https://www.easyrdf.org/converter

https://www.easyrdf.org/converter

https://www.ldf.fi/service/rdf-grapher

http://rdfvalidator.mybluemix.net/

https://www.w3.org/RDF/Validator/

<1>

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

```
<rdf:Description rdf:about="#Continent">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
</rdf:Description>
```

```
<rdf:Description rdf:about="#Continent">
<rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#class"/>
</rdf:Description>
```

N-TRIPLES

```
<#Continent> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://www.w3.org/2000/01/rdf-schema#class>.
```

```
<rdfs:class rdf:about="#Continent">
</rdfs:class>
```

:Continent a owl:Class .

:Continent a rdfs:Class .

<2>

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:owl="http://www.w3.org/2002/07/owl#" xmlns:rdfs="http://www.w3.org/2000/01/rdf-
schema#" xmlns="http://example.com/ex#" xmlns:xsd="http://www.w3.org/2001/XMLSchema#" >

<owl:Class rdf:ID="LivingBeing">
 <owl:unionOf rdf:parseType="Collection">
   <owl:Class rdf:about="#Plant"/>
   <owl:Class rdf:about="#Animal"/>
 </owl:unionOf>
</owl:Class>
</rdf:RDF>
```

RDF -XML Format.

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

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

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

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

<#Plant> a    owl:Class .

<#Animal> a   owl:Class .

<#LivingBeing> a   owl:Class ;
     owl:unionOf  ( <#Plant> <#Animal> ) .
```

```
@prefix owl: <http://www.w3.org/2002/07/owl#> .
<http://njh.me/#LivingBeing>
  a owl:Class ;
  owl:unionOf (
   <http://njh.me/#Plant>
   <http://njh.me/#Animal>
 ) .
<http://njh.me/#Plant> a owl:Class .<http://njh.me/#Animal> a
owl:Class .
```

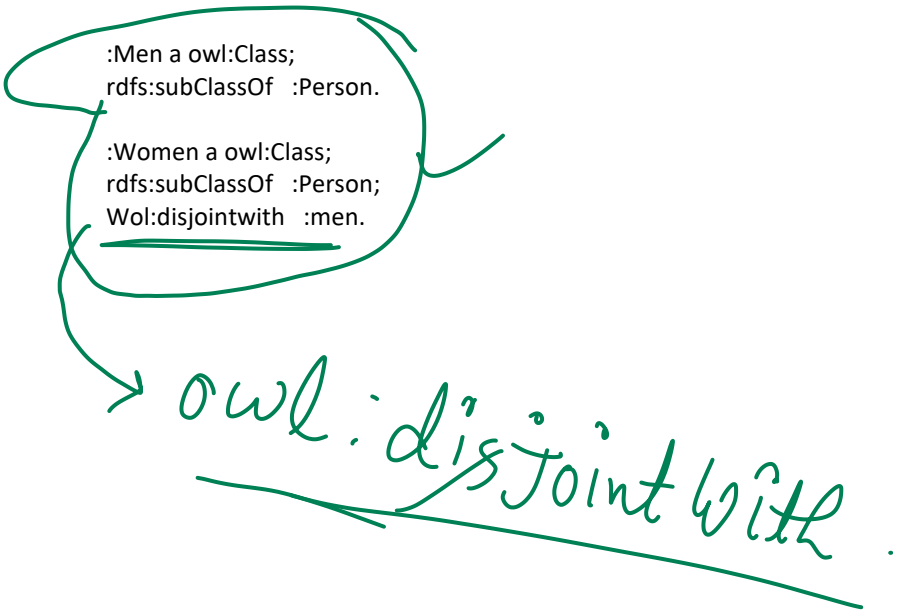
```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns="http://example.com/ex#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#" >
  <rdf:Description rdf:about="#Plant">
```

```
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
 </rdf:Description>
 <rdf:Description rdf:nodeID="A0">
  <rdf:first rdf:resource="#Animal"/>
  <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
 </rdf:Description>
 <rdf:Description rdf:about="#Animal">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
 </rdf:Description>
 <rdf:Description rdf:about="#LivingBeing">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
  <owl:unionOf rdf:nodeID="A1"/>
 </rdf:Description>
 <rdf:Description rdf:nodeID="A1">
  <rdf:first rdf:resource="#Plant"/>
  <rdf:rest rdf:nodeID="A0"/>
 </rdf:Description>
</rdf:RDF>
```

```
<?xml version="1.0" encoding="utf-8" ?><rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:owl="http://www.w3.org/2002/07/owl#">

  <owl:Class rdf:about="http://njh.me/#LivingBeing">
    <owl:unionOf>
      <rdf:Description>
        <rdf:first>
          <owl:Class rdf:about="http://njh.me/#Plant">
          </owl:Class>
        </rdf:first>

        <rdf:rest>
          <rdf:Description>
            <rdf:first>
              <owl:Class rdf:about="http://njh.me/#Animal">
              </owl:Class>
            </rdf:first>

            <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-
ns#nil"/>
          </rdf:Description>
        </rdf:rest>

      </rdf:Description>
    </owl:unionOf>

  </owl:Class>
</rdf:RDF>
```

<3>

```
<owl:Class rdf:about="#Man">
  <owl:disjointWith rdf:resource="#Woman"/>
</owl:Class>
```

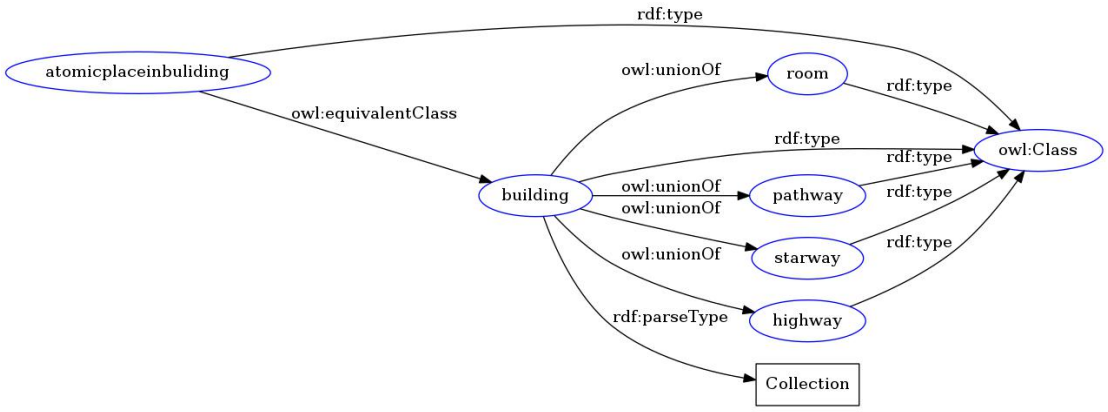


<4>

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:owl="http://www.w3.org/2002/07/owl#" xmlns:rdfs="http://www.w3.org/2000/01/rdf-
schema#" xmlns="http://example.com/ex#" xmlns:xsd="http://www.w3.org/2001/XMLSchema#" >

<owl:Class rdf:ID="AtomicPlaceInBuilding">

 <rdfs:subClassOf>
 <owl:Class rdf:about="#AtomicPlace"/>
 </rdfs:subClassOf>
 <owl:equivalentClass>
 <owl:Class>
 <owl:unionOf rdf:parseType="Collection">
 <owl:Class rdf:about="#Room"/>
 <owl:Class rdf:about="#Hallway"/>
 <owl:Class rdf:about="#Stairway"/>
 <owl:Class rdf:about="#OtherPlaceInBuilding"/>
 </owl:unionOf>

 </owl:Class>
 </owl:equivalentClass>
 </owl:Class>
</rdf:RDF>
```

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


<#Stairway>  a  owl:Class .
<#Room>  a     owl:Class .
<#AtomicPlaceInBuilding>    a              owl:Class ;
      rdfs:subClassOf     <#AtomicPlace> ;
      owl:equivalentClass  [ a         owl:Class ;
                  owl:unionOf  ( <#Room> <#Hallway> <#Stairway> <#OtherPlaceInBuilding> )
                ] .
<#AtomicPlace>  a  owl:Class .
<#Hallway>  a   owl:Class .
<#OtherPlaceInBuilding>
     a      owl:Class .
```

```
@prefix owl: <http://www.w3.org/2002/07/owl#> .@prefix rdfs:
<http://www.w3.org/2000/01/rdf-schema#> .
<http://njh.me/#AtomicPlaceInBuilding>
  a owl:Class ;
  rdfs:subClassOf <http://njh.me/#AtomicPlace> ;
  owl:equivalentClass [
    a owl:Class ;
    owl:unionOf (
     <http://njh.me/#Room>
     <http://njh.me/#Hallway>
     <http://njh.me/#Stairway>
     <http://njh.me/#OtherPlaceInBuilding>
   )
  ] .
<http://njh.me/#AtomicPlace> a owl:Class .<http://njh.me/#Room> a
owl:Class .<http://njh.me/#Hallway> a
owl:Class .<http://njh.me/#Stairway> a
owl:Class .<http://njh.me/#OtherPlaceInBuilding> a owl:Class .
```



Namespaces:
http://example.com/ex#
rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
owl: http://www.w3.org/2002/07/owl#
xsd: http://www.w3.org/2001/XMLSchema#
rdfs: http://www.w3.org/2000/01/rdf-schema#

These are another technique to write this above given question: ????

```
@prefix :     <http://example.com/ex#> .
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix owl:  <http://www.w3.org/2002/07/owl#> .
@prefix xsd:  <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
:room     a    owl:Class .
:pathway  a    owl:Class .
:starway  a    owl:Class .
:highway  a    owl:Class .
:atomicplaceinbuliding  a    owl:Class ;
owl:equivalentClass  :building  .
:building  a    owl:Class ;
rdf:parseType "Collection";
owl:unionOf  :room,
 :pathway,
:starway,
 :highway.
```

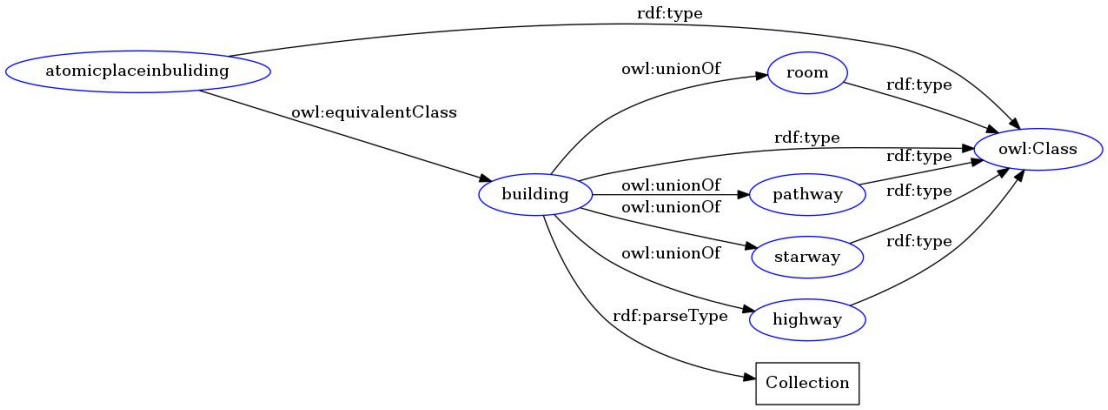
X

Wrong

```
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
<http://example.com/ex#room> a owl:Class.
<http://example.com/ex#pathway> a owl:Class .
<http://example.com/ex#starway> a owl:Class .
<http://example.com/ex#highway> a owl:Class .
<http://example.com/ex#atomicplaceinbuliding> a owl:Class ;
  owl:equivalentClass <http://example.com/ex#building> .
<http://example.com/ex#building>  a owl:Class ;
```

```
  rdf:parseType "Collection" ;
  owl:unionOf <http://example.com/ex#room>,
<http://example.com/ex#pathway>,
 <http://example.com/ex#starway>,
 <http://example.com/ex#highway> .
```

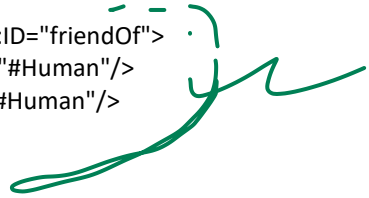
```
<?xml version="1.0" encoding="utf-8" ?><rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:owl="http://www.w3.org/2002/07/owl#">

  <owl:Class rdf:about="http://example.com/ex#room">
  </owl:Class>

  <owl:Class rdf:about="http://example.com/ex#pathway">
  </owl:Class>

  <owl:Class rdf:about="http://example.com/ex#starway">
  </owl:Class>

  <owl:Class rdf:about="http://example.com/ex#highway">
  </owl:Class>

  <owl:Class rdf:about="http://example.com/ex#atomicplaceinbuliding">
    <owl:equivalentClass>
      <owl:Class rdf:about="http://example.com/ex#building">
        <rdf:parseType>Collection</rdf:parseType>
        <owl:unionOf rdf:resource="http://example.com/ex#room"/>
        <owl:unionOf rdf:resource="http://example.com/ex#pathway"/>
        <owl:unionOf rdf:resource="http://example.com/ex#starway"/>
        <owl:unionOf rdf:resource="http://example.com/ex#highway"/>
      </owl:Class>
    </owl:equivalentClass>

  </owl:Class>
</rdf:RDF>
```



Namespaces:
http://example.com/ex#
rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
owl: http://www.w3.org/2002/07/owl#
xsd: http://www.w3.org/2001/XMLSchema#
rdfs: http://www.w3.org/2000/01/rdf-schema#

<5>

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:owl="http://www.w3.org/2002/07/owl#" xmlns:rdfs="http://www.w3.org/2000/01/rdf-
schema#" xmlns="http://example.com/ex#" xmlns:xsd="http://www.w3.org/2001/XMLSchema#" >

<owl:Class rdf:about="#friendOf">
</owl:Class>
<owl:Class rdf:about="#Human">
</owl:Class>
<owl:SymmetricProperty rdf:ID="friendOf">
  <rdfs:domain rdf:resource="#Human"/>
  <rdfs:range  rdf:resource="#Human"/>
</owl:SymmetricProperty>
</rdf:RDF>
```



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

<#Human> a    owl:Class .

<#friendOf> a     owl:Class , owl:SymmetricProperty ;
    rdfs:domain  <#Human> ;
    rdfs:range   <#Human> .
```

```
<?xml version="1.0" encoding="utf-8" ?><rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:owl="http://www.w3.org/2002/07/owl#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">

  <owl:Class rdf:about="http://njh.me/#friendOf">
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#SymmetricProperty"/>
    <rdfs:domain rdf:resource="http://njh.me/#Human"/>
    <rdfs:range rdf:resource="http://njh.me/#Human"/>
  </owl:Class>

  <owl:Class rdf:about="http://njh.me/#Human">
  </owl:Class>
```

```
</rdf:RDF>
```

- There exist two **predefined classes**
  - `owl:Thing` (class that contains all individuals)
  - `owl:Nothing` (empty class)

- **Definition of a class**

```
:GreenhouseGas a owl:Class .
```

This is OWL in RDF/Turtle serialization

OWL Individuals

- **Definition of individuals** via class membership

```
:JosephFourier a :Person .
```

`Person(JosephFourier)`

- Individuals can also be defined **without class membership** as **named individuals**

```
:HaraldSack a owl:NamedIndividual .
```

- **Object properties** have classes as range

```
:discoverer a owl:ObjectProperty .
```

- **Domain** and **Range** of object properties

```
:discoverer a owl:ObjectProperty ;
            rdfs:domain owl:Thing ;
            rdfs:range  :Person .
```

$\exists \text{discoverer}.\top \sqsubseteq \top$

$\top \sqsubseteq \forall \text{discoverer}.\text{Person}$

- **Datatype properties** have datatypes as range

```
:discoveredIn a owl:DatatypeProperty .
```

- **Domain** and **Range** of datatype properties

```
:discoveredIn a owl:DatatypeProperty ;
              rdfs:domain owl:Thing ;
              rdfs:range  xsd:date .
```

$\exists \text{discoveredIn}.\top \sqsubseteq \top$

$\top \sqsubseteq \forall \text{discoveredIn}.\text{Date}$

OWL Properties and Individuals - Example

OWL TBox

```
:AtmosphericProcess a owl:Class .
:Person a owl:Class .

:discoverer a owl:ObjectProperty ;
            rdfs:domain owl:Thing ;
            rdfs:range  :Person .

:discoveredIn a owl:DatatypeProperty ;
              rdfs:domain owl:Thing ;
              rdfs:range  xsd:date .
```

OWL ABox

```
:JosephFourier a Person .
:GreenhouseEffect a :AtmosphericProcess ;
                  :discoverer :JosephFourier ;
                  :discoveredIn "1824-00-00"^^xsd:date .
```

OWL Class Hierarchies

```
:Physicist a owl:Class ;
       rdfs:subClassOf :Scientist .
:Scientist a owl:Class ;
      rdfs:subClassOf :Person .
:Person a owl:Class .
```

we don't need to define a new subClassOf property for owl, we simply reuse rdfs:subClassOf

$\text{Physicist} \sqsubseteq \text{Scientist}$
$\text{Scientist} \sqsubseteq \text{Person}$

- Via **inference** it can be entailed that `:Physicist` is also a subclass of `:Person`.

```
:ChemicalSubstance a owl:Class .
:Person a owl:Class .
:GreenhouseGas a owl:Class ;
       rdfs:subClassOf :ChemicalSubstance .
:Scientist a owl:Class ;
      rdfs:subClassOf :Person .

:ChemicalSubstance owl:disjointWith :Person .
```

In owl everything might be potentially identical if we don't explicitly state the difference

$GreenhouseGas \sqsubseteq ChemicalSubstance$
$Scientist \sqsubseteq Person$
$ChemicalSubstance \sqcap Person \sqsubseteq \bot$

- Via **inference** it can be entailed that `:GreenhouseGas` and `:Scientist` are also disjoint classes.

```
:Scientist  a owl:Class .
:Researcher a owl:Class .
:Physicist  a owl:Class ;
      rdfs:subClassOf :Scientist .

:Scientist  owl:equivalentClass :Researcher .
```

```
<rdfs:class rdf:about="#Continent">
</rdfs:class>
```

```
<rdfs:class rdf:about="#Person">
</rdfs:class>
```

```
<rdfs:class rdf:about="#happyperson">
<owl:subclassof rdf:resource=”#person”>
</rdfs:class>
```

```
<rdfs:class rdf:about="#happyperson">
<owl:subclassof rdf:resource=”#person”>
</rdfs:class>
```

```
<owl:ObjectProperty rdf:about="#happyperson">
<rdfs:domain rdf:resource="#person1’/>
<rdfs:range rdf:resource="#child"/>

</owl:ObjectProperty>
```

SEMANTIC WEB CLASS **DATE-28/10/2021**

<1> question

HAPPPYPERSON IS A PERSON ,WHO HAS_CHILD.

Method1===rdf-xml

```
<rdfs:class rdf:about="#Person">
</rdfs:class>

<rdfs:class rdf:about="#happyperson">
<owl:subclassof rdf:resource=”#person”/>
</rdfs:class>

<owl:ObjectProperty rdf:about="#haschild">
<rdfs:domain rdf:resource="#person"/>
<rdfs:range rdf:resource="#child"/>
</owl:ObjectProperty>

<rdfs:class rdf:about="#child">
<owl:subclassof rdf:resource=”#person”/>
<owl:Restriction>
<owl:onproperty  rdf:resource=”age”/>
<owl:maxCardinality  rdf:datatype=”&xsd;NonNegative”> 15</owl:maxCardinality>
</owl:Restriction>
</rdfs:class>
```

```
:person a owl:class.
:child a owl:class;
  rdf:subClassOf  :person.
:person a owl:class;
  rdf:subClassOf  :person.
```

<2>QUESTION

There are 2 types of train.Those are passenger Train and goods train and both of them are disjoint.Passenger train carries passenger and goods train carries goods.

Mtehod--TURTLE

```
:train  a owl:Class.
:person a owl:Class.

:pass_train a owl:Class;
rdfs:subClassOf   :train.

:goods_train a owl:Class;
rdfs:subClassOf   :train;
owl:disjointwith   :pass_train.

:pass a owl:Class;
rdfs:subClassOf   :person.

:good a owl:Class.

:Good_pass  a owl:Class ;
  owl:unionOf (
<#goods>
<#pass>
 ) .
:carrier a  owl:ObjectProperty;
rdfs:domain :train;
rdfs:range  :Good_pass.
```

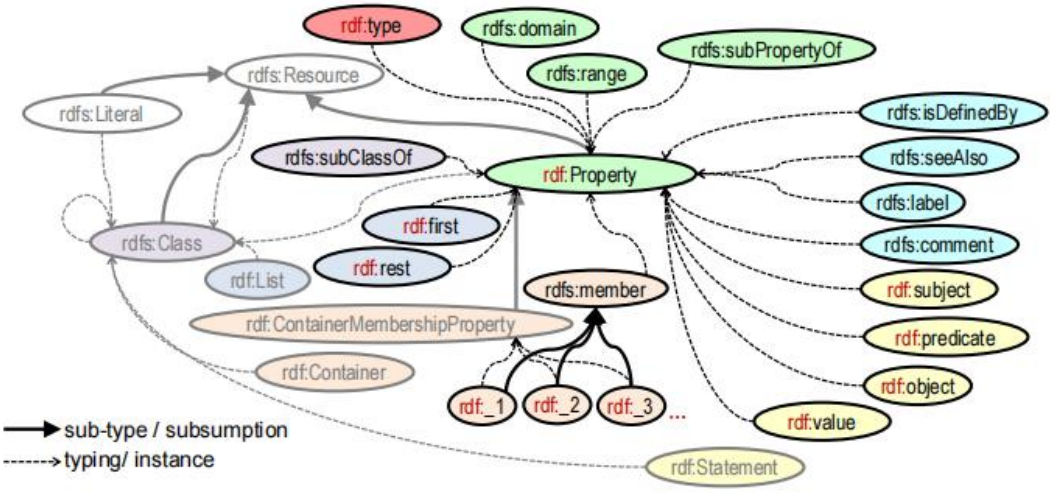
```
:Good_pass  a owl:Class ;
 owl:unionOf   :goods,   :pass.
```

METHOD-2 ---RDF-XML

```
<owl:class rdf:about  = "Train"/>
<owl:class rdf:about = "Person"/>
<owl:class    rdf:about = "Passenger_Train">
  <rdfs:subClassOf rdf:resource="Train"/>

<owl:class rdf:about = "Goods_Train">
  <rdfs:subClassOf rdf:resource="Train"/>
       <owl:disjointWith rdf:resource="#Passenger_Train"/>
       </owl:class>

<owl:class rdf:about = "Passenger">
  <rdfs:subClassOf rdfs:resource="Person"/>

<owl:class rdf:about  = "Goods"/>

<owl:objectPrperty rdf:about=#carries">
 <rdfs:domain rdf:resource="Passenger_Train">
 <rdfs:range rdf:resource="Passenger">
 <rdfs:range rdf:resource="Good">
</owl:objectPrperty >
```

RDFS properties

meta-properties and some of their links



Quantification can be contrary to our intuition.

- Universal quantification over an empty set is true
- `:John` may belong to the class `:OnlyDaughterParent` if he has no child at all and we describe that class as:

```
:OnlyDaughterParent  rdf:type     owl:Class ;
      owl:equivalentClass  [
         rdf:type             owl:Restriction ;
         owl:onProperty       :hasChild ;
         owl:allValuesFrom    :Female
      ] .
```

instances of rdfs:Class

the class of classes is in RDFS namespace.

```
<rdf:RDF xml:base="http://inria.fr/2005/humans.rdfs"
 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:ID="Man">
  <rdfs:subClassOf rdf:resource="#Person"/>
  <rdfs:subClassOf rdf:resource="#Male"/>
 </rdfs:Class>
</rdf:RDF>
```

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@base <http://inria.fr/2005/humans.rdfs> .
<Man> a rdfs:Class ;
      rdfs:subClassOf <Person>, <Male> .
```

semantics

1. Every class is a subclass of rdfs:Resource
   IF `c rdf:type rdfs:Class` THEN
   `c rdfs:subClassOf rdfs:Resource`
2. Type propagation
   IF `c2 rdfs:subClassOf c1` AND `x rdf:type c2`
   THEN `x rdf:type c1`
3. Reflexivity of subsumption
   IF `c rdf:type rdfs:Class`
   THEN `c rdfs:subClassOf c`
4. Transitivity of subsumption
   IF `c2 rdfs:subClassOf c1` AND `c3 rdfs:subClassOf c2`
   THEN `c3 rdfs:subClassOf c1`

instances of rdf:Property

the class of properties was placed in the RDF namespace because triples are a construction of RDF.

```
<rdf:RDF xml:base="http://inria.fr/2005/humans.rdfs"
 xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
 <rdf:Property rdf:ID="hasMother">
  <rdfs:subPropertyOf rdf:resource="#hasParent"/>
 </rdf:Property>
</rdf:RDF>
```

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@base <http://inria.fr/2005/humans.rdfs> .
<hasMother> a rdf:Property ;
  rdfs:subPropertyOf <hasParent> .
```

semantics

1. Type propagation
   IF `p2 rdfs:subPropertyOf p1` AND `x p2 y`
   THEN `x p1 y`
2. Reflexivity of subsumption
   IF `p rdf:type rdf:Property`
   THEN `p rdfs:subPropertyOf p`
3. Transitivity of subsumption
   IF `p2 rdfs:subPropertyOf p1` AND `p3 rdfs:subPropertyOf p2`
   THEN `p3 rdfs:subPropertyOf p1`

domain and range

Class of departure or domain: rdfs:domain
Class of arrival, co-domain or range: rdfs:range

```
<rdf:RDF xml:base="http://inria.fr/2005/humans.rdfs"
 xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
 <rdf:Property rdf:ID="hasMother">
  <rdfs:subPropertyOf rdf:resource="#hasParent"/>
  <rdfs:domain rdf:resource="#Human"/>
  <rdfs:range  rdf:resource="#Woman"/>
 </rdf:Property>
</rdf:RDF>
```

domain and range

Class of departure or domain: rdfs:domain
Class of arrival, co-domain or range: rdfs:range

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@base <http://inria.fr/2005/humans.rdfs> .
<hasMother> a rdf:Property ;
  rdfs:subPropertyOf <hasParent> ;
  rdfs:domain <Human> ;
  rdfs:range  <Woman> .
```

rdfs:label

a resource may have one or more labels in one or more natural language

```
<rdf:Property rdf:ID='name'>
 <rdfs:domain rdf:resource='Person'/>
 <rdfs:range rdf:resource='&rdfs;Literal'/>
 <rdfs:label xml:lang='fr'>nom</rdfs:label>
 <rdfs:label xml:lang='fr'>nom de famille</rdfs:label>
 <rdfs:label xml:lang='en'>name</rdfs:label>
</rdf:Property>
```

```
<name> a rdf:Property ;
  range rdfs:Literal ; domain <Person> ;
  label "nom"@fr, "nom de famille"@fr, "name"@en .
```

textual labels attached to resources

any resource may have one or more labels in one or more languages

```
<rdf:RDF xml:base="http://inria.fr/2005/humans.rdfs"
 xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
 <rdf:Property rdf:ID='name'>
  <rdfs:label xml:lang='fr'>nom</rdfs:label>
  <rdfs:label xml:lang='fr'>nom de famille</rdfs:label>
  <rdfs:label xml:lang='en'>name</rdfs:label>
 </rdf:Property>
</rdf:RDF>
```

textual labels attached to resources

any resource may have one or more labels in one or more languages

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@base <http://inria.fr/2005/humans.rdfs> .
<name> a rdf:Property ;
       rdfs:label "nom"@fr, "nom de famille"@fr, "name"@en .
```

rdfs:comment & rdfs:seeAlso

comments provide definitions and explanations in natural language

```
<rdfs:Class rdf:about='#Woman'>
  <rdfs:subClassOf rdf:resource="#Person"/>
  <rdfs:comment xml:lang='fr'>une personne adulte du
                              sexe féminin</rdfs:comment>
  <rdfs:comment xml:lang='en'>a female adult person
  </rdfs:comment>
</rdfs:Class>
```

```
<Woman> a rdfs:Class ; rdfs:subClassOf <Person> ;
  rdfs:comment "adult femal person"@en ;
  rdfs:comment "une adulte de sexe féminin"@fr .
```

see also...

```
<rdfs:Class rdf:about='#Man'>
    <rdfs:seeAlso rdf:resource='#Woman'/>
</rdfs:Class>
```

```
<Man> a rdfs:Class ; rdfs:seeAlso <Woman> .
```

example of RDFS classes

```
<rdf:RDF xml:base  ="http://inria.fr/2005/humans.rdfs"
 xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
 xmlns     ="http://www.w3.org/2000/01/rdf-schema#>
 <Class rdf:ID="Man">
  <subClassOf rdf:resource="#Person"/>
  <subClassOf rdf:resource="#Male"/>
   <label xml:lang="en">man</label>
   <comment xml:lang="en">an adult male person</comment>
 </Class>
```

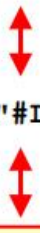
```
<Man> a Class ; subClassOf <Person>, <Male> .
```

example of RDFS properties

```
<rdf:RDF xml:base  ="http://inria.fr/2005/humans.rdfs"
  xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns     ="http://www.w3.org/2000/01/rdf-schema#>
  <rdf:Property rdf:ID="hasMother">
    <subPropertyOf rdf:resource="#hasParent"/>
    <range rdf:resource="#Female"/>
    <domain rdf:resource="#Human"/>
    <label xml:lang="en">has for mother</label>
    <comment xml:lang="en">to have for parent a female.
                                                  </comment>
  </rdf:Property>
```

```
<hasMother> a rdf:Property ;
  subPropertyOf <hasParent> ;
  range <Female> ; domain <Human> .
```

```
<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="http://inria.fr/2005/humans.rdfs#"
  xml:base=" http://inria.fr/2005/humans.rdfs-instances" >

  <rdf:Description rdf:ID="Lucas">
     <rdf:type
  rdf:resource="http://inria.fr/2005/humans.rdfs#Man"/>
     <hasMother rdf:resource="#Laura"/>
  </rdf:Description>

  <Man rdf:ID="Lucas">
     <hasMother rdf:resource="#Laura"/>
  </Man>
```



```
 <Luca> a Man; hasMother <Laura> .
```

example of RDF using this schema

usage and references to schemas

in a resource description

```
<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:h="http://inria.fr/2005/humans.rdfs#"
 xml:base="http://inria.fr/2005/humans.rdfs-instances" >
 <rdf:Description rdf:ID="Lucas">
   <rdf:type rdf:resource="http://inria.fr/2005/humans.rdfs#Man"/>
   <h:hasMother rdf:resource="#Laura"/>
 </rdf:Description>
</rdf:RDF>
```

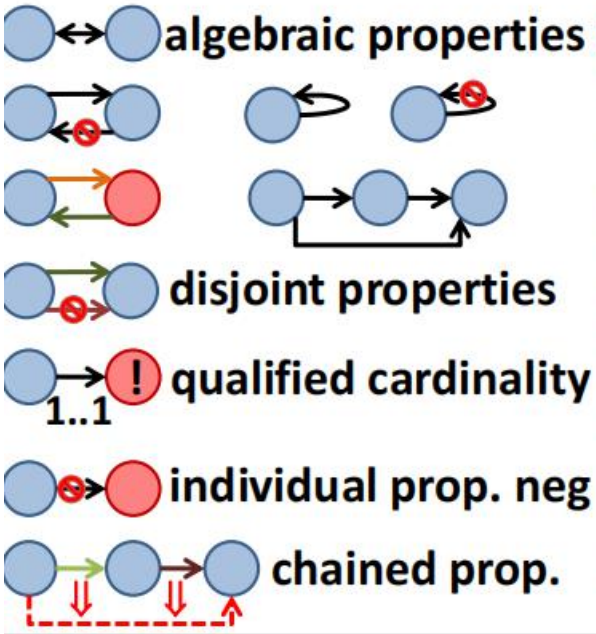
usage and references to schemas

in a resource description

```
<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:h="http://inria.fr/2005/humans.rdfs#"
 xml:base=" http://inria.fr/2005/humans.rdfs-instances" >
 <h:Man rdf:ID="Lucas">
   <h:hasMother rdf:resource="#Laura"/>
 </h:Man>
</rdf:RDF>
```


```
@prefix h: <http://inria.fr/2005/humans.rdfs#> .
@base <http://inria.fr/2005/humans.rdfs-instances> .
<Lucas> a h:Man; h:hasMother <Laura> .
```

OWL in one...

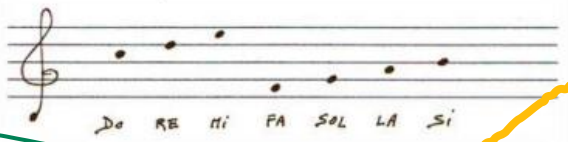


enumerated class {a,b,c,d,e}

define a class by providing all its members

Collection

```
<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> ) .
```

classes defined by union

of other 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> ) .
```

classes defined by intersection

of other classes

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

intersection-Of

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

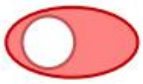
No comma

complement

of a class

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

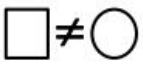
```
<Inedible> rdf:type owl:Class ;
  owl:complementOf <Edible> .
```



disjunction

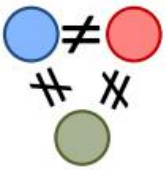
of classes

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

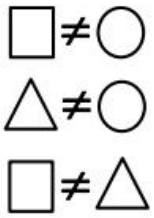


```
<Square> rdf:type owl:Class ;
  owl:disjointWith <Circle> .
```

disjunction
of several classes



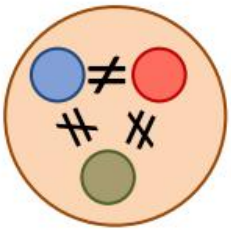
```
<owl:AllDisjointClasses>
   <owl:members rdf:parseType="Collection">
     <owl:Class rdf:about="#Square"/>
     <owl:Class rdf:about="#Circle"/>
     <owl:Class rdf:about="#Triangle"/>
   </owl:members>
 </owl:AllDisjointClasses>
```



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

Blank Node

disjoint union
of several classes



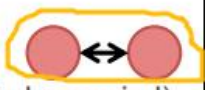
```
<owl:Class rdf:about="Passenger">
  <owl:disjointUnionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#Adult"/>
    <owl:Class rdf:about="#Child"/>
    <owl:Class rdf:about="#Pet"/>
  </owl:disjointUnionOf>
</owl:Class>
```

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

types of properties

- `owl:ObjectProperty` are relations between resources only e.g. hasParent(#thomas,#stephan)
- `owl:DatatypeProperty` have a literal value possibly typed ex:hasAge(#thomas,16^^xsd:int)
- `owl:AnnotationProperty` are ignored in inferences and used for documentation and extensions

symmetric property



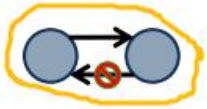
a relation that, as soon as it exists, exists in both directions (e.g. to be maried)

$x R y \Rightarrow y R x$

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

```
<hasSpouse> a owl:SymmetricProperty .
```

asymmetric property

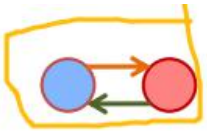


a relation that, as soon as it exists, exists in only one direction (e.g. parent)

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

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

```
<hasChild> a owl:AsymmetricProperty .
```



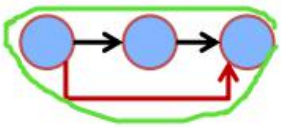
inverse property

two relations that exist simultaneously and inversly (ex. parent_of / child_of)

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

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

```
<hasChild> owl:inverseOf <hasParent> .
```



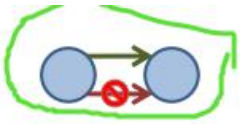
transitive property

a property propagated from peers to peers (e.g ancestors)

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

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

```
<hasAncestor> a owl:TransitiveProperty
```



disjoint properties

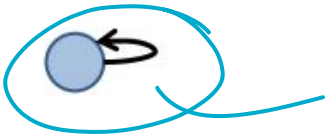
relations that cannot exist together on the same subject and the same object

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

```
<hasSon> owl:propertyDisjointWith <hasDaughter> .
```

reflexive property

a relation that links all individuals to themselves



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

<hasRelative> a owl:ReflexiveProperty .
```

irreflexives proprieties

relations never link resources to themselves

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

<hasParent> a  owl:IrreflexiveProperty .
```

functional property

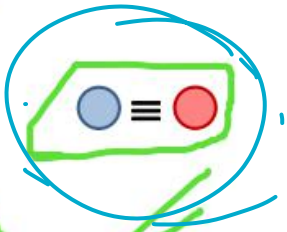
a relation for which a resource can have only one value (e.g. birth date)

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

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

<birthDate> a owl:FunctionalProperty .
```

equivalent classes

two classes containing exactly the same resources.



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

mit:Student owl:equivalentClass keio:Gakusei .
```

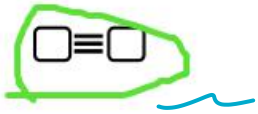
equivalent properties

two properties representing exactly the same relation.

```
ex:name owl:equivalentProperty my:label
```

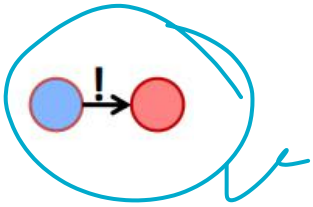
identical resources

two URIs identifying exactly the same thing.



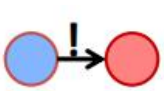
```
ex:Bill owl:sameAs ex:William
```

restriction on all values



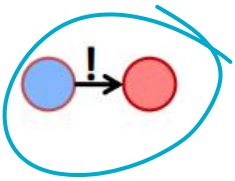
```
<owl:Class rdf:ID="Herbivore">
  <subClassOf rdf:resource="#Animal"/>
  <subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#eats" />
      <owl:allValuesFrom rdf:resource="#Plant" />
    </owl:Restriction>
  </subClassOf>
</owl:Class>
```

restriction on some values



```
<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>
```

restriction to an exact value



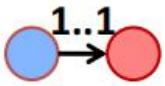

```
<owl:Class rdf:ID="Bike">
 <subClassOf>
  <owl:Restriction>
   <owl:onProperty rdf:resource="#nbWheels" />
   <owl:hasValue>2</owl:hasValue>
  </owl:Restriction>
 </subClassOf>
</owl:Class>
```

self restriction

classes where instances have themselves as value of a property


```
ex:NarcisticPerson rdfs:subClassOf
[ a owl:Restriction ;
   owl:onProperty ex:love ;
   owl:hasSelf true ]
```

restriction on cardinality



how many times a property is used for
a same subject but with different values

- Constraints: minimum, maximum, exact number
- Exemple

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

OWL Class Hierarchies and Equivalence

```
:Scientist  a owl:Class .
:Researcher a owl:Class .
:Physicist  a owl:Class ;
      rdfs:subClassOf :Scientist .

:Scientist  owl:equivalentClass :Researcher .
```

```
Physicist ⊑ Scientist
Scientist ≡ Researcher
```

- Via **inference** it can be entailed that `:Physicist` is also a `:Researcher`.

OWL Individuals - Identity and Distinctiveness

```
:CarbonDioxide a :GreenhouseGas ;
    :discoverer :JosephBlack ;
    :discoveredIn "1750-00-00"^^xsd:date ;
    owl:sameAs :ARX012345 .
:GreenhouseGas a owl:Class ;
       rdfs:subClassOf :ChemicalSubstance .

:ChemicalSubstance a owl:Class.
```

For identical individuals: `owl:sameAs`
For identical classes: `owl:equivalentClass`

- Via **inference** it can be entailed that `:ARX012345` is a `:ChemicalSubstance`.
- **Difference of Individuals** via `owl:differentFrom`.

```
:ARX012345 a :GreenhouseGas ;
           owl:differentFrom :ARX012346 .
```

OWL Complex Classes - Nominals

```
:Chemist   a owl:Class .
:Physicist a owl:Class .
:JosephFourier a :Physicist .
:JanBaptistVanHelmont a :Physicist .
:JosephBlack a Chemist .

:CarbonDioxideClub a owl:Class ;
     owl:oneOf
     ( :JosephFourier
       :JanBaptistVanHelmont
       :JosephBlack ) .
```



```
CarbonDioxideClub ⊑ { JosephFourier,
                      JanBaptistVanHelmont,
                      JosephBlack }
```

No Comma inside.

- There are only three scientists in the Carbon Dioxide Club.

OWL Logical Class Constructors

- logical AND (conjunction): `owl:intersectionOf` ⊓
- logical OR (disjunction): `owl:unionOf` ⊔
- logical negation: `owl:complementOf` ¬

Used to create complex classes from atomic classes.

OWL Logical Class Constructors - Intersection


```
:Scientist a owl:Class .
:ClimateActivist a owl:Class .
:Scientist4Future a owl:Class ;
      owl:intersectionOf (:Scientist :ClimateActivist) .
```

Scientist4Future ≡ Scientist ⊓ ClimateActivist

- Scientists4Future are Scientists who are also ClimateActivists.

OWL Logical Class Constructors - Union


```
:Environmentalist a owl:Class ;
      owl:equivalentClass [
            owl:unionOf ( :ClimateActivist
                          :AnimalRightsActivist
                          :EnergySaver )
] .
```

Environmentalist ≡ ClimateActivist ⊔ AnimalRightsActivist ⊔ EnergySaver

- Climate Activists, Animal Rights Activists, and Energy Savers are all Environmentalists.

OWL Logical Class Constructors - Negation

```
:Pacifist a owl:Class .

:Warmonger a owl:Class ;
     owl:complementOf (:Pacifist) .
```

Warmonger ≡ ¬Pacifist

- warmongers are the opposite of Pacifists.

OWL Property Restrictions

- **OWL property restrictions** are used to describe **complex classes** via **properties**
- Restrictions on values:
  - owl:hasValue
  - owl:allValuesFrom
  - owl:someValuesFrom
- Restrictions on cardinality:
  - owl:cardinality
  - owl:minCardinality
  - owl:maxCardinality

OWL Property Restrictions with Constants

```
:FouriersDiscoveries a owl:Class ;
              rdfs:subClassOf
                [ a owl:Restriction ;
                  owl:onProperty :discoverer ;
                  owl:hasValue :JosephFourier ] .
```

FouriersDiscoveries ⊑ discoverer.(JosephFourier)

- The class :FouriersDiscoveries is described via fixed value assignment (=constant) of the individual :JosephFourier to the property :discoverer.
- Fourier's Discoveries have been discovered by Joseph Fourier.

OWL Properties Restriction with Strict Binding

```
:VegetarianDish a owl:Class ;
      rdfs:subClassOf
         [ a owl:Restriction ;
           owl:onProperty :ingredient ;
           owl:allValuesFrom :VegetarianFood ] .
```

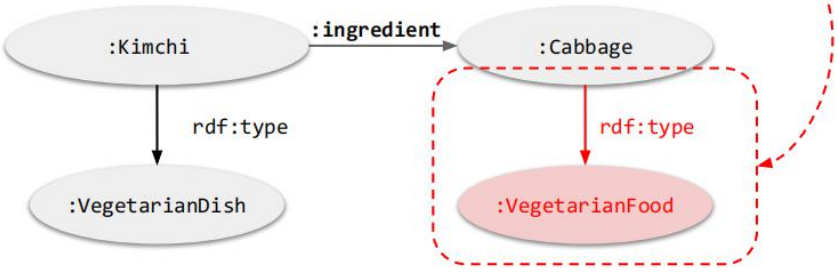
$\text{VegetarianDish} \sqsubseteq \forall \text{ingredient}.\text{VegetarianFood}$

- owl:allValuesFrom fixes all instances of a specific class C as allowed range for a property p (strict binding) $\forall p.C$

- The ingredients of a vegetarian dish are only vegetarian food.

OWL Properties Restriction with Strict Binding


```
:VegeratianDish a owl:Class ;
       rdfs:subClassOf
         [ a owl:Restriction ;
           owl:onProperty :ingredient ;
           owl:allValuesFrom :VegetarianFood ] .
```



Logical inference

OWL Property Restriction with Loose Binding

```
:Reader a owl:Class ;
       rdfs:subClassOf
         [ a owl:Restriction ;
           owl:onProperty :reads ;
           owl:someValuesFrom :Book ] .
```

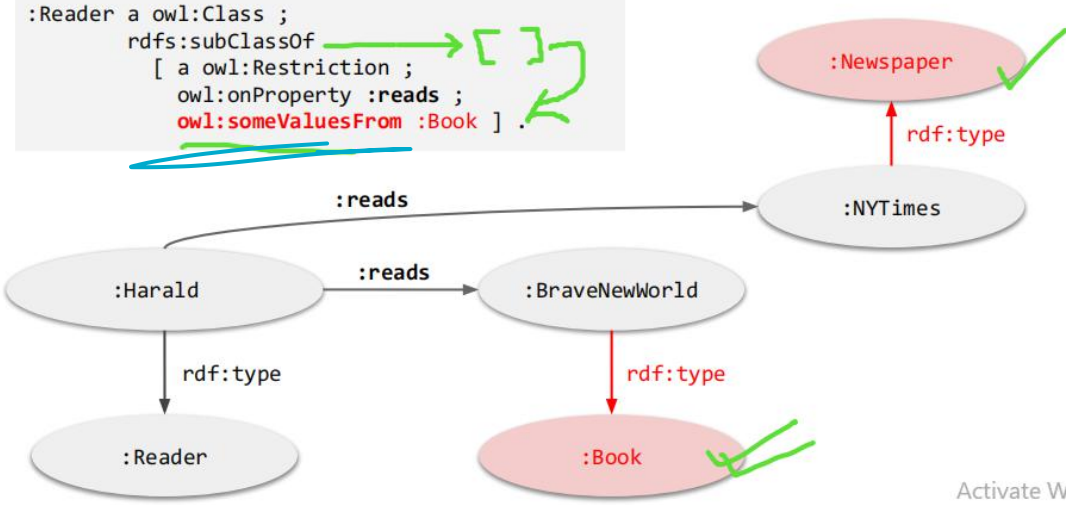
$\text{Reader} \sqsubseteq \exists \text{reads}.\text{Book}$

- owl:someValuesFrom describes that there must exist an individual for p and fixes its range to class C (loose binding) $\exists p.C$

A reader reads amongst other things also books.

OWL Property Restriction with Loose Binding

```
:Reader a owl:Class ;
        rdfs:subClassOf
          [ a owl:Restriction ;
            owl:onProperty :reads ;
            owl:someValuesFrom :Book ] .
```



OWL Property Restrictions with Cardinality

```
:StringQuartet a owl:Class ;
     rdfs:subClassOf
        [ a owl:Restriction ;
          owl:onProperty :member ;
          owl:cardinality "4"^^<http://www.w3.org/2001/XMLSchema#int> ] .
```

StringQuartett ⊑ =4.member.⊤

- A String Quartet has exactly 4 members.
- Any instance of Class `:StringQuartet` must have exactly 4 values for the property `:member`.
- For `owl:maxCardinality` and `owl:minCardinality` the restriction gives **upper and lower bounds** on property value cardinalities.

Loose or Strict Binding?

- "A vegetarian does not eat animals"
- How to model this restriction with the classes `:Vegetarian`, `:Animal` and the property `:eats` ?

```
Vegetarian a owl:Class ;
      rdfs:subClassOf [
           a owl:Restriction ;
           owl:onProperty :eats ;
           owl:allValuesFrom [
                owl:complementOf :Animal .
           ] .
]
```

```
Vegetarian a owl:Class ;
rdfs:subClassOf [
        a owl:Restriction ;
        owl:onProperty :eats ;
        owl:allValuesFrom [
                owl:complementOf :Animal .
                ] .
]
```

Loose or Strict Binding?

- "A Driver is somebody who drives a car."
- How to model this restriction with the classes :Driver, :Car and the property :drives ?

```
Driver a owl:Class ;
      rdfs:subClassOf [
            a owl:Restriction ;
            owl:onProperty :drives ;
            owl:someValuesFrom :Car .
]
```

OWL Property Relationships

- **Property hierarchies** can be created via specializations: rdfs:subPropertyOf.
- **Inverse properties** are defined via owl:inverseOf.
- **Identical properties** are defined via owl:equivalentProperty.

```
:isMadeOf a owl:ObjectProperty ;
      rdfs:subPropertyOf :consistsOf .

:reads a owl:ObjectProperty ;
      owl:inverseOf :isReadBy .

:composedOf a owl:ObjectProperty ;
      owl:equivalentProperty :consistsOf .
```

```
isMadeOf   ⊑ consistsOf
reads-     ≡ isReadBy
composedOf ≡ consistsOf
```

OWL Property Relationships

- `owl:TransitiveProperty`
  e.g.: if `isPartOf(a,b)` and `isPartOf(b,c)` then it holds that `isPartOf(a,c)`.
- `owl:SymmetricProperty`
  e.g.: if `isNeighborOf(a,b)` then it holds that `isNeighborOf(b,a)`.
- `owl:FunctionalProperty`
  e.g.: if `hasMother(a,b)` and `hasMother(a,c)` then it holds that `b=c`.
- `owl:InverseFunctionalProperty`
  e.g.: if `isMotherOf(b,a)` and `isMotherOf(c,a)` then it holds that `b=c`.

OWL Transitive Properties

```
:isPublishedBefore a owl:ObjectProperty ;
                   a owl:TransitiveProperty ;
                   rdfs:domain owl:Book ;
                   rdfs:range owl:Book .

:AnimalFarm a owl:Book ;
            :isPublishedBefore :NineteenEightyfour .

:BraveNewWorld a :Book ;
            :isPublishedBefore :AnimalFarm .
```

- Via inference it can be entailed that
  `:BraveNewWorld :isPublishedBefore :NineteenEightyfour .`

More Property Relationships

- **Asymmetric properties** via `owl:AsymmetricProperty`
  e.g.: if it holds that `isLeftOf(a,b)`
  then it is not possible that `isLeftOf(b,a)` .
- **Reflexive properties** via `owl:ReflexiveProperty`
  e.g.: `isRelatedTo(x,x)` .
- **Irreflexive properties** via `owl:IrreflexiveProperty`
  e.g.: if `isParentOf(x,y)` then $x \neq y$.

Disjunctive Properties

- Two properties R and S are **disjunctive**,
  if two individuals x,y are never related via both properties

```
:hasSibling a owl:ObjectProperty ;
            owl:propertyDisjointWith :hasChild .
```

- **Shortcut** for several **disjunctive properties**

```
[] rdf:type owl:AllDisjointProperties ;
   owl:members
      ( :hasSibling
        :hasChild
        :hasGrandchild ) .
```

OWL RDF Syntax: Complex Classes

```
:Mother  owl:equivalentClass  [
  rdf:type              owl:Class ;
  owl:intersectionOf    ( :Woman :Parent )
] .
```

Mother ≡ Woman ⊓ Parent

```
:Parent  owl:equivalentClass  [
  rdf:type      owl:Class ;
  owl:unionOf   ( :Mother :Father )
] .
```

Parent ≡ Mother ⊔ Father

```
:ChildlessPerson  owl:equivalentClass  [
  rdf:type              owl:Class ;
  owl:intersectionOf    ( :Person  [ owl:complementOf  :Parent ] )
] .
```

ChildlessPerson ≡ Person ⊓ ¬Parent

```
:Grandfather  rdfs:subClassOf  [
  rdf:type              owl:Class ;
  owl:intersectionOf    ( :Man  :Parent )
] .
```


Person ⊓ ¬Parent (Jack)

```
:Jack  rdf:type  [
  rdf:type              owl:Class ;
  owl:intersectionOf    ( :Person
                          [ rdf:type           owl:Class ;
                            owl:complementOf   :Parent     ]
                        )
] .
```

```
:Parent  owl:equivalentClass  [
  rdf:type            owl:Restriction ;
  owl:onProperty      :hasChild ;
  owl:someValuesFrom  :Person
] .
```

$\text{Parent} \equiv \exists \text{hasChild}.\text{Person}$

```
:Orphan  owl:equivalentClass  [
  rdf:type            owl:Restriction ;
  owl:onProperty      [ owl:inverseOf  :hasChild ] ;
  owl:allValuesFrom   :Dead
] .
```

$\text{Orphan} \equiv \forall\, \text{hasChild}^{-}.\text{Dead}$

```
:JohnsChildren  owl:equivalentClass  [
  rdf:type          owl:Restriction ;
  owl:onProperty    :hasParent ;
  owl:hasValue      :John
] .
```

$\text{JohnsChildren} \equiv \exists \text{hasParent}.\{\text{John}\}$

```
:NarcisticPerson owl:equivalentClass  [
  rdf:type          owl:Restriction ;
  owl:onProperty    :loves ;
  owl:hasSelf       "true"^^xsd:boolean .
] .
```

$\text{NarcisticPerson} \equiv \exists \text{loves}.\text{Self}$

```
:John  rdf:type  [
  rdf:type          owl:Restriction ;
  owl:cardinality   "5"^^xsd:nonNegativeInteger ;
  owl:onProperty    :hasChild
] .
```

$=5\ hasChild.\top\ (John)$

```
:MyBirthdayGuests  owl:equivalentClass  [
  rdf:type   owl:Class ;
  owl:oneOf  ( :Bill  :John  :Mary )
] .
```

$MyBirthdayGuests \equiv \{Bill, John, Mary\}$

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

```
:Orphan  owl:equivalentClass  [
  rdf:type           owl:Restriction ;
  owl:onProperty     [ owl:complementOf  :hasChild ] ;
  owl:allValuesFrom  :Dead
] .
```

$Orphan \equiv \forall\ hasChild^{-}.Dead$

```
:hasSpouse  rdf:type  owl:SymmetricProperty .
:hasChild  rdf:type  owl:AsymmetricProperty .
:hasParent  owl:propertyDisjointWith  :hasSpouse .
:hasRelative  rdf:type  owl:ReflexiveProperty .
:parentOf  rdf:type  owl:IrreflexiveProperty .
:hasHusband  rdf:type  owl:FunctionalProperty .
:hasHusband  rdf:type  owl:InverseFunctionalProperty .
:hasAncestor  rdf:type  owl:TransitiveProperty .
```

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

```
<http://example.com/owl/families>
owl:imports <http://example.org/otherOntologies/families/> .
```

```
:Mary       owl:sameAs               otherOnt:MaryBrown .
:John       owl:sameAs               otherOnt:JohnBrown .
:Adult      owl:equivalentClass      otherOnt:Grownup .
:hasChild   owl:equivalentProperty   otherOnt:child .
:hasAge     owl:equivalentProperty   otherOnt:age .
```

Each class, property, or individual needs to be declared.

```
:John     rdf:type owl:NamedIndividual .
:Person   rdf:type owl:Class .
:hasWife  rdf:type owl:ObjectProperty .
:hasAge   rdf:type owl:DatatypeProperty .
```

Punning:

Same URI can stand e.g. for both an individual and a class:

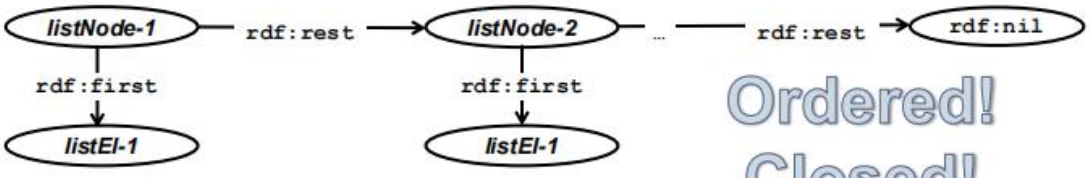
```
:John     rdf:type    :Father .
:Father   rdf:type    :SocialRole .
```

Semantics: This is semantically interpreted as if the two occurrences of `Father` were in fact distinct.

Not allowed: E.g. use of a URI for both object and datatype property.

Features of RDF: containers (aka. lists)

```
ex:summerSchool   ex:1stDaySchedule _:dayOne
_:dayOne          rdf:first         "Enrico's Keynote"
_:dayOne          rdf:rest          _:dayOneB
_:dayOneB         rdf:first         "Session 1"
_:dayOneB         rdf:rest          _:dayOneC
_:dayOneC         rdf:first         "Session 2"
_:dayOneC         rdf:rest          _:dayOneD
_:dayOneD         rdf:first         "Hands On 1"
_:dayOneD         rdf:rest          rdf:nil
```



Ordered!
Closed!

N-Triples

Full URIs in '<>'angle brackets

Terms delimited by space or tab

Triples ended by '.', delimited by new line

```
<http:/ex.org/#aidan> <http:/ns.com/worksAt> <http:/ex.org/#DERI> .
<http:/ex.org/#aidan> <http:/ns.com/author> _:bnode1 .
<http:/ex.org/#aidan> <http:/ns.com/author> _:bnode2 .
_:bnode2 <http://ns.com/title> "On Blank Nodes"@en .
_:bnode2 <http://ns.com/pages> "16"^^<http://w3.org/...#integer> .
```

Blank nodes with '_:' prefix

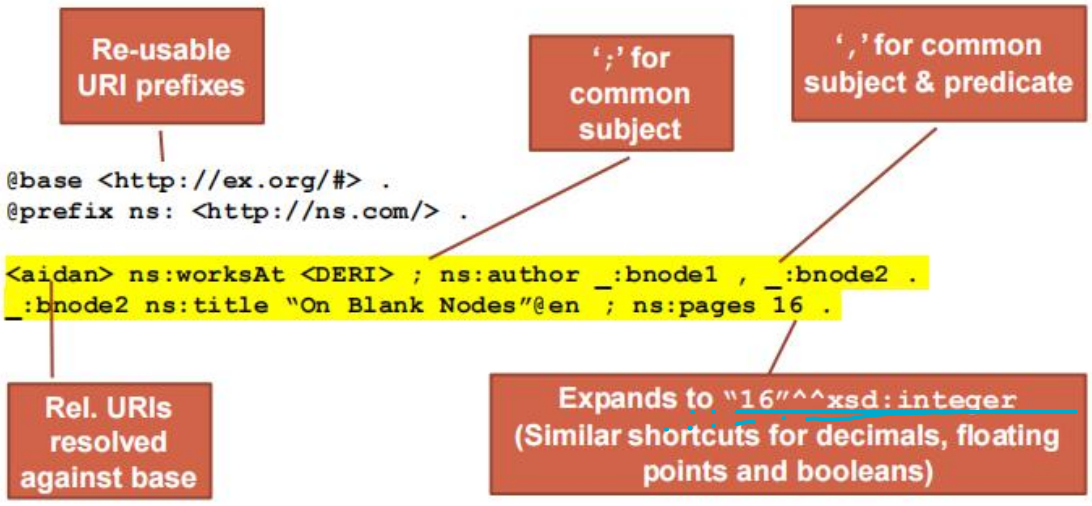
Literals enclosed with quotes

Optional datatype with '^^'

Optional lang-tag with '@'

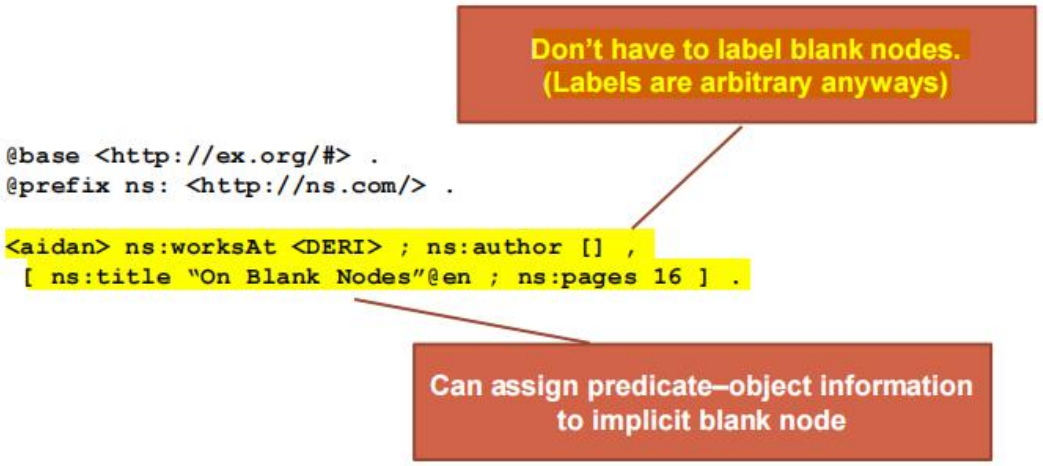
Turtle (Terse RDF Triple Language)

```
<http:/ex.org/#aidan> <http:/ns.com/worksAt> <http:/ex.org/#DERI> .
<http:/ex.org/#aidan> <http:/ns.com/author> _:bnode1 .
<http:/ex.org/#aidan> <http:/ns.com/author> _:bnode2 .
_:bnode2 <http://ns.com/title> "On Blank Nodes"@en .
_:bnode2 <http://ns.com/pages> "16"^^<http://w3.org/…#integer> .
```



Turtle (Blank node abbreviation)

```
<http:/ex.org/#aidan> <http:/ns.com/worksAt> <http:/ex.org/#DERI> .
<http:/ex.org/#aidan> <http:/ns.com/author> _:bnode1 .
<http:/ex.org/#aidan> <http:/ns.com/author> _:bnode2 .
_:bnode2 <http://ns.com/title> "On Blank Nodes"@en .
_:bnode2 <http://ns.com/pages> "16"^^<http://w3.org/…#integer> .
```



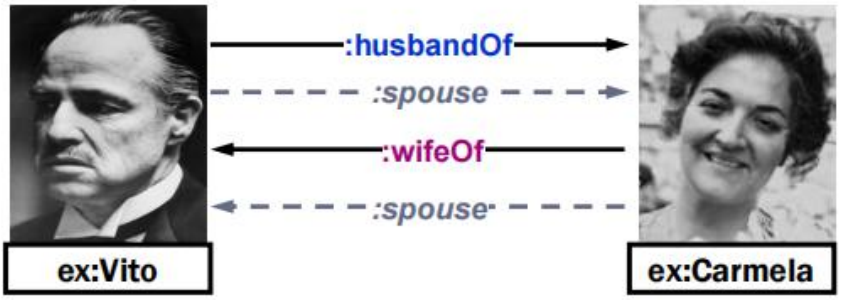
Turtle (RDF collection / list abbreviation)

```
<http://ex.org/#summerSchool> <http://ex.org/#1stDaySchedule> _:dayOne .
_:dayOne <http://w3.org/…/22-rdf-syntax#first> "Enrico's Keynote" .
_:dayOne <http://w3.org/…/22-rdf-syntax#rest> _:dayOneB .
_:dayOneB <http://w3.org/…/22-rdf-syntax#first> "Session 1" .
_:dayOneB <http://w3.org/…/22-rdf-syntax#rest> _:dayOneC .
_:dayOneC <http://w3.org/…/22-rdf-syntax#first> "Session 2" .
_:dayOneC <http://w3.org/…/22-rdf-syntax#rest> _:dayOneD .
_:dayOneD <http://w3.org/…/22-rdf-syntax#first> "Hands-on 1"
_:dayOneD <http://w3.org/…/22-rdf-syntax#first> <http://…/22-rdf-syntax#nil> .
```

```
@base <http://ex.org/#> .

<summerSchool> <1stDaySchedule>
   ( "Enrico's Keynote" "Session 1" "Session 2" "Hands-on 1" ) .
```

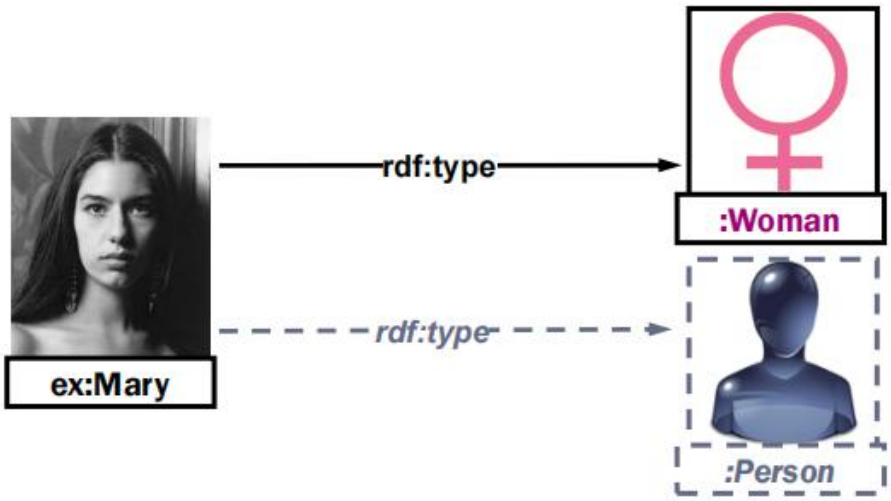
Enclose list elements in parentheses, and you're done!

rdfs:subPropertyOf



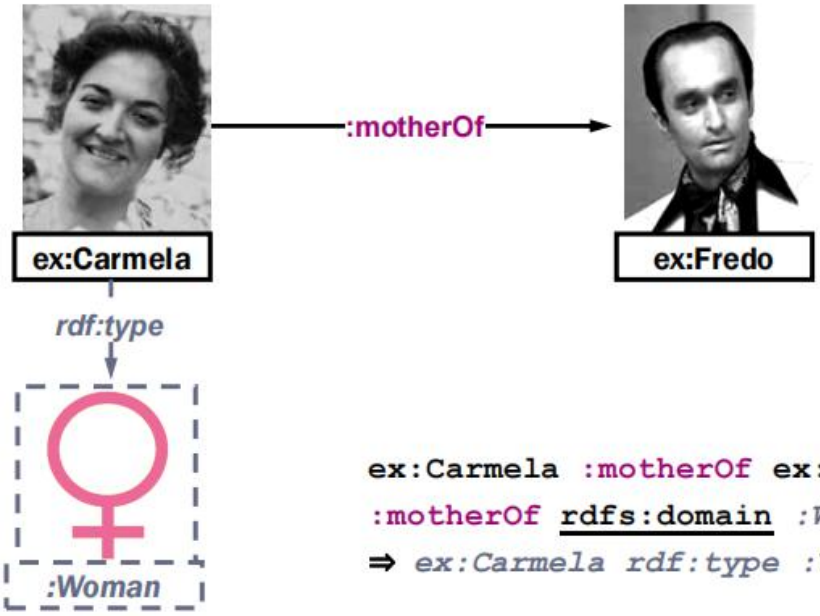

```
ex:Vito :husbandOf ex:Carmela .
:husbandOf rdfs:subPropertyOf :spouse .
⇒ ex:Vito :spouse ex:Carmela .

ex:Carmela :wifeOf ex:Vito .
:wifeOf rdfs:subPropertyOf :spouse .
⇒ ex:Carmela :spouse ex:Vito .
```

rdfs:subClassOf



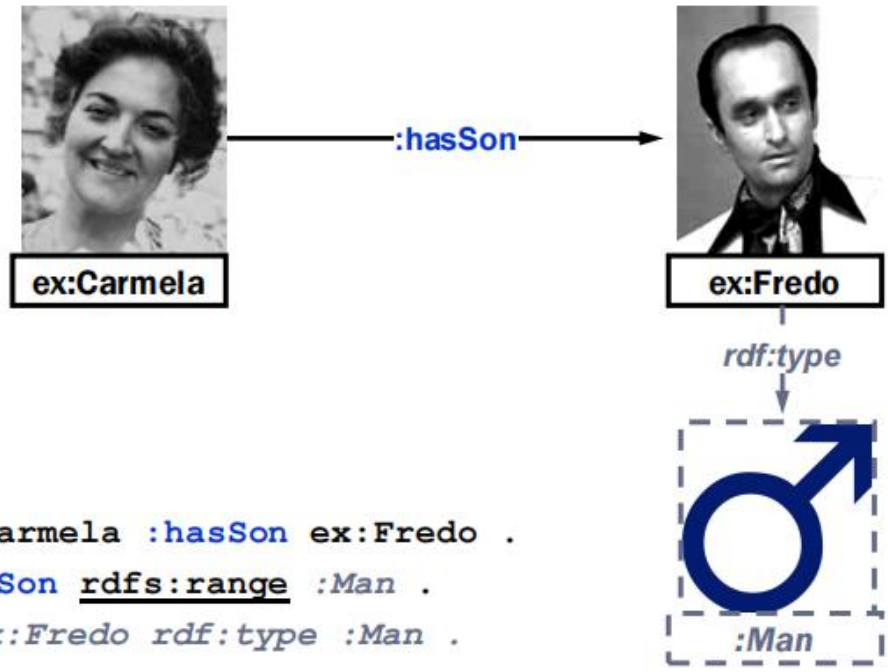

```
ex:Mary rdf:type :Woman .
:Woman rdfs:subClassOf :Person .
⇒ ex:Mary rdf:type :Person .
```

rdfs:domain




```
ex:Carmela :motherOf ex:Fredo .
:motherOf rdfs:domain :Woman .
⇒ ex:Carmela rdf:type :Woman .
```

rdfs:range




```
ex:Carmela :hasSon ex:Fredo .
:hasSon rdfs:range :Man .
⇒ ex:Fredo rdf:type :Man .
```