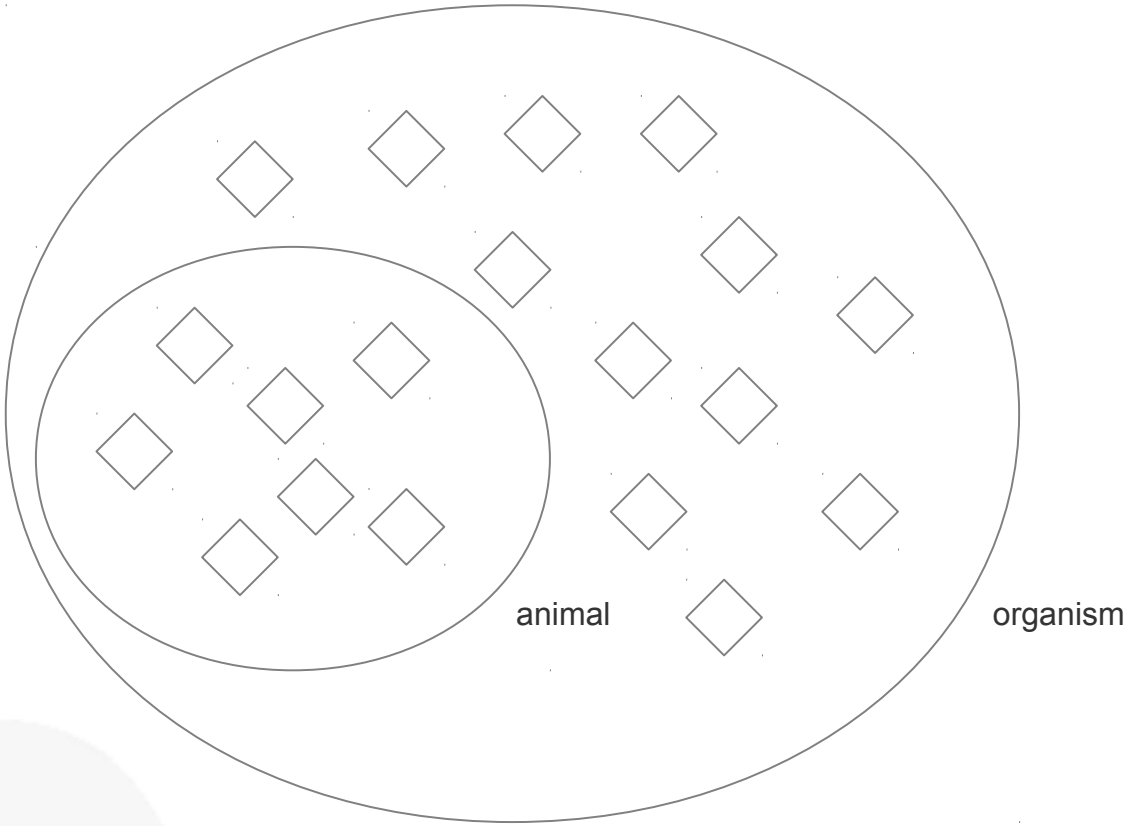
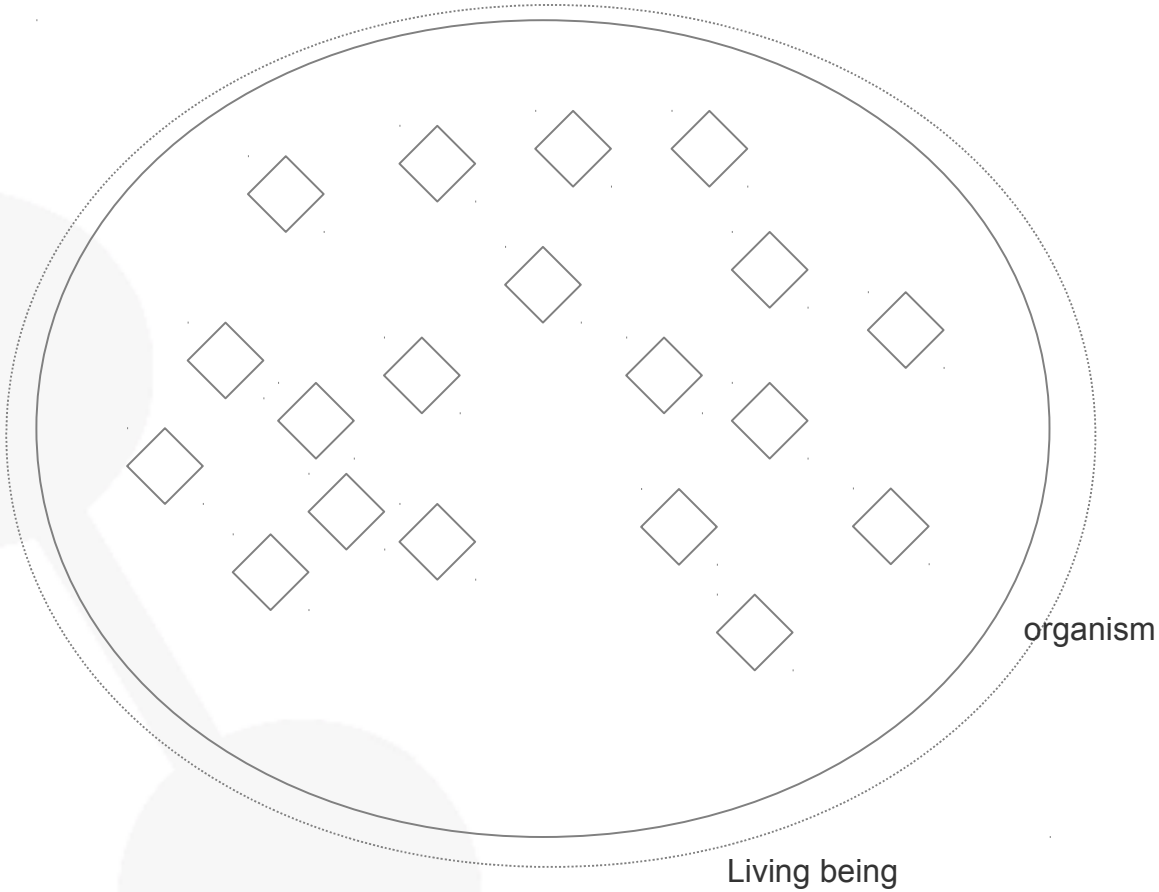
Classes

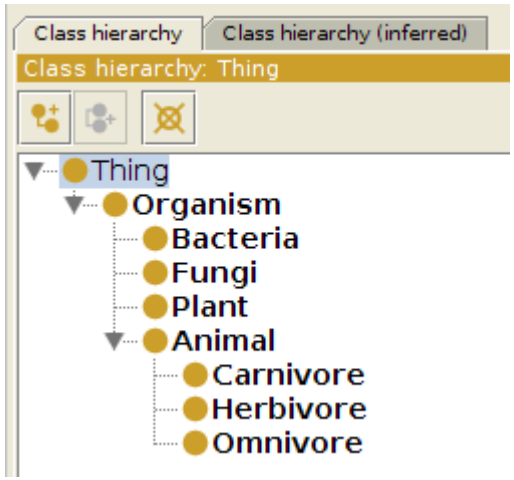
Classes can be subclasses of other classes (subClassOf): all the instances of the subclass are also instances of the superclass (But no the other way around)

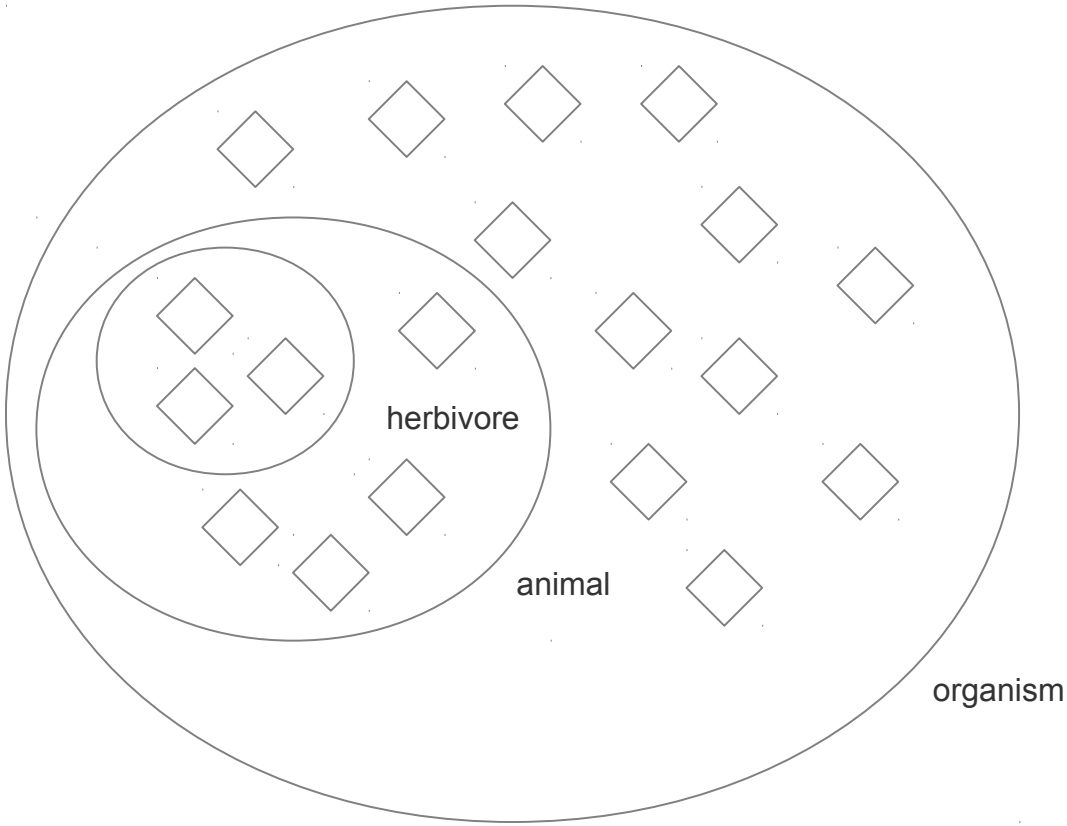


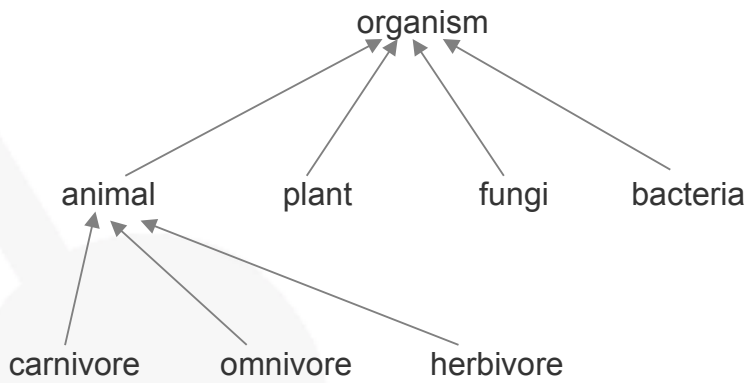
Classes are equivalent if the extent of their sets is exactly the same (equivalentTo): all the instances of A are also instances of B and the other way around



A hierachy can be built combining different subClassOf 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

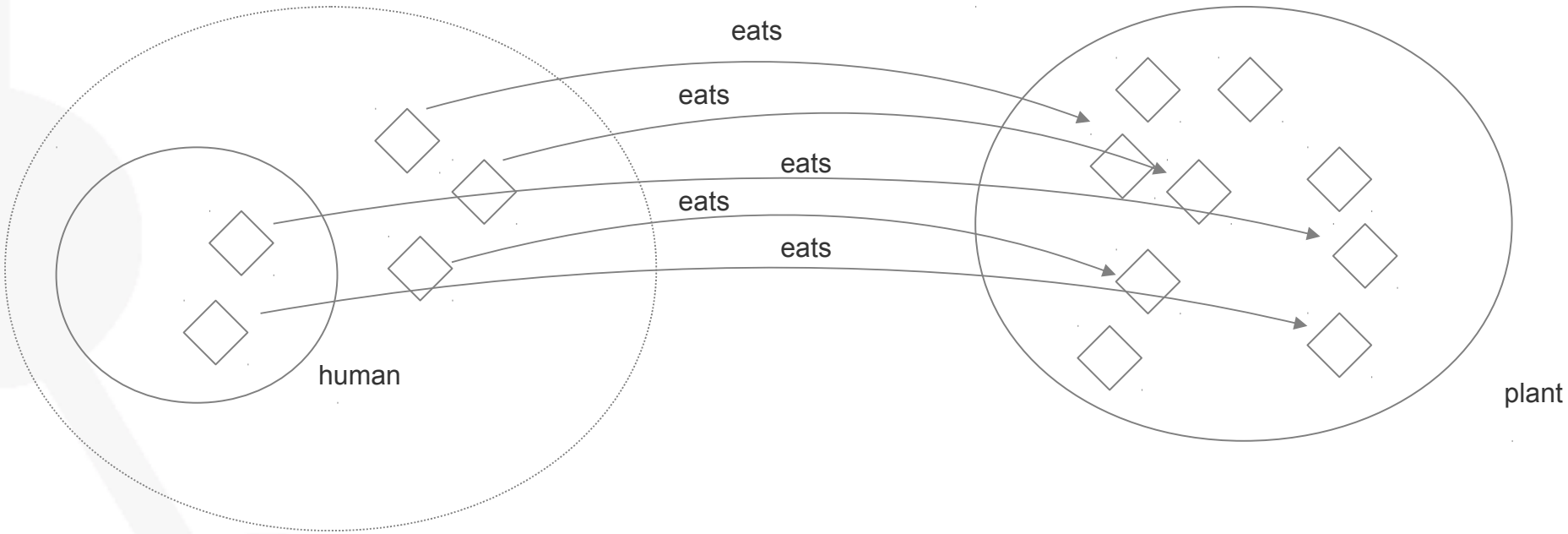
For example, we can state (In our ontology!) that in order to be human something must eat plants

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

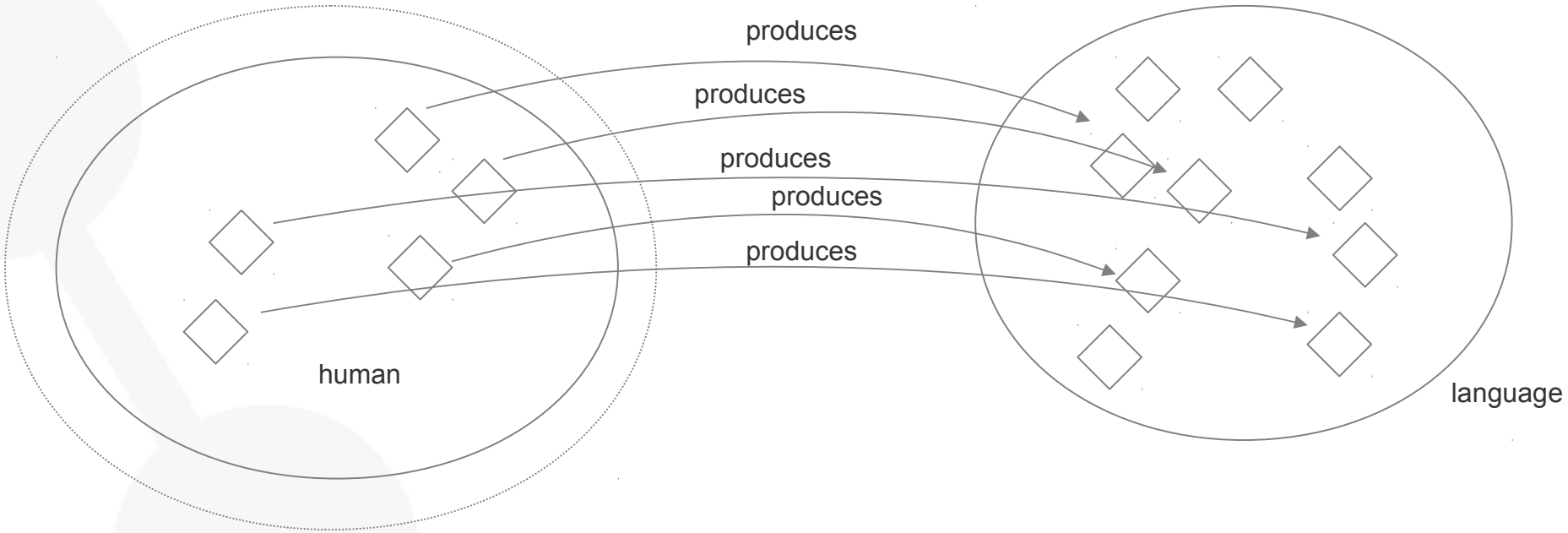
We can also define a *necessary and sufficient* condition: producing language is a unique quality of humans: if we find an individual (Organism) capable of producing language we can infer that is human, since no other organism does it

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 human is a subclass (N) of the anonymous class comprised of the individuals that have at least one eats binary relation with an individual of the class plant

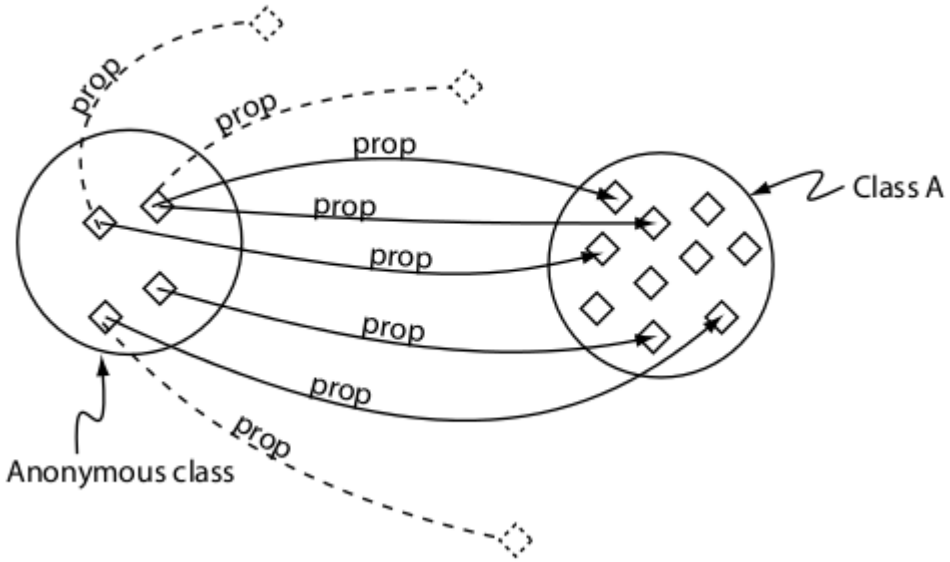


The class human is equivalent (N+S) to the anonymous class comprised of the individuals that have at least on relation with the property produces with and individual of the class language



Existential restriction

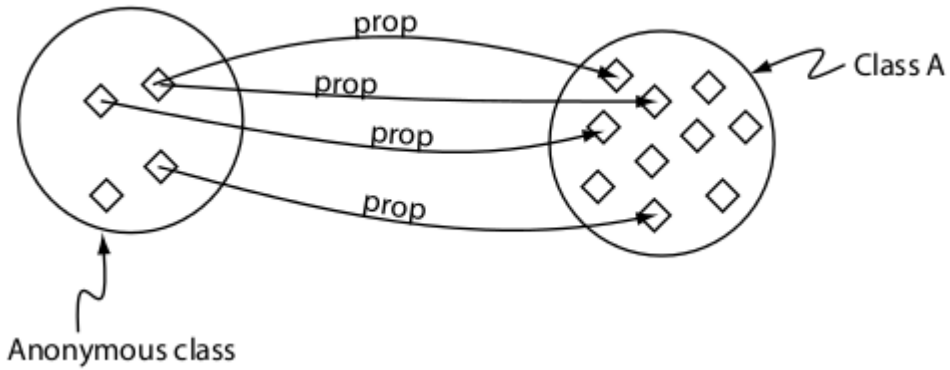
owl:someValuesFrom: the anonymous class comprised of the individuals that, ammongst other things, have at least one relation to an individual of a given class with a given property: human subClassOf eats some plant



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

Universal restriction

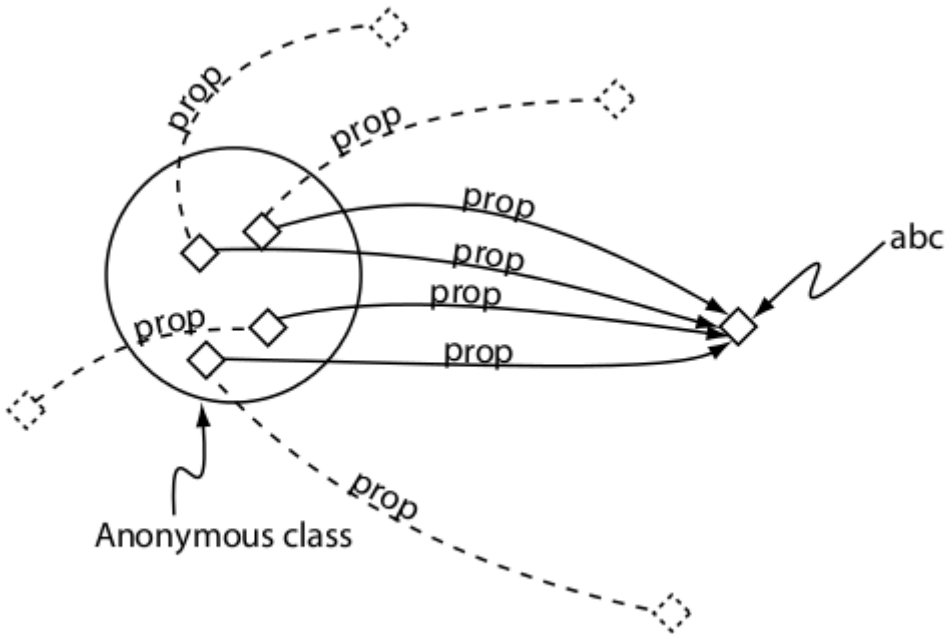
owl:allValuesFrom: the anonymous class comprised of the individuals that, if having a relation with a given property, must be to an individual of a concrete class or *none*: human subClassOf eats only organism



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

hasValue

the anonymous class comprised of the individuals that have a relation to a concrete individual human subClassOf eats value pig-peggy



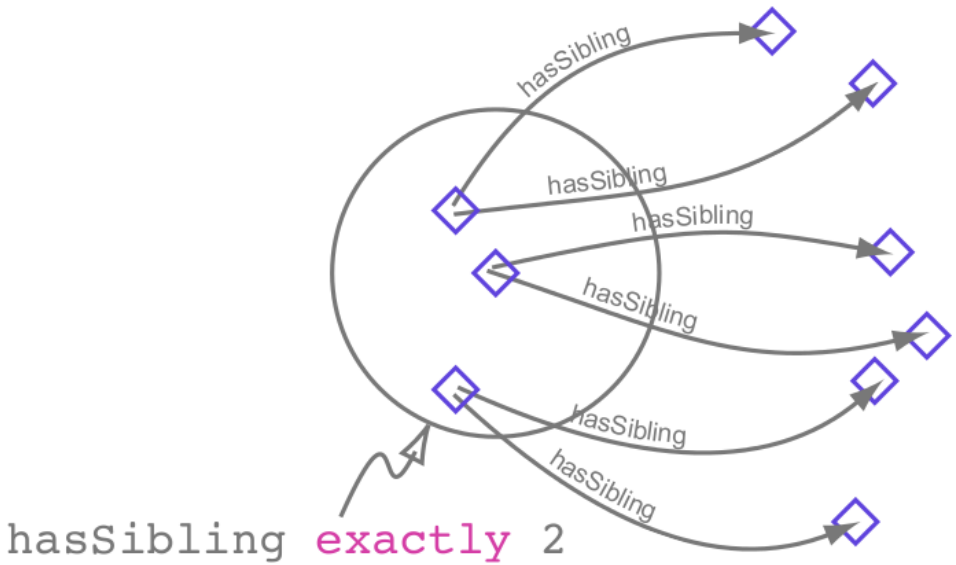
http://owl.cs.manchester.ac.uk/tutorials/protegeowltutorial/

Cardinal restrictions:

Min: human subClassOf eats min 1

Max: human subClassOf eats max 5

Exactly: human subClassOf eats exactly 3



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

QCR (Qualified Cardinality Constraint):

Min: human subClassOf eats min 1 plant

Max: human subClassOf eats max 5 plant

Exactly: human subClassOf eats exactly 3 plant

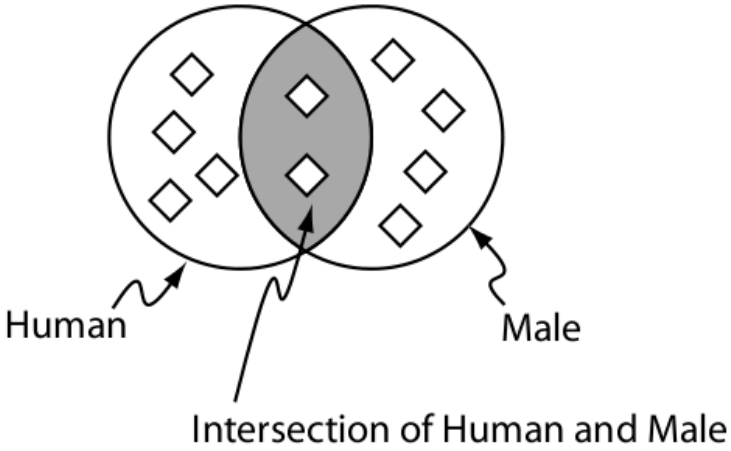
We can state that a class is different to other class (They don't have any individual in common) using disjointFrom: human disjointFrom plant

We can state that two classes are the same (They have the same extent of individuals) using equivalentTo: human equivalentTo person

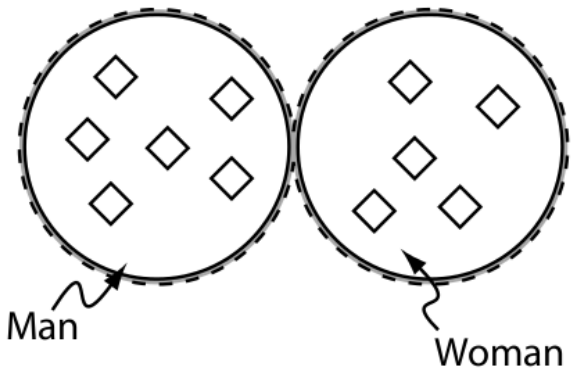
Booleans

Not: human subClassOf not (eats some stone)

And (Intersection):
man equivalentTo human and male

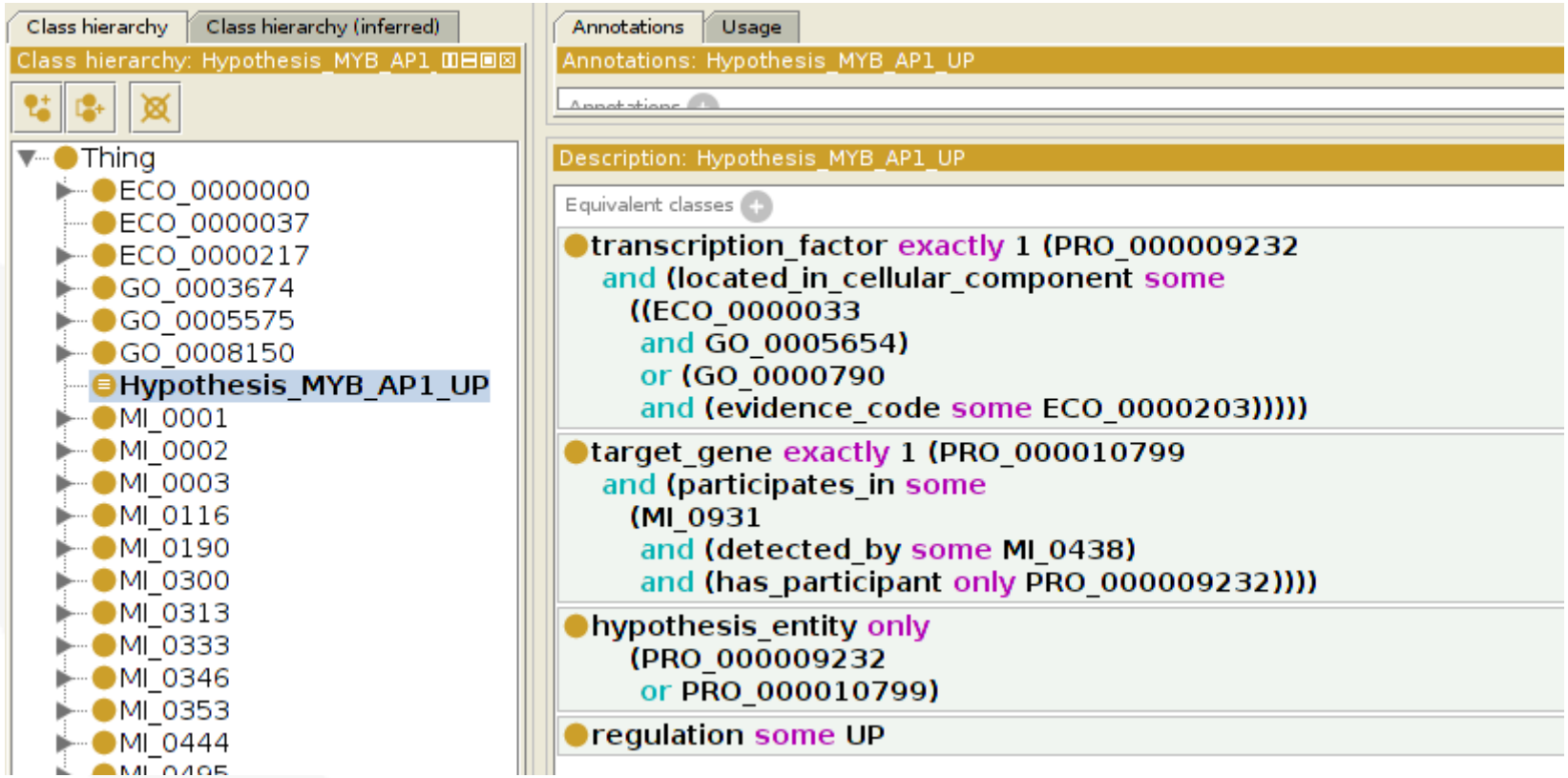


Or (Union):
human equivalentTo woman or man



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

Conditions can be very complex, combining different OWL elements

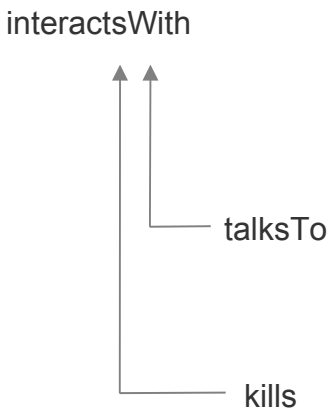


Object Properties

Property hierarchy:

Sub/SuperProperties

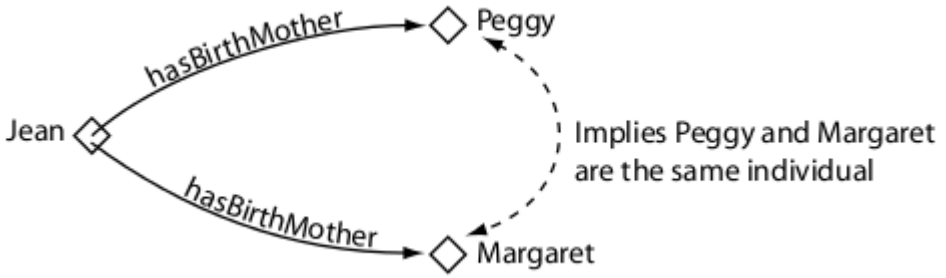
If Tom talks to Martin, he interacts with him
If Tom interacts with Michael, he doesn't talk to him (he might!)



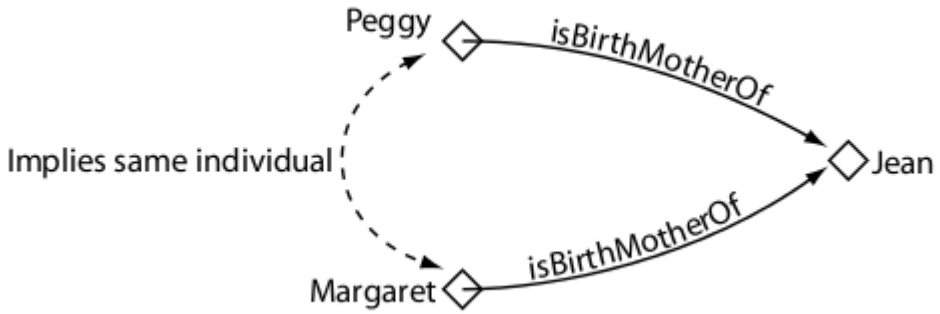
Equivalent Properties

Disjoint Properties

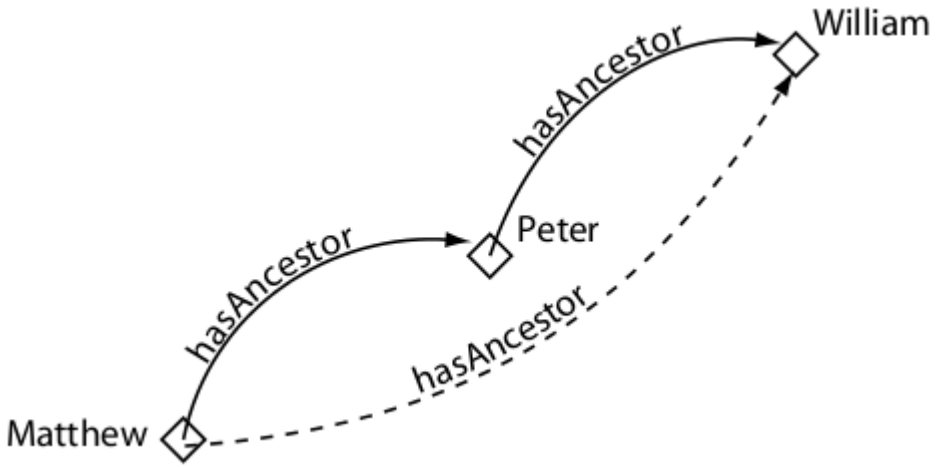
Functional



Inverse functional

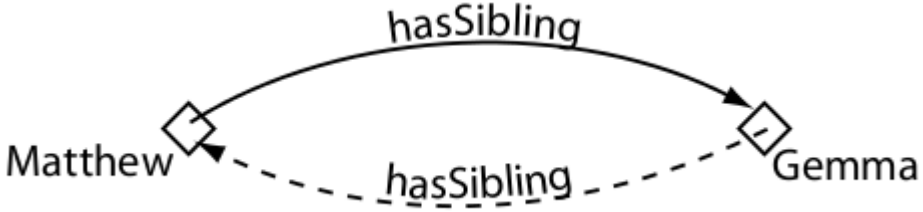


Transitive

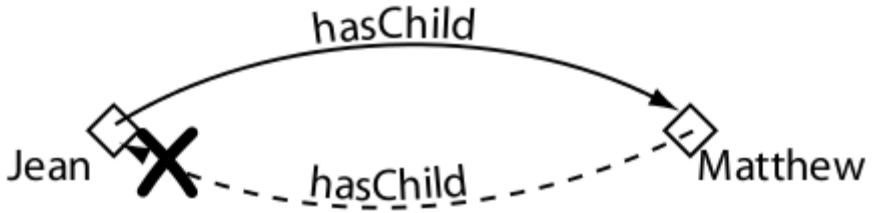


http://owl.cs.manchester.ac.uk/tutorials/protegeowltutorial/

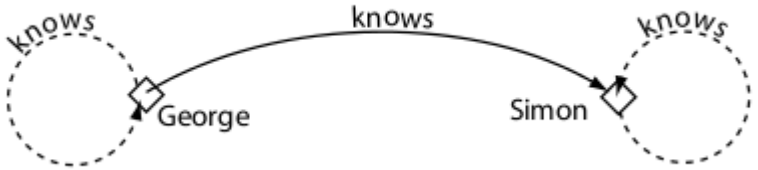
Symmetric



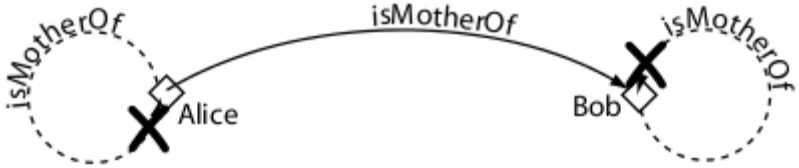
Antisymmetric



Reflexive

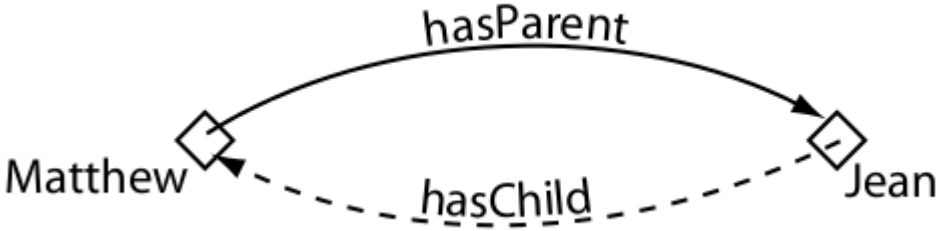


Irreflexive



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

Inverse properties



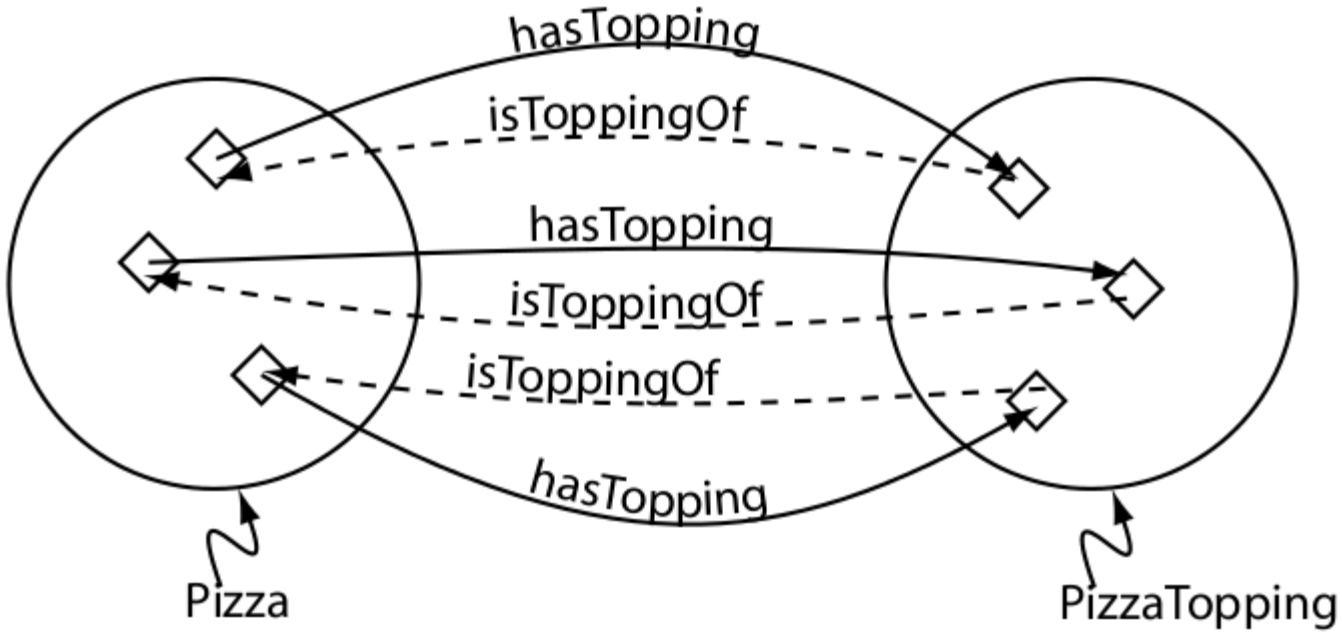
http://owl.cs.manchester.ac.uk/tutorials/protegeowltutorial/

Domain and Range:

Usually classes or class unions

But any anonymous expression class can be used

They are not constraints, they are axioms



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

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