



Web Ontology Language (OWL)

Athens 2011

Mikel Egaña Aranguren

3205 Facultad de Informática
Universidad Politécnica de Madrid (UPM)
Campus de Montegancedo
28660 Boadilla del Monte
Spain

<http://www.oeg-upm.net>

megana@fi.upm.es
<http://mikeleganaaranguren.com>



Research and fill the wiki

<http://delicias.dia.fi.upm.es/athens/index.php/OWL>



Idea sharing



Hands-on

Today:

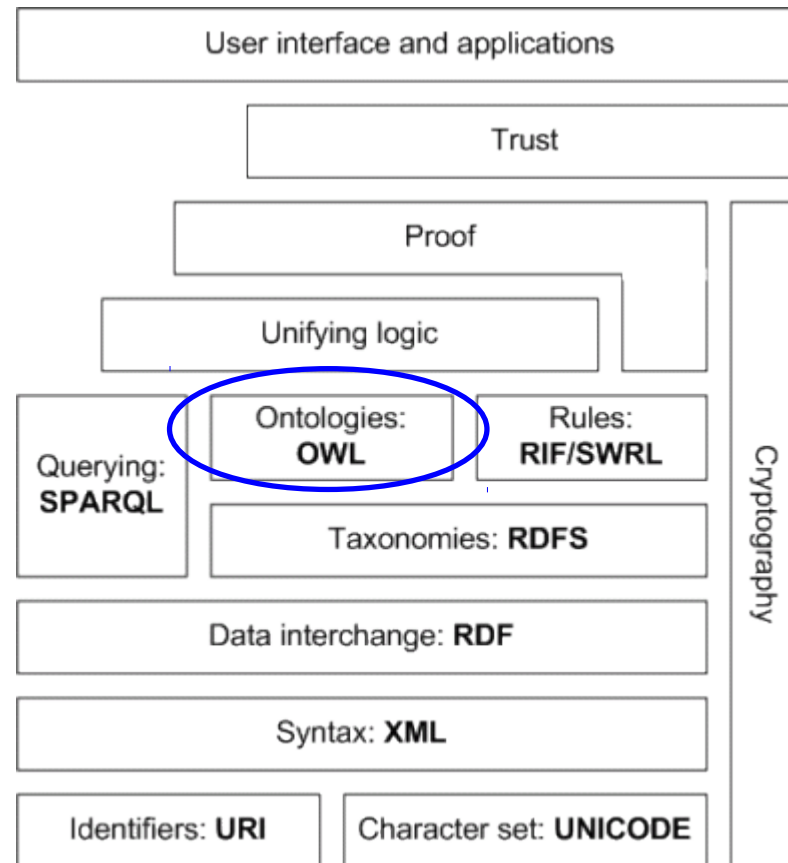
- Introduction to OWL (15 min.)
- Research OWL entities (5 min.)
- Idea sharing OWL entities (5 min.)
- Research OWL axioms (30 min.)
- Idea sharing OWL axioms (15 min.)
- Hands-on (40 min.)
- Hands-on idea-sharing (10 min.)

Tomorrow:

OWL reasoning

Introduction to OWL

OWL is a Knowledge Representation language proposed by the W3C as a standard to codify ontologies in a prospective Semantic Web



OWL is based in Description Logics

We can represent a knowledge domain computationally in an OWL ontology, in order to:

Apply automated reasoning: infer “new” knowledge, queries, consistency, classify entities against the ontology, ...

Integrate knowledge from different resources

Everything about OWL 2:

<http://www.w3.org/standards/techs/owl>

Document overview:

<http://www.w3.org/TR/2009/REC-owl2-overview-20091027/>

Primer:

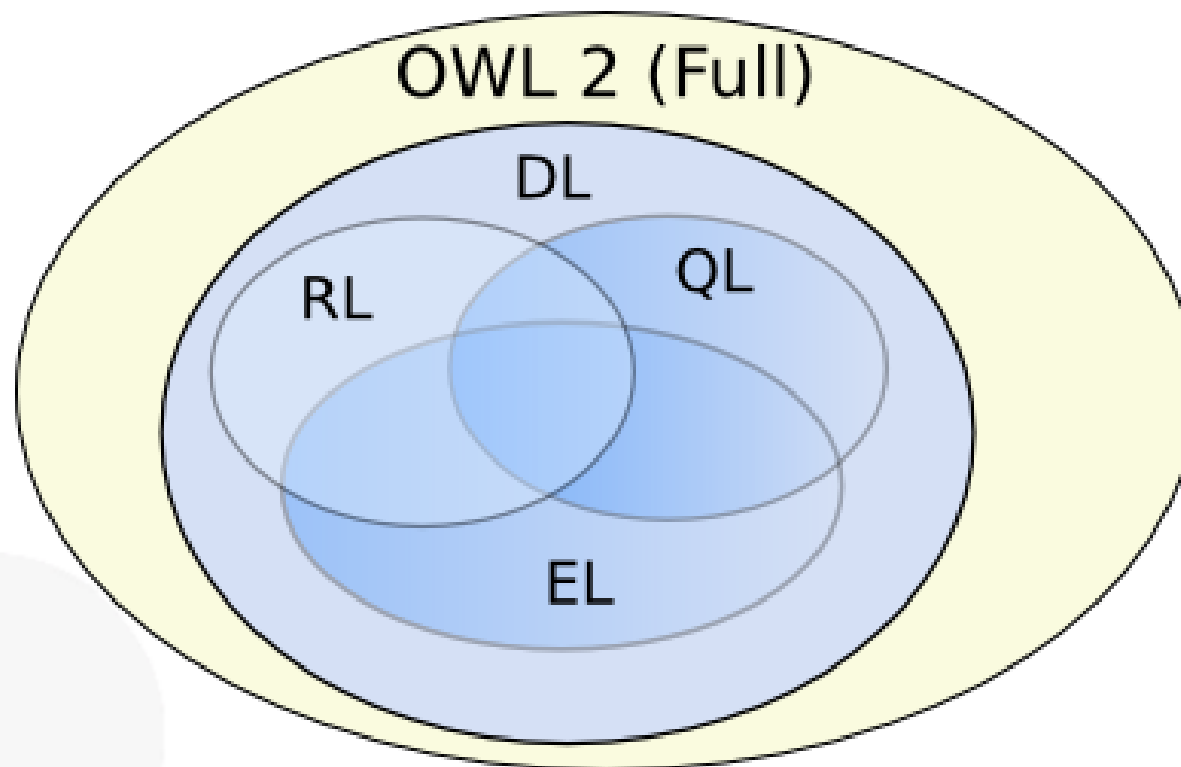
<http://www.w3.org/TR/2009/REC-owl2-primer-20091027/>

Manchester OWL + Protégé tutorial (Copied some examples :-):

<http://owl.cs.manchester.ac.uk/tutorials/protegeowltutorial/>

“OWL 1”: OWL lite, OWL DL, OWL Full

OWL 2 profiles



For computers: RDF/XML, OWL/XML, ...

RDF/XML:

```
<owl:Class rdf:about="#arm">  
  <rdfs:subClassOf>  
    <owl:Restriction>  
      <owl:onProperty rdf:resource="#part_of"/>  
      <owl:someValuesFrom rdf:resource="#body"/>  
    </owl:Restriction>  
  </rdfs:subClassOf>  
</owl:Class>
```

For humans: Manchester OWL Syntax, functional, ...

Manchester OWL Syntax: `arm` subClassOf `art_of` some `body`

http://www.co-ode.org/resources/reference/manchester_syntax/

Ontology editors:

Protégé: <http://protege.stanford.edu/>

TopBraid composer:

http://www.topquadrant.com/products/TB_Composer.html

NeOn toolkit: <http://neon-toolkit.org>

APIs:

OWL API: <http://owlapi.sourceforge.net/>

Reasoners:

Pellet: <http://clarkparsia.com/pellet/>

HermiT: <http://hermit-reasoner.com/>

FaCT++: <http://code.google.com/p/factplusplus/>

Racer: <http://www.racer-systems.com/>

OWL semantics

An OWL ontology comprises:

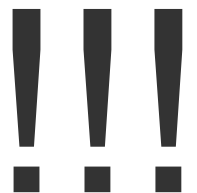
Entities: the named elements from the knowledge domain, created by the ontology creator. Entities are identified using URIs (To work in a web setting)

Axioms: axioms relate the entities to each other using the OWL logic vocabulary

An OWL ontology can import other ontologies ([owl:import](#)): the entities of the imported ontology can be referenced by axioms on our ontology

OWL is “Axiom-centric”

Entities only “exist” as part of axioms, and therefore the only way of creating an entity in an ontology is by adding an axiom that refers to it. We cannot create the class **A**, but we can state that **A subClassOf owl:Thing**



There are three types of entities in an OWL ontology:

Individuals

Properties

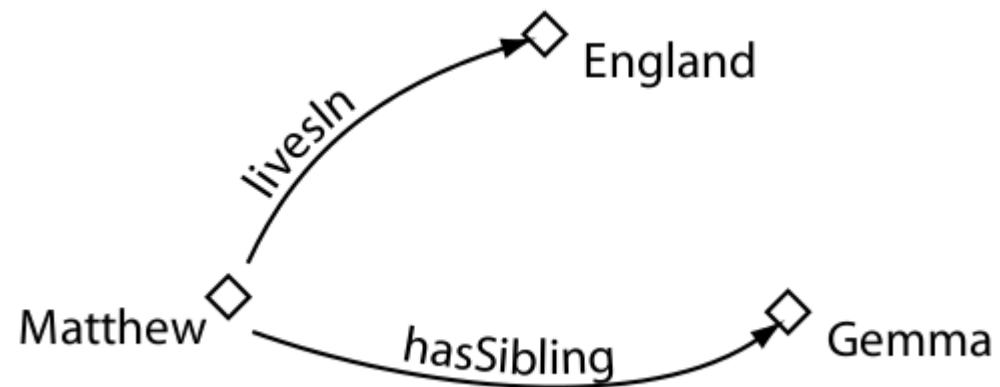
Classes

Individuals: the objects of the knowledge domain



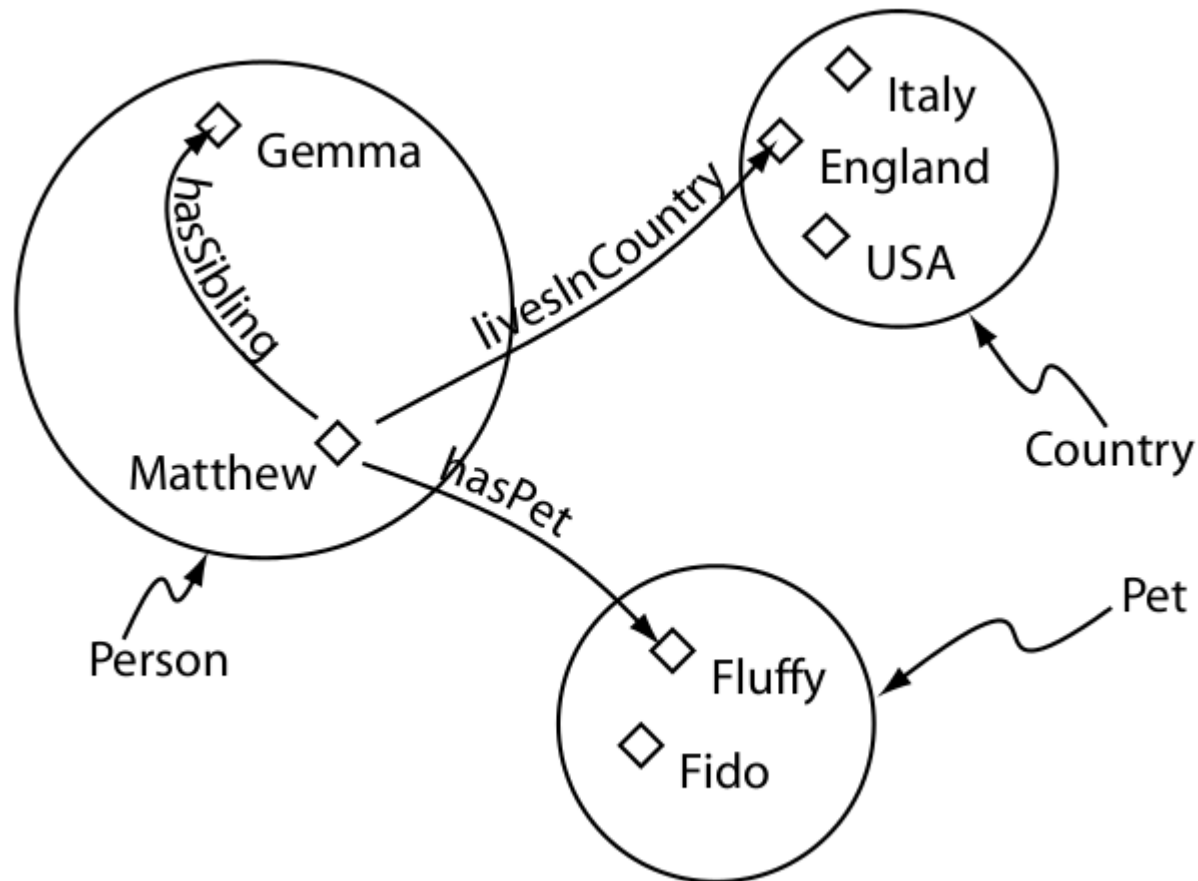
(Tutorial Manchester)

Properties: they can be used to link individuals in binary relations



(Tutorial Manchester)

Classes: sets of individuals with common characteristics

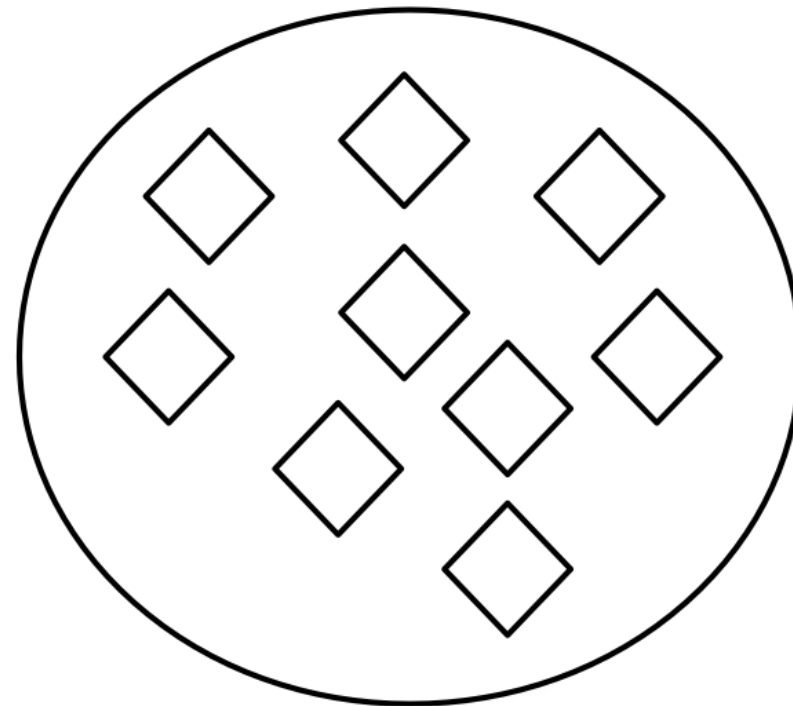


(Tutorial Manchester)

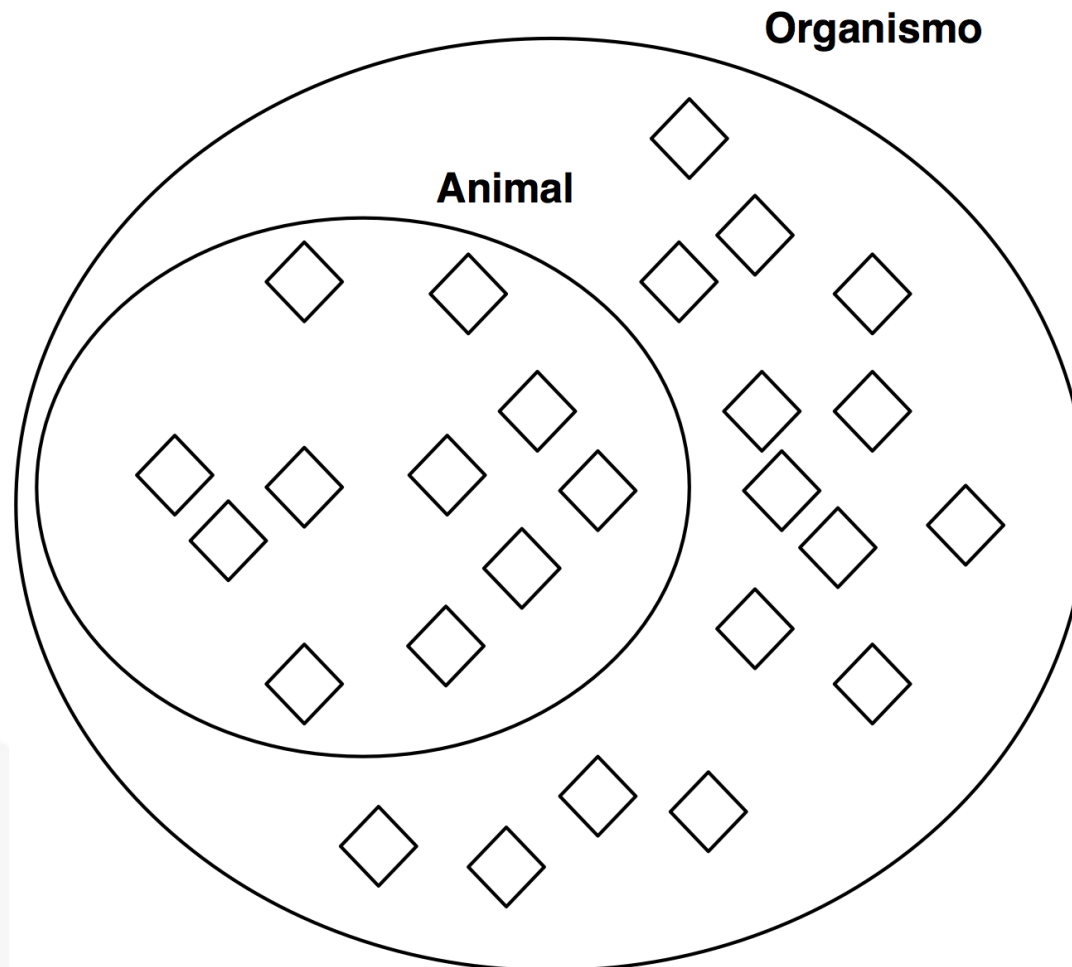
Classes

Classes: Sets of individuals

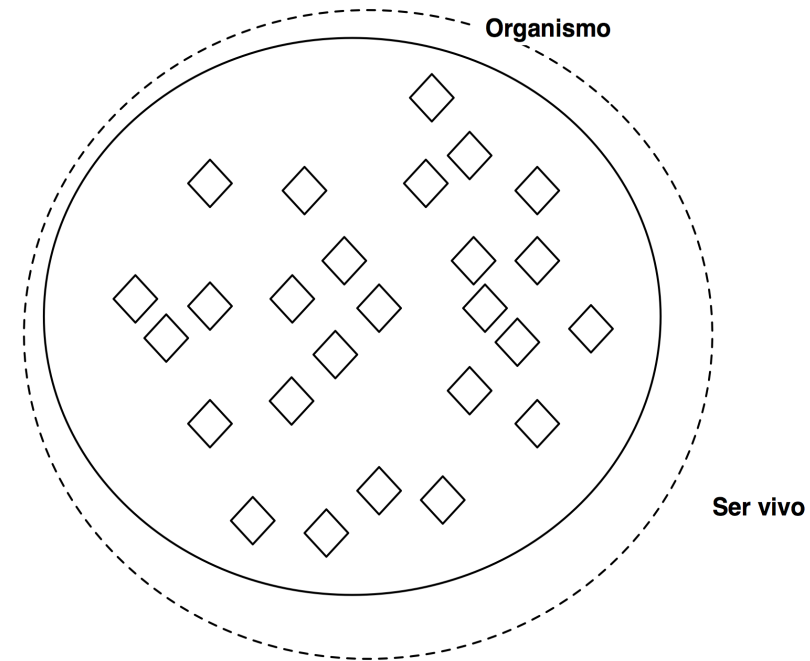
Organismo



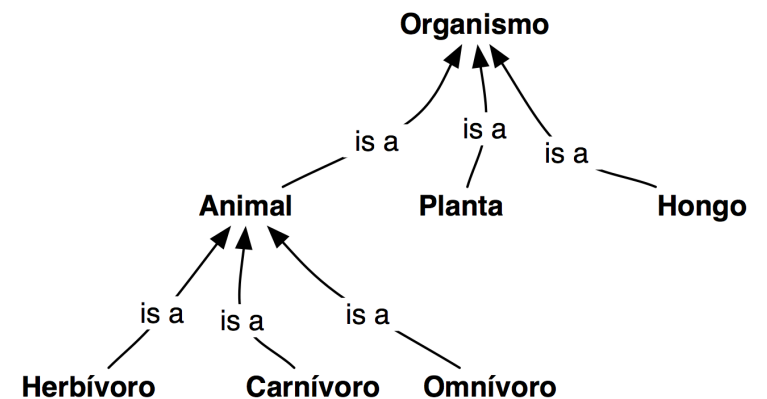
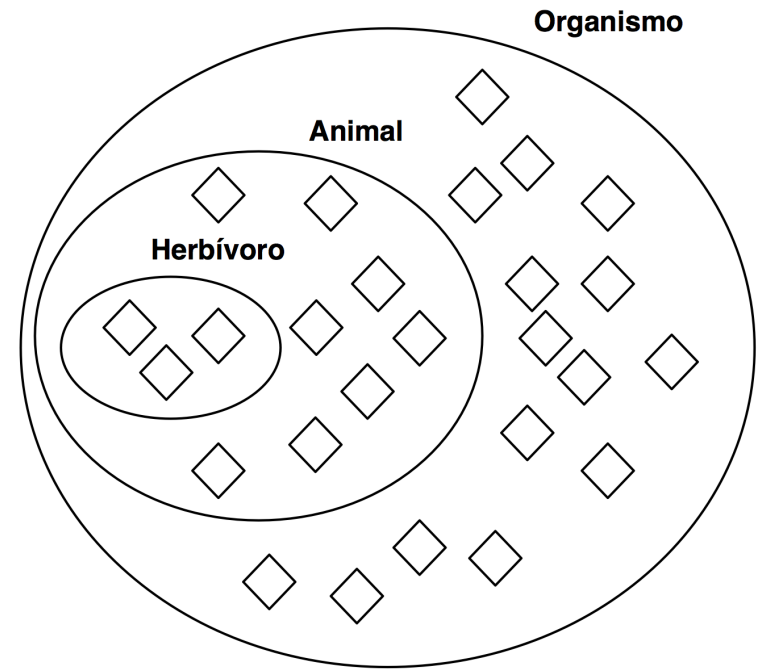
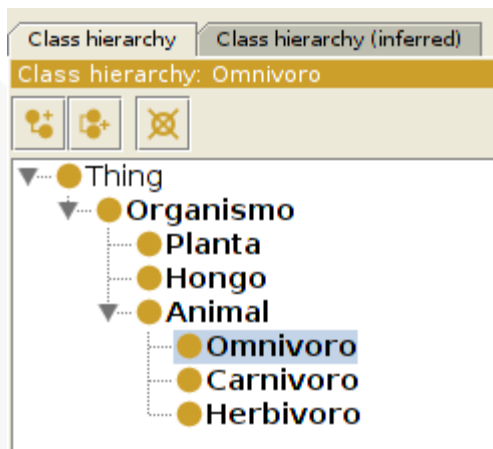
Classes can be subclasses of other classes: all the instances of the subclass are also instances of the superclass (But not the other way around)



Classes are equivalent if the extent of their sets is exactly the same: all the instances of A are also instances of B and the other way around



A taxonomy can be built combining different class-subclass axioms



In order to define the qualities that the individuals of a class must hold to be members of that class, *restrictions* on the number and type of binary relations are used

Thus, the restrictions define the conditions that must be fulfilled to be a member of a given class

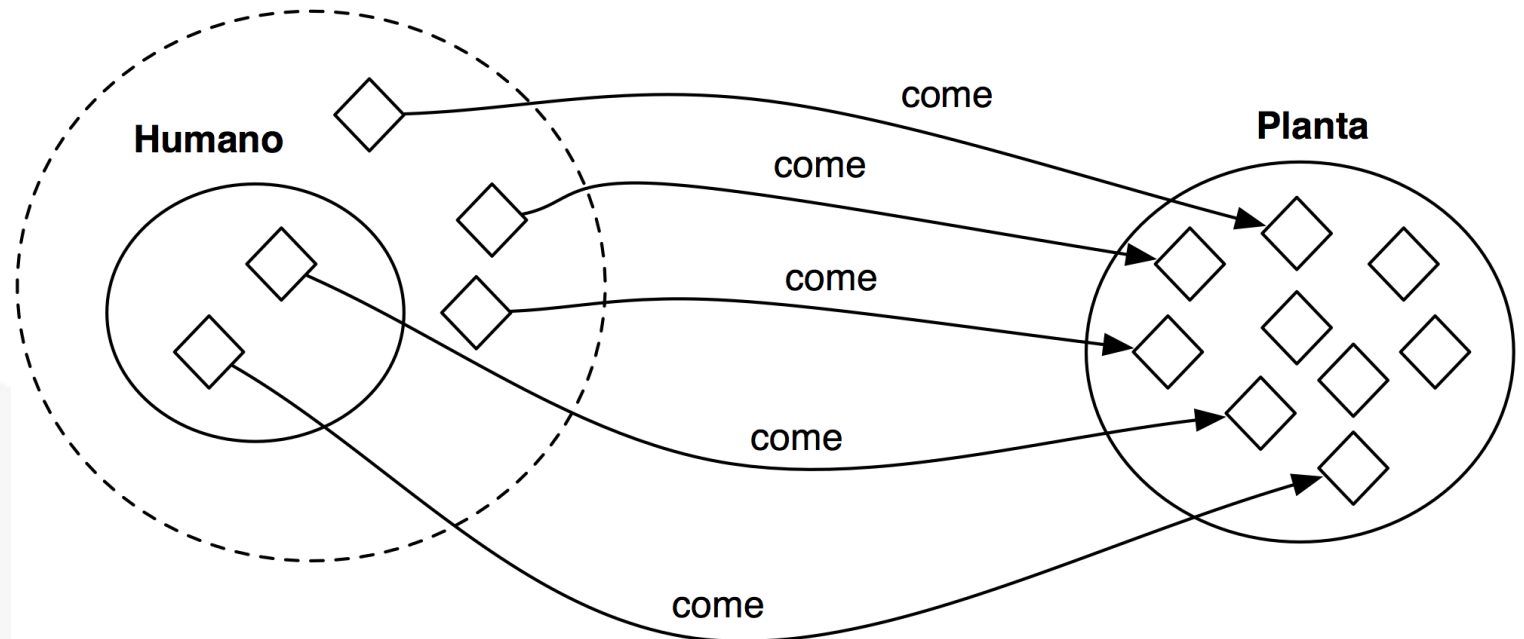
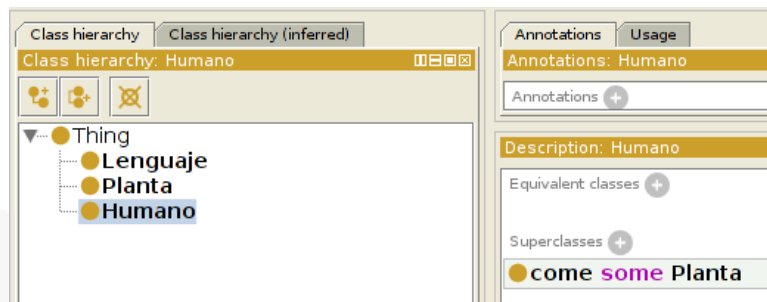
For example, we can state (In our ontology!) that in order to be human something must eat plants

Eating plants is a *necessary condition* to be human: all the humans eat plants, but there are other organisms that also eat plants that are not humans

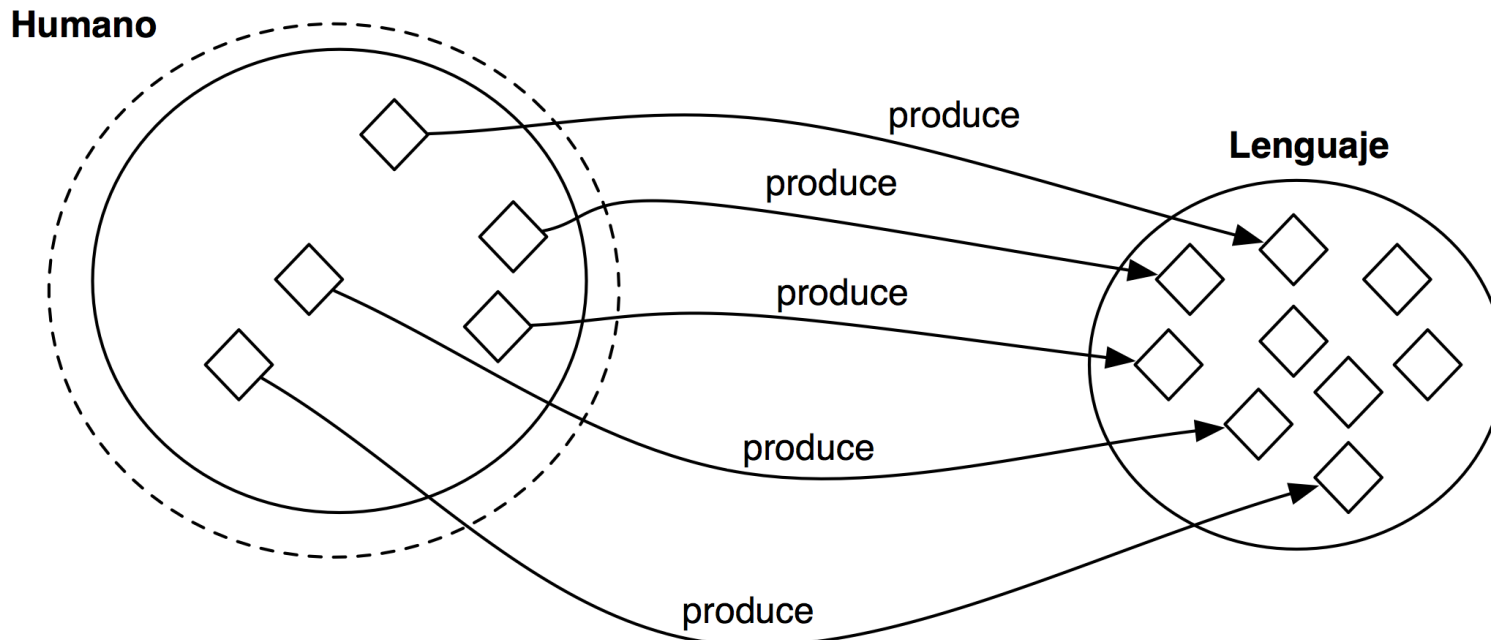
We can also define a *necessary and sufficient* condition: producing language is a unique quality of humans: if we find an individual (Organism) capable of producing language we can infer that is human, since no other organism does it

Conditions are anonymous classes: the named class we are defining with such conditions can be a subclass (Necessary) or equivalent class (Necessary and sufficient) to the anonymous class

The class **Humano** is a subclass (N) of the anonymous class comprised of the individuals that have at least one **come** binary relation with an individual of the class **Planta**



The class **Humano** is equivalent (N+S) to the anonymous class comprised of the individuals that have at least on relation with the property **produce** with and individual of the class **Lenguaje**

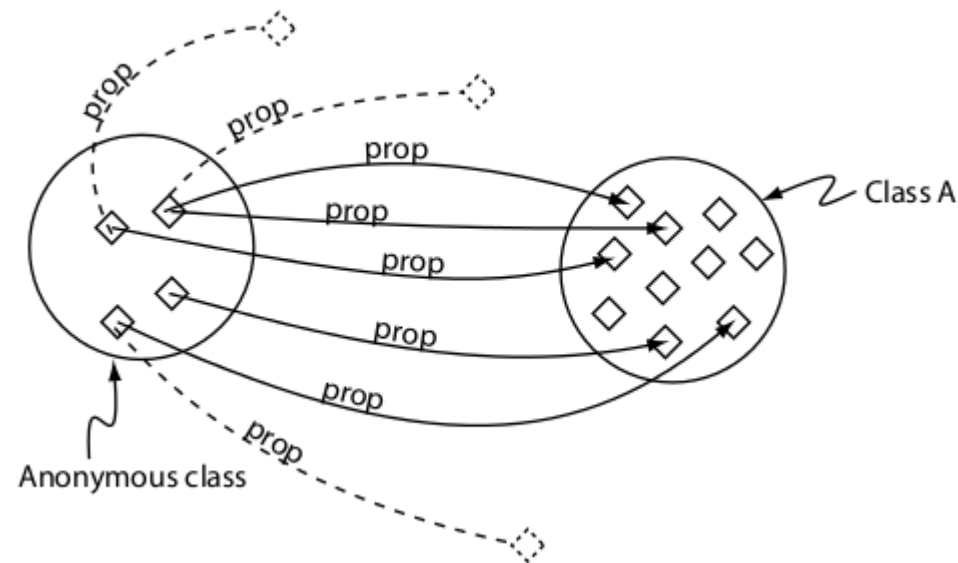


The classes with necessary and sufficient conditions are *defined* classes, and they are exploited for automated reasoning

The classes with only necessary conditions are *primitive* classes

Existential restrictions

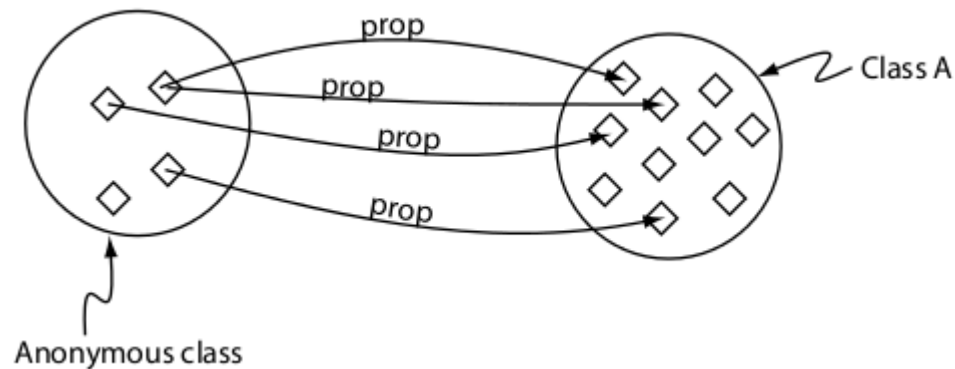
owl:someValuesFrom: the anonymous class comprised of the individuals that, amongst other things, have at least one relation to an individual of a given class with a given property: [humano subClassOf come some Planta](#)



(Tutorial Manchester)

Universal restriction

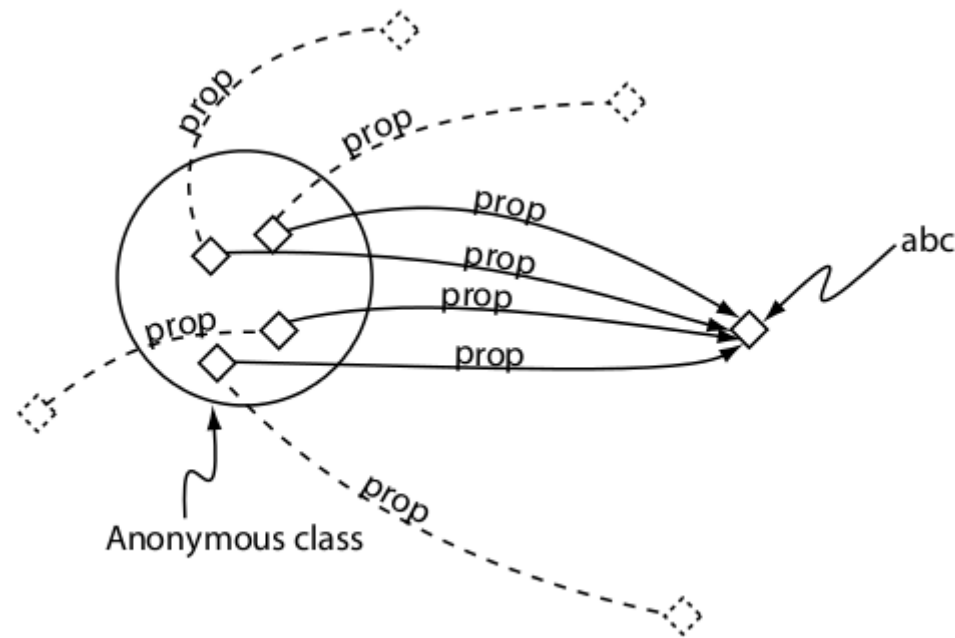
owl:allValuesFrom: the anonymous class comprised of the individuals that, if having a relation with a given property, must be to an individual of a concrete class or *none*: [humano subClassOf come only Organismo](#)



(Tutorial Manchester)

hasValue

the anonymous class comprised of the individuals that have a relation to a concrete individual `humano subClassOf come value este tomate`



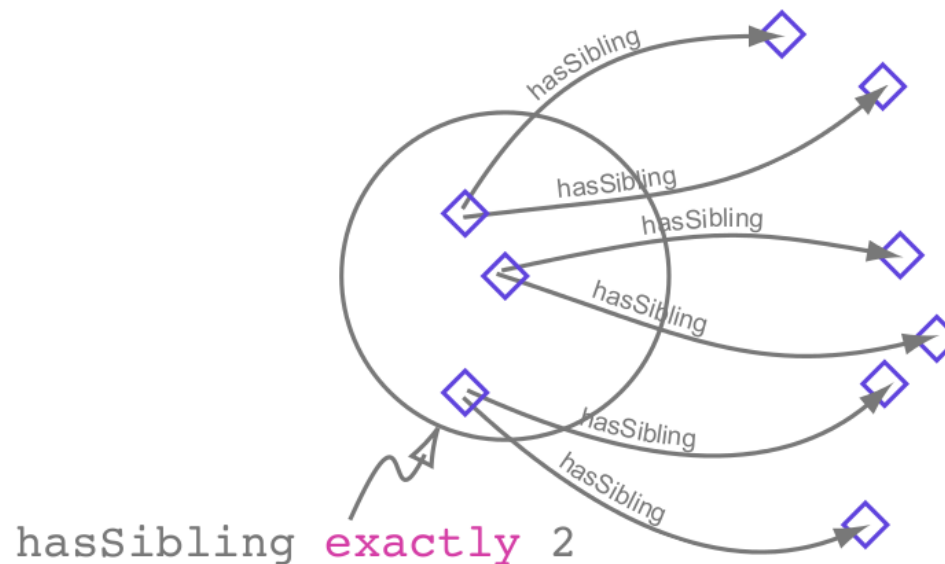
(Tutorial Manchester)

Cardinal restrictions:

Min: `humano subClassOf come min 1`

Max: `humano subClassOf come max 5`

Exactly: `humano subClassOf come exactly 3`



(Tutorial Manchester)

QCR (Qualified Cardinality Constraint):

Min: `humano subClassOf come min 1 Planta`

Max: `humano subClassOf come max 5 Planta`

Exactly: `humano subClassOf come exactly 3 Planta`

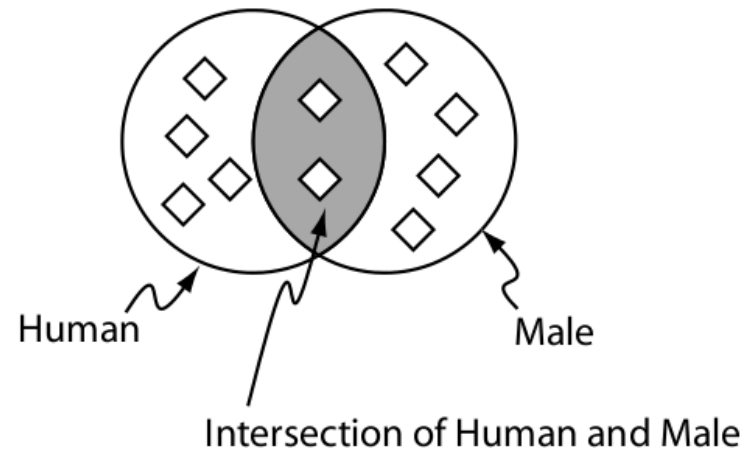
We can state that a class is different to other class (They don't have any individual in common) using disjointFrom: `humano disjointFrom planta`

We can state that two classes are the same (They have the same extent of individuals) using equivalentTo: `humano equivalentTo persona`

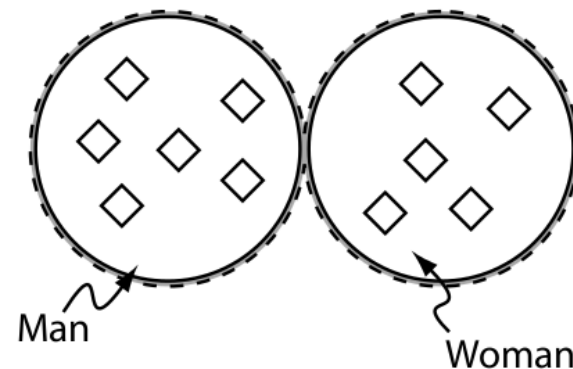
Booleans

Not: `humano subClassOf not (come some electrodomestico)`

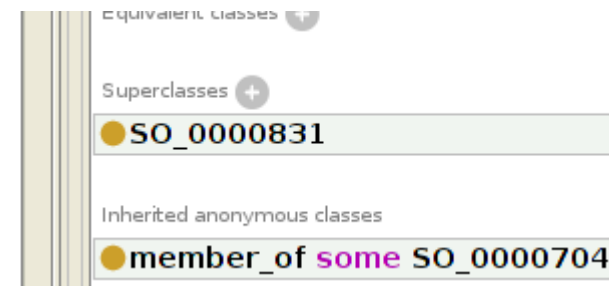
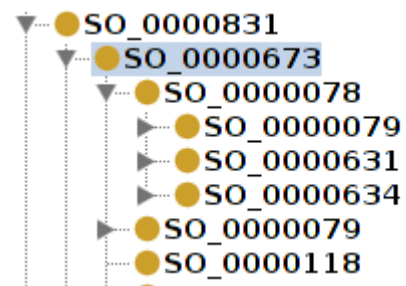
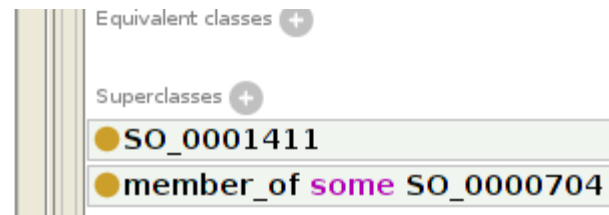
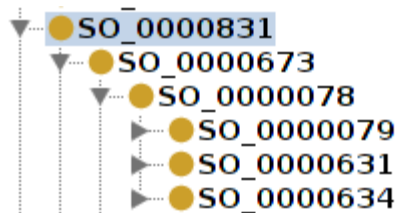
And (Intersection):
`man equivalentTo human and male`



Or (Union):
`human equivalentTo woman or man`



In a class hierarchy, the subclass “inherits” the conditions of the superclass: it can have further conditions but not a condition that conflicts with the conditions of the superclass



Conditions can be very complex, combining different OWL elements

The screenshot displays a web ontology editor interface. On the left, a class hierarchy is shown with 'Hypothesis_MYB_AP1_UP' selected. On the right, the 'Usage' tab shows the class description and its equivalent classes.

Class hierarchy: Hypothesis_MYB_AP1_UP

- Thing
 - ECO_000000
 - ECO_000037
 - ECO_0000217
 - GO_0003674
 - GO_0005575
 - GO_0008150
 - Hypothesis_MYB_AP1_UP**
 - MI_0001
 - MI_0002
 - MI_0003
 - MI_0116
 - MI_0190
 - MI_0300
 - MI_0313
 - MI_0333
 - MI_0346
 - MI_0353
 - MI_0444
 - MI_0495

Annotations: Hypothesis_MYB_AP1_UP

Description: Hypothesis_MYB_AP1_UP

Equivalent classes:

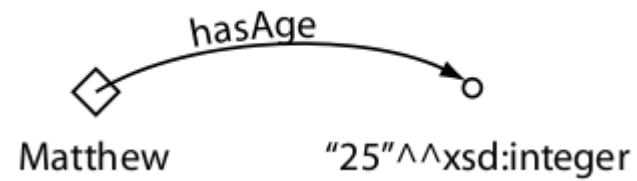
- transcription_factor exactly 1 (PRO_00009232 and (located_in_cellular_component some ((ECO_000033 and GO_0005654) or (GO_0000790 and (evidence_code some ECO_0000203))))))
- target_gene exactly 1 (PRO_000010799 and (participates_in some (MI_0931 and (detected_by some MI_0438) and (has_participant only PRO_00009232))))))
- hypothesis_entity only (PRO_00009232 or PRO_000010799)
- regulation some UP

Properties

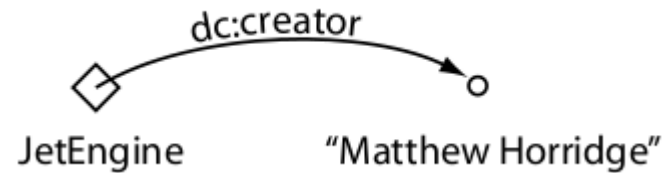
Object Properties



Data Type Properties



Annotation Properties*



Object Properties

Property hierarchy:

Sub/SuperProperties

p SubPropertyOf q

If $A p B$, $A q B$

But if $D q F$, not $D p F$

Equivalent Properties

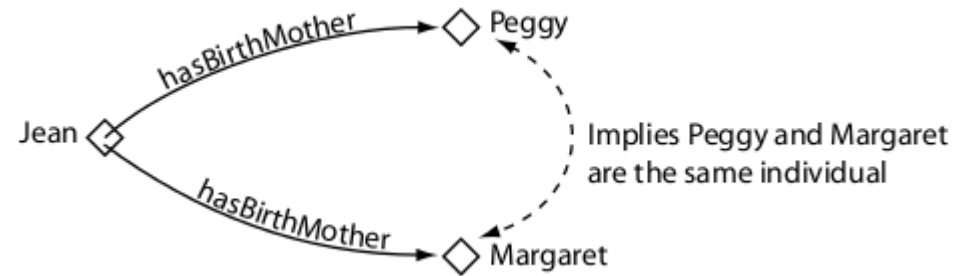
Disjoint Properties

The screenshot displays the Protégé OWL editor interface. The top menu bar includes File, Edit, View, Reasoner, Tools, Refactor, Window, and Help. The address bar shows the ontology URL: <http://www.semanticweb.org/ontologies/2011/3/Ontology1301827823935.owl>. The main workspace is divided into several panes:

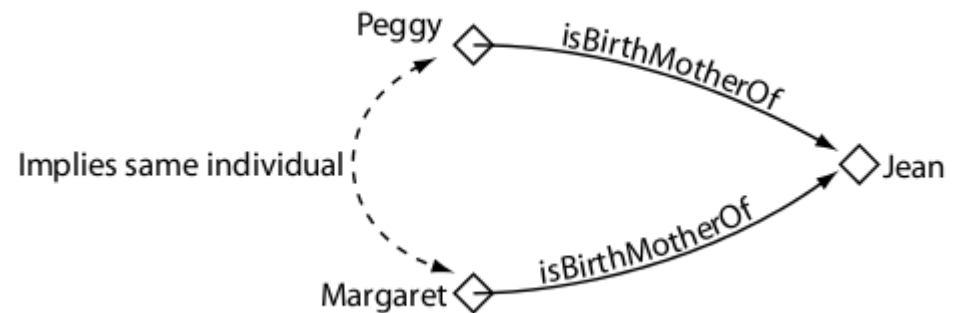
- Object property hierarchy:** A tree view showing the hierarchy of object properties. The selected property is `interacciona_con`, which is a sub-property of `parte_de`. Other visible properties include `es_localizacion_de`, `tiene_parte`, `componente_de`, `localizado_en`, `mata_a`, and `estrangula_a`.
- Annotations:** A pane for viewing and editing annotations for the selected property. It currently shows no annotations.
- Characteristics:** A list of checkboxes for property characteristics:
 - Functional
 - Inverse funcional
 - Transitive
 - Symmetric
 - Asymmetric
 - Reflexive
 - Irreflexive
- Description:** A pane for viewing and editing the description of the property. It includes expandable sections for:
 - Domains (intersection)
 - Ranges (intersection)
 - Equivalent object properties
 - Super properties
 - Inverse properties
 - Disjoint properties
 - Property chains

At the bottom right, there is a status bar with the text: "To use the reasoner click Reasoner->Start Reasoner" and a checked checkbox for "Show Inferences".

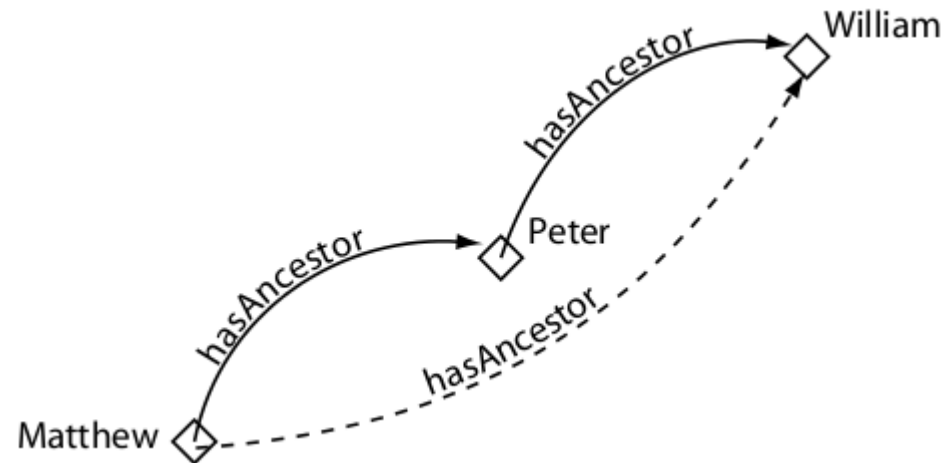
Functional



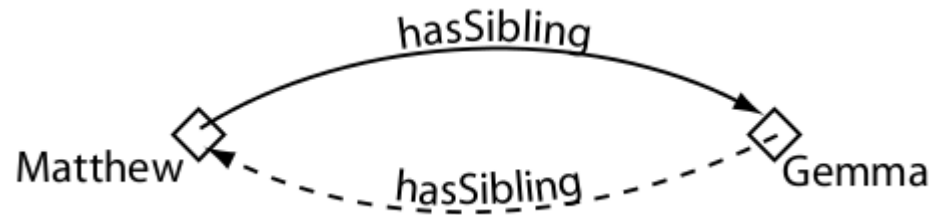
Inverse functional



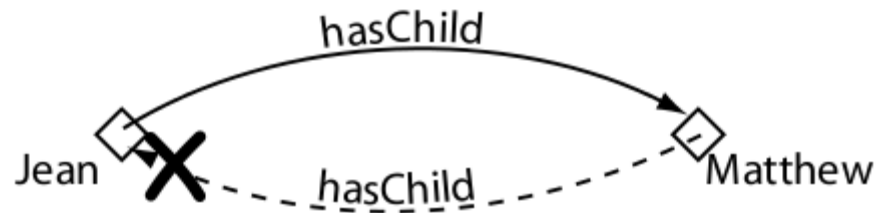
Transitive



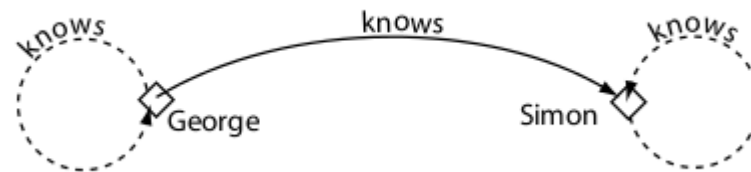
Symmetric



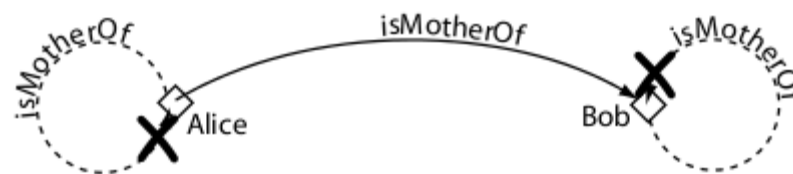
Antisymmetric*



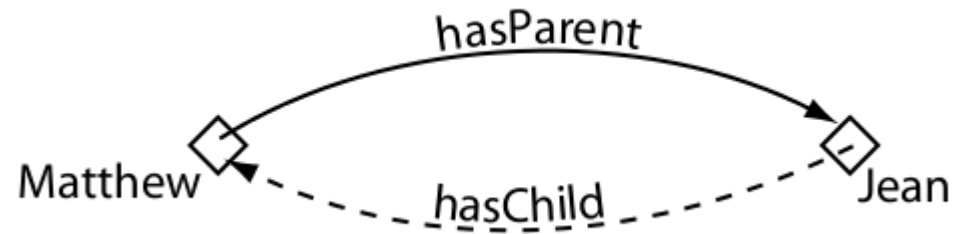
Reflexive



Irreflexive*



Inverse properties



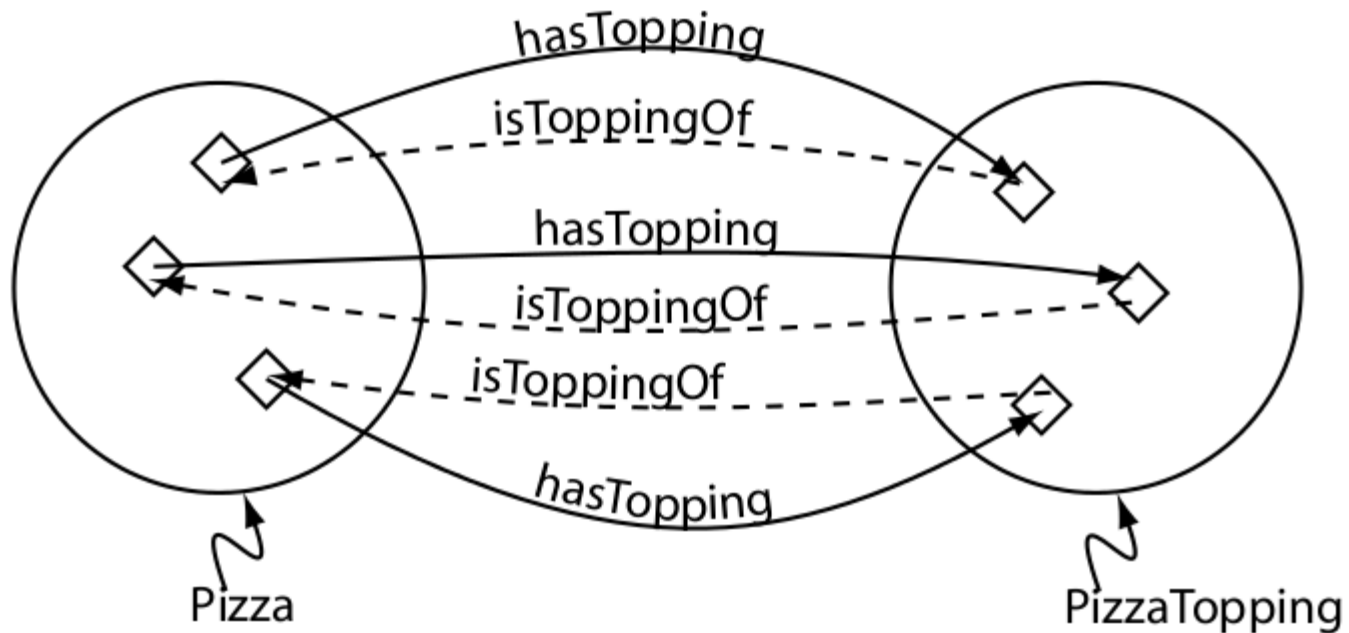
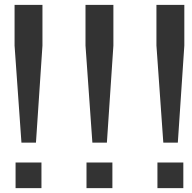
(Tutorial Manchester)

Domain and Range:

Usually classes or class unions

But any anonymous expression class can be used

They are not constraints, they are axioms



Data Type Properties

The screenshot displays an OWL editor interface with the following components:

- Menu Bar:** File, Edit, View, Reasoner, Tools, Refactor, Window, Help.
- Address Bar:** Ontology1301827823935 (<http://www.semanticweb.org/ontologies/2011/3/Ontology1301827823935.owl>)
- Navigation Tabs:** Active Ontology, Entities, Classes, Object Properties, Data Properties, Individuals, OWLViz, DL Query, OntoGraf.
- Data property hierarchy: topDataPrope...**
 - topDataProperty
 - cantidad_de_soldados
 - distancia
 - distancia_de_ataque
 - alcance_de_misil
 - alcance_de_cañon

- Annotations: topDataPrope...**
- Annotations +
- Characteristics: topDat**
- Functional
- Description: topDataPrope...**
- Domains (intersection) +
- Ranges +
- Equivalent properties +
- Super properties +
- Disjoint properties +

To use the reasoner click Reasoner->Start Reasoner Show Inferences

Equivalent / sub-super / disjoint

Only Functional (No transitive, inverse functional, ...)

Domain: ~ Object Properties

Range:

Built-in datatypes

Data range expression

Annotation Properties

Add non-semantic annotations in natural language to entities, axioms or the ontology

rdfs:label, rdfs:comment, ...

Dublin Core (<http://dublincore.org/>)

Custom annotation properties

Language (en, es, ...) and type (xsd:string, ...)

The screenshot displays an ontology editor interface. On the left, a 'Class hierarchy' panel shows a tree structure starting with 'Thing', which includes 'Coche', 'CocheAudi', 'Componente', and 'fabricante'. The 'Coche' class is selected. On the right, an 'Annotations' panel for 'Coche' shows two annotations: a 'label' with the value '"Car"@en' and a custom annotation property 'notas_compra' with the value '"El dependiente era muy simpatico"@es'.

Individuals

An individual can be a member of one or more anonymous or named classes (**Types**)

An individual can be the same as other individual (**SameAs**)

An individual can be different from another individual (**DifferentFrom**)

Individuals can be related in binary relations (Object Properties):

```
my_wheel part_of my_car  
my_wheel not part_of your_car
```

Individuals can be related with data (Data Type properties):

```
my_car has_power "90"^^xsd:positiveInteger  
my_car not has_power "90"^^xsd:positiveInteger
```

The screenshot displays the OWL editor interface for the class **Mi_retrovisor**. The interface is divided into several panels:

- Members list:** Shows the class **Mi_retrovisor** with a plus icon and a minus icon.
- Annotations:** Shows the class **Mi_retrovisor** with an **Annotations** section containing a plus icon.
- Description:** Shows the class **Mi_retrovisor** with a **Types** section containing **Recambio** and **Retrovisor**, and **Same individuals** containing **Retrovisor_de_mi_novia**, and **Different individuals** containing **retrovisor_de_pedro**.
- Property assertions:** Shows the class **Mi_retrovisor** with **Object property assertions** containing **parte_de mi_coche** and **parte_de coche_de_pedro**, **Data property assertions** containing **tiene_precio 20**, and **Negative data property assertions** containing **tiene_precio 30**.

Some extra constructs

OWL oneOf

The screenshot displays an ontology editor interface. On the left, a class hierarchy is shown under the 'Class hierarchy (inferred)' tab. The hierarchy is: Thing (parent) -> Familia (child) -> FamiliaMikel (child of Familia). FamiliaMikel is highlighted. On the right, the 'Annotations' and 'Usage' tabs are active. The 'Annotations' section shows 'Annotations: FamiliaMikel' and a '+' button. The 'Description' section shows 'Description: FamiliaMikel'. The 'Equivalent classes' section shows a set: {Iker, JuanRamon, Mikel}. The 'Superclasses' section shows 'Familia'. The 'Members' section lists: Iker, JuanRamon, Mikel, and Pedro.

Class hierarchy (inferred)	Annotations	Usage
Class hierarchy: FamiliaMikel	Annotations: FamiliaMikel	
Thing	Annotations +	
Familia		
FamiliaMikel		
	Description: FamiliaMikel	
	Equivalent classes +	
	{Iker, JuanRamon, Mikel}	
	Superclasses +	
	Familia	
	Inherited anonymous classes	
	Members +	
	Iker	
	JuanRamon	
	Mikel	
	Pedro	

Role chains

The screenshot displays the Protege OWL editor interface. At the top, there are tabs for 'Active Ontology', 'Entities', 'Classes', 'Object Properties', 'Data Properties', 'Individuals', 'OWLviz', 'DL Query', and 'OntoGraf'. The 'Object Properties' tab is active.

On the left, the 'Object property hierarchy' shows a tree structure under 'topObjectProperty' with three sub-properties: 'tiene_padre', 'tiene_primo', and 'tiene_sobrino'. The 'tiene_primo' property is selected.

The main area is divided into two columns. The left column has tabs for 'Annotations' and 'Usage'. The 'Annotations' tab is active, showing 'Annotations: tiene_primo' and a '+' icon to add more annotations.

The right column has tabs for 'Characteristi' and 'Description'. The 'Description' tab is active, showing 'Description: tiene_primo'. Below this, there are several expandable sections: 'Domains (intersection)', 'Ranges (intersection)', 'Equivalent object properties', 'Super properties', 'Inverse properties', 'Disjoint properties', and 'Property chains'. Each section has a '+' icon to expand it.

At the bottom of the 'Description' tab, a list of property relationships is shown. The selected relationship is highlighted in blue: 'tiene_padre o tiene_sobrino SubPropertyOf tiene_primo'.

OWL Self

The screenshot displays an OWL editor interface with two main panels. The left panel, titled 'Class hierarchy', shows a tree structure under the 'Familia' class. It includes 'Thing' as a superclass, 'Familia' as the current class, and 'FamiliaMikel' as a subclass. The right panel, titled 'Annotations' and 'Usage', shows the 'Familia' class selected. It displays 'Annotations: Familia' and 'Description: Familia'. Under the 'Superclasses' section, there is a self-referencing property: 'hace_negocios_con some Self'.

OWL keys

<http://www.w3.org/TR/2009/REC-owl2-primer-20091027/#Keys>

~ “datatype inverse functional”

The screenshot shows a web ontology editor interface. On the left, a 'Class hierarchy' panel displays a tree structure with 'Thing' as the root and 'Persona' as a child class. On the right, a detailed view for the 'Persona' class is shown, including sections for 'Annotations', 'Description', 'Members', and 'Keys'. The 'Members' section lists three individuals: Maria, Mikel, and Oscar. The 'Keys' section lists a single property: 'numero_seguridad_social'.

numero_seguridad_social "7"^^xsd:integer
 numero_seguridad_social "8"^^xsd:integer
 numero_seguridad_social "7"^^xsd:integer