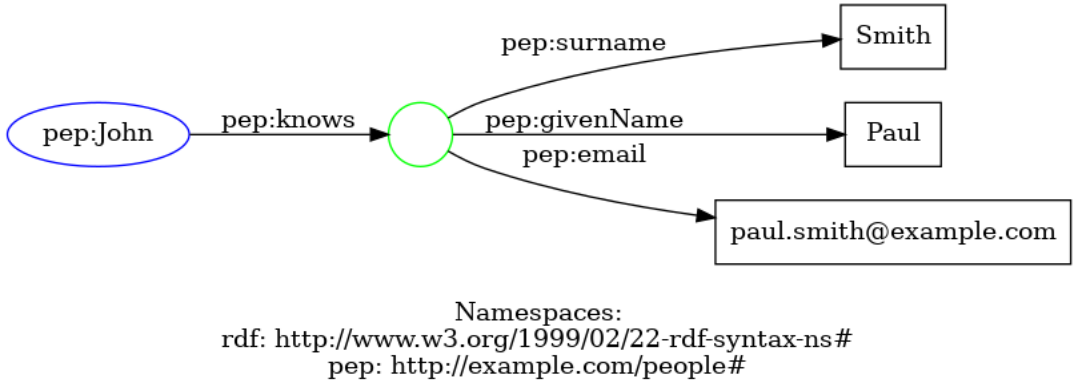
- rdf:type
- rdf:first
- rdf:rest
- rdf:value
- rdf:subject
- rdf:predicate
- rdf:object

https://www.w3.org/TR/rdf12-schema/

Blank nodes

- When a node does not need to be visible outside, there is no need to assign to it an IRI
- It will be a *blank node* or *bnode*, or *anonymous node*
- Blank nodes are represented by an empty node
- Usually, an internal ID is generated to differentiate a blank node from other blank nodes
- So, an *object* in a triple (subject, predicate, object) can be
  - IRI
  - Literal
  - Blank node

Example: blank node



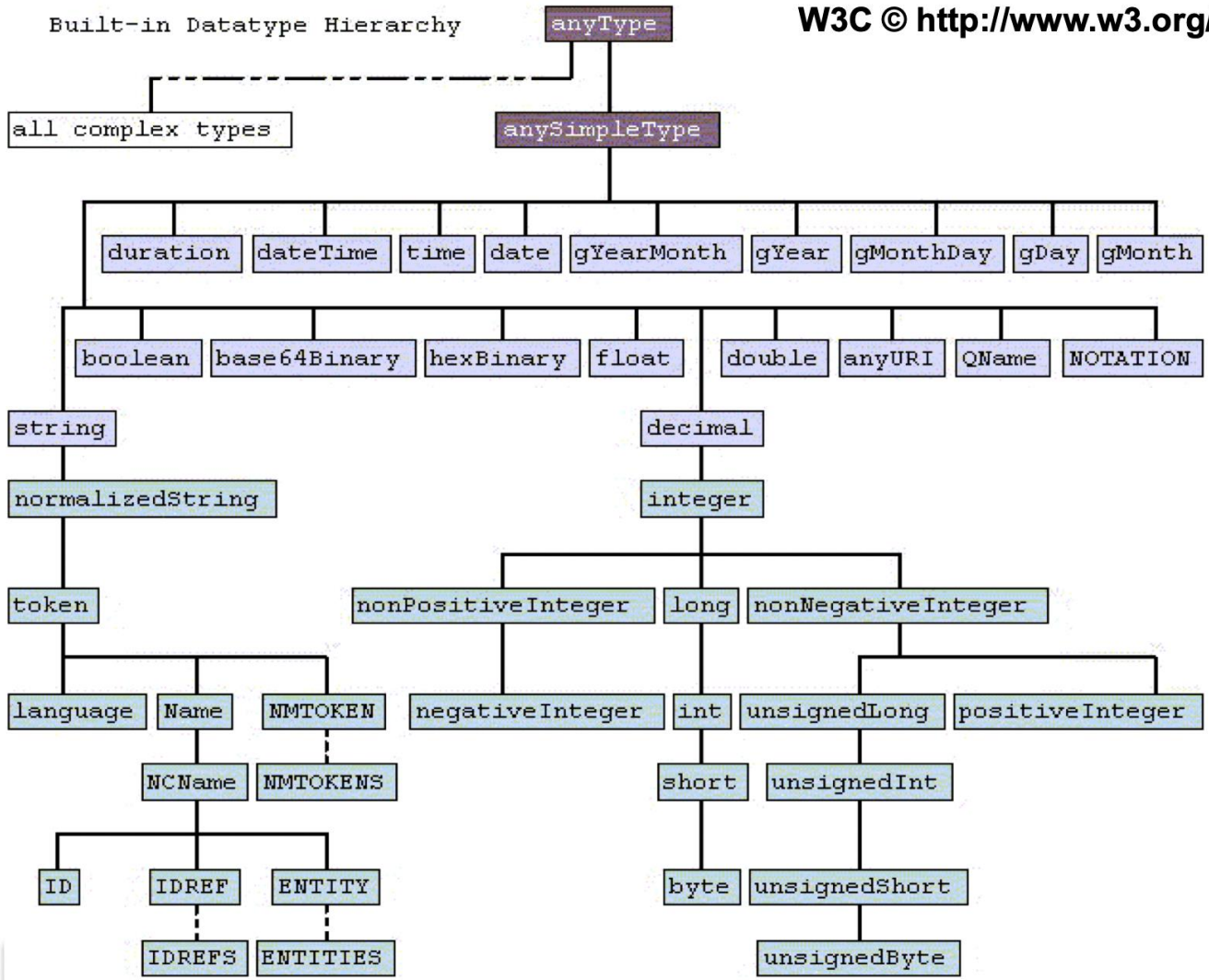
RDF/XML

```
<?xml version="1.0"?>
<rdf:RDF
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:pep="http://example.com/people#"
 xml:base="http://example.com/">

 <rdf:Description rdf:about="pep:John">
   <pep:knows>
      <rdf:Description>
            <pep:surname>Smith</pep:surname>
            <pep:givenName>Paul</pep:givenName>
            <pep:email>paul.smith@example.com</pep:email>
      </rdf:Description>
   </pep:knows>
 </rdf:Description>
</rdf:RDF>
```

Turtle

```
@prefix pep: <http://example.com/people#> .
<pep:john>
    pep:knows [
            pep:surname "Smith" ;
            pep:givenName "Paul";
            pep:email "paul.smith@example.com"
] .
```

XML Schema Datatypes for Literals



W3C © http://www.w3.org/TR/xmlschema-2/

RDF limitations

- RDF triples represent a knowledge graph but without much restrictions on the use of terms
  - Example, nothing prevents using the property “last name” for a car!
- This is due to the fact there is no type mechanism in RDF
  - We cannot specify which relation are allowed and for what type of subject and object
- This leads to two problems
  - Interpretability: different people understand predicates in different ways and that lead to statements that are difficult to interpret
  - Automatization: software applications cannot be developed to generate rigorous knowledge graphs or make accurate inference

RDFS: RDF schema

- Also known as RDF Vocabulary Description Language
- It is an extension of RDF
- It defines the vocabulary used in RDF models
- It defines a metamodel with
  - Concepts: Resource, Literal, Class, Datatype
  - Relationships: subClassOf, subPropertyOf, domain, range
- "*RDFS provides a means for defining the classes (concepts or entities), properties, and relationships in an RDF model and organizing these concepts and relationships into hierarchies*"
- It defines axioms (entailments or rules) for concepts and relationships. These rules allow inferring new triples from existing ones

RDFS classes

Do not confuse
: an attribute to define the object of a property

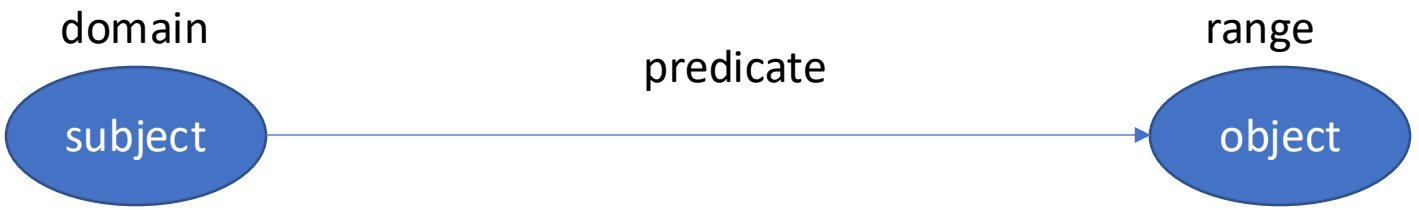
Class	Description
rdfs:Class	the class (or meta class) that defines all classes. Instance of rdfs:Class
rdfs:Resource	the class that defines all resources. Instance of rdfs:Class
rdfs:Literal	the class that defines all literal values (integers, strings…). Instance of rdfs:class and subclass of rdfs:Resource
rdfs:Property	the class that defines all properties. Instance of rdfs:Class
rdfs:Datatype	the class of datatypes. Instance and subclass of rdfs:Class. Each instance of rdfs:Datatype is subclass of rdfs:Literal

Do not confuse

rdf:resource	an attribute to define the object of a property, example *<rdf:Description rdf:about="http://example.org/Bob">* *<foaf:address rdf:resource="http://example.org/address1"/>* *</rdf:Description>*	**Turtle** (no rdf:resource) *<http://example.org/Bob>* *foaf:address* *<http://example.org/address1> .*
rdfs:Resource	A class that defines all resources (anything is resource, and any resource is subclass of rdfs:Resource) *<http://example.org/Bob> rdf:type <http://example.org/Person> .* *<http://example.org/Bob> rdf:type rdfs:Resource .*	

RDFS properties

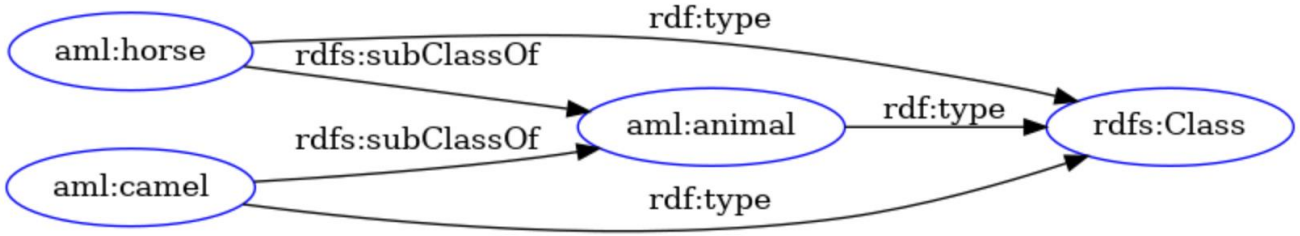
property	
rdfs:subClassOf	Relate a class to a superclass (there might be many). Allows creating a hierarchy of classes. This relation is transitive.
rdfs:subPropertyOf	Relate a property to a superproperty (there might be many). This relation is transitive.
rdfs:domain	Declares the class of the subject for a predicate
rdfs:range	Declares the class or datatype of the of the object of a predicate



RDFS metadata (information about the resources)

property	description
rdfs:comment	typically provides a longer text description of the resource
rdfs:label	associates the resource with a human-friendly name
rdfs:isDefinedBy	relates a resource to the place where its definition, typically an RDF schema
rdfs:seeAlso	relates a resource to another resource that explains it

RDFS, Example1



Namespaces:
rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs: http://www.w3.org/2000/01/rdf-schema#
aml: http://www.animals.fake/animals#

Note, it is an RDF document

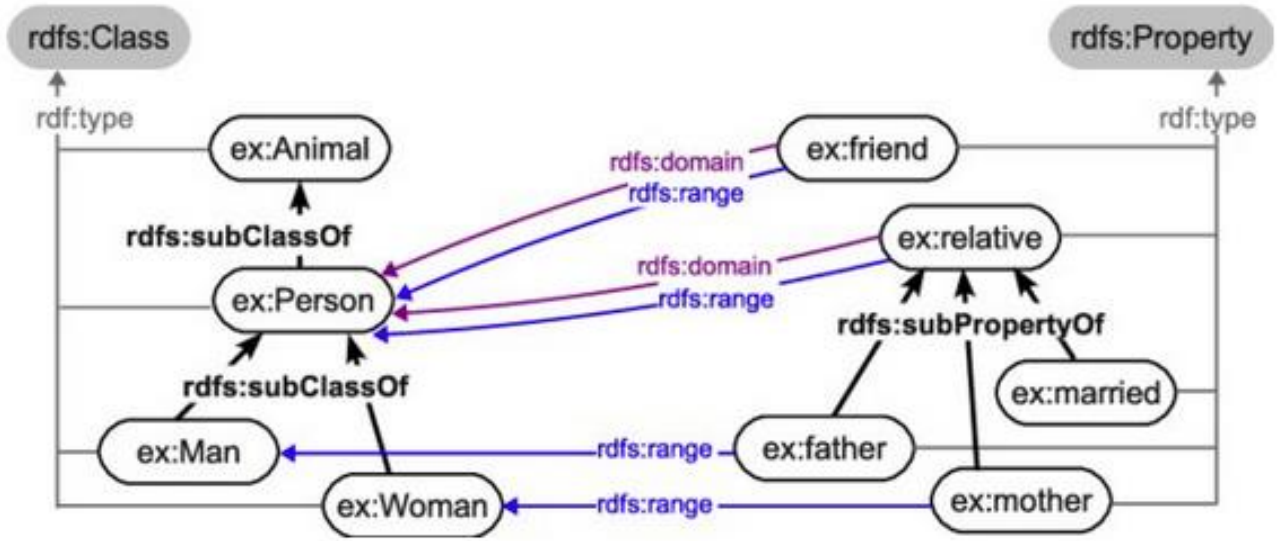
```
<?xml version="1.0"?>
<rdf:RDF
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
 xmlns:aml="http://www.animals.fake/animals#"
 xml:base="http://www.animals.fake/animals">

 <rdfs:Class rdf:ID="animal"/>
  <rdfs:Class rdf:ID="horse">
     <rdfs:subClassOf rdf:resource="aml:animal"/>
  </rdfs:Class>
  <rdfs:Class rdf:ID="camel">
    <rdfs:subClassOf rdf:resource="aml:animal"/>
  </rdfs:Class>
</rdf:RDF>
```

Turtle

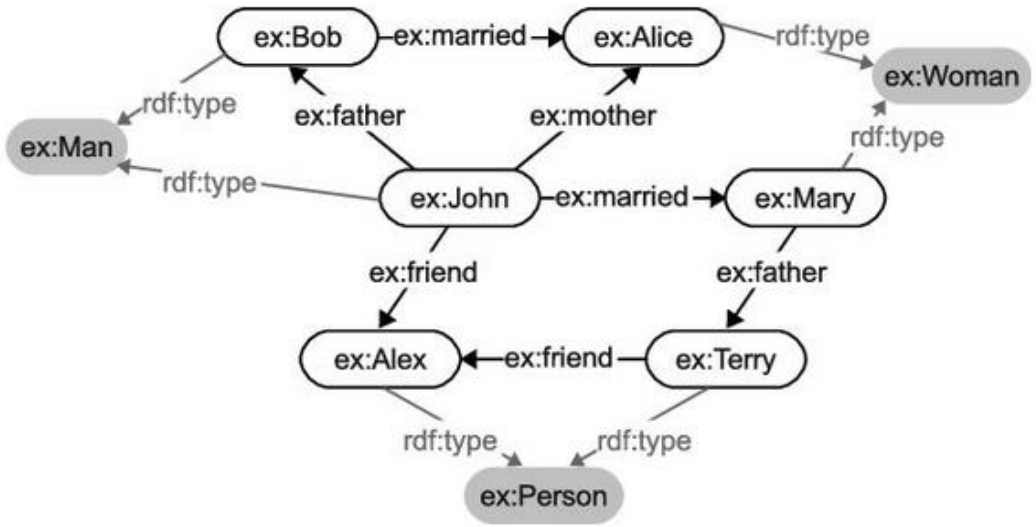
```
@base <http://www.animals.fake/animals> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix aml: <http://www.animals.fake/animals#> .
<aml#animal> a rdfs:Class .
<aml#horse> a rdfs:Class ;
            rdfs:subClassOf <aml#animal> .
<aml#camel> a rdfs:Class ;
            rdfs:subClassOf <aml#animal> .
```

RDFS, Example 2

RDFS



RDF



RDFS example serialised (RDF/XML)

```
<?xml version="1.0" encoding="utf-8" ?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">

  <rdfs:Class rdf:about="http://www.people.fake/people#pep:animal">
  </rdfs:Class>

  <rdfs:Class rdf:about="http://www.people.fake/people#pep:person">
    <rdfs:subClassOf rdf:resource="pep:animal"/>
  </rdfs:Class>

  <rdfs:Class rdf:about="http://www.people.fake/people#pep:man">
    <rdfs:subClassOf rdf:resource="pep:person"/>
  </rdfs:Class>

  <rdfs:Class rdf:about="http://www.people.fake/people#pep:woman">
    <rdfs:subClassOf rdf:resource="pep:person"/>
  </rdfs:Class>

  <rdfs:Property rdf:about="http://www.people.fake/people#pep:friend">
    <rdfs:domain rdf:resource="pep:person"/>
    <rdfs:range rdf:resource="pep:person"/>
  </rdfs:Property>

  <rdfs:Property rdf:about="http://www.people.fake/people#pep:relative">
    <rdfs:domain rdf:resource="pep:person"/>
    <rdfs:range rdf:resource="pep:person"/>
  </rdfs:Property>

  <rdfs:Property rdf:about="http://www.people.fake/people#pep:father">
    <rdfs:subPropertyOf rdf:resource="pep:relative"/>
    <rdfs:range rdf:resource="pep:man"/>
  </rdfs:Property>

  <rdfs:Property rdf:about="http://www.people.fake/people#pep:mother">
    <rdfs:subPropertyOf rdf:resource="pep:relative"/>
    <rdfs:range rdf:resource="pep:woman"/>
  </rdfs:Property>

  <rdfs:Property rdf:about="http://www.people.fake/people#pep:married">
    <rdfs:subPropertyOf rdf:resource="pep:relative"/>
  </rdfs:Property>

</rdf:RDF>
```

RDFS example serialised (Turtle)

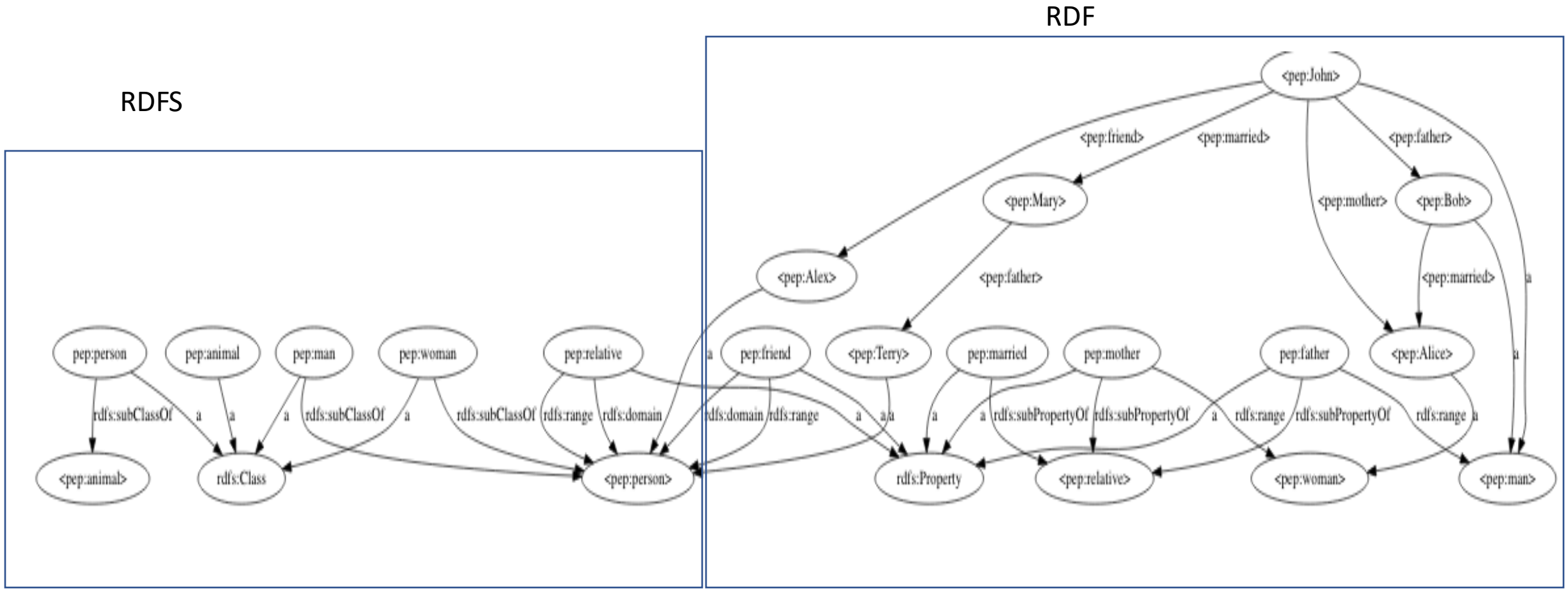
```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

<http://www.people.fake/people#pep:animal> a rdfs:Class .
<http://www.people.fake/people#pep:person>
  a rdfs:Class ;
  rdfs:subClassOf <pep:animal> .

<http://www.people.fake/people#pep:man>
  a rdfs:Class ;
  rdfs:subClassOf <pep:person> .

<http://www.people.fake/people#pep:woman>
  a rdfs:Class ;
  rdfs:subClassOf <pep:person> .

<http://www.people.fake/people#pep:friend>
  a rdfs:Property ;
  rdfs:domain <pep:person> ;
  rdfs:range <pep:person> .
```

```
<http://www.people.fake/people#pep:relative>
  a rdfs:Property ;
  rdfs:domain <pep:person> ;
  rdfs:range <pep:person> .

<http://www.people.fake/people#pep:father>
  a rdfs:Property ;
  rdfs:subPropertyOf <pep:relative> ;
  rdfs:range <pep:man> .

<http://www.people.fake/people#pep:mother>
  a rdfs:Property ;
  rdfs:subPropertyOf <pep:relative> ;
  rdfs:range <pep:woman> .

<http://www.people.fake/people#pep:married>
  a rdfs:Property ;
  rdfs:subPropertyOf <pep:relative> .
```

RDF in Turtle format

```
<pep:John>
  rdf:type <pep:man>;
 <pep:father> <pep:Bob>;
 <pep:mother> <pep:Alice>;
 <pep:married> <pep:Mary>;
 <pep:friend> <pep:Alex>.
<pep:Bob>
  rdf:type <pep:man>
 <pep:married> <pep:Alice>.
 <pep:Alice>
  rdf:type <pep:woman>.
<pep:Mary>
  <pep:father> <pep:Terry>.
 <pep:Terry>
  rdf:type <pep:person>.
 <pep:Alex>
  rdf:type <pep:person>.
```

The whole graph together



Schema vs. instance

- Schema
  - The **types of things** we can talk about (Classes)
  - What we can say about those things (Properties)
  - With RDFS, it is a shared terminology (vocabulary)
- Instance
  - The **actual things** we are describing
  - It is the actual data
  - It must respect the principles of open linked data
    - Use of IRI to identify things
    - resolvable HTTP IRIs
    - machine understandable format (RDF)
    - Data linked with external data (HTTPS IRIs)

Inference with RDFS

- Inference means discovering facts that are not explicitly stated by applying logical rules and properties (transitivity, symmetry …)
- With RDFS, we can do two basic types of inference:
  - Type inheritance (discover the class of an entity by following the inheritance tree)

```
:alice rdf:type foaf:Person
foaf:Person rdfs:subClassOf foaf:Agent
```

⇒

```
:alice rdf:type foaf:Agent
```

  - Type inference (discover discover the class of an entity by applying constraints)

```
foaf:Person rdf:knows foaf:Person
:alice rdfs:knows :bob
```

⇒

```
:alice rdf:type foaf:Person
:bob rdf:type foaf:Person
```

Characteristics of RDFS/RDF

- No distinction between classes and instances (individuals)
  - <Species, type, Class>
  - <Lion, type, Species>
  - <Leo, type, Lion>
- Properties can themselves have properties
  - <hasDaughter, subPropertyOf, hasChild>
  - <hasDaughter, type, familyProperty>
- No distinction between language constructors and ontology vocabulary, so constructors can be applied to themselves/ each other
  - <type, range, Class>
  - <Property, type, Class>
  - <type, subPropertyOf, subClassOf>

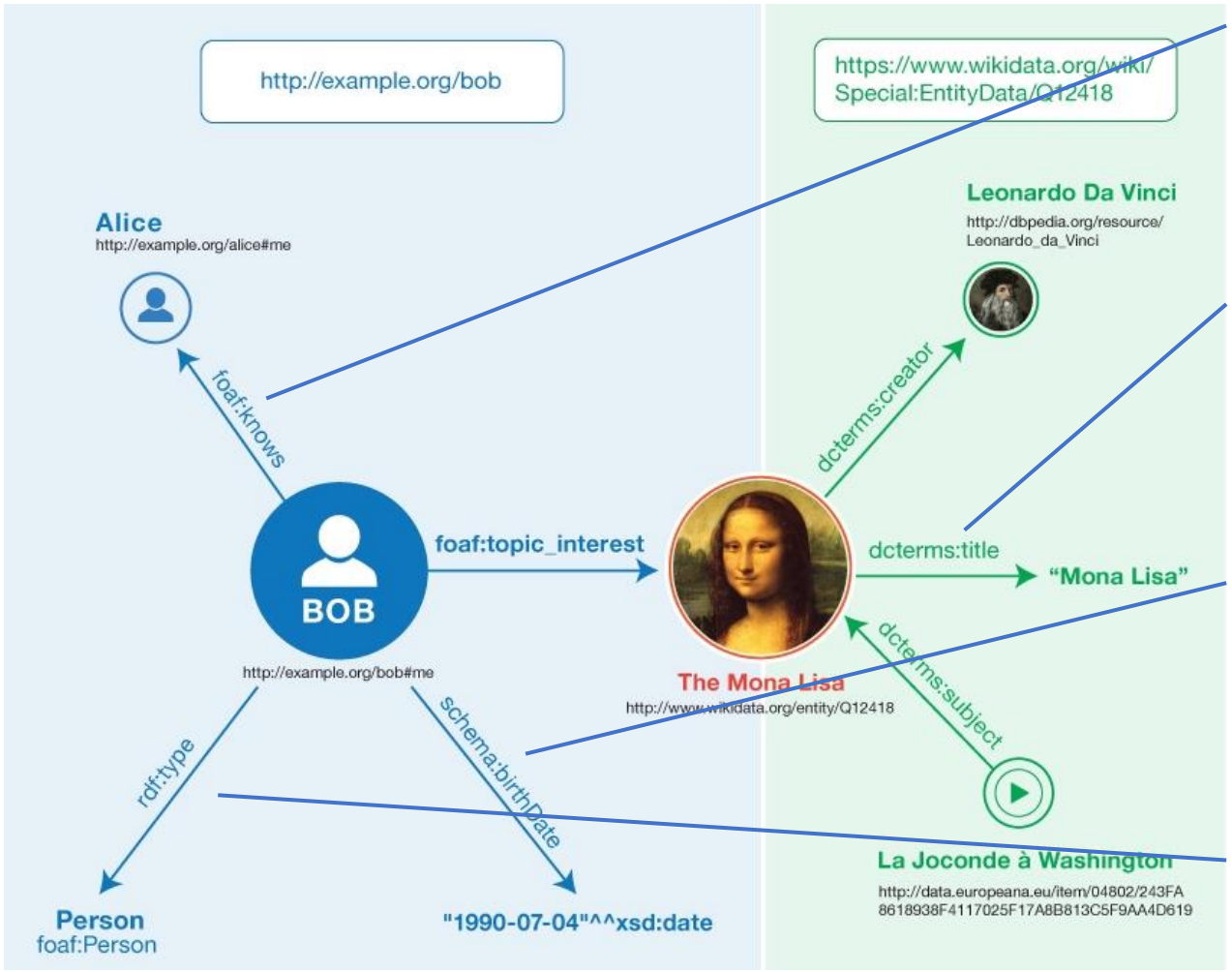
RDFS vocabulary

- RDFS allows defining vocabularies. A vocabulary is a data model
- A vocabulary sets of *terms* used to describe things, defined using classes, relationships and attributes
- **Class:** Represents things in the real world
  - a person, an organisation, concepts such as “health”, “freedom”…
- **Relationship**: A link between two classes
  - link between a document and the organisation that published it (i.e. organisation publishes document),
  - link between a map and the geographic region it depicts (i.e. map depicts geographic region)
- **Attribute**: A characteristic of a class in a particular dimension
  - the legal name of an organisation or the date and time that an observation was made

Vocabularies used with RDF/RDFS

- RDF uses various existing namespaces (vocabularies)
  - RDF: http://www.w3.org/1999/02/22-rdf-syntax-ns#
    - Elements and properties of RDF
  - DublinCore: http://purl.org/dc/elements/1.1/
    - Document metadata, such as author, title…
  - SKOS: http://www.w3.org/2004/02/skos/core#
    - Structing vocabularies (taxonomy, thesauri…)
  - FOAF: http://xmlns.com/foaf/0.1/
    - Persons, their activities and their relations to objects

Example of RDF and external terms



FOAF

DCTerms

Schema.org

RDF

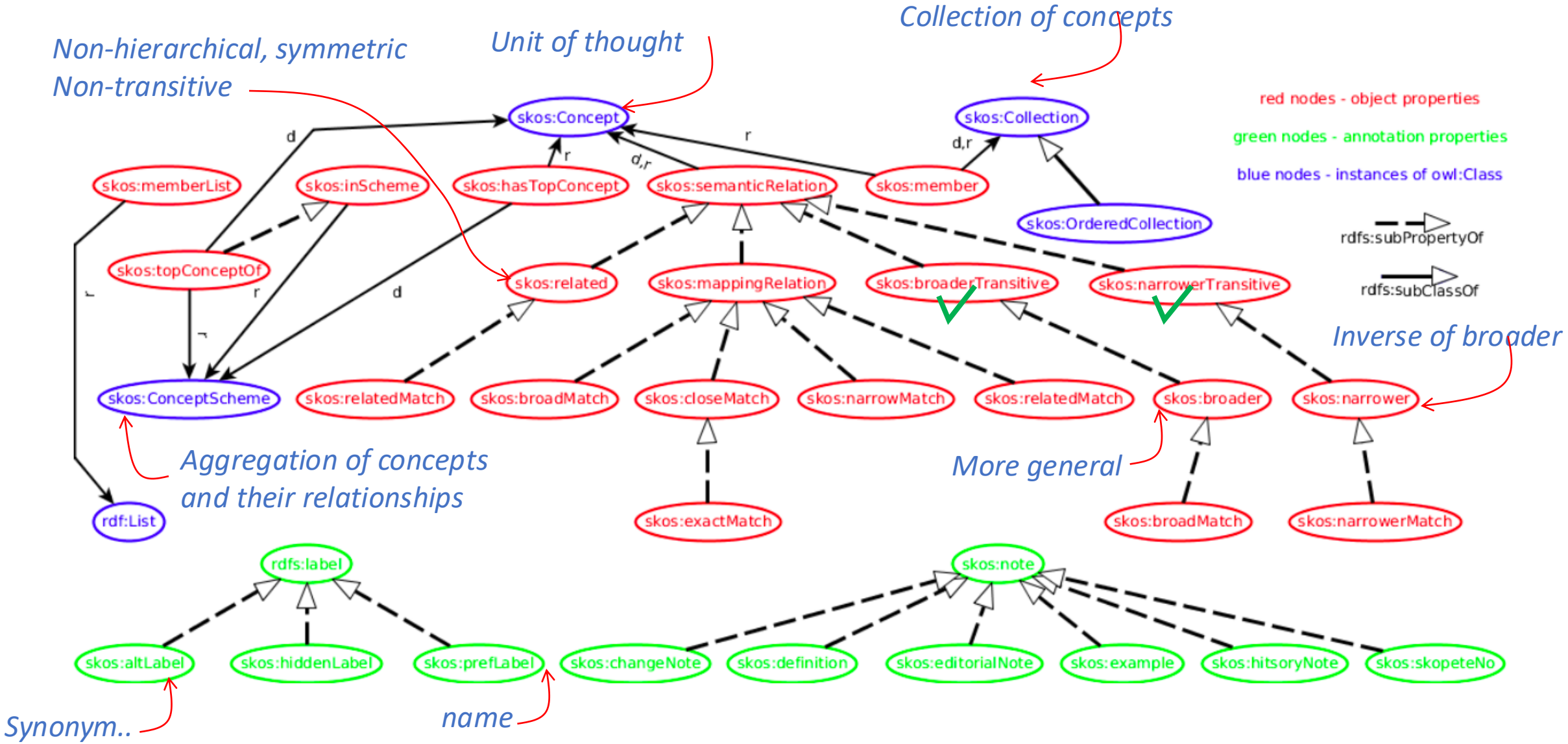
Dublin core

- Defines basic metadata that can be used to briefly describe most resources (files, data, services …)
  - E.g.
    - The **title** of the resource: **dc:title**
    - The **creator/author** of the resource: **dc:creator**
    - A human-readable **description**: **dc:description**
    - **Date** of creation: **dc:date**
- Complete list on http://dublincore.org

SKOS: Simple Knowledge Organization System

- A standard RDF vocabulary (2009)
- It defines knowledge organisation systems
- It allows to express thesauri, classification systems, subject headings, lists ..
- Controlled vocabularies do not have relationships
- SKOS allows defining thesauri and taxonomies that have relationships

Partial SKOS model



SKOS: Simple Knowledge Organization System

The fundamental element of the SKOS vocabulary is the *concept*.

```
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix ex: <http://www.example.com/> .
 ex:cat rdf:type skos:Concept.
```

The Concept is defined by a label in natural language.

```
ex:cat rdf:type skos:Concept;
        skos:prefLabel "cat"@en;
```

SKOS: Simple Knowledge Organization System

Semantic relationships:

- Hierarchical via properties skos:broader, skos:narrower

  For explicit transitivity use

  skos:broaderTransitive

  skos:narrowerTransitive
- Associative via property skos:related
  - is not defined as a transitive property

SKOS and transitivity

- Relationship p is transitive if

```
:r1 :p :r2
:r2 :p :r3
```

⇒

```
:r1 :p :r3
```

- Relationship p is intransitive if

```
:r1 :p :r2
:r2 :p :r3
```

⇒

```
Not(:r1 :p :r3)
```

- Relationship p is neither transitive nor intransitive

```
:r1 :p :r2
:r2 :p :r3
```

⇒

```
Nothing is known on p(r1,r3)
```

- SKOS relationships are neither transitive nor intransitive

SKOS Transitive relationships

- SKOS defines explicit transitive relationships
  - skos:broaderTransitive, skos:tnarrowerTransitive
- Example

```
:animal rdf:type skos:Concept.
:mammal rdf:type skos:Concept;
   skos:broaderTransitive :animal.
:cat rdf:type skos:Concept;
   skos:broaderTransitive :mammal.
```

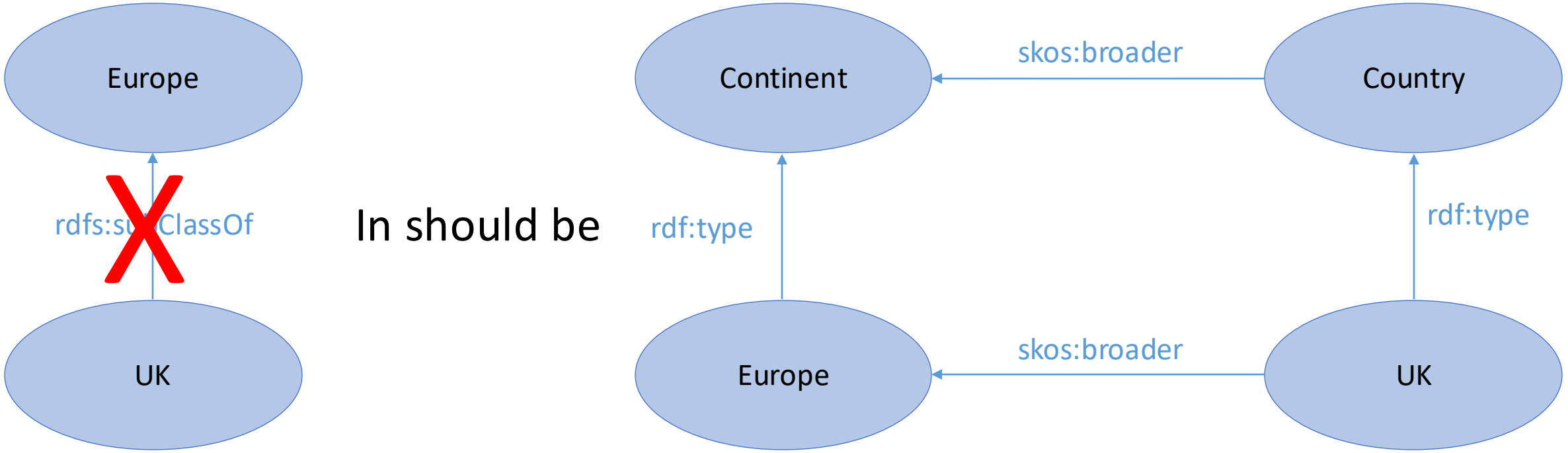
⇒

```
:cat skos:broader :animal.
```

SKOS: Simple Knowledge Organization System

```
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix ex: <http://www.example.com/> .

ex:cat rdf:type skos:Concept;
  skos:prefLabel "cat"@en;
  skos:prefLabel "Katze"@de;
  skos:altLabel "kitten"@en;
  skos:narrower ex:wildcat.

ex:wildcat rdf:type skos:Concept;
  skos:prefLabel "wildcat"@en;
  skos:broader ex:cat.
```

Typical mistake



Disjoint properties

- Two properties are disjoint if they cannot occur at the same time
- In SKOS, skos:related is disjoint with both skos:broader and skos:narrower
- The following is inconsistent

```
:r1 rdf:type skos:Concept;
      skos:related :r3.
:r2 rdf:type skos:Concept;
      skos:narrowerTransitive :r1.
:r3 rdf:type skos:Concept;
      skos:narrowerTransitive :r2.
```

Because by transitivity

```
:r3 skos:narrower :r1.
```

Therefore, we have at the same time

```
:r1 rdf:type skos:Concept;
      skos:related :r3.
:r3 skos:narrower :r1.
```

Where narrower and related are disjoint

SKOS and similarity between concepts

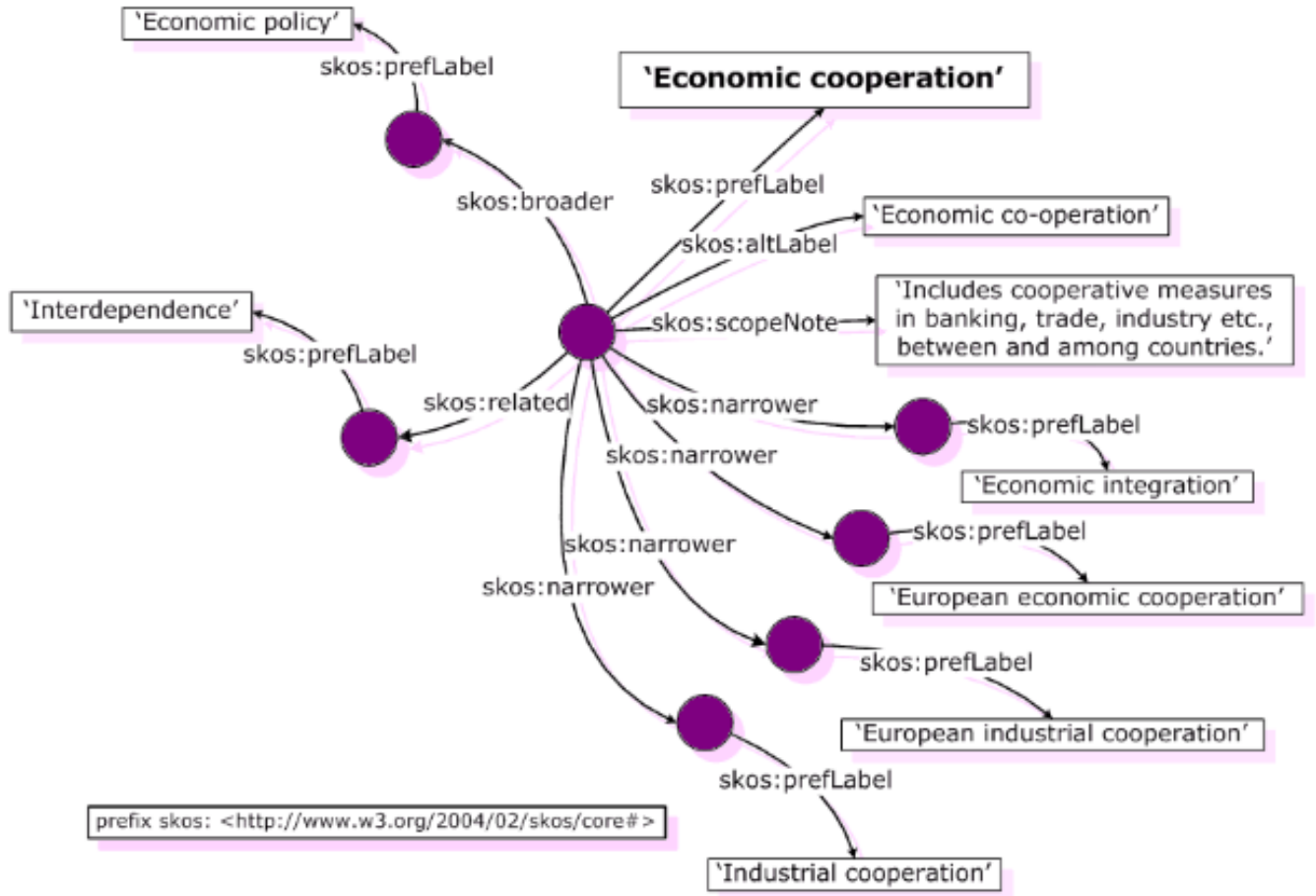
property	description
:c1 skos:exactMatch :c2	c1 and c2 similar concepts, can be used interchangeably
:c1 skos:closeMatch :c2	c1 and c2 almost similar concepts, can be used interchangeably in some situations
:c1 owl:sameAs :c2	**c1 and c2 are one and the same thing**

This relationships can be used to align (or integrate) two ontologies (to be seen later)

Skos example

Part of the UK Archival Thesaurus (UKAT)

Associated RDF/SKOS

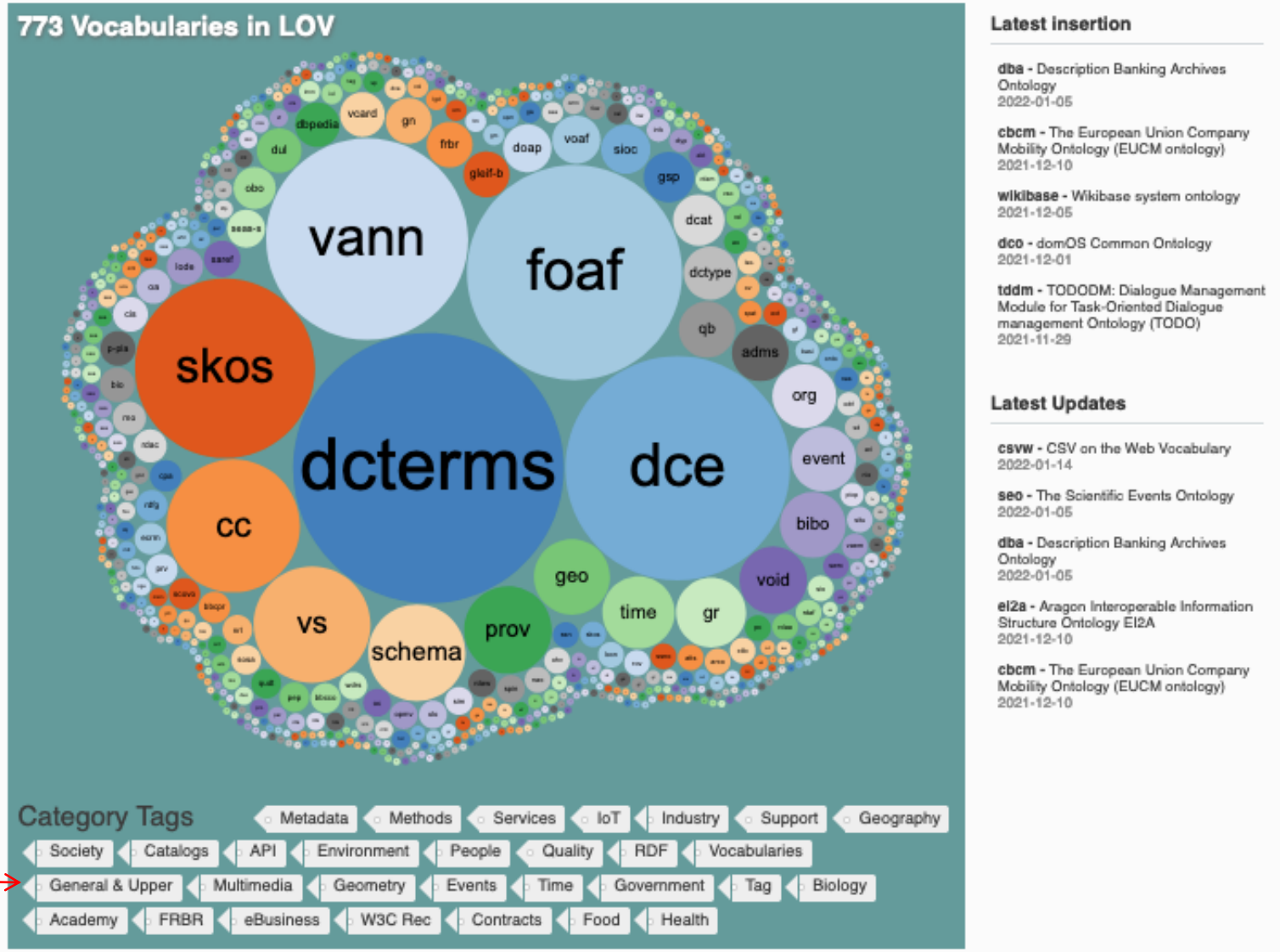


SKOS reasoning

- With SKOS, we can do the following reasoning
- Check consistency
  - Example, x broader than y, and y broader than x
- Trivial entailments
  - Example, x broader than y, then y narrower than x
- Transitivity
  - Example, x narrowerTransitive than y, y narrowerTransitive than z, then x narrower than z

Linked open vocabularies

Repository of vocabularies (a European project): https://lov.linkeddata.es/dataset/lov/



Searchable by category

Latest insertion

dba - Description Banking Archives Ontology
2022-01-05

cbcm - The European Union Company Mobility Ontology (EUCM ontology)
2021-12-10

wikibase - Wikibase system ontology
2021-12-05

dco - domOS Common Ontology
2021-12-01

tddm - TODODM: Dialogue Management Module for Task-Oriented Dialogue management Ontology (TODO)
2021-11-29

Latest Updates

csvw - CSV on the Web Vocabulary
2022-01-14

seo - The Scientific Events Ontology
2022-01-05

dba - Description Banking Archives Ontology
2022-01-05

ei2a - Aragon Interoperable Information Structure Ontology EI2A
2021-12-10

cbcm - The European Union Company Mobility Ontology (EUCM ontology)
2021-12-10

Limitations of RDFS

- RDFS cannot describe resources with sufficient details
  - *cannot express that the range of hasChild is person when applied to persons and elephant when applied to elephants*
    - Requires **localised range and domain** constraints
  - *cannot express that all* instances *of person have a mother that is also a person, or that persons have exactly 2 parents*
    - Requires **existence/cardinality** constraints
  - *cannot express that isPartOf is a transitive property, that hasPart is the inverse of isPartOf or that touches is symmetrical*
    - Requires **transitive, inverse or symmetrical** properties
- Difficult to provide reasoning support
  - No “native” reasoners

Next