

# OWL: Ontology Web Language

Catherine Faron [faron@i3s.unice.fr](mailto:faron@i3s.unice.fr)

# Ontology Web Language (OWL)

- W3C recommendation
- provides additional primitives for expressing more complex ontologies
- enables richer class and property definitions
- enables to infer more facts, to perform more inferences

# namespaces and prefix to use OWL

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

- namespace of OWL primitives
- same principle than for RDFS
- usual prefix: `owl` (used in the following)

# Course outline

1. Property characteristics
2. Class relationships
3. Equivalences and alignements
4. Property restrictions
5. Ontology management
6. OWL profiles

# three different kinds of properties

## 1. `owl:ObjectProperty`

relations between resources

## 2. `owl:DatatypeProperty`

relations having literal (typed) values

## 3. `owl:AnnotationProperty`

relations ignored by reasoners, used to document the ontology or for extensions

no need to specify domain and range for annotation property

# symmetric properties

relations which, when they hold, hold in both directions

$$x R y \Rightarrow y R x$$

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

```
:hasSpouse rdf:type owl:SymmetricProperty .
```



# asymmetric properties

relations which, when they hold, cannot hold in both directions

$$x R y \Rightarrow \neg y R x$$

```
<owl:AsymmetricProperty rdf:ID="hasChild" />
```

```
:hasChild a owl:AsymmetricProperty .
```



# inverse properties

two relations holding together in the opposite direction

$$x R_1 y \Leftrightarrow y R_2 x$$

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

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





# transitive properties

relations which propagate from one resource to its neighbour

$$x R y \ \& \ y R z \Rightarrow x R z$$

---

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

```
:hasAncestor a owl:TransitiveProperty .
```

---

# disjoint properties

relations which cannot hold between the same subject and object

```
<owl:ObjectProperty rdf:about="hasSon">  
  <owl:propertyDisjointWith rdf:resource="hasDaughter"/>  
</owl:ObjectProperty>
```

```
:hasSon owl:propertyDisjointWith :hasDaughter .
```



# reflexive properties

relations which link all their subjects to themselves

```
<owl:ReflexiveProperty rdf:about="hasRelative"/>
```

```
:hasRelative a owl:ReflexiveProperty .
```



# irreflexive properties

relations which cannot link a resource to itself

```
<owl:IrreflexiveProperty rdf:about="hasParent"/>
```

```
:hasParent a owl:IrreflexiveProperty .
```

# property chains

a chain of relations can imply another relation

$$x P y \ \& \ y Q z \Rightarrow x R z$$

```
<owl:ObjectProperty rdf:ID="uncle">
  <owl:propertyChainAxiom rdf:parseType="Collection">
    <owl:ObjectProperty rdf:about="#parent" />
    <owl:ObjectProperty rdf:about="#brother" />
  </owl:propertyChainAxiom>
</owl:ObjectProperty>
```

```
:uncle a owl:ObjectProperty ;
  owl:propertyChainAxiom (:parent :brother) .
```

# functional properties

relations for which a resource can only have a single value

$$x R \underline{y} \ \& \ x R \underline{z} \Rightarrow \underline{y = z}$$

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

```
:birthDate a owl:FunctionalProperty .
```

Birthdate is a functional prop

# inverse functional properties

relations for which the same value implies the same subject

$$x R y \ \& \ z R y \Rightarrow x = z$$

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

```
:socialSecurityNumber a owl:InverseFunctionalProperty .
```



# Course outline

1. Property characteristics
- 2. Class relationships**
3. Equivalences and alignements
4. Property restrictions
5. Ontology management
6. OWL profiles



# enumerated classes



class defined by enumerating its instances

```
<owl:Class rdf:id="EyeColor">
  <owl:oneOf rdf:parseType="Collection">
    <owl:Thing rdf:ID="Blue"/>
    <owl:Thing rdf:ID="Green"/>
    <owl:Thing rdf:ID="Brown"/>
    <owl:Thing rdf:ID="Black"/>
  </owl:oneOf>
</owl:Class>
```

```
:EyeColor rdf:type owl:Class ;
  owl:oneOf
    ( :Blue :Green :Brown :Black ) .
```

# class union



class defined as the set of resources which are instances of at least one of the given classes

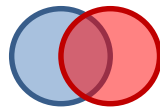
```
<owl:Class rdf:id="LegalAgent">
  <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#Person" />
    <owl:Class rdf:about="#Group" />
  </owl:unionOf>
</owl:Class>
```

```
:LegalAgent rdf:type owl:Class ;
  owl:unionOf ( :Person :Group ) .
```

---

# class intersection

class defined as the set of resources which are instances of all the given classes



```
<owl:Class rdf:id="Man">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#Person"/>
    <owl:Class rdf:about="#Male"/>
  </owl:intersectionOf>
</owl:Class>
```



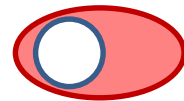
```
:Man rdf:type owl:Class ;
  owl:intersectionOf ( :Person :Male ) .
```

# class negation

class defined as the set of resources which are not instance of a given class

```
<owl:Class rdf:ID="Inedible">  
  <owl:complementOf rdf:resource="#Edible"/>  
</owl:Class>
```

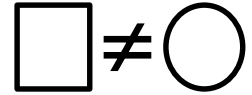
```
:Inedible rdf:type owl:Class ;  
  owl:complementOf :Edible .
```



# disjonction between two classes

A resource cannot belong to both classes

```
<owl:Class rdf:ID="Square">  
  <owl:disjointWith rdf:resource="#Circle"/>  
</owl:Class>
```

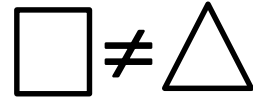
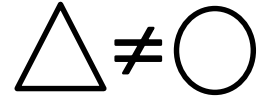
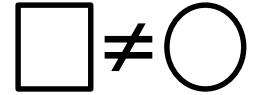


```
:Square rdf:type owl:Class ;  
  owl:disjointWith :Circle .
```

# disjonction between several classes

a resource can belong at the most to one of the disjoint classes

```
<owl:AllDisjointClasses>
  <owl:members rdf:parseType="Collection">
    <owl:Class rdf:about="#Square"/>
    <owl:Class rdf:about="#Circle"/>
    <owl:Class rdf:about="#Triangle"/>
  </owl:members>
</owl:AllDisjointClasses>
```



```
[ ] rdf:type owl:AllDisjointClasses ;
    owl:members
      ( :Square :Circle :Triangle ) .
```

# disjoint union

division of a class into a complete partition of subclasses

```
<owl:Class rdf:about="Passenger">  
  <owl:disjointUnionOf rdf:parseType="Collection">  
    <owl:Class rdf:about="#Adult"/>  
    <owl:Class rdf:about="#Child"/>  
    <owl:Class rdf:about="#Pet"/>  
  </owl:disjointUnionOf>  
</owl:Class>
```



```
:Passenger rdf:type owl:Class ;  
  owl:disjointUnionOf  
    ( :Adult :Child :Pet ) .
```



# Course outline

1. Property characteristics
2. Class relationships
- 3. Equivalences and alignements**
4. Property restrictions
5. Ontology management
6. OWL profiles



# equivalent classes



two classes gathering exactly the same resources

```
ex:Human owl:equivalentClass foaf:Person.
```

---

# equivalent properties



two property types expressing exactly the same relation

ex:name owl:equivalentProperty my:label.

A blue line that starts under the 'o' in 'owl' and extends to the right, ending under the 'y' in 'my:label.', with a slight upward curve at the end.

# same resources

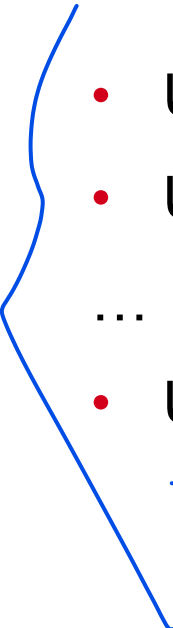



two URI identifying exactly the same thing

`ex:Bill owl:sameAs ex:William.`

A single, smooth, blue curved line that starts on the left side of the slide and curves downwards and to the right, ending in the middle of the slide.

# propagation of the identity (transitivity)

- 
- $\text{URI}_1$  owl:sameAs  $\text{URI}_2$
  - $\text{URI}_2$  owl:sameAs  $\text{URI}_3$
  - ...
  - $\text{URI}_1$  owl:sameAs  $\text{URI}_3$
- 
-

# different resources

□ ≠ □

two URI which are known as identifying two different things

`ex:Good owl:differentFrom ex:Evil.`



## identification by keys

two instances having the same key value(s) are the same instance

$$x \text{ C}_1 \text{ V}_1 ; \text{C}_2 \text{ V}_2 \ \& \ y \text{ C}_1 \text{ V}_1 ; \text{C}_2 \text{ V}_2 \Rightarrow x = y$$

```
<owl:Class rdf:about="Person">  
  <owl:hasKey rdf:parseType="Collection">  
    <owl:DataProperty rdf:about="hasSSN"/>  
  </owl:hasKey>  
</owl:Class>
```

```
:Person owl:hasKey ( :hasSSN ) .
```

# Course outline

1. Property characteristics
2. Class relationships
3. Equivalences and alignements
4. **Property restrictions**
5. Ontology management
6. OWL profiles

# restriction of property values

for the instances of the defined class, all the values of a given property are of a same given type, i.e. instances of a same given class

```
<owl:Class rdf:ID="Herbivore">
  <rdfs:subClassOf rdf:resource="#Animal"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#eats" />
      <owl:allValuesFrom rdf:resource="#Plant" />
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```



```
:Herbivore a owl:Class; rdfs:subclassOf :Animal,
[a owl:Restriction;
  owl:onProperty :eats; owl:allValuesFrom :Plant]
```

class



# restriction of some property values

for the instances of the defined class, at least one value of a given property is instance of a given class

```
<owl:Class rdf:ID="Sportive">
  <owl:equivalentClass>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hobby" />
      <owl:someValuesFrom rdf:resource="#Sport" />
    </owl:Restriction>
  </owl:equivalentClass>
</owl:Class>
```



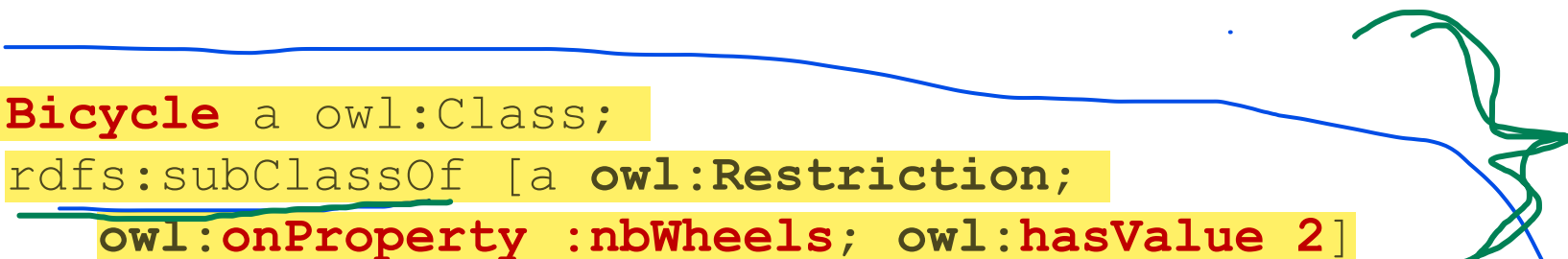
```
:Sportive a owl:Class;
  owl:equivalentClass [a owl:Restriction;
    owl:onProperty :hobby; owl:someValuesFrom :Sport]
```

# restriction to a single property value

the instances of the defined class can only have a given single value for the given property

```
<owl:Class rdf:ID="Bicycle">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#nbWheels" />
      <owl:hasValue>2</owl:hasValue>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

```
:Bicycle a owl:Class;
  rdfs:subClassOf [a owl:Restriction;
    owl:onProperty :nbWheels; owl:hasValue 2]
```

Hand-drawn blue and green lines, possibly representing a signature or decorative element, located at the bottom right of the slide.

# restriction of a property value to its subject

class defined as the set of instances having themselves as value of a given property

```
:NarcisticPerson rdfs:subClassOf
```

```
[ a owl:Restriction ;  
  owl:onProperty :love ;  
  owl:hasSelf true ]
```

# cardinality restriction

constraint on the number of times that a property can be used with different values for a given subject: minimum, maximum, exact number

```
<owl:Class rdf:ID="Person">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#name" />
      <owl:maxCardinality>1</owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

```
:Person a owl:Class;
rdfs:subClassOf [a owl:Restriction;
  owl:onProperty :name; owl:maxCardinality 1]
```



# qualified cardinality restriction

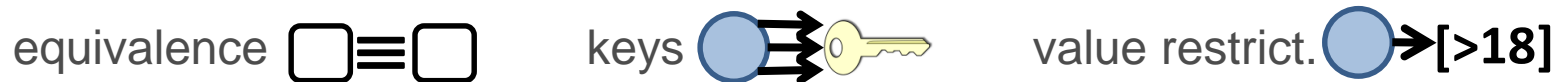
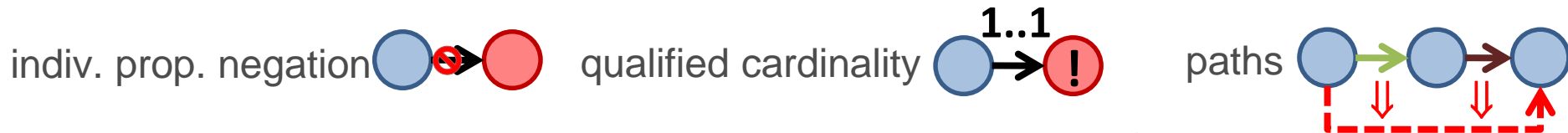
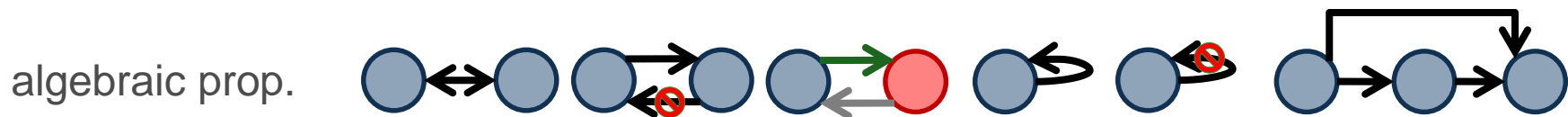
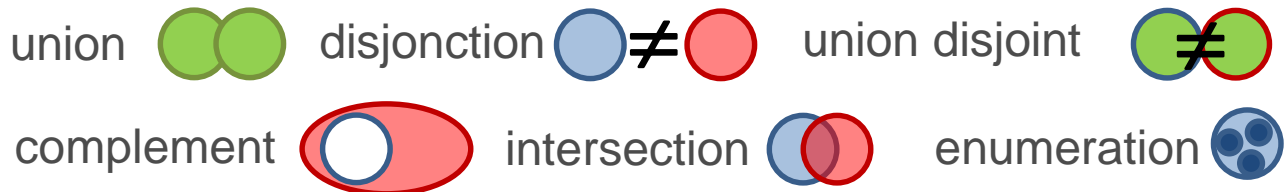
constraint on the number of times that a property can be used with different values of a given type for a given subject: minimum, maximum, exact number

```
<owl:Class rdf:ID="Human">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasParent" />
      <owl:onClass rdf:resource="#Male" />
      <owl:qualifiedCardinality>1</owl:qualifiedCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

```
:Human a owl:Class;
  rdfs:subClassOf [a owl:Restriction;
    owl:onProperty :hasParent; owl:onClass :Male;
    owl:qualifiedCardinality 1]
```

# OWL in one...

a graphical view of OWL constructs



# Course outline

1. Property characteristics
2. Class relationships
3. Equivalences and alignements
4. Property restrictions
- 5. Ontology management**
6. OWL profiles

# documenting ontologies

- an ontology is also a resource
- an ontology can be identified by a URI and then be described in RDF
- OWL provides primitives to describe this special kind of resources which are ontologies



# description of an ontology

one class (owl:Ontology) and several properties (owl:imports, owl:versionInfo, owl:priorVersion, owl:backwardCompatibleWith, owl:incompatibleWith)

```
<owl:Ontology rdf:about="http://inria.fr/2005/humans/">
  <rdfs:comment>An example OWL ontology</rdfs:comment>
  <owl:priorVersion
    rdf:resource="http://inria.fr/2004/humans/" />
  <owl:imports rdf:resource="http://cnrs.fr/animals/" />
  <rdfs:label>Bio Ontology</rdfs:label>
</owl:Ontology>
```

```
<http://inria.fr/2005/humans/> a owl:Ontology ;
  rdfs:comment "An example OWL ontology";
  owl:priorVersion <http://inria.fr/2004/humans/>;
  owl:imports <http://cnrs.fr/animals/>;
  rdfs:label "Bio Ontology".
```

# changes in classes or properties

mark a class or a property as being obsolete

```
<rdf:RDF xml:base="http://inria.fr/2005/humans/"  
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"  
  xmlns:owl="http://www.w3.org/2002/07/owl#">  
  <owl:DeprecatedClass rdf:ID="mammals"/>  
  <owl:DeprecatedProperty rdf:ID="age"/>  
</rdf:RDF>
```

```
:mammals a owl:DeprecatedClass.
```

```
:age a owl:DeprecatedProperty.
```

# Course outline

1. Property characteristics
2. Class relationships
3. Equivalences and alignements
4. Property restrictions
5. Ontology management
6. **OWL profiles**

# RDFS entailment (reminder)

- Type inference (instanciation)

based on class hierarchies

```
i rdf:type C1 . C1 rdfs:subclassOf C2 => i rdf:type C2
```

based on property signatures

```
i p o . p rdfs:domain C => i rdf:type C
```

- Property inference based on property hierarchies

```
i p1 o . P1 rdfs:subPropertyOf p2 => i p2 o
```

- Inference of hierarchical relations between classes (classification)

based on the transitivity of `rdfs:subclassOf`

```
C1 rdfs:subclassOf C2 . C2 rdfs:subclassOf C3 =>  
C1 rdfs:subclassOf C3
```

<https://www.w3.org/TR/rdf11-mt/#rdfs-entailment>

<https://www.w3.org/TR/sparql11-entailment/#RDFSEntRegime>

# OWL entailment

- Type inference (instanciation) **based on class definitions**
- Property inference **based on property characteristics**

`i p o . p a owl:SymmetricProperty => o p i`

- Inference of hierarchical relations between classes (classification) **based on class definitions**
- Consistency checking

<https://www.w3.org/TR/sparql11-entailment/#OWLRDFBSEntRegime>

# different profiles = different expressivities

- each profile corresponds to a subset of OWL primitives
- choosing a profile means choosing an expressivity to define an ontology
- the higher the expressivity, the more complex the inferences

# OWL 1 profiles

- **Lite**: mainly simple hierarchies
- **DL**: more expressive but still with complete reasoning.
- **Full**: maximal expressivity but reasoning may be incomplete

# OWL 2 profiles

---

- **EL:** large ontology, with many properties and/or classes, polynomial time
- **QL:** large dataset, RDB-like conjunctive queries, LOGSPACE
- **RL:** reasoning scaling without losing too much expressivity; inference rules, polynomial time
- **DL:** the most expressive