

## Short notes about OWL<sup>1</sup>

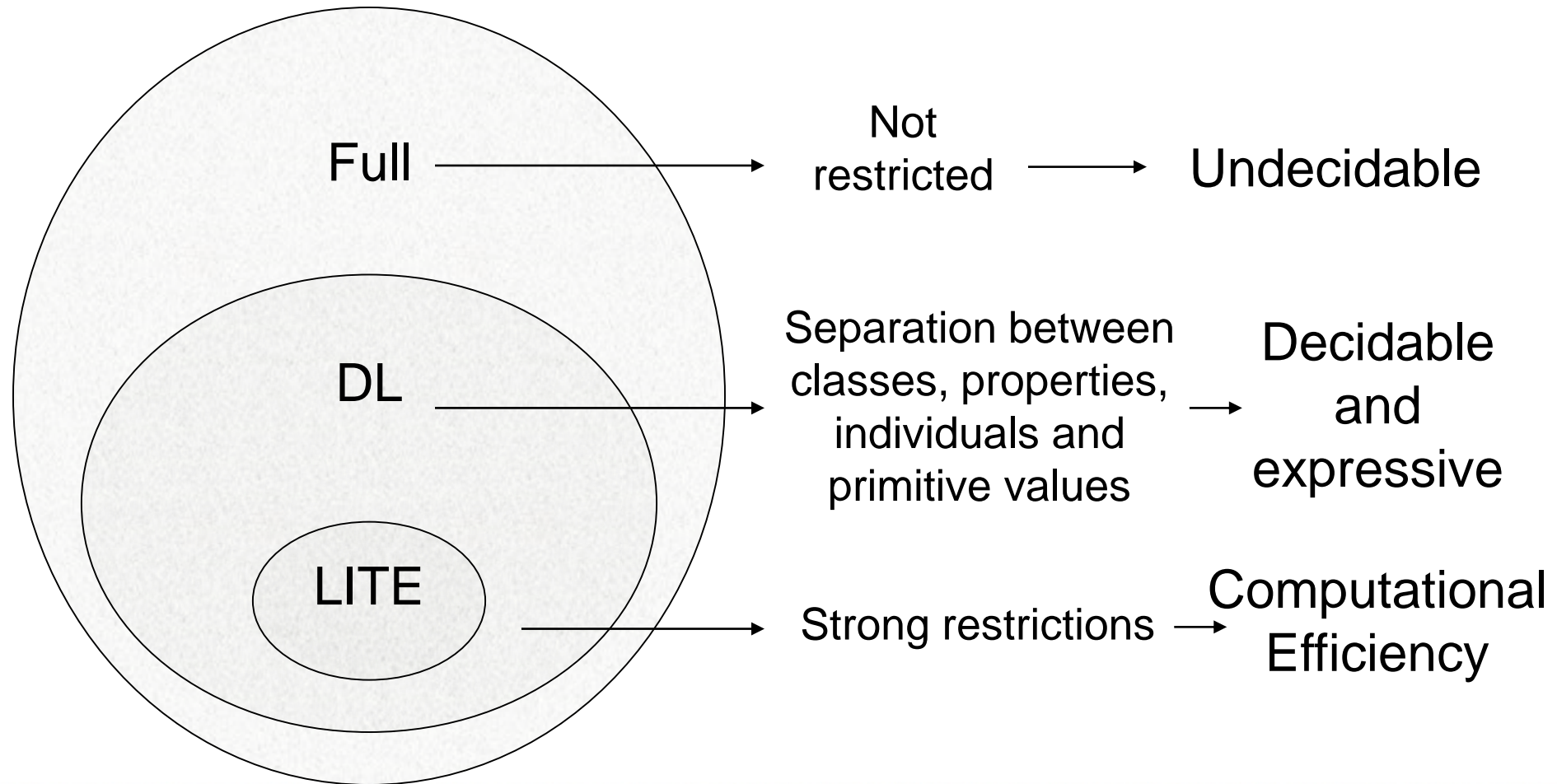
**Manuel Fiorelli**

**fiorelli@info.uniroma2.it**

[1] this presentation is limited to OWL 1 features. A [new version of OWL \(OWL 2\)](#), which adds further features (thus remaining backward-compatible with the original OWL), has reached the status of W3C recommendation in 2012

# OWL Sublanguages

Sublanguages defined upon restrictions on the use of OWL constructs



# Classes

Can be introduced by simply giving them a name

```
<owl:Class rdf:ID="Person" />
```

*rdf:ID="Person" is equivalent to *rdf:about="#Person"* with the additional check that the same name is not used in another attribute *rdf:ID* in the scope of an *xml:base* value (or document, if none is given)*

which can be used in order to describe their instances

```
<Person rdf:ID="manuel" />
```

or

```
<owl:Thing rdf:ID="manuel">
```

```
  <rdf:type rdf:resource="#Person" />
```

```
</owl:Thing>
```

# Properties

- *datatype property*, relate individuals to literals
- *object property*, relate individuals among them
- *annotation property*, out of ontology semantics; useful in order to provide comments/documentation to the **ontology** (predefined annotation properties are: *owl:versionInfo*, *rdfs:label*, *rdfs:comment*, *rdfs:seeAlso*, and *rdfs:isDefinedBy*)
- *ontology property* (e.g. *owl:imports*), must have the class *owl:Ontology* as domain and range

# Object Property

```
<owl:ObjectProperty rdf:ID="loves">  
  <rdfs:domain rdf:resource="#Person" />  
  <rdfs:range rdf:resource="#Person" />  
  <rdfs:subPropertyOf rdf:resource="#knows" />  
</owl:ObjectProperty>
```

# Datatype Property

*#Person* is a relative IRI, which is then resolved relative to the current base IRI

```
<owl:DatatypeProperty rdf:ID="name">
```

```
<rdfs:domain rdf:resource="#Person" />
```

```
<rdfs:range rdf:resource="&xsd:string" />
```

```
</owl:DatatypeProperty>
```

In RDF/XML we can't use prefixed names in attribute values. However, we can use DTD entities (e.g. &xsd;) that are defined with a value equal to the desired namespace IRI

# Annotation Property

In **OWL DL** annotation properties have the following constraints:

- An annotation property must be explicitly typed as *owl:AnnotationProperty* (unless it is a predefined annotation property)
- Annotation properties are disjoint with other types of properties
- Annotation properties cannot be used in property axioms  
(e.g. it is not possible to state a domain, a range or a super property) ◦ ◦ ◦
- The value of an annotation property is either an individual, a data literal or a URI reference

OWL 2 relaxed  
this constraint

# Describing individuals

```
<Person rdf:ID="armando">
    <knows rdf:resource="#manuel" />
    <name rdf:datatype="&xsd:string">Armando</name>
</Person>

<owl:Thing rdf:ID="manuel">
    <rdf:type rdf:resource="#Person" />
    <name rdf:datatype="&xsd:string">Manuel</name>
</owl:Thing >
```



# owl:differentFrom

```
<Person rdf:ID="armando">  
    <owl:differentFrom rdf:resource="#manuel" />  
    <owl:differentFrom rdf:resource="#andrea" />  
</Person>  
  
<owl:Thing rdf:ID="manuel">  
    <owl:differentFrom rdf:resource="#andrea" />  
</owl:Thing>  
  
<owl:Thing rdf:ID="andrea" />
```

```
<owl:AllDifferent>  
  
  <owl:distinctMembers  
rdf:parseType="Collection">  
  
    <owl:Thing rdf:about="#armando" />  
  
    <owl:Thing rdf:about="#manuel" />  
  
    <owl:Thing rdf:about="#andrea" />  
  
  </owl:distinctMembers>  
  
</owl:AllDifferent>
```

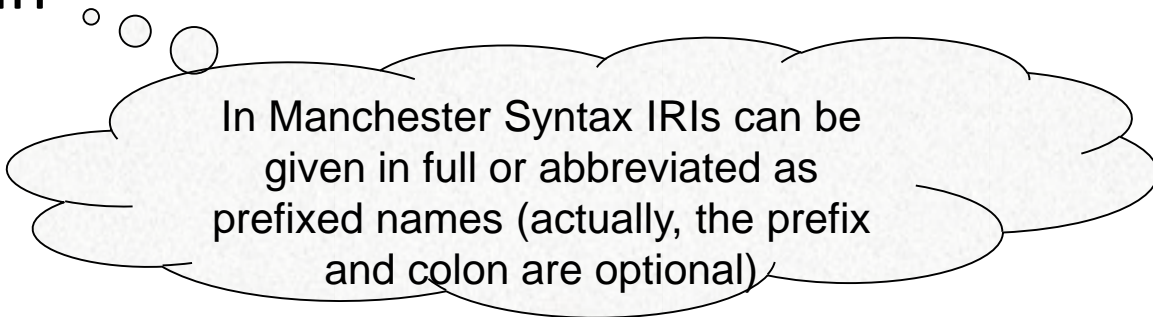
Different kind of *class descriptions*

- A *class name* (URI)
- An exhaustive *enumeration* of its instances
- A *restriction on a property*
- *Intersection* of two or more classes
- *Union* of two or more classes
- The *complement* to a class

```
<owl:Class rdf:ID="Human" />
```

In *Description Logics (DL)* and *Manchester Syntax*

Human



In Manchester Syntax IRIs can be given in full or abbreviated as prefixed names (actually, the prefix and colon are optional)

We define a class through the set of individuals belonging to its extension

```
<owl:Class>  
  <owl:oneOf rdf:parseType="Collection">  
    <owl:Thing rdf:about="#Europe"/>  
    <owl:Thing rdf:about="#Africa"/>  
    <owl:Thing rdf:about="#Asia"/>  
    <owl:Thing rdf:about="#America"/>  
    <owl:Thing rdf:about="#Australia"/>  
    <owl:Thing rdf:about="#Antarctica"/>  
  </owl:oneOf>  
</owl:Class>
```

In *DL* and *Manchester Syntax*: {Europe, Africa, Asia, America, Australia, Antarctica}

# Restriction on a property

A class is defined as the set of all individuals satisfying certain conditions on the use of a property.

- Value constraints
- Cardinality constraints

# Value Restriction: owl:allValuesFrom

We define the class of individuals that have all values on a given property belonging to a certain class (if the property is an owl:ObjectProperty) or to a datarange (if an owl:DatatypeProperty)

```
<owl:Restriction>
  <owl:onProperty rdf:resource="#hasParent" />
  <owl:allValuesFrom rdf:resource="#Human" />
</owl:Restriction>
```

In *DL*:  $\forall \text{ hasParent . Human}$

In *Manchester Syntax*: hasParent only Human

# Value Restriction owl:someValuesFrom

We define the class of individuals that have at least a value of a given property belonging to a certain class (if the property is an owl:ObjectProperty) or to a datarange (if an owl:DatatypeProperty)

```
<owl:Restriction>
  <owl:onProperty rdf:resource="#hasParent" />
  <owl:someValuesFrom rdf:resource="#Physician" />
</owl:Restriction>
```

In *DL*:  $\exists$  hasParent . Physician

In *Manchester Syntax*: hasParent some Physician



# Value Restriction owl:hasValue

We define the class of individuals that have a given property with at least a value semantically identical to a given one

```
<owl:Restriction>
  <owl:onProperty rdf:resource="#hasParent" />
  <owl:hasValue rdf:resource="#clinton" />
</owl:Restriction>
```

In *DL*:  $\text{hasParent} \sqsupseteq \text{clinton}$

In *Manchester Syntax*:  $\text{hasParent value clinton}$

# Cardinality Restriction owl:maxCardinality

The class of all individuals that have maximum N semantically different values on a given property

```
<owl:Restriction>
  <owl:onProperty rdf:resource="#hasParent" />
  <owl:maxCardinality
rdf:datatype="&xsd;nonNegativeInteger">2</owl:maxCardinality>
</owl:Restriction>
```

In *DL*:  $\leq 2$  hasParent

In Manchester Syntax: hasParent max 2

# Cardinality Restriction owl:minCardinality

The class of all individuals that have at least N semantically different values on a given property

```
<owl:Restriction>
  <owl:onProperty rdf:resource="#hasParent" />
  <owl:minCardinality
rdf:datatype="&xsd;nonNegativeInteger">2</owl:minCardinality>
</owl:Restriction>
```

In *DL*:  $\geq 2$  hasParent

In *Manchester Syntax*: hasParent min 2

# Cardinality Restriction owl:cardinality

The class of all individuals having exactly N semantically different values on a given property

```
<owl:Restriction>  
  <owl:onProperty rdf:resource="#hasParent" />  
  <owl:cardinality  
rdf:datatype="&xsd;nonNegativeInteger">2</owl:cardinality>  
</owl:Restriction>
```

In *DL*: = 2 hasParent

In *Manchester Syntax*: hasParent exactly 2

# Intersection owl:intersectionOf

The class of all individuals belonging to all of the classes listed in the intersection

```
<owl:Class>
```

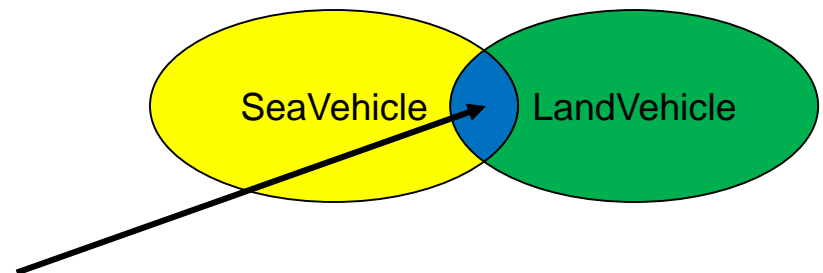
```
  <owl:intersectionOf rdf:parseType="Collection">
```

```
    <owl:Class rdf:about="#LandVehicle" />
```

```
    <owl:Class rdf:about="#SeaVehicle" />
```

```
  </owl:intersectionOf>
```

```
</owl:Class>
```



In *DL*: LandVehicle  $\sqcap$  SeaVehicle

In *Manchester Syntax*: LandVehicle and SeaVehicle

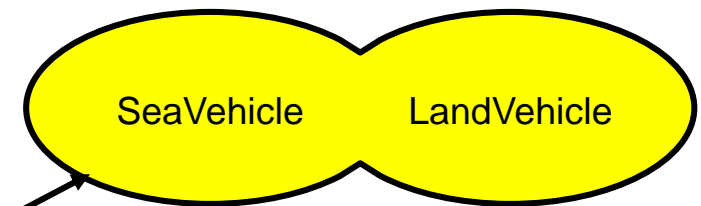
In Manchester Syntax, instead of using “and” it is possible to use “that” to write descriptions like the following (consisting in a collection of property restrictions):

Animal that eats min 1 and eats only Vegetables

# Union owl:unionOf

The class of all individuals belonging to at least one of the classes listed in the union

```
<owl:Class>
  <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#LandVehicle" />
    <owl:Class rdf:about="#SeaVehicle" />
  </owl:unionOf>
</owl:Class>
```



In *DL*: LandVehicle  $\sqcup$  SeaVehicle

In *Manchester Syntax*: LandVehicle or SeaVehicle

# Complement owl:complementOf

The class of all individuals not belonging to a given class

```
<owl:Class>
```

```
  <owl:complementOf>
```

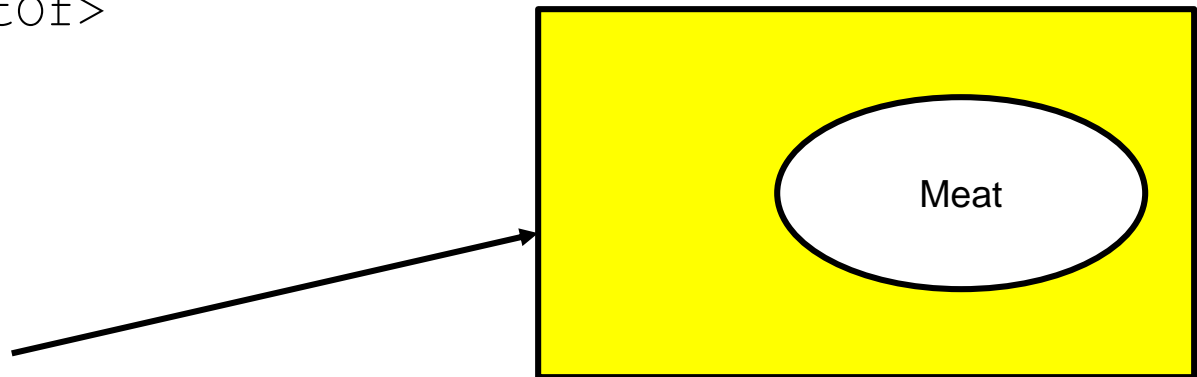
```
    <owl:Class rdf:about="#Meat"/>
```

```
  </owl:complementOf>
```

```
</owl:Class>
```

In *DL*:  $\neg$ Meat

In *Manchester Syntax*: not Meat





# Operator Precedence

In decreasing order of precedence

- SOME, ONLY, VALUE, MIN, MAX, EXACTLY, THAT
- NOT
- AND
- OR

Italian AND Actor OR Musician

*is parsed as*

(Italian AND Actor) OR Musician

eats ALL Vegetable OR Meat

*is parsed as*

(eats ALL Vegetable) OR Meat

OWL supports the following axioms concerning classes

- `rdfs:subClassOf`
- `owl:equivalentClass`
- `owl:disjointWith`

# rdfs:subClassOf

```
<owl:Class rdf:ID="aClass">
```

```
  <rdfs:subClassOf>
```

```
    class expression
```

```
  </rdfs:subClassOf>
```

```
</owl:Class>
```

```
<owl:Class rdf:ID="aClass">
```

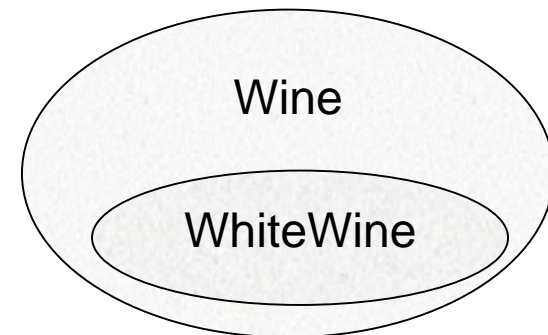
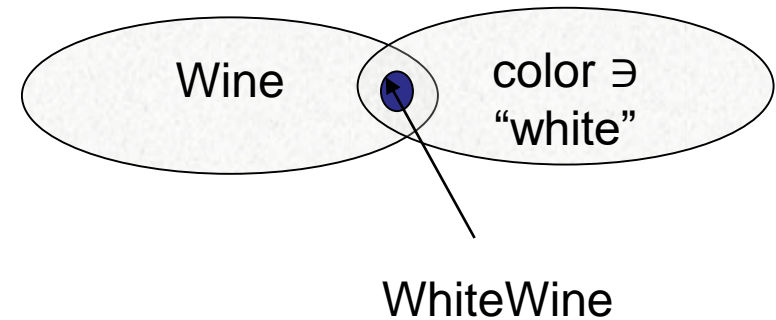
```
  <rdfs:subClassOf rdf:resource="class" />
```

```
</owl:Class>
```

# rdfs:subClassOf

```
<owl:Class rdf:ID="WhiteWine">
  <rdfs:subClassOf>
    <owl:Class>
      <owl:intersectionOf parseType="Collection">
        <owl:Class rdf:about="#Wine" />
        <owl:Restriction>
          <owl:onProperty rdf:resource="color" />
          <owl:hasValue rdf:datatype="xsd:string">white</owl:hasValue>
        </owl:Restriction>
      </owl:intersectionOf>
    </owl:Class>
  </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:id="WhiteWine">
  <rdfs:subClassOf rdf:resource="Wine" />
</owl:Class>
```



# owl:equivalentClass

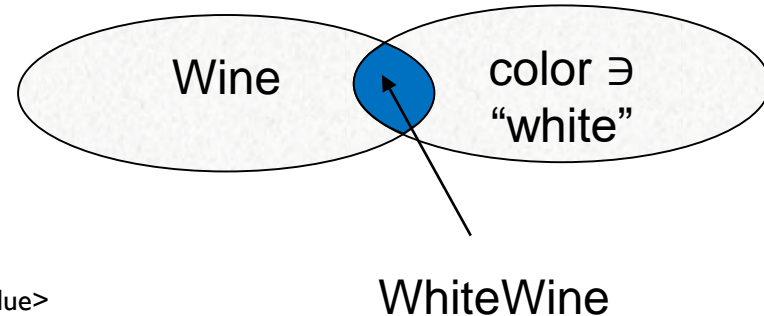
```
<owl:Class rdf:ID="aClass">
  <owl:equivalentClass>
    class expression
  </owl:equivalentClass >
</owl:Class>
```

```
<owl:Class rdf:ID="aClass">
  <owl:equivalentClass rdf:resource="class" />
</owl:Class>
```

# owl:equivalentClass

```
<owl:Class rdf:id="WhiteWine">
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf parseType="Collection">
        <owl:Class rdf:about="#Wine" />
        <owl:Restriction>
          <owl:onProperty rdf:resource="color" />
          <owl:hasValue rdf:datatype="&xsd:string">white</owl:hasValue>
        </owl:Restriction>
      </owl:intersectionOf>
    </owl:Class>
  </owl:equivalentClass>
</owl:Class>

<owl:Class rdf:id="WhiteWine">
  <owl:equivalentClass rdf:resource="http://other.com/WhiteWine" />
</owl:Class>
```



# owl:disjointWith

```
<owl:Class rdf:id="aClass">
```

```
  <owl:DisjointWith>
```

```
    class expression
```

```
  </owl:disjointWith>
```

```
</owl:Class>
```

```
<owl:Class rdf:id="aClass">
```

```
  <owl:disjointWith rdf:resource="class" />
```

```
</owl:Class>
```

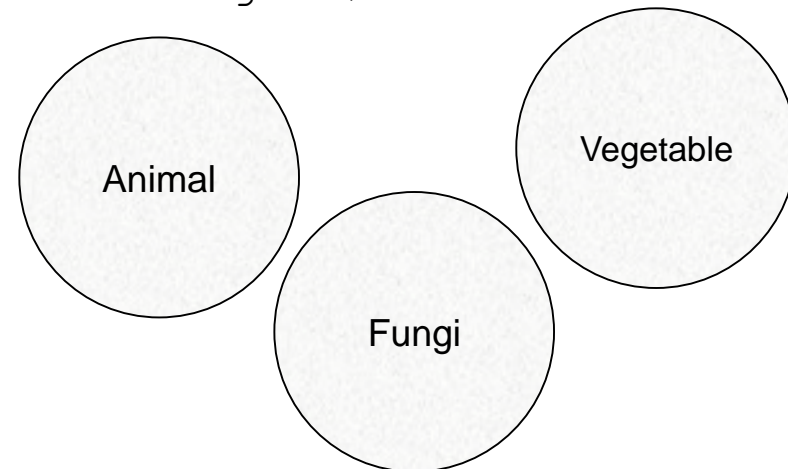
# owl:disjointWith

```
<owl:Class rdf:ID="Animal">
    <owl:disjointWith rdf:resource="#Vegetable" />
    <owl:disjointWith rdf:resource="#Fungi" />
</owl:Class>

<owl:Class rdf:ID="Vegetable">
    <owl:disjointWith rdf:resource="#Fungi" />
</owl:Class>

<owl:Class rdf:ID="Fungi" />
```

The above classes cannot  
have any instance in common





# Property Facets

With OWL it is possible to express different characteristics (facets) of properties:

- `owl:TransitiveProperty`
- `owl:SymmetricProperty`
- `owl:FunctionalProperty`
- `owl:inverseOf`
- `owl:InverseFunctionalProperty`

# owl:TransitiveProperty

if a property  $P$  is declared to be *transitive* then, for each:  $x, y, z$ , it holds that:

$$\frac{x P y \quad y P z}{x P z}$$

# owl:TransitiveProperty

```
<owl:TransitiveProperty rdf:ID="partOf" />
```

or

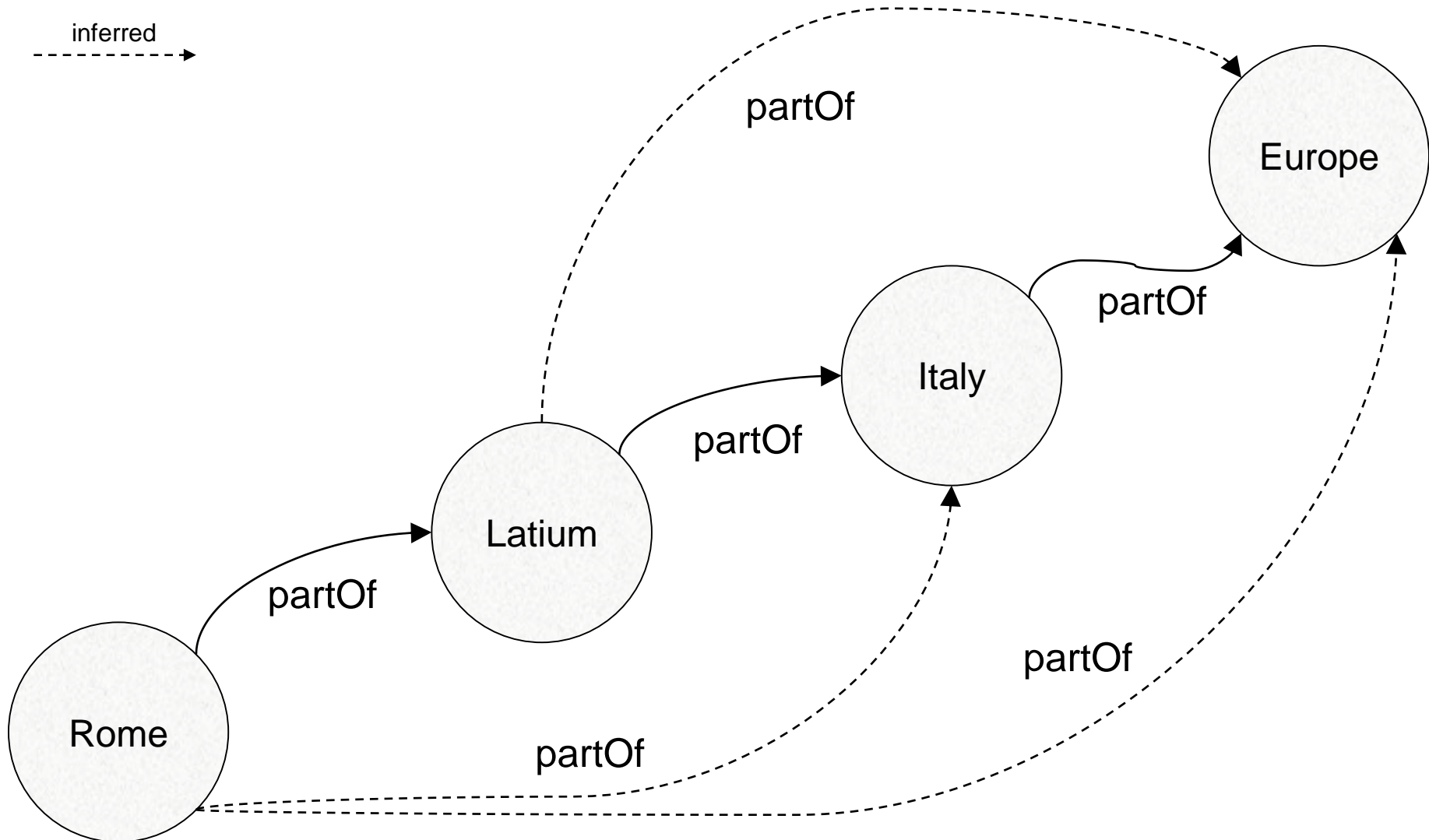
```
<owl:ObjectProperty rdf:ID="partOf" />
```

```
  <rdf:type
```

```
    rdf:resource="&owl;TransitiveProperty" />
```

```
</owl:ObjectProperty>
```

# owl:TransitiveProperty



# owl:SymmetricProperty

if a property  $P$  is declared to be *symmetric* then, for each:  $x, y$  it holds that:

$$\frac{x P y}{y P x}$$

# owl:SymmetricProperty

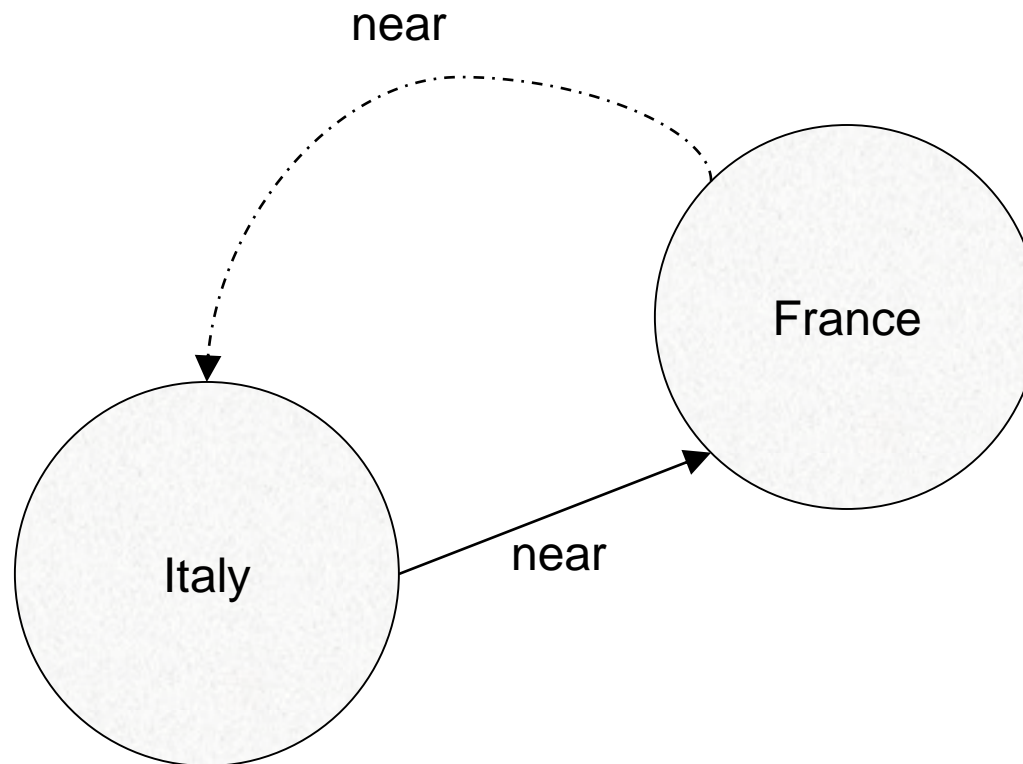
```
<owl:SymmetricProperty rdf:ID="near" />
```

or

```
<owl:ObjectProperty rdf:ID="near" />
  <rdf:type rdf:resource="&owl;
SymmetricProperty" />
</owl:ObjectProperty>
```

# owl:SymmetricProperty

inferred  
----->



if a property  $P$  is declared to be *functional* then, for each:  $x, y, z$  it holds that:

$$\frac{x P y \quad x P z}{y = z}$$

Any resource can have (at most) one value (unique) for a functional property



# owl:FunctionalProperty

```
<owl:FunctionalProperty rdf:ID="produttore" />
```

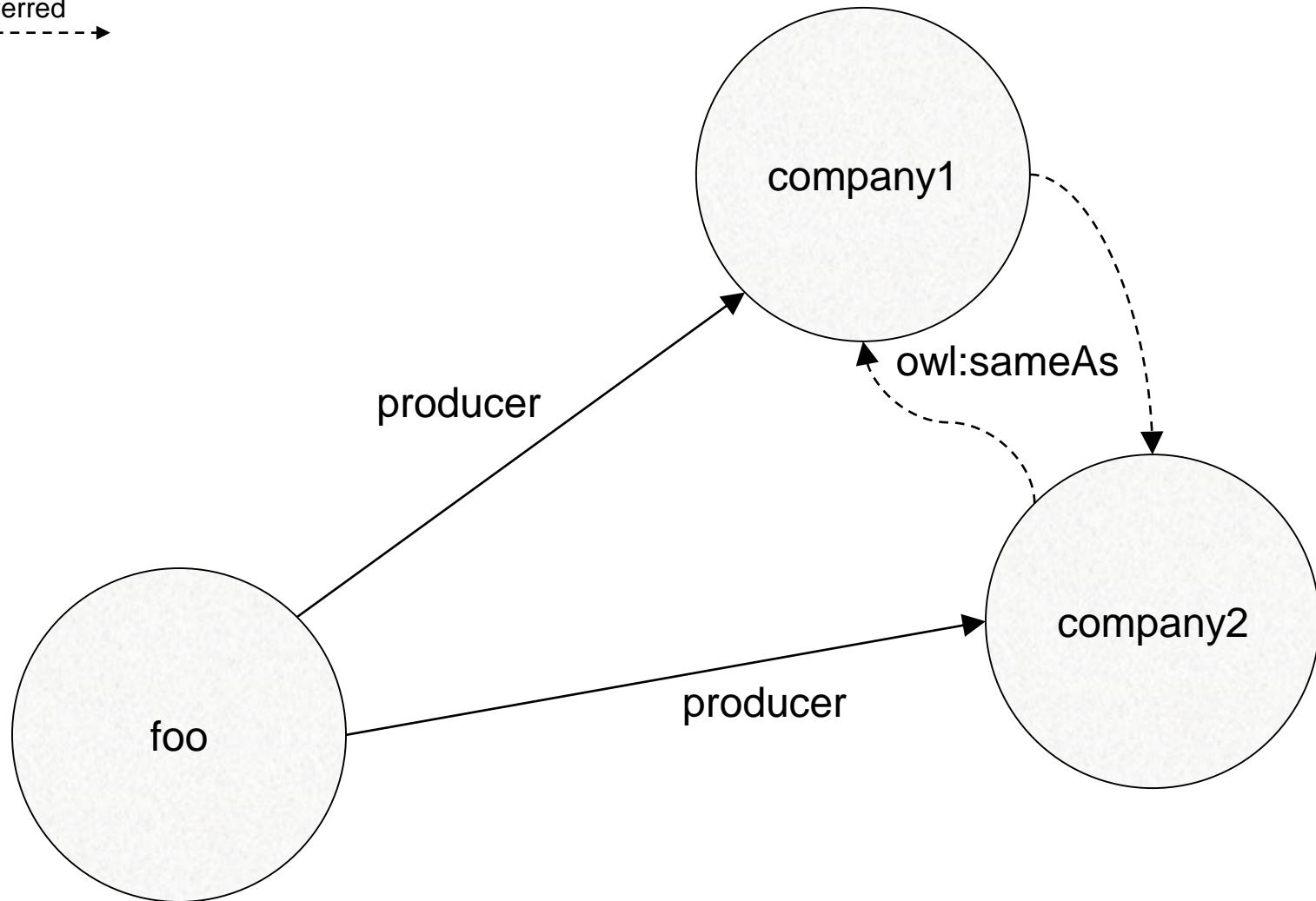
or

```
<owl:ObjectProperty rdf:ID="produttore" />
  <rdf:type rdf:resource="&owl;
FunctionalProperty" />
</owl:ObjectProperty>
```

Even datatype properties can be functional

# owl:FunctionalProperty

inferred  
----->



X hasChild Y **if and only if** Y hasParent X

```
<owl:ObjectProperty rdf:ID="hasChild">
```

```
  <owl:inverseOf rdf:resource="#hasParent"/>
```

```
</owl:ObjectProperty>
```

# owl:InverseFunctionalProperty

if a property  $P$  is declared to be *inverseFunctional* then,  
for each:  $x, y, z$  it holds that:

$$\frac{x P z \quad y P z}{x = y}$$

The object of an inverse functional property uniquely determines the subject, i.e. it's a unique identifier

# owl:InverseFunctionalProperty

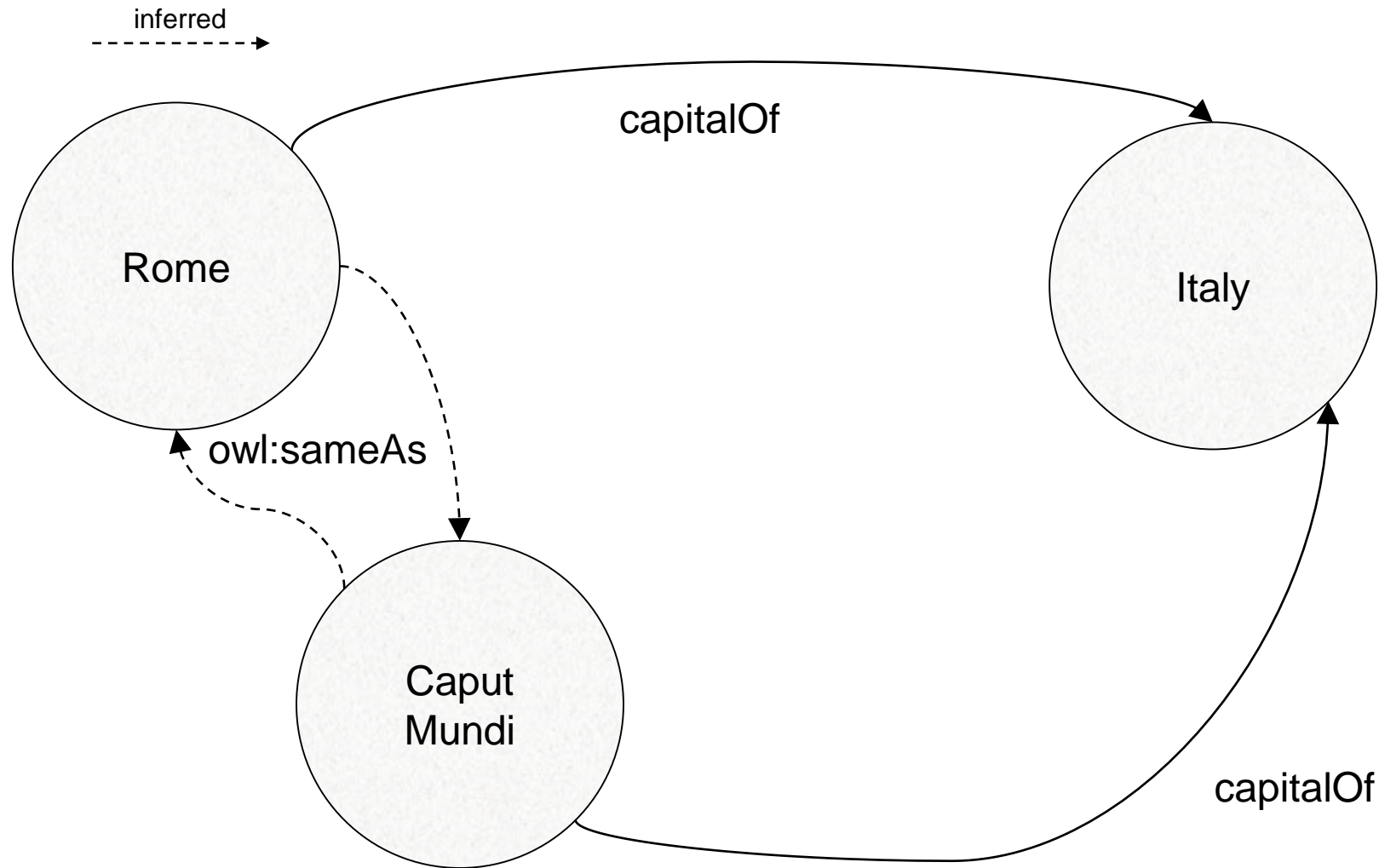
```
<owl:InverseFunctionalProperty rdf:ID="capitalOf" />
```

or

```
<owl:ObjectProperty rdf:ID="capitalOf" />
  <rdf:type rdf:resource="&owl;
InverseFunctionalProperty" />
</owl:ObjectProperty>
```

In OWL DL: only object properties can be inverseFunctional

# owl:InverseFunctionalProperty



An instance of the class *owl:Ontology* can be used to express metadata about an ontology (e.g. creator, version, etc...)

It can be also used to (transitively) import other ontologies (in order to include their axioms)

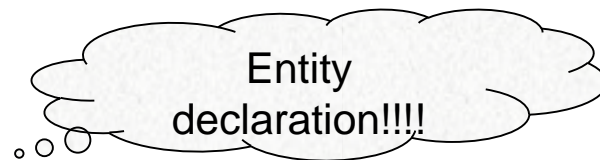
```
<owl:Ontology rdf:about="">  
  <rdfs:label xml:lang="en">An example ontology</rdfs:label>  
  <owl:imports rdf:resource="http://purl.org/dc/elements/1.1/" />  
  <dc:creator>Manuel Fiorelli</dc:creator>  
  
</owl:Ontology>
```

# An example RDF/XML file (1/4)

```
<?xml version="1.0"?>
```

```
<!DOCTYPE rdf:RDF [
```

```
  <!ENTITY foaf "http://xmlns.com/foaf/0.1/">
```





```
<rdf:RDF xmlns="http://example.org#"
  xml:base="http://example.org"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:foaf="http://xmlns.com/foaf/0.1/"
  xmlns:base="http://example.org"> <!-- continues on the next slide -->
```



# An example RDF/XML file (2/4)

```
<owl:Ontology rdf:about="http://example.org">
```

```
  <owl:imports rdf:resource="http://xmlns.com/foaf/0.1/" />
```

```
</owl:Ontology>
```

```
<owl:Class rdf:ID="EducationalInstitution">
```

```
  <rdfs:subClassOf rdf:resource="&foaf;Organization" />
```

```
  <rdfs:label xml:lang="en">educational institution</rdfs:label>
```

```
  <rdfs:label xml:lang="it">istituto d'istruzione</rdfs:label>
```

```
</owl:Class>
```

<!-- continues on the next slide -->

# An example RDF/XML file (3/4)

```
<owl:Class rdf:ID="Student">
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <rdf:Description rdf:about="&foaf;Person" />
        <owl:Restriction>
          <owl:onProperty rdf:resource="#enrolledIn" />
          <owl:someValuesFrom rdf:resource="#EducationalInstitution" />
        </owl:Restriction>
      </owl:intersectionOf>
    </owl:Class>
  </owl:equivalentClass>
  <rdfs:label xml:lang="en">student</rdfs:label>
  <rdfs:label xml:lang="it">studente</rdfs:label>
</owl:Class>

<!-- continues on the next slide -->
```

# An example RDF/XML file (4/4)

```
<owl:ObjectProperty rdf:ID="enrolledIn">  
  <rdfs:label xml:lang="en">enrolled in</rdfs:label>  
  <rdfs:label xml:lang="it">iscritto a</rdfs:label>  
</owl:ObjectProperty>  
</rdf:RDF>
```

# An example Turle file (1/3)

@prefix : <http://example.org#> .

@prefix owl: <http://www.w3.org/2002/07/owl#> .

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

@prefix xml: <http://www.w3.org/XML/1998/namespace> .

@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

@prefix base: <http://example.org> .

@prefix foaf: <http://xmlns.com/foaf/0.1/> .

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

@base <http://example.org> .

<http://example.org> rdf:type owl:Ontology ;

owl:imports foaf: .

# continues on the next slide

## An example Turle file (2/3)

```
:enrolledIn rdf:type owl:ObjectProperty ;
```

```
  rdfs:label "enrolled in"@en ,
```

```
    "iscritto a"@it .
```

```
:EducationalInstitution rdf:type owl:Class ;
```

```
  rdfs:subClassOf foaf:Organization ;
```

```
  rdfs:label "educational institution"@en ,
```

```
    "istituto d'istruzione"@it .
```

# continues on the next slide

# An example Turtle file (3/3)

```
:Student rdf:type owl:Class ;
    owl:equivalentClass [ owl:intersectionOf ( foaf:Person
        [ rdf:type owl:Restriction ;
          owl:onProperty :enrolledIn ;
          owl:someValuesFrom :EducationalInstitution
        ]
      ) ;
    rdf:type owl:Class
  ] ;
  rdfs:label "student"@en ,
    "studente"@it .
```

## OWL I

- OWL Web Ontology Language. Guide  
(<https://www.w3.org/TR/owl-guide/>)
- OWL Web Ontology Language. Reference  
(<https://www.w3.org/TR/owl-ref/>)