OWL - Part 1/2

Class Axioms, Class Assertions, Property Axioms, Property Types, Entailment Rules, OWL-Specific Rules

CLASS AXIOMS

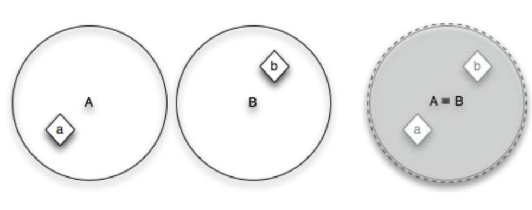
General

- owl:Class
- owl:Thing
- owl:NamedIndividual
- owl:Nothing

Equivalence

owl:*equivalentClass*

Equivalent classes contain the same individuals, and have the same definition.



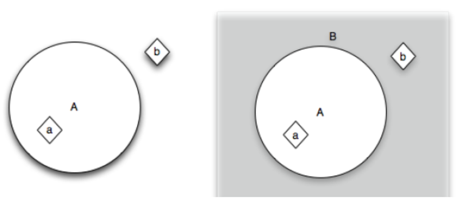
ex:A	owl:equivalentClass	ex:B .
ex:a	rdf:type	ex:A .
ex:b	rdf:type	ex:B .
ex:a	rdf:type	ex:B .
ex:b	rdf:type	ex:A .

Complement

Analgous to a "select inverse" function

owl:*complementOf*

The complement of a class contains all individuals that are **not** in the class.

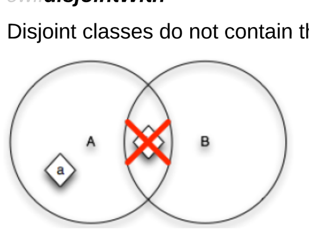


ex:B	owl:complementOf	ex:A .
ex:b	rdf:type	owl:Thing .
ex:b	rdf:type	ex:B .

Disjointness

owl:*disjointWith*

Disjoint classes do not contain the same individuals.



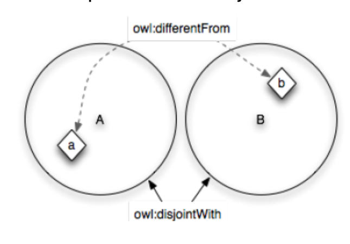
ex:A	owl:disjointWith	ex:B .
ex:a	rdf:type	ex:A .
ex:b	rdf:type	ex:A, ex:B .

inconsistent

Difference

owl:*differentFrom*

The complement and disjointness allow us to infer that individuals are different.

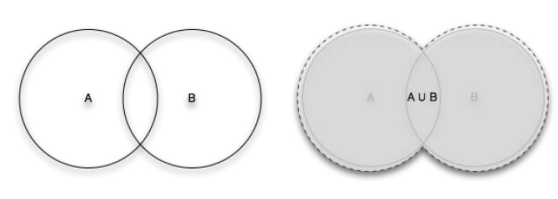


ex:A	owl:disjointWith	ex:B .
ex:a	rdf:type	ex:A .
ex:b	rdf:type	ex:B .
ex:a	owl:differentFrom	ex:b .

Union

owl:*unionOf*

The union contains all individuals that belong to the classes of the union.

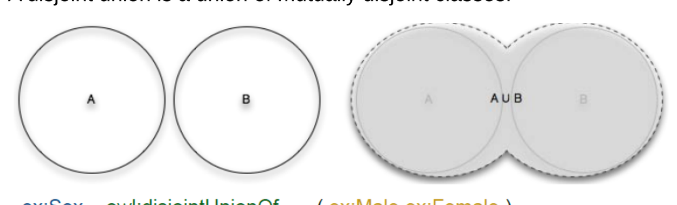


ex:C	owl:unionOf	(ex:A ex:B).
ex:a	rdf:type	ex:A .
ex:b	rdf:type	ex:B .
ex:a	rdf:type	ex:C .
ex:b	rdf:type	ex:C .

Disjoint Union

owl:*disjointUnionOf*

A disjoint union is a union of mutually disjoint classes.

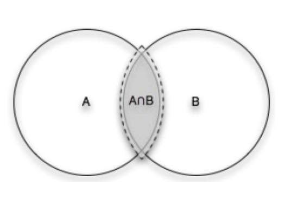


ex:Sex owl:disjointUnionOf (ex:Male ex:Female) .

Intersection

owl:*intersectionOf*

The intersection contains individuals that each belong to both classes.

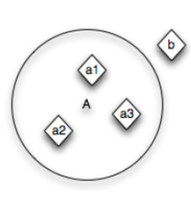


ex:C	owl:intersectionOf	(ex:A ex:B).
ex:a	rdf:type	ex:A .
ex:a	rdf:type	ex:B .
ex:a	rdf:type	ex:C .

Enumeration

owl:*oneOf*

We can enumerate all members of a class.



ex:A	owl:oneOf	(ex:a1 ex:a2 ex:a3) .
ex:a1	rdf:type	owl:Thing .
ex:a 1	rdf:type	ex:A .

CLASS ASSERTIONS

Same As

owl:*sameAs*

ex:Venus owl:sameAs ex:MorningStar, ex:EveningStar .

Different From

owl:*differentFrom*

ex:Venus owl:differentFrom ex:AlphaCentauri .

ex:MorningStar owl:sameAs ex:Venus, ex:EveningStar ;
owl:differentFrom ex:AlphaCentauri .
ex:EveningStar owl:sameAs ex:Venus, ex:MorningStar ;
owl:differentFrom ex:AlphaCentauri .

Negation

built with special construct

You can say that a property does not hold between two individuals

_:x	rdf:type	owl:NegativePropertyAssertion ;
	owl:sourceIndividual	ex:Amsterdam ;
	owl:assertionProperty	ex:locatedIn ;
	owl:targetIndividual	ex:Brussels .

PROPERTY AXIOMS

Inverse Properties

owl:*InverseOf*

Used to specify that one property is always the inverse of another property.

ex:p	owl:inverseOf	ex:q .
ex:x	ex:p	ex:y .
ex:y	ex:q	ex:x .

Equivalent Properties

owl:*equivalentProperty*

Used to specify that two properties always co-occur.

ex:p	owl:equivalentProperty	ex:q .
ex:x	ex:p	ex:y .
ex:x	ex:q	ex:y .

Disjoint Properties

owl:*disjointProperty*

Used to specify that two properties never co-occur.

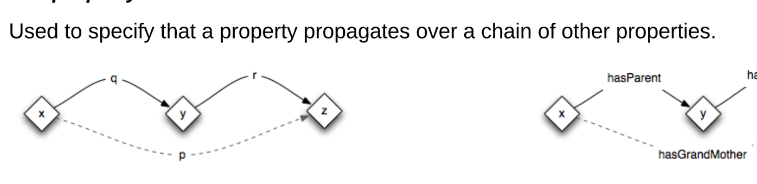
ex:p	owl:disjointProperty	ex:q .
ex:x	ex:p	ex:y .
ex:x	ex:q	ex:y .

inconsistent

Property Chain

owl:*propertyChainAxiom*

Used to specify that a property propagates over a chain of other properties.



ex:p	owl:propertyChainAxiom	(ex:q ex:r) .
ex:x	ex:q	ex:y .
ex:y	ex:r	ex:z .
ex:x	ex:p	ex:z .

PROPERTY TYPES

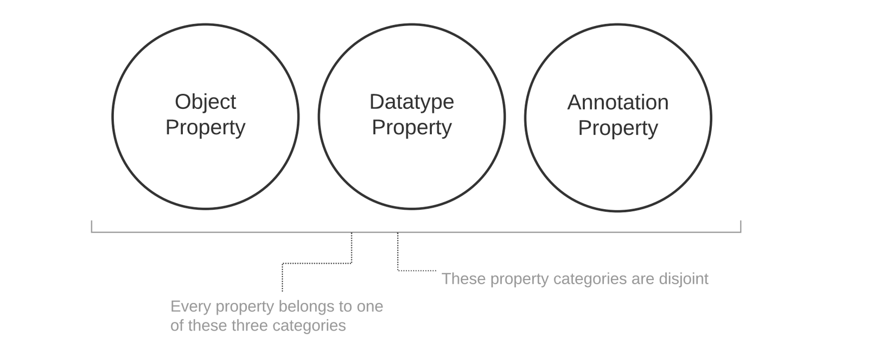
General

- owl:ObjectProperty
  - rdfs:subPropertyOf
  - owl:topObjectProperty
- owl:DatatypeProperty
  - owl:subPropertyOf
  - owl:topDatatypeProperty
- owl:AnnotationProperty

Have only non-literals as range

Have only literals as range

Cannot be used in restrictions



Symmetric

owl:*SymmetricProperty*

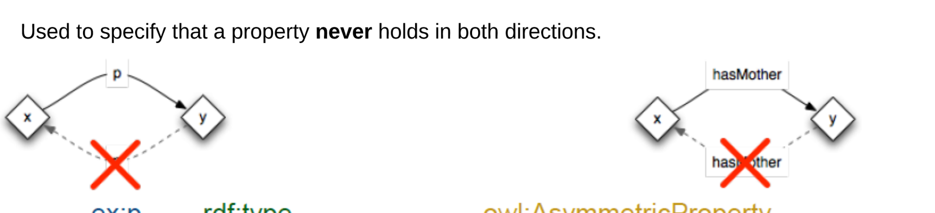
Used to specify that a property always holds in both directions.

ex:p	rdf:type	owl:SymmetricProperty .
ex:x	ex:p	ex:y .
ex:y	ex:p	ex:x .

Assymmetric

owl:*AssymetricProperty*

Used to specify that a property **never** holds in both directions.

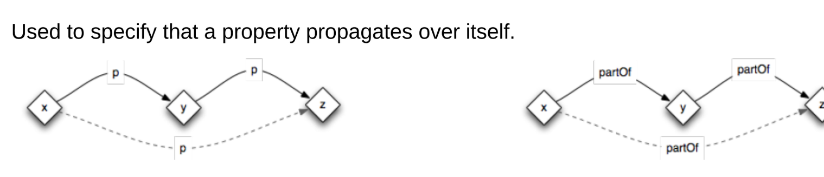


ex:p	rdf:type	owl:AsymmetricProperty .
ex:x	ex:p	ex:y .
ex:y	ex:p	ex:x .
ex:x	owl:differentFrom	ex:x .

Transitive

owl:*TransitiveProperty*

Used to specify that a property propagates over itself.



ex:p	rdf:type	owl:TransitiveProperty .
ex:x	ex:p	ex:y .
ex:y	ex:p	ex:z .
ex:x	ex:p	ex:z .

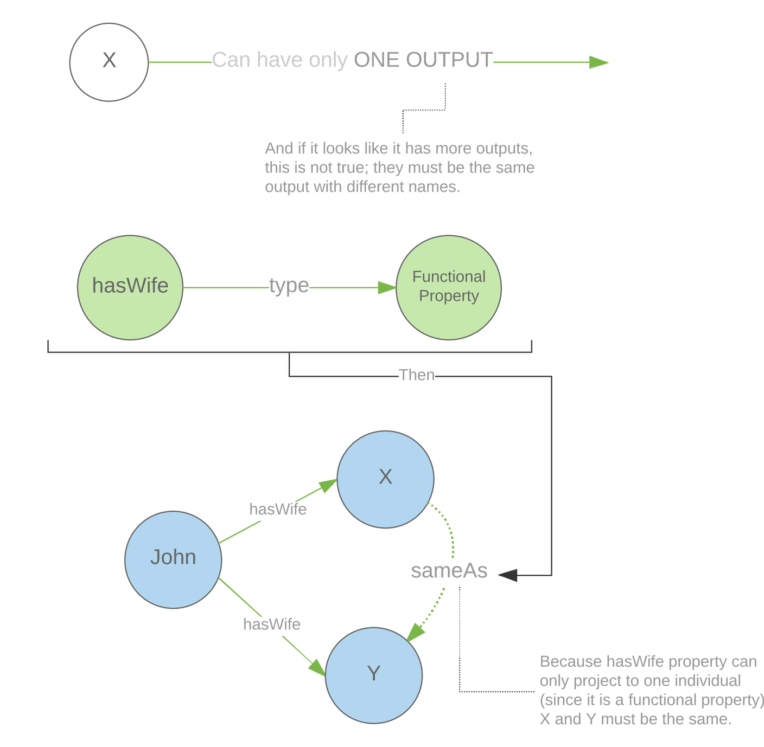
Functional

owl:*FunctionalProperty*

Read as "Unique Rangeness"

Used to specify that a property has only one value for any particular instance.

ex:p	rdf:type	owl:FunctionalProperty .
ex:x	ex:p	ex:y .
ex:x	ex:p	ex:z .
ex:y	owl:sameAs	ex:z .



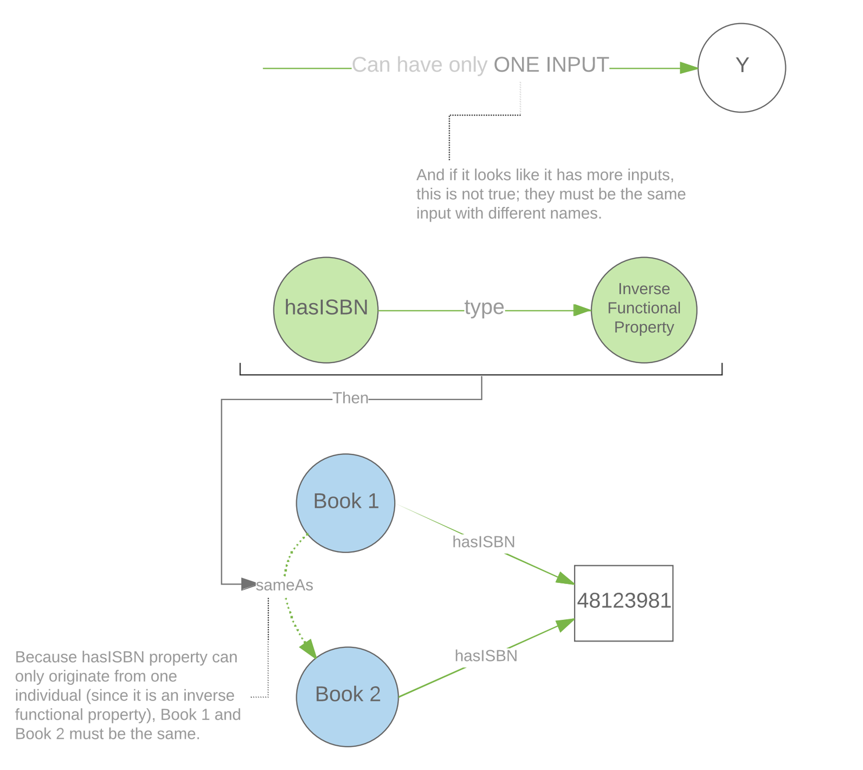
Inverse Functional

owl:*InverseFunctionalProperty*

Read as "Unique Domainness"

Used to specify that a value for the property uniquely identifies an instance.

ex:p	rdf:type	owl:InverseFunctionalProperty .
ex:x	ex:p	ex:y .
ex:z	ex:p	ex:y .
ex:x	owl:sameAs	ex:z .



Reflexive

owl:*ReflexiveProperty*

Used to specify that every individual is always related to itself by that property.

ex:p	rdf:type	owl:ReflexiveProperty .
ex:x	ex:p	ex:x .

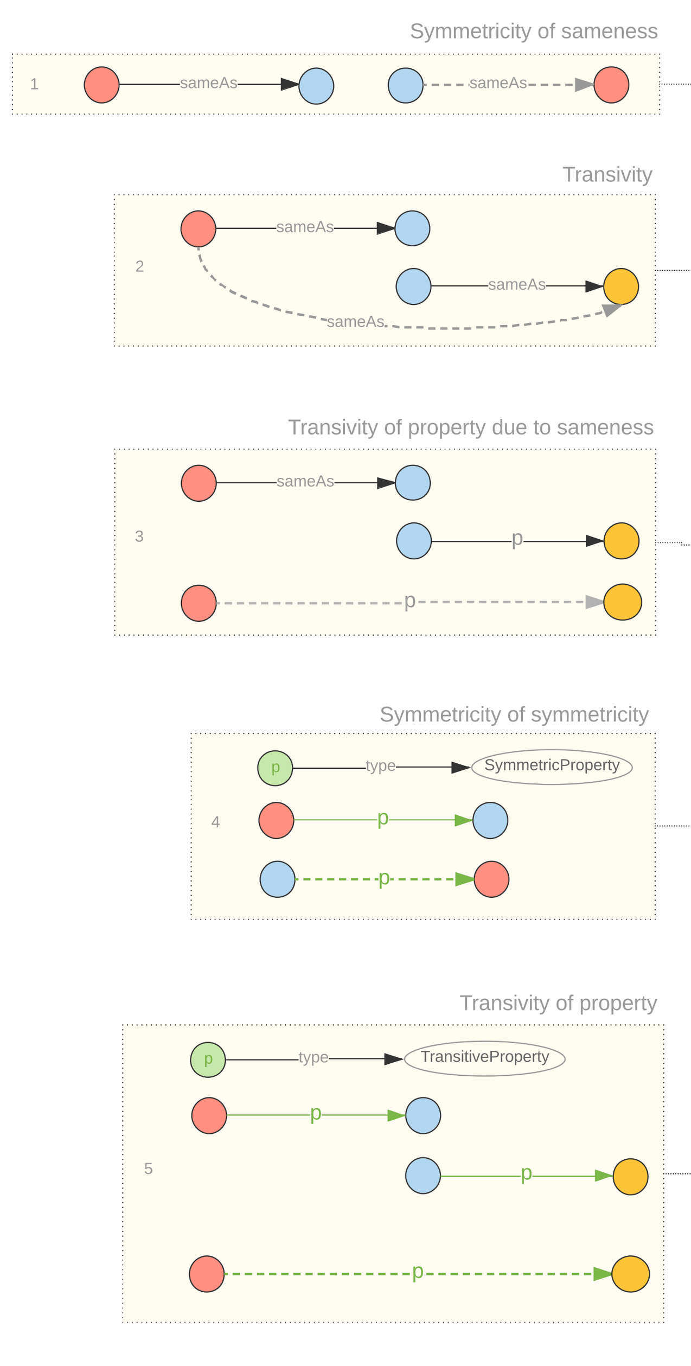
Irreflexive

owl:*IrreflexiveProperty*

Used to specify that no individual is ever related to itself via that property.

ex:p	rdf:type	owl:IrreflexiveProperty .
ex:x	ex:p	ex:x .
ex:x	owl:differentFrom	ex:x .

ENTAILMENT RULES FOR AXIOMS AND TYPES




```
1.  v  owl:sameAs        w            ⟺  w  owl:sameAs  v

2.  u  owl:sameAs        v
    v  owl:sameAs        w            ⟺  u  owl:sameAs  w

3.  v  owl:sameAs        w
    v      p             o            ⟺  w      p       o

4.  p  rdf:type   owl:SymmetricProperty
    v      p             w            ⟹  w      p       v

5.  p  rdf:type   owl:TransitiveProperty
    u      p             v
    v      p             w            ⟹  u      p       w
```

This is an extension of the RDF inference rule: "A class is always a subclass of itself".

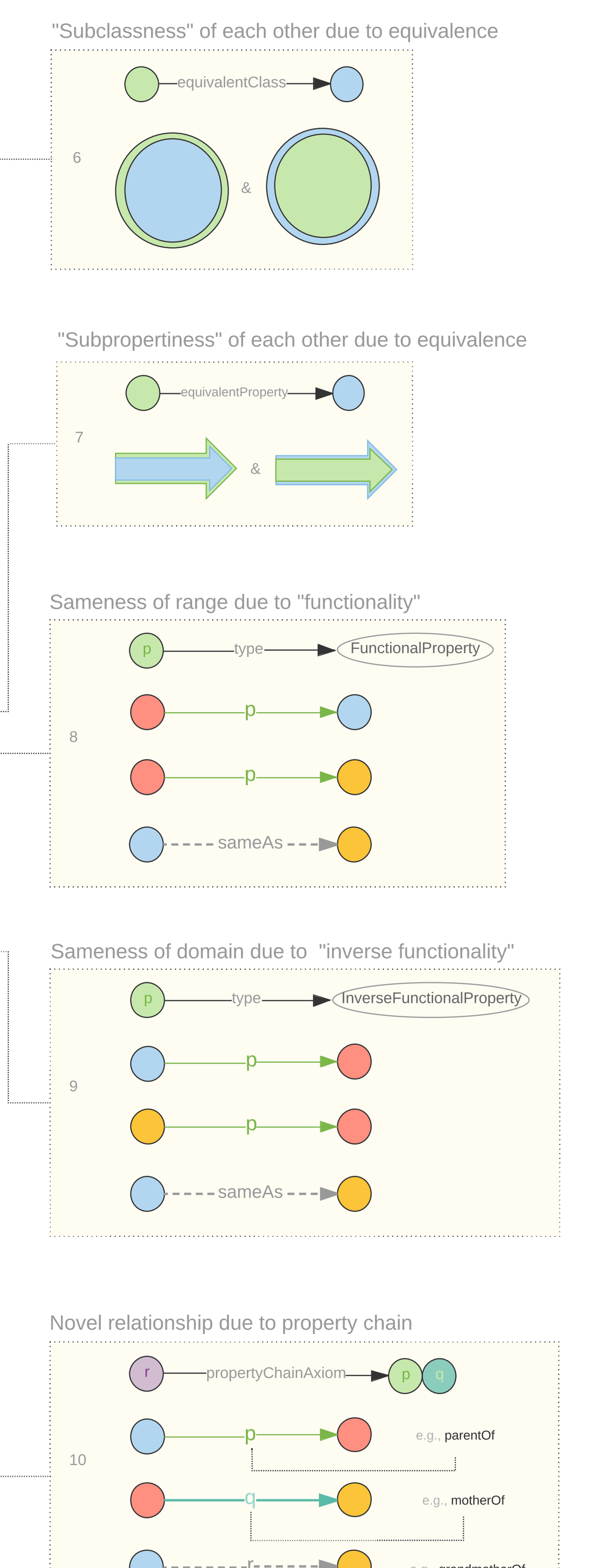
```
6.  v  owl:equivalentClass        w      ⟺  v  rdfs:subClassOf     w
                                             w  rdfs:subClassOf     v

7.  p  owl:equivalentProperty     q      ⟺  p  rdfs:subPropertyOf  q
                                             q  rdfs:subPropertyOf  p

8.  p     rdf:type    owl:FunctionalProperty
    x        p             v
    x        p             w             ⟹  v     owl:sameAs       w

9.  p     rdf:type    owl:InverseFunctionalProperty
    v        p             x
    w        p             x             ⟹  v     owl:sameAs       w

10. r  owl:propertyChainAxiom    (p q)
    v        p             w
    w        q             x             ⟹  v        r             x
```



OWL-SPECIFIC RULES

Open World Assumption

"Nothing is assumed to be true or false unless it is explicit knowledge or derivable from axioms or known facts. "

OWL does not assume that the ontology contains all possible knowledge, and behaves conservatively when classifying. Unless ***sufficient*** evidence required for classification is not specified (e.g., by using 'equivalentClass'), OWL would not automatically classify objects based on ***necessity*** criteria (i.e., which could have been established by using 'subClassOf').

For instance, meeting all the ***necessary*** criteria for being a car by having four wheels and the right size would not lead OWL to categorize a thing as a car (e.g., it may be a car model, but not an actual car), unless it is specifically stated that these two criteria is ***sufficient*** evidence for being categorized as a car.

Punning

OWL does not allow an entity to be used as class instances and objects at the same time (while RDF/S does).

To make things a bit easier, thouh, OWL automatically infers an URI's type (i.e., whether it is a class instance or object) from the position of a URI in an axiom. This is called 'punning'.

Reusing entities as objects, classes, and properties this way is likely to lead to reasoning errors, and is not allowed by OWL.