Previously on Introduction to Linked Data…

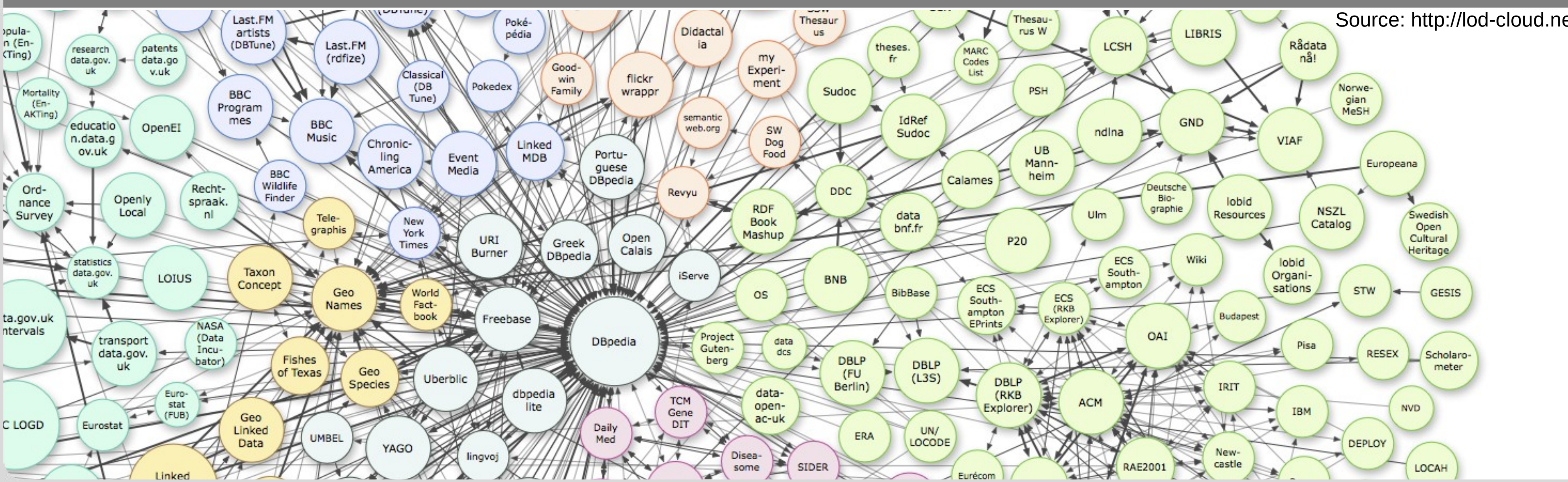
- Data modelling and vocabulary descriptions in RDF and RDFS
  - Simple, *D*-, RDF and RDFS entailment is defined based on semantic conditions on interpretations
  - Implementing RDF and RDFS entailment involves entailment patterns (if-then)
- Major features in RDFS are subclass/subproperty relations and domain and range of properties
  - Semantics is defined in a way to “expand” interpretations so that the semantic conditions are satisfied
  - E.g., domain restrictions cause subject to be of a specified type
  - E.g., range restrictions cause object to be of a specified type
- Unsatisfiability as last resort
  - Up to RDFS entailment, unsatisfiability always involves ill-typed literals

C10 Data Modelling with the Web Ontology Language
How to describe things in a way that supports data integration?

Version 2022-07-05

Lecturer: Prof. Dr. Andreas Harth

CHAIR OF TECHNICAL INFORMATION SYSTEMS



Source: http://lod-cloud.net

CC - Creative Commons Licensing

- This set of slides is part of the lecture „Semantic Web Technologies“ held at Karlsruhe Institute of Technology
- The content of the lecture was prepared by PD Dr. Andreas Harth based on his book „Introduction to Linked Data“
- The slides were prepared by Maribel Acosta and Andreas Harth with input from Lars Gleim

Outline

1. **OWL Overview**
2. Datatype Support
3. Equality and Inequality Characteristics
4. Class Characteristics
5. Property Characteristics
6. Additional Vocabulary Terms
7. OWL for Data Integration

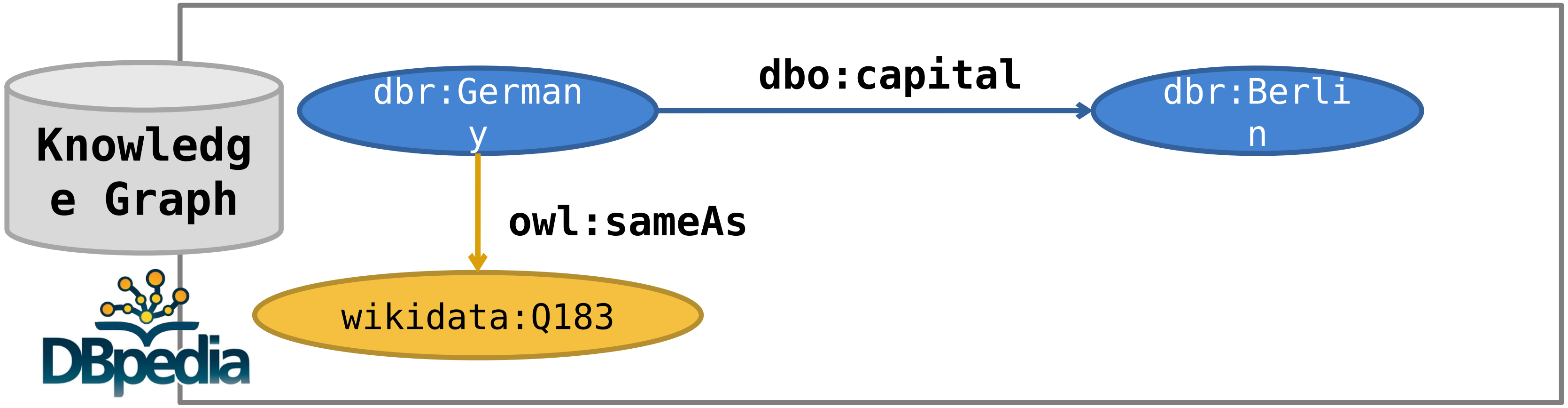
Motivation

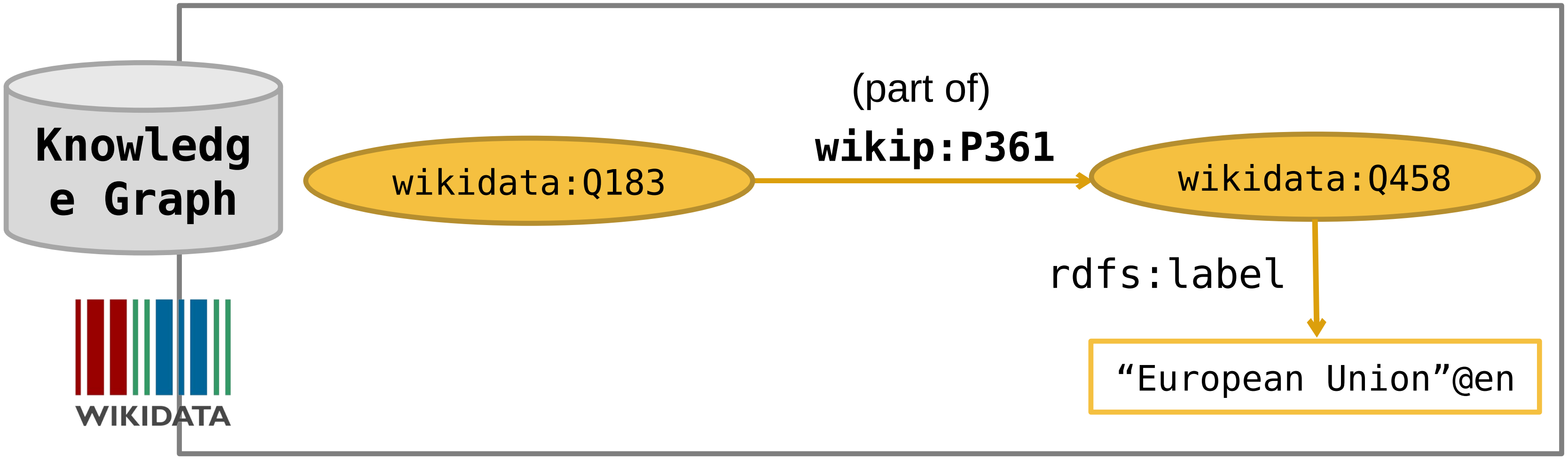
- The web is based on a decentralised architecture
- Anybody can say anything about everything
- Links connect disparate sites

- We have seen how to specify equivalent classes via reciprocal rdfs:subClassOf statements
- Useful for mapping classes from different sources

- We need a way to specify equality on the instance level
- Useful for mapping instances from different sources
- The single-most used construct from the OWL vocabulary on the Linked Data web is owl:sameAs

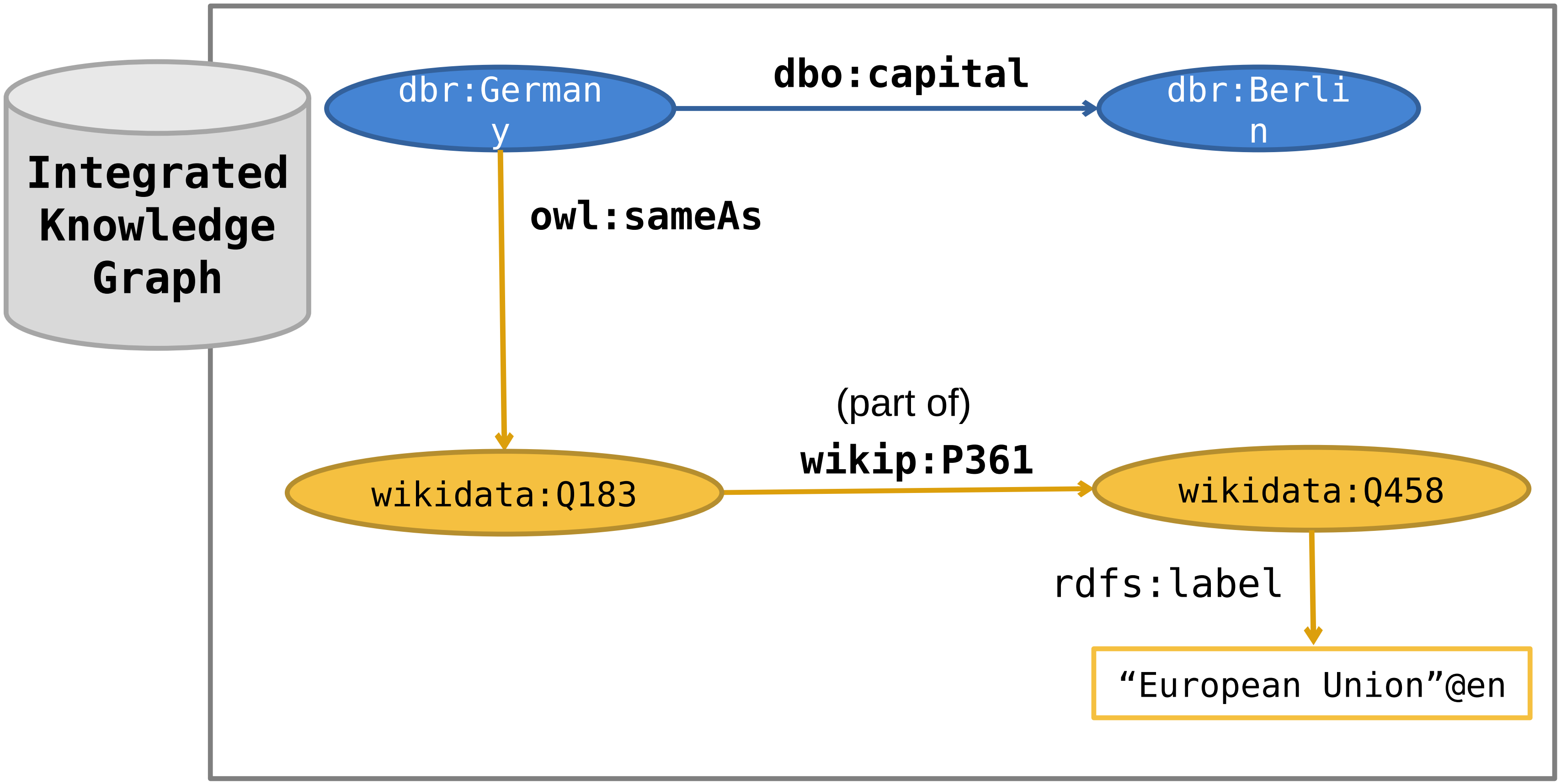
Motivation: Data Integration (1)





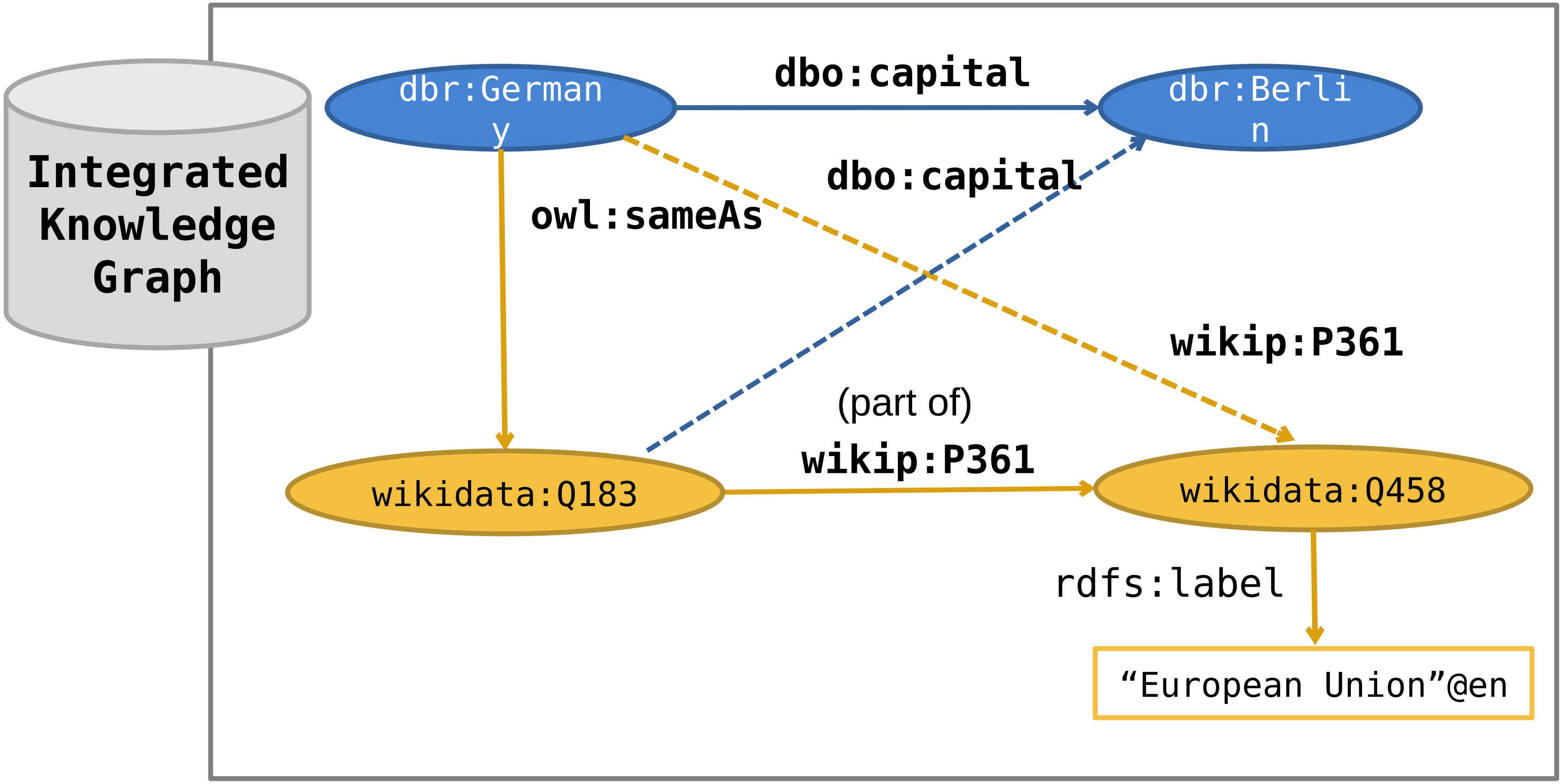
Source: Maribel Acosta. Semantic Data Management: Challenges and Opportunities. Tutorial at the Wissensmanagement Konferenz. 2017

Motivation: Data Integration (2)

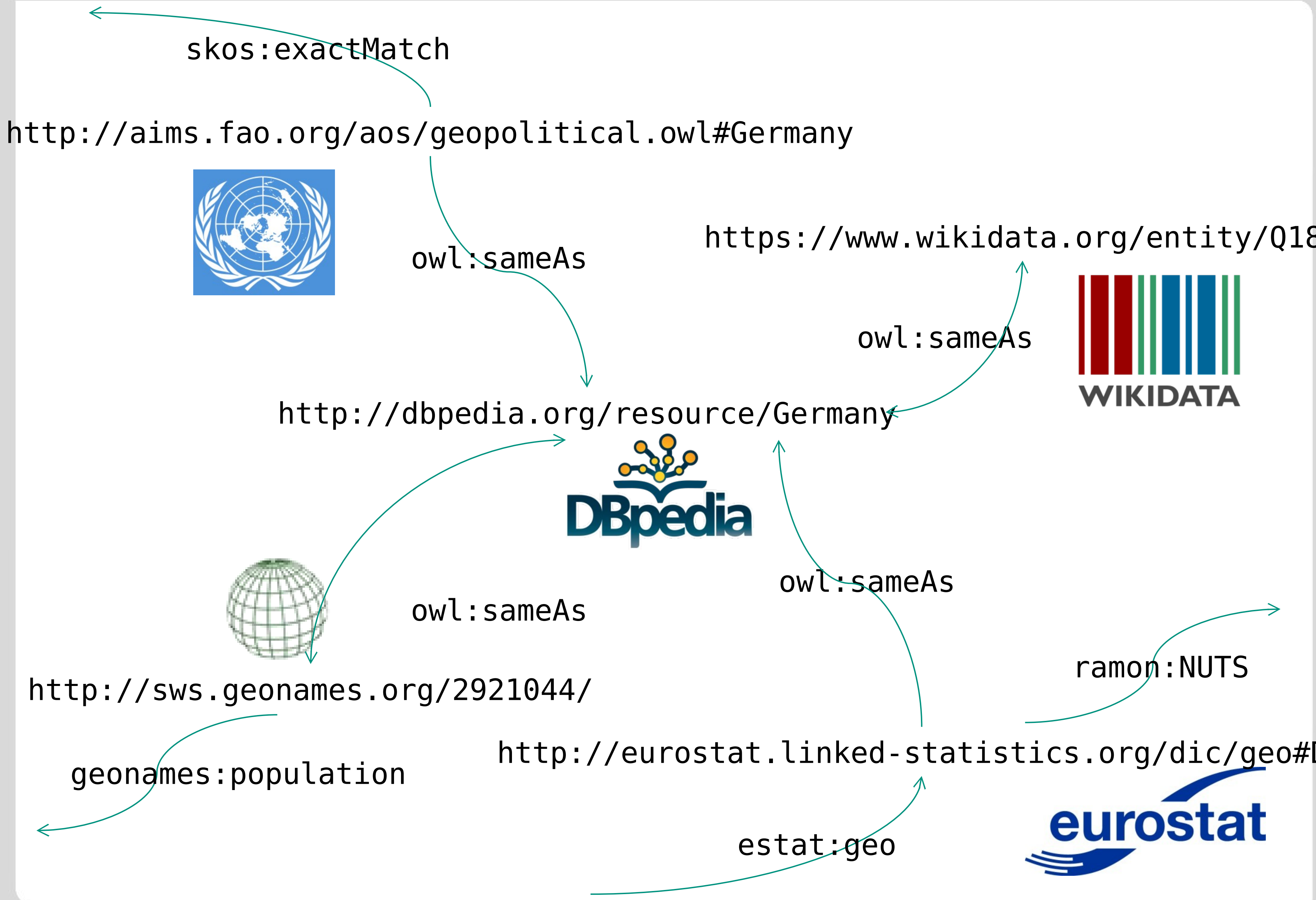


Source: Maribel Acosta. Semantic Data Management: Challenges and Opportunities. Tutorial at the Wissensmanagement Konferenz. 2017

Motivation: Data Integration & Reasoning



Source: Maribel Acosta. Semantic Data Management: Challenges and Opportunities. Tutorial at the Wissensmanagement Konferenz. 2017


skos:exactMatch
http://aims.fao.org/aos/geopolitical.owl#Germany
owl:sameAs
https://www.wikidata.org/entity/Q18
owl:sameAs
WIKIDATA
http://dbpedia.org/resource/Germany
DBpedia
owl:sameAs
owl:sameAs
ramon:NUTS
http://sws.geonames.org/2921044/
geonames:population
http://eurostat.linked-statistics.org/dic/geo#D
eurostat
estat:geo

OWL: The Web Ontology Language

- OWL acronym for Web Ontology Language, more easily pronounced than WOL
- Family of languages for authoring ontologies
- Since 2004, OWL 2 since 2009
- Works on the RDF triple data model

- Semantic fragment of first order logic
- Formal logic with a computational character are always a compromise between expressivity and computational complexity
- OWL comes in **different profiles** which balance the expressivity with computational complexity.
- OWL 1: OWL Lite, OWL DL, OWL Full
- OWL 2: OWL 2 EL, OWL 2 QL, OWL 2 RL

Two Families of OWL Profiles

- Formal languages need a clear semantics
- Two families of OWL profiles based on different ways to specify the semantics of terms in OWL
- One can reason with OWL 2 ontologies under either
  - the RDF-based Semantics or
  - the Direct Semantics.
- The Direct Semantics requires that OWL 2 ontologies adhere to certain syntactic restrictions
- The RDF-based semantics builds on definitions around RDF model theory, RDF and RDFS. We will use the RDF-based semantics.

OWL Linked Data

- We will focus on the OWL LD profile, a subset of OWL 2 RL with semantics given compatible with the semantics of the RDF and RDFS vocabularies.

- Like RDFS, OWL has the concepts of class, property and instance. The OWL vocabulary is made up of terms which provide for:
  - Equality and Inequality Characteristics
  - Class Characteristics
  - Property Characteristics

- OWL supports datatypes
- OWL includes partial support for RDF and RDFS vocabularies

- OWL allows for more advanced entailment than RDF and RDFS

OWL LD Vocabulary Terms

(In-)Equality	Classes	Properties	RDF Schema URIs
owl:sameAs	owl:equivalentClass	owl:equivalentProperty	rdfs:subClassOf
owl:differentFrom	owl:disjointWith	owl:SymmetricProperty	rdfs:subPropertyOf
		owl:TransitiveProperty	rdfs:domain
		owl:inverseOf	rdfs:range
		owl:FunctionalProperty	
		owl:InverseFunctionalProperty	
		owl:AsymmetricProperty	
		owl:IrreflexiveProperty	
		owl:propertyDisjointWith	

Individuals, Classes, and Properties

- Ontologies consists of the following:
  - **Individuals:**
    Concrete objects in a modelled world
  - **Classes:**
    Set of individuals
  - **Roles/Properties:**
    Associations between two individuals

Think-Pair-Share

- In the Marvel Universe, there are human and mutant persons. People have names. Magneto is a mutant person, Moira MacTaggert is a human person.

```
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix : <#> .
```

Think-Pair-Share

- In the Marvel Universe, there are human and mutant persons. People have names. Magneto is a mutant person, Moira MacTaggert is a human person.

```
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix : <#> .

:MutantPerson  rdfs:subClassOf  :Person .
:HumanPerson   rdfs:subClassOf  :Person .
:name          rdfs:domain      :Person ;
               rdfs:range       xsd:string .
:Magneto       rdf:type         :MutantPerson ;
               :name            "Magneto"^^xsd:string .
:Moira         rdf:type         :HumanPerson ;
               :name            "Moira MacTaggert"^^xsd:string .
```

Outline

1. OWL Overview
2. **Datatype Support**
3. Equality and Inequality Characteristics
4. Class Characteristics
5. Property Characteristics
6. Additional Vocabulary Terms
7. OWL for Data Integration

Datatypes

- Similar to *D*-entailment, OWL LD supports datatypes built on XML Schema Datatypes (XSD)

- **xsd:anyURI**
- **xsd:base64Binary**
- **xsd:boolean**
- **xsd:dateTime**
  - xsd:dateTimeStamp
- **xsd:decimal**
  - xsd:integer
    - xsd:long
      - xsd:int
        - xsd:short
          - xsd:byte
    - xsd:nonNegativeInteger
      - xsd:positiveInteger
      - xsd:unsignedLong
        - xsd:unsignedInt
          - xsd:unsignedShort
            - xsd:unsignedByte
    - xsd:nonPositiveInteger
      - xsd:negativeInteger
- **xsd:double**
- **xsd:float**
- **xsd:hexBinary**
- **xsd:string**
  - xsd:normalizedString
    - xsd:token
      - xsd:language
      - xsd:Name
        - xsd:NCName
    - xsd:NMTOKEN

Datatype Entailment Patterns

	if G contains	then G OWL LD-entails
dt-type1	(for each supported datatype ?dt in D)	?dt a rdfs:Datatype .
dt-type2	(for each literal ?lt in the value space of datatype ?dt)	?lt a ?dt .
dt-eq	(for all ?lt1 and ?lt2 with the same data value)	?lt1 owl:sameAs ?lt2 .
dt-diff	(for all ?lt1 and ?lt2 with different data values)	?lt1 owl:differentFrom ?lt2 .
dt-not-type	?lt a ?dt . (where ?lt is not in the value space of ?dt)	false

Outline

1. OWL Overview
2. Datatype Support
3. **Equality and Inequality Characteristics**
4. Class Characteristics
5. Property Characteristics
6. Additional Vocabulary Terms
7. OWL for Data Integration

Equality and Inequality

- OWL individuals represent instances of classes
- They are related to their class by the **`rdf:type`** property (RDF Schema)
- We can explicitly state that two individuals are the same.

```
dbr:Germany owl:sameAs wikidata:Q183 .
```

- We can explicitly state that two individuals are different.

```
dbr:Germany owl:differentFrom dbr:German_Empire .
```

Entailment Patterns

	if G contains	then G OWL LD-entails
eq-ref	?s ?p ?o .	?s owl:sameAs ?s . ?p owl:sameAs ?p . ?o owl:sameAs ?o .
eq-sym	?x owl:sameAs ?y .	?y owl:sameAs ?x .
eq-trans	?x owl:sameAs ?y . ?y owl:sameAs ?z .	?x owl:sameAs ?z .
eq-rep-s	?s owl:sameAs ?so . ?s ?p ?o .	?so ?p ?o .
eq-rep-p	?p owl:sameAs ?po . ?s ?p ?o .	?s ?po ?o .
eq-rep-o	?o owl:sameAs ?oo . ?s ?p ?o .	?s ?p ?oo .
eq-diff1	?x owl:sameAs ?y ; owl:differentFrom ?y .	false

Outline

1. OWL Overview
2. Datatype Support
3. Equality and Inequality Characteristics
4. **Class Characteristics**
5. Property Characteristics
6. Additional Vocabulary Terms
7. OWL for Data Integration

OWL Class Characteristics

- **Equivalent relationship** (classes have the same individuals).
  - Example: *Every human is a person, and every person is a human*.

```
:Human owl:equivalentClass :Person .
:Alice rdf:type :Human .
:Alice rdf:type :Person .
```

- **Disjointness** (classes have no shared individuals).
  - Example: *Men are not dogs.*

```
:Man owl:disjointWith :Dog .
```

Entailment Patterns

	if G contains	then G OWL LD-entails
cax-sco	?c1 rdfs:subClassOf ?c2 . ?x a ?c1 .	?x a ?c2 .
cax-eqc1	?c1 owl:equivalentClass ?c2 . ?x a ?c1 .	?x a ?c2 .
cax-eqc2	?c1 owl:equivalentClass ?c2 . ?x a ?c2 .	?x a ?c1 .
cax-dw	?c1 owl:disjointWith ?c2 . ?x a ?c1 , ?c2 .	false

Outline

1. OWL Overview
2. Datatype Support
3. Equality and Inequality Characteristics
4. Class Characteristics
5. **Property Characteristics**
6. Additional Vocabulary Terms
7. OWL for Data Integration

Property Characteristics

- Apart from the sub-property relationship from RDFS, OWL also allows for expressing other types of property axioms.

Property Characteristics

- **Equivalent** properties
- **Symmetric** property
  r(a, b) implies r(b, a)
- **Transitive** property
  r(a, b) and r(b, c) implies r(a,c)
- **Inverse** properties
- **Functional** property
  r(a, b) and r(a, c) implies b=c
- **Inverse functional** property
  r(a, b) and r(c, b) implies a=c
- **Asymmetric** properties
- **Irreflexive** property
- **Disjoint** properties

Equivalent Properties

- Equivalent properties have the same values
- Specified with `owl:equivalentProperty`
- Example: *If somebody is an university employee* (`:isUniEmployee`)*, then they work for the university* (`:worksForUni`) *and vice versa.*

```
      :isUniEmployee
owl:equivalentProperty :worksForUni.
```

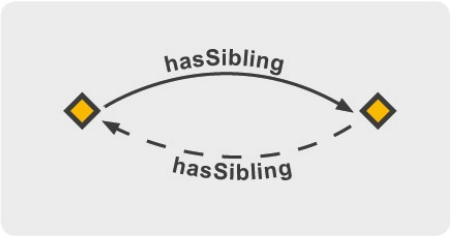
- **Entailment:**

```
      :isUniEmployee
owl:equivalentProperty :worksForUni.
       :Maribel :isUniEmployee :KIT .
       :Maribel :worksForUni :KIT .
```

Symetric, Funcational and Inverse Functional Properties

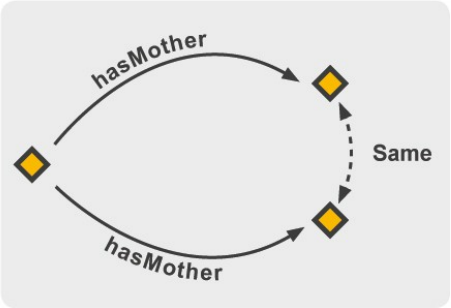
- **Symmetry** (if X p Y then Y p X)

```
:hasSibling rdf:type owl:SymmetricProperty .
:Bob :hasSibling :Charlie .
:Charlie :hasSibling :Bob .
```



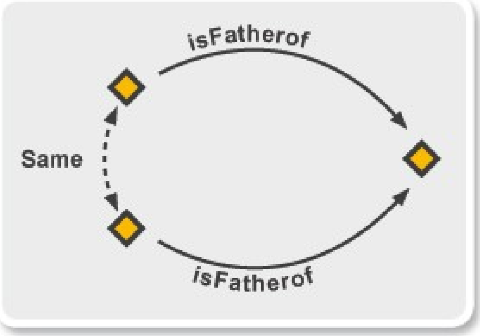
- **Functional** (if X p Y and X p Z then Y = Z)

```
:hasMother rdf:type owl:FunctionalProperty
:Bob :hasMother :Alice .
:Bob :hasMother :Al .
:Alice owl:sameAs :Al .
```



- **Inverse Functional** (if X p Z and Y p Z then X = Y)

```
:isFatherOf rdf:type owl:InverseFunctionalPrope
:Bob :isFatherOf :Mike .
:Robert :isFatherOf :Mike .
:Bob owl:sameAs :Robert .
```



Inverse Properties

- Useful to define relations between subjects/objects and objects/subjects

- Specified with `owl:inverseOf`

- Example: *The relation "has child" is the inverse of "has parent".*

```
:hasChild owl:inverseOf :hasParent .
```

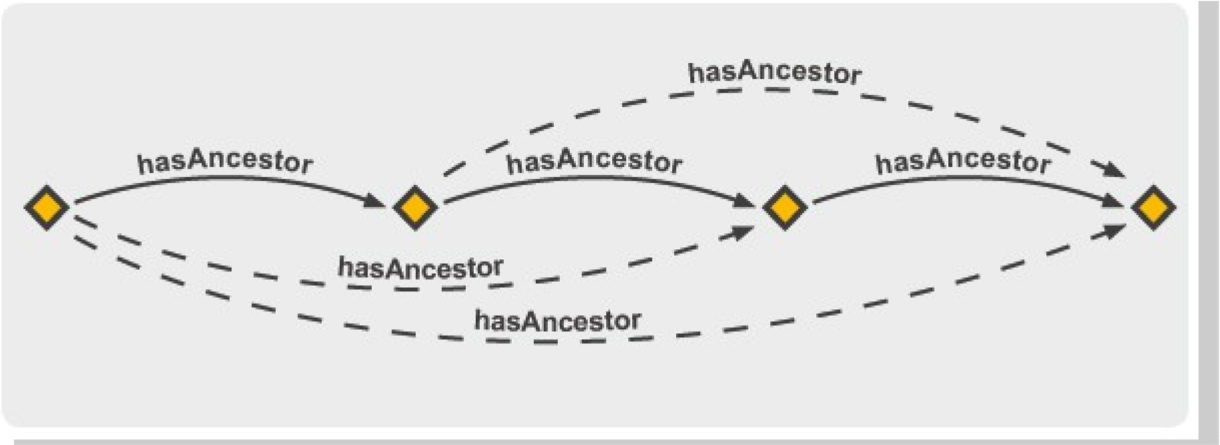
- **Entailment:**

```
:hasChild owl:inverseOf :hasParent .
:Alice owl:hasChild :Bob .
:Bob :hasParent :Alice .
```

Transitive Property

- OWL introduces property characteristics for more expressivity in inferring characteristics relating to instances and their properties

- **Transitivity** (if X p Y and Y p Z, then X p Z)




```
:hasAncestor rdf:type owl:TransitiveProperty .
:Alice :hasAncestor :Tim .
:Tim :hasAncestor :Rob .
:Alice :hasAncestor :Rob .
```

Entailment Patterns

	if *G* contains	then *G* OWL LD-entails
prp-dom	?p rdfs:domain ?c . ?x ?p ?y .	?x a ?c .
prp-rng	?p rdfs:range ?c . ?x ?p ?y .	?y a ?c .
prp-fp	?p a owl:FunctionalProperty . ?x ?p ?y1 , ?y2 .	?y1 owl:sameAs ?y2 .
prp-ifp	?p a owl:InverseFunctionalProperty . ?x1 ?p ?y . ?x2 ?p ?y .	?x1 owl:sameAs ?x2 .
prp-irp	?p a owl:IrreflexiveProperty . ?x ?p ?x .	false
prp-symp	?p a owl:SymmetricProperty . ?x ?p ?y .	?y ?p ?x .
prp-asyp	?p a owl:AsymmetricProperty . ?x ?p ?y . ?y ?p ?x .	false
prp-trp	?p a owl:TransitiveProperty . ?x ?p ?y . ?y ?p ?z .	?x ?p ?z .
prp-spo1	?p1 rdfs:subPropertyOf ?p2 . ?x ?p1 ?y .	?x ?p2 ?y .
prp-eqp1	?p1 owl:equivalentProperty ?p2 . ?x ?p1 ?y .	?x ?p2 ?y .
prp-eqp2	?p1 owl:equivalentProperty ?p2 . ?x ?p2 ?y .	?x ?p1 ?y .
prp-pdw	?p1 owl:propertyDisjointWith ?p2 . ?x ?p1 ?y ; ?p2 ?y .	false
prp-inv1	?p1 owl:inverseOf ?p2 . ?x ?p1 ?y .	?y ?p2 ?x .
prp-inv2	?p1 owl:inverseOf ?p2 . ?x ?p2 ?y .	?y ?p1 ?x .

Outline

1. OWL Overview
2. Datatype Support
3. Equality and Inequality Characteristics
4. Class Characteristics
5. Property Characteristics
6. **Additional Vocabulary Terms**
7. OWL for Data Integration

Schema Vocabulary Entailment Patterns

	if G contains	then G OWL LD-entails
scm-sco	?c1 rdfs:subClassOf ?c2 . ?c2 rdfs:subClassOf ?c3 .	?c1 rdfs:subClassOf ?c3 .
scm-eqc1	?c1 owl:equivalentClass ?c2 .	?c1 rdfs:subClassOf ?c2 . ?c2 rdfs:subClassOf ?c1 .
scm-eqc2	?c1 rdfs:subClassOf ?c2 . ?c2 rdfs:subClassOf ?c1 .	?c1 owl:equivalentClass ?c2 .
scm-op	?p a owl:ObjectProperty .	?p rdfs:subPropertyOf ?p ; owl:equivalentProperty ?p .
scm-dp	?p a owl:DatatypeProperty .	?p rdfs:subPropertyOf ?p ; owl:equivalentProperty ?p .
scm-spo	?p1 rdfs:subPropertyOf ?p2 . ?p2 rdfs:subPropertyOf ?p3 .	?p1 rdfs:subPropertyOf ?p3 .
scm-eqp1	?p1 owl:equivalentProperty ?p2 .	?p1 rdfs:subPropertyOf ?p2 . ?p2 rdfs:subPropertyOf ?p1 .
scm-eqp2	?p1 rdfs:subPropertyOf ?p2 . ?p2 rdfs:subPropertyOf ?p1 .	?p1 owl:equivalentProperty ?p2 .
scm-dom1	?p rdfs:domain ?c1 . ?c1 rdfs:subClassOf ?c2 .	?p rdfs:domain ?c2 .
scm-dom2	?p2 rdfs:domain ?c . ?p1 rdfs:subPropertyOf ?p2 .	?p1 rdfs:domain ?c .
scm-rng1	?p rdfs:range ?c1 . ?c1 rdfs:subClassOf ?c2 .	?p rdfs:range ?c2 .
scm-rng2	?p2 rdfs:range ?c . ?p1 rdfs:subPropertyOf ?p2 .	?p1 rdfs:range ?c .

OWL Classes

- Two pre-defined classes:
  - `owl:Thing` (class that contains all individuals)
  - `owl:Nothing` (empty class)

- The relation between `owl:Class` and `rdfs:Class` is complicated[1]

- Short answer: keep using `rdfs:Class`

[1] https://lists.w3.org/Archives/Public/www-rdf-comments/2003JulSep/0331.html

Unsatisfiability

- A statement in an OWL ontology must be satisfiable with regards to the other statements in the ontology.

- e.g. if an individual belongs to two disjoint classes.

```
:Cat owl:disjointWith Dog .
 :Bob a :Cat, :Dog .
```

- e.g., if two individuals are the same and different at the same time.

```
:Bob owl:sameAs :Robert .
:Bob owl:differentFrom :Robert .
```

- The previous statements lead to an **unsatisfiable** ontology.

Think-Pair-Share

- Dedice whether the following lead to an unsatisfiable graph under OWL LD entailment or not.

a)

```
:hasMother a owl:FunctionalProperty .
:Bob :hasMother :Al, :Alice .
```

b)

```
:hasMother a owl:FunctionalProperty .
:Bob :hasMother :Al, :Alice .
:Al owl:differentFrom :Alice .
```

Outline

1. OWL Overview
2. Datatype Support
3. Equality and Inequality Characteristics
4. Class Characteristics
5. Property Characteristics
6. Additional Vocabulary Terms
7. **OWL for Data Integration**

Example of <u>Non-integrated</u> Knowledge Graphs

```
dbr:Germany rdf:type dbo:Country ;
            dbo:capital dbr:Berlin ;
            dbo:language dbr:German_language .
dbo:language rdfs:label “language“@en, “Sprache“@de
```

Knowledg
e Graph

```
wikidata:Q183 rdf:type wikidata:Q6256,
wikidata:Q4209223;
              rdfs:label “Germany“@en ;
              wikidata-prop:P37 wikidata:Q188 .
wikidata:Q6256 rdfs:label “country“@en .
wikidata:Q4209223 rdfs:label “legal state“@en .
wikidata-prop:P37 rdfs:label “official language “@
```

Knowledg
e Graph

Example of Integrated Knowledge Graphs

```
dbr:Germany rdf:type dbo:Country ;
            dbo:capital dbr:Berlin ;
            dbo:language dbr:German_language .
dbo:language rdfs:label “language“@en, “Sprache“@de .
```

Knowledge Graph

```
dbr:Germany owl:sameAs wikidata:Q183 .
dbo:Country owl:equivalentClass wikidata:Q6256  .
wikidata-prop:P37 rdfs:subPropertyOf dbo:language .
```

Mappings for data integration

```
wikidata:Q183 rdf:type wikidata:Q6256,
wikidata:Q4209223;
              rdfs:label “Germany“@en ;
              wikidata-prop:P37 wikidata:Q188 .
wikidata:Q6256 rdfs:label “country“@en .
wikidata:Q4209223 rdfs:label “legal state“@en .
wikidata-prop:P37 rdfs:label “official language “@
```

Knowledge Graph

Learning Goals

- G 10.1 Describe the two broad groups of OWL profiles and outline the differences between the two.
- G 10.2 Categorise properties as symmetric, inverse, transitive, functional, inverse functional, irreflexive or asymmetric.
- G 10.3 Model ontologies in Turtle syntax based on a textual description, using URIs from the OWL vocabulary.
- G 10.4 Identify triples that lead to unsatisfiable graphs under OWL LD entailment.
- G 10.5 Map URIs of classes, properties and individuals from multiple RDF graphs using terms from the RDFS and OWL LD vocabularies.

Outlook

- In the next lectures, we consider user agents which combine SPARQL query processing with entailment
- We also consider user agents that follow links to discover new resources