Cours 1

The OWL2 Web Ontology Language

G. Falquet

OWL 2

- A language in the Description Logics family
- with
  - A rich set of class constructors and property constructors
  - Several relations to define axioms
    - on classes (subclass, equivalent, disjoint, ...)
    - on properties (transitive, functional, ...)
    - on individuals (same, different)
- Can be expressed in RDF (https://www.w3.org/TR/owl2-mapping-to-rdf/)

https://www.w3.org/TR/owl2-primer/

OWL 2 is an ontology language for the Semantic Web

- with formally defined meaning
- provide classes, properties, individuals, and data values

OWL 2 ontologies

- can be expressed in RDF (https://www.w3.org/TR/owl2-mapping-to-rdf/)
  - owl RDF vocabulary (@prefix owl: http://www.w3.org/2002/07/owl#)

Knowledge representation primitives in OWL

- individuals
- classes
- datatypes
- properties
  - object properties
  - datatype properties
- axioms

Individuals, classes and instances

Named individual declaration

rdf:type

```
:Mary a owl:NamedIndividual .
```

Class declaration

```
:Woman a owl:Class
```

$\sqsubseteq$ rdfs:class

Instance axiom

```
:Mary a :Woman
```

Axioms for class hierarchies

Subclass

- `:Mother rdfs:subClassOf :Woman`
- `:Woman rdfs:subClassOf :Person`

Equivalence

- `:Person owl:equivalentClass :Human`

Disjointness

- `:Woman owl:disjointWith :House`

Object properties

Object property declaration

```
hasWife  rdf:type owl:ObjectProperty
```

Object property assertion

```
:John :hasWife :Mary
```

Subproperty

```
:hasWife rdfs:subPropertyOf :hasSpouse
```

Object properties

In OWL (DL in general) what is absent can be either true or false (unknown).

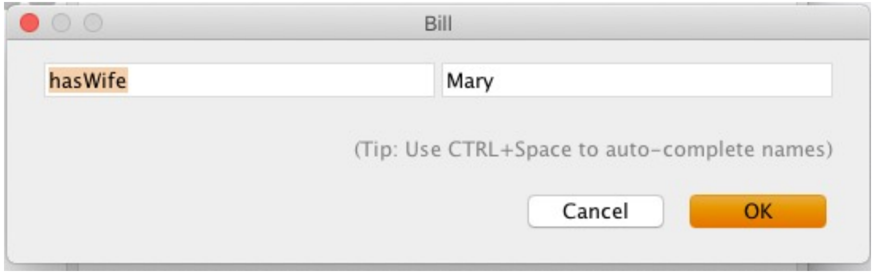
Negative property assertions provide a way to make statements where we know something that is not true.

~~bill hasWife mary~~

Negative object property assertion

```
_:x rdf:type owl:NegativePropertyAssertion .
_:x owl:sourceIndividual :Bill .
_:x owl:assertionProperty :hasWife .
_:x owl:targetIndividual :Mary .
```

Good news! Ontology editors, like Protégé, generate that for you



Datatype properties

Datatype property declaration

birthDate *rdf:type owl:DatatypeProperty*

Datatype *property assertion*

:John :birthDate "1980-10-22"^^xsd:date

Class Constructors

DL notation	in RDF/Turtle
$\top$	`owl:Thing`
$\bot$	`owl:Nothing` (always empty)
a concept name C	C a `owl:Class`
$C = C1 \sqcap C2$	C a `owl:Class ; owl:intersectionOf` (C1 C2)
$C = C1 \sqcup C2$	C a `owl:Class ; owl:unionOf` (C1 C2)
$C = \neg\, C1$	C a `owl:Class ; owl:complementOf` C1

Examples with class axioms

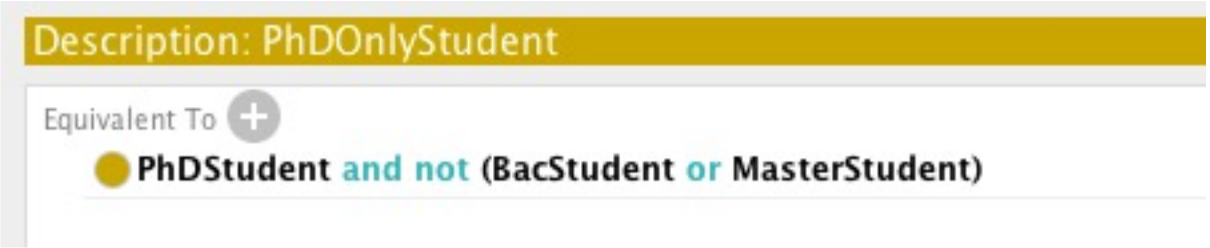
$\text{Student} \equiv \text{BacStudent} \sqcup \text{MasterStudent} \sqcup \text{PhDStudent}$

(RDF syntax)

```
:Student owl:equivalentClass [
    a owl:Class ;
    owl:unionOf (:BachelorStudent :MasterStudent :PhDStudent) ] .
```

PhDOnlyStudent ≡ PhDStudent and not (BacStudent or MasterStudent)

```
:PhDOnlyStudent rdfs:equivalentClass [ a owl:Class ;
   owl:intersectionOf (:PhDStudent
      [a owl:Class ;  owl:complementOf
         [a owl:Class ; owl:unionOf (:BacStudent :MasterStudent)]])
```



Disjoint Union

```
:LivingThing owl:equivalentClass [
   a owl:Class ;
   owl:disjointUnionOf (:Plant :Animal :Human) ].
```

is equivalent to

```
:LivingThing owl:equivalentClass [
     a owl:Class ;
     owl:unionOf (:Plant :Animal :Human) ].
:Plant owl:disjointWith :Animal, :Human .
:Human owl:disjointWith :Animal .
```

Restriction Constructors

existential restriction: P **some** D $(C = \exists\, P.D)$

`[ a ow:Restriction; owl:onProperty P; owl:someValuesFrom D ]`

universal restriction: C = P **only** D $(C = \forall\, P.D)$

`[ a ow:Restriction;  owl:onProperty P; owl:allValuesFrom D ]`

Example

$$\mathrm{Human} \sqsubseteq \forall\, \mathrm{hasOffspring.\,Human}$$

```
:Human rdfs:subClassOf
   [a owl:Restriction; owl:onProperty :hasOffspring;
    owl:allValuesFrom :Human] .
```

$$\exists\, \mathrm{hasOffspring.\,Cat} \sqsubseteq \mathrm{Cat}$$

```
 [a owl:Restriction; owl:onProperty :hasOffspring;
  owl:someValuesFrom :Cat]
                                  rdfs:subClassOf :Cat .
```

One Of Class Constructor

```
GriffinFamilyMember rdf:type owl:Class ;
    owl:equivalentClass
          [ rdf:type owl:Class ;
            owl:oneOf (:Peter :Lois :Stevie :Meg :Chris :Brian ) ]
```

- The Griffin family consists exactly of Peter, Lois, Steie, Meg, Chris, and Brian.

Value Restriction

```
:SwissProduct rdf:type owl:Class ;
         rdfs:equivalentClass [ rdf:type owl:Restriction ;
                                owl:onProperty :madeIn ;
                                owl:hasValue :Switzerland
                               ] .
```

A swiss product is something made in Switzerland

Self-Restriction

```
:SelfDriving rdf:type owl:Class ;
   owl:equivalentClass [ rdf:type owl:Restriction ;
                         owl:onProperty :drivenBy ;
                         owl:hasSelf "true"^^xsd:boolean
                        ] .
```

Something is self-driving if it's driven by itself.

Property axioms

- P `rdfs:subPropertyOf` Q
- P `owl:propertyDisjointWith` Q
- P `owl:equivalentProperty` Q
- P `owl:inverseOf` Q
- P `owl:propertyChainAxiom` (Q1 Q2 … Qn)

Example

$$parent \circ parent \sqsubseteq grandParent$$

```
:grandParent    rdf:type owl:ObjectProperty ;
                owl:propertyChainAxiom ( :parent :parent ) .

:hasAunt        rdf:type owl:ObjectProperty ;
                owl:propertyChainAxiom ( :hasParent :hasSister ) .
```

Property Characteristics

owl:FunctionalProperty	$x \mathrm{P} y1 \wedge x \mathrm{P} y2 \rightarrow y1 = y2$
owl:InverseFunctionalProperty	$x1 \mathrm{P} y \wedge x2 \mathrm{P} y \rightarrow x1 = x2$
owl:ReflexiveProperty	$x \mathrm{P} x$
owl:IrreflexiveProperty	$\neg(x \mathrm{P} x)$
owl:SymmetricProperty	$x \mathrm{P} y \rightarrow y \mathrm{P} x$
owl:AsymmetricProperty	$x \mathrm{P} y \rightarrow \neg(y \mathrm{P} x)$
owl:TransitiveProperty	$x \mathrm{P} y \wedge y \mathrm{P} z \rightarrow x \mathrm{P} z$

Individual Axioms

- `owl:`**`sameAs`**
- `owl:`**`differentFrom`**

There is no unique name assumption

⇒ different IRIs may be interpreted as the same thing

Examples

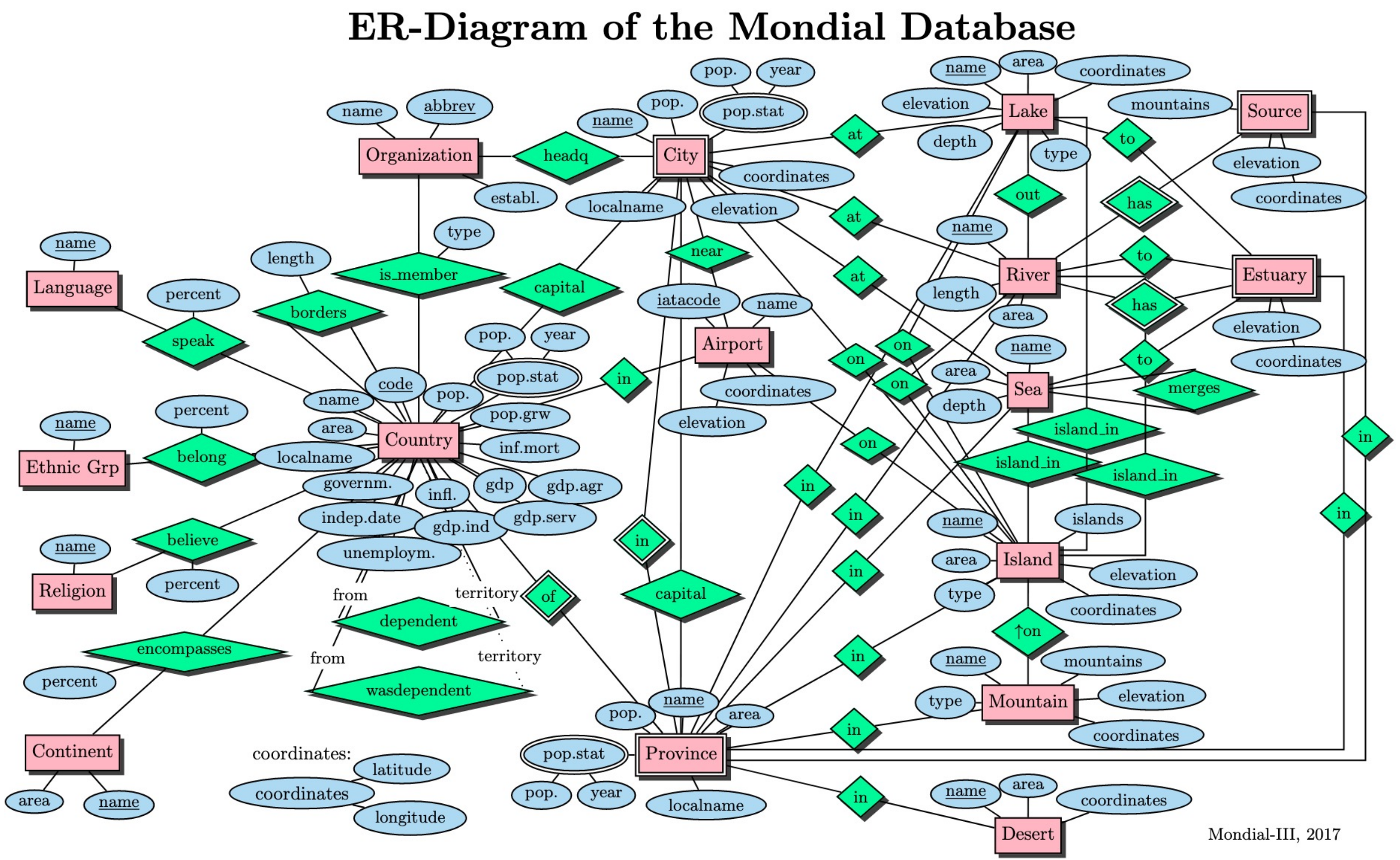
```
:livesIn a owl:FunctionalProperty .
:Bob :livesIn :Geneva .
:Bob :livesIn :GVA .
```

is consistent.

It entails

```
    :Geneva owl:sameAs :GVA
```

Example: The OWL2 TBox for Mondial



Sample Data

```
@base <http://www.semwebtech.org/mondial/10/>.
...
<countries/AL/> rdf:type :Country ;
:name "Albania" ;
:capital <countries/AL/cities/Tirana/> ;
:population 2821977 ;
:hadPopulation
   [ a :PopulationCount; :year "1950"^^xsd:gYear; :value 1214489] ,
   [ a :PopulationCount; :year "1960"^^xsd:gYear; :value 1618829] ,
...
:populationGrowth 0.3 ;
:independenceDate '1912-11-28'^^xsd:date ;
:wasDependentOf <politicalbodies/Ottoman+Empire/> .

<politicalbodies/Ottoman+Empire/> rdf:type :PoliticalBody .
```

Class hierarchy

- AdministrativeArea
- AdministrativeSubdivision
  - City
- AnthropoGeographicalRelationship
  - BelievedBy
  - EthnicProportion
  - SpokenBy
- AnthropoGeographicalThing
  - EthnicGroup
  - Language
  - Religion
- Area
  - LargeArea
    - Continent
    - Country
    - EncompassedArea
    - Province
    - Sea
- GeographicalNonPoliticalThing
- GeographicalThing
  - City
- Geometrical
- Line
  - Border
  - River
- Location
- MondialThing
- NonGeographicalPoliticalThing
- Organization
- Place
  - Estuary
  - Mountain
    - Volcano
  - Source
- PoliticalOrGeographicalThing
- PoliticalThing
  - PoliticalBody
- ReifiedThing
  - Encompassed
  - Membership
- SmallArea
  - Archipelago
  - City
  - Desert
  - Island
  - Lake
  - Mountains
- TimeSeriesElement
  - Measurement
  - YearlyMeasurement
    - PopulationCount
- Water

Some axiomes

Description: River

Equivalent To

SubClass Of

- hasEstuary **exactly** 1 owl:Thing
- hasSource **exactly** 1 owl:Thing
- Line
- locatedIn **only** (**not** (City))

Description: City

Equivalent To

SubClass Of

- AdministrativeSubdivision
- area **exactly** 0 rdfs:Literal
- cityIn **exactly** 1 Country
- cityIn **max** 1 Province
- GeographicalThing
- isCapitalOf **max** 1 Country
- locatedIn **exactly** 1 Country
- locatedIn **max** 1 Province
- SmallArea

Description: Continent

Equivalent To

SubClass Of

- LargeArea
- locatedIn only owl:Nothing
- GeographicalNonPoliticalThing

General class axioms

- GeographicalThing and (not (Continent)) SubClassOf locatedIn min 1 Country

Description: PopulationCount

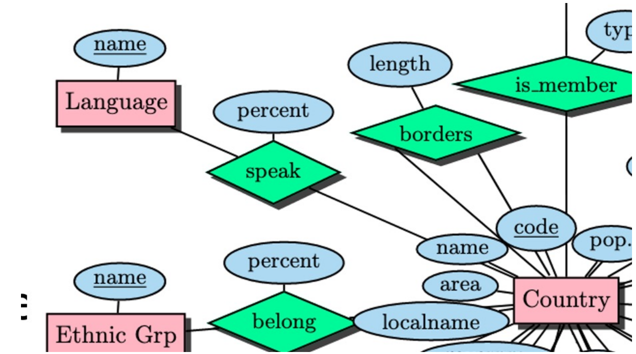
Equivalent To

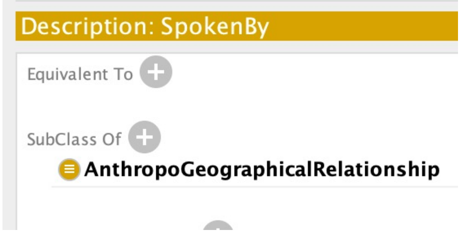
SubClass Of

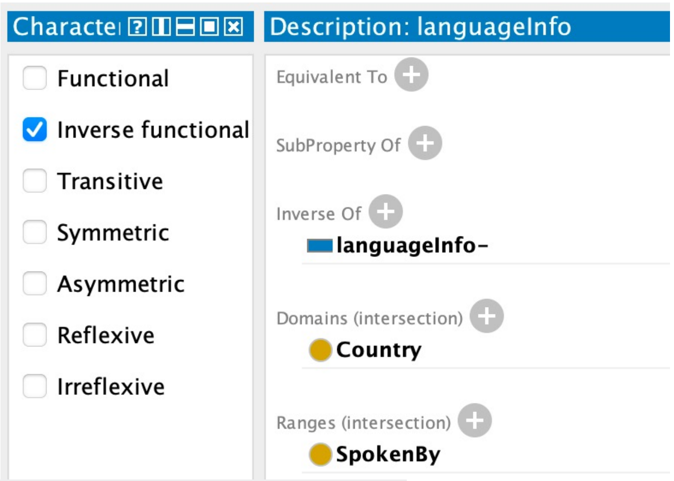
- ofObject only (City or Country or Province)
- YearlyMeasurement

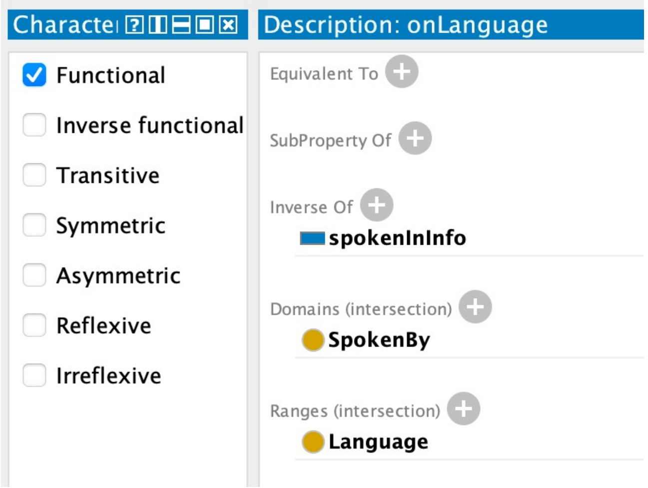
General class axioms

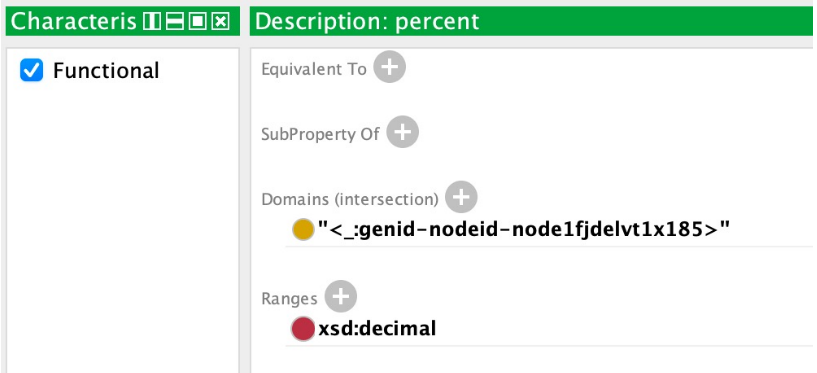
- PopulationCount and (ofObject some owl:Thing) SubClassOf inverse (hadPopulation) some owl:Thing












```
:percent a owl:DatatypeProperty; a owl:FunctionalProperty;
    rdfs:domain [owl:unionOf
        (:SpokenBy :BelievedBy :EthnicProportion :Encompassed)];
    rdfs:range xsd:decimal.
```

Enhancing the river data

add a sequence no.

```
<rivers/Vuoksi/> :flowsThrough <lakes/Saimaa/> .
<rivers/Rhein/> :flowsThrough <lakes/Bodensee/> .
<rivers/Aare/> :flowsThrough <lakes/Brienzersee/> .
<rivers/Aare/> :flowsThrough <lakes/Thunersee/> .
<rivers/Reuss/> :flowsThrough <lakes/Vierwaldstättersee/> .
<rivers/Rhone/> :flowsThrough <lakes/Lac+Leman/> .
```

- **class** FlowThrough
  - subclass of
    - river **only** River
    - through **only** Lake
    - seq **only** xsd:integer

- **functional properites**: river, through, seq
  - domain: FlowThrough

Metamodeling (by punning)

DL limitation: entities are either classes or individuals, not both.

In some situations this causes modeling problems

In OWL2, an IRI I can be used to refer to more than one type of entity.

Goal: To state facts about classes and properties themselves.

> entities that share the same IRI should be understood as different "views" of the same underlying notion identified by the IRI.

Example

(1) :Brian a :Dog .

(2) :Dog a :Species .

in (1) *:Dog* is a class

in (2) *:Dog* is an individual, member of *x:Species*

The individual *x:Dog* and the class *x:Dog* should be understood as two "views" of one and the same IRI — *x:Dog*.

The OWL 2 Direct Semantics treats the different uses of the same name as **completely separate**, as is required in DL reasoners.

Metamodelling and Annotations

Two means to associate additional information with classes and properties.

- Metamodeling should be used when the information attached to entities should be considered a part of the domain.
- Annotations should be used when the information attached to entities should not be considered a part of the domain and when it should not contribute to the logical consequences of an ontology.

Profiles

An OWL 2 *profile* (commonly called a *fragment* or a *sublanguage* in computational logic) is a trimmed down version of OWL 2 that trades some expressive power for the efficiency of reasoning.

EL : polynomial time reasoning (large ontologies)

QL : LOGSPACE reasoning ("database" applications)

RL : polynomial time reasoning with a rule-based (database) system

OWL 2 EL Profile

- for applications employing ontologies that define very large numbers of classes and/or properties,
- captures the expressive power used by many such ontologies,
- consistency, class expression subsumption, and instance checking can be decided in polynomial time.

Constructs not supported in EL

R only C

R min n C, R max n C

C or D

not C

$\{a_1, a_2, \ldots, a_n\}$ with $n>1$

On properties:

disjoint, irreflexive, inverse, fonctional and inverse functional, symetric, asymetric

QL Profile

- sound and complete query answering is in $\mathbf{L}$ (LOGSPACE) $\subseteq$ **P**
- many of the main features necessary to express conceptual models such as UML class diagrams and ER diagrams
- contains the intersection of RDFS and OWL 2 DL
- assertions stored in a standard relational database system can be queried through an ontology via a simple rewriting mechanism
  - rewriting the query into an SQL query

Restrictions on axiom structures

a class R some Thing R some DataRange	$\sqsubseteq$	a class C and D not C R some C R some DataRange

Constructs not supported

R **only** C

R **value** V

{a, b, ...}

R **min/max** n C

C **or** D

...

Axioms not supported

P1 o P2 o ... subPropertyOf P

sameIndividual(a, b)

not P(a, b)

Rewriting examples

Q ≡ A and (R some C)

C ≡ D and E

Find all the instances of Q:

```
select TA.id
from TA, TR, TC, TD, TE
where TA.id = TR.from and TR.to = C.id
      and C.id = D.id and C.id = E.id
```

OWL 2 RL

OWL 2 RL enables the implementation of polynomial time reasoning algorithms (in ABox size) using rule-extended database technologies operating directly on RDF triples;

~ mapping to Datalog programs

" For applications where relatively lightweight ontologies are used to organize large numbers of individuals and where it is useful or necessary to operate directly on data in the form of RDF triples. "

Restrictions on axiom structures

a class {a, b, c, ...} C and D C orD R some C R some DataRange R value a ...	$\sqsubseteq$	a class C and D not C R only C R value a R max 0/1 C R some DataRange ...

Sample rule: $\exists p.c \sqsubseteq d \land p(x,y) \land c(y) \rightarrow d(x)$

IF the ontology contains

- e_1 `rdfs:subClassOf` d
- e_1 `a owl:Restriction; owl:onProperty` p`; owl:someValuesFrom` c
- $x\ p\ y$
- y rdf:type c

THEN infer

- x rdf:type d