OWL - Part 2/2

Class Restrictions with Examples: Necessity, Sufficiency, Some, Only, Value, Cardinatlity, Minimum/Maximum, Local Reflexivity

Sheet 6

CLASS RESTRICTIONS

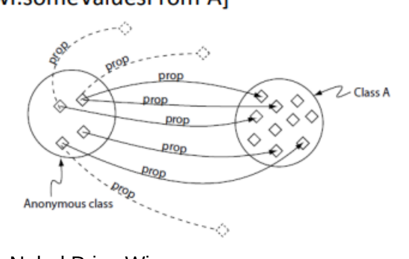
The Tools

Ways of determining class memberships via restricting properties

The "Only" keyword

owl:someValuesFrom

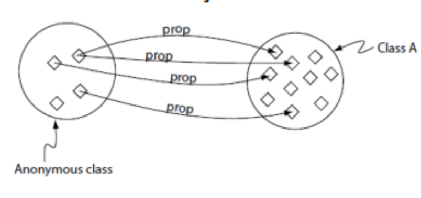
[a **owl:Restriction**;
owl:onProperty prop;
owl:someValuesFrom A]



e.g.: Nobel Prize Winners are persons who won a Nobel Prize, regardless of whether they have won other prizes or not.

owl:allValuesFrom

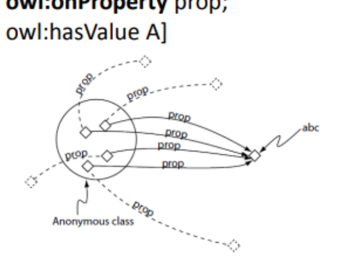
[a **owl:Restriction**;
owl:onProperty prop;
owl:allValuesFrom A]



e.g.: Nobel Prize Winners are persons who won a **only** a Nobel Prize (and no other prizes).

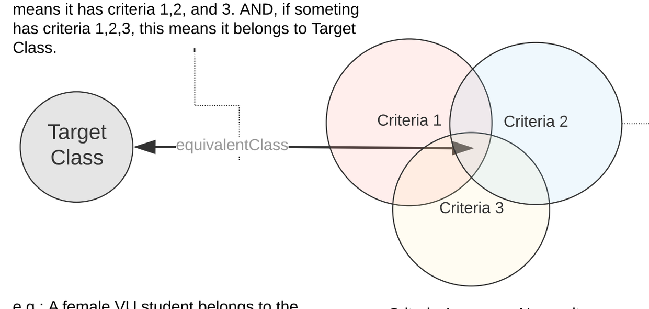
owl:hasValue

[a **owl:Restriction**;
owl:onProperty prop;
owl:hasValue A]



owl:equivalentClass — The keyword of "sufficiency" — Bidirectional inference

equivalentClass is a bi-directional relationship: If something belongs to the "Target Class", this means it has criteria 1,2, and 3. AND, if someting has criteria 1,2,3, this means it belongs to Target Class.

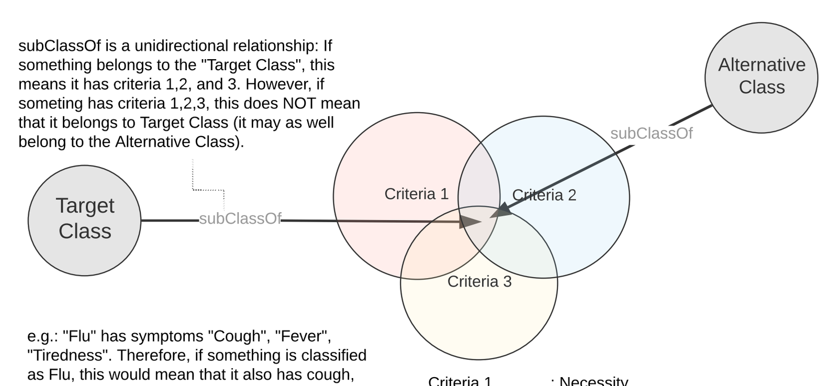


e.g.: A female VU student belongs to the subclasses of "female", "student", "located at VU". Likewise, if someone belongs to the classes "female", "student", and "located at VU", this is ***sufficient*** to be considered a female VU student.

Criteria 1 : Necessity
Criteria 2 : Necessity
Criteria 3 : Necessity
Criteria 1 + 2 + 3 : Sufficiency

rdf:subClassOf — The keyword of "necessity" — Unidirectional inference

subClassOf is a unidirectional relationship: If something belongs to the "Target Class", this means it has criteria 1,2, and 3. However, if someting has criteria 1,2,3, this does NOT mean that it belongs to Target Class (it may as well belong to the Alternative Class).



e.g.: "Flu" has symptoms "Cough", "Fever", "Tiredness". Therefore, if something is classified as Flu, this would mean that it also has cough, fever, and tiredness. However, if something is not classified, but it has these three symptoms does not warrant being classified as flu. After all, these symptoms could point out to more than one ilness.

Criteria 1 : Necessity
Criteria 2 : Necessity
Criteria 3 : Necessity
Criteria 1 + 2 + 3 : (Still) Necessity

Sufficiency: Equivalent & Some

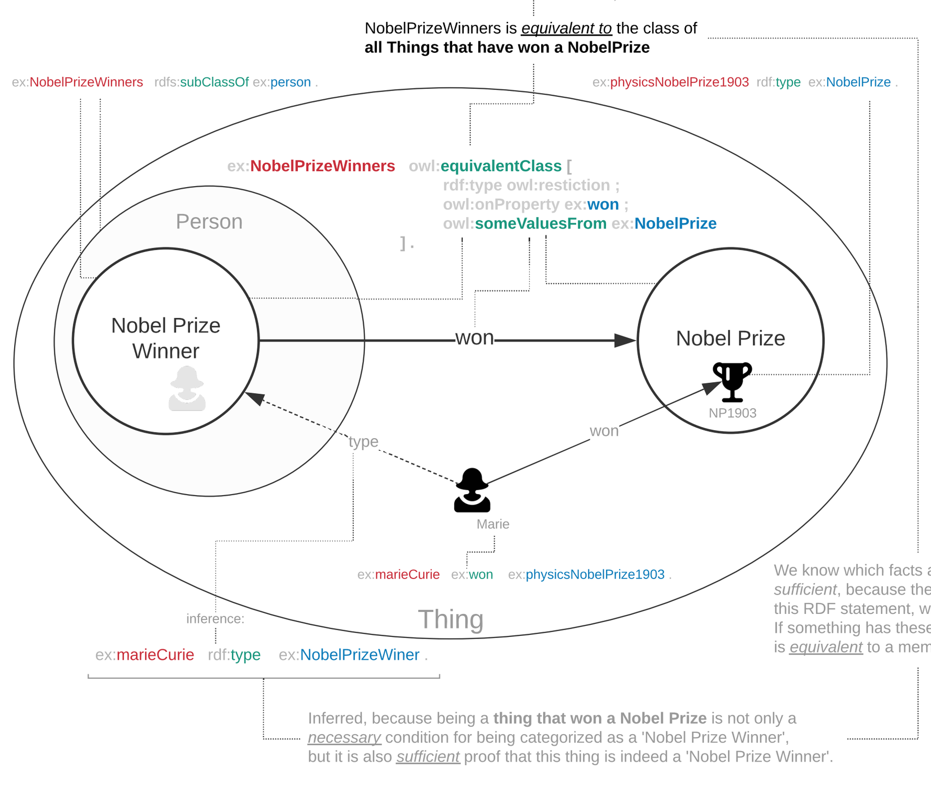
Creating a sufficiency rule

*owl:**equivalentClass*** | *owl:**someValuesFrom***

Wining a Nobel Prize is ***sufficient*** proof for being classified as a Nobel Prize winner.

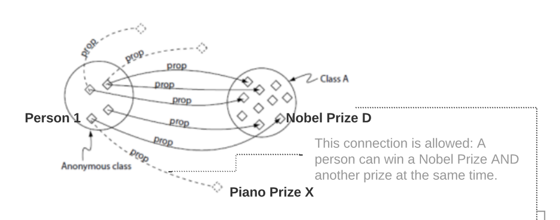
In more plain words: If something has won a Nobel Prize, it can be considered a Nobel Prize winner.

NobelPrizeWinners is *equivalent to* the class of **all Things that have won a NobelPrize**



We know which facts are *necessary* and *sufficient*, because they are specified by this RDF statement, which says: If something has these qualities, then it is *equivalent* to a member of this class.

Inferred, because being a **thing that won a Nobel Prize** is not only a *necessary* condition for being categorized as a 'Nobel Prize Winner', but it is also *sufficient* proof that this thing is indeed a 'Nobel Prize Winner'.



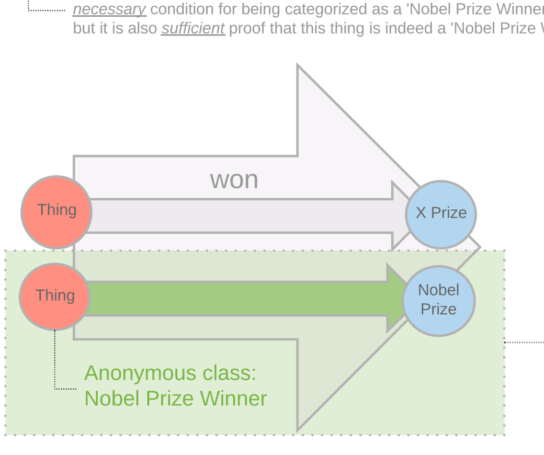
This connection is allowed: A person can win a Nobel Prize AND another prize at the same time.

To be considered a Nobel Prize winner — it is **sufficient** — to have won a Nobel Prize (among other possible prizes)

```
ex:NobelPrizeWinners    rdfs:subClassOf      ex:Person ;
                        owl:equivalentClass  [
                                 rdf:type            owl:Restriction ;
                                 owl:onProperty      ex:won ;
                                 owl:someValuesFrom  ex:NobelPrize ] .
ex:marieCurie             ex:won     ex:physicsNobelPrize1903 .
ex:physicsNobelPrize1903  rdf:type   ex:NobelPrize .
-----------------------------------------------------------------
ex:marieCurie             rdf:type   ex:NobelPrizeWinner .
```

So, given this reasoning...
IF
Marie Curie has won something named "Physics Nobel Prize 1903"
AND if this thing is of type "Nobel Prize",
THEN
Marie Curie must be a "Nobel Prize Winner".

This is inferred, because to be considered a Nobel Prize winner, it is ***sufficient*** to have won a Nobel Prize (regardless of other prizes that may have been won.)

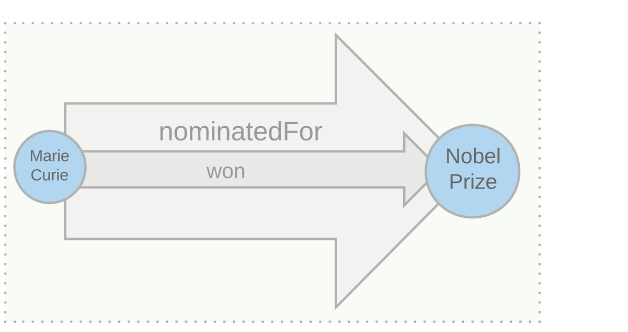


Necessity: Subclass & Some

Creating a necessity rule

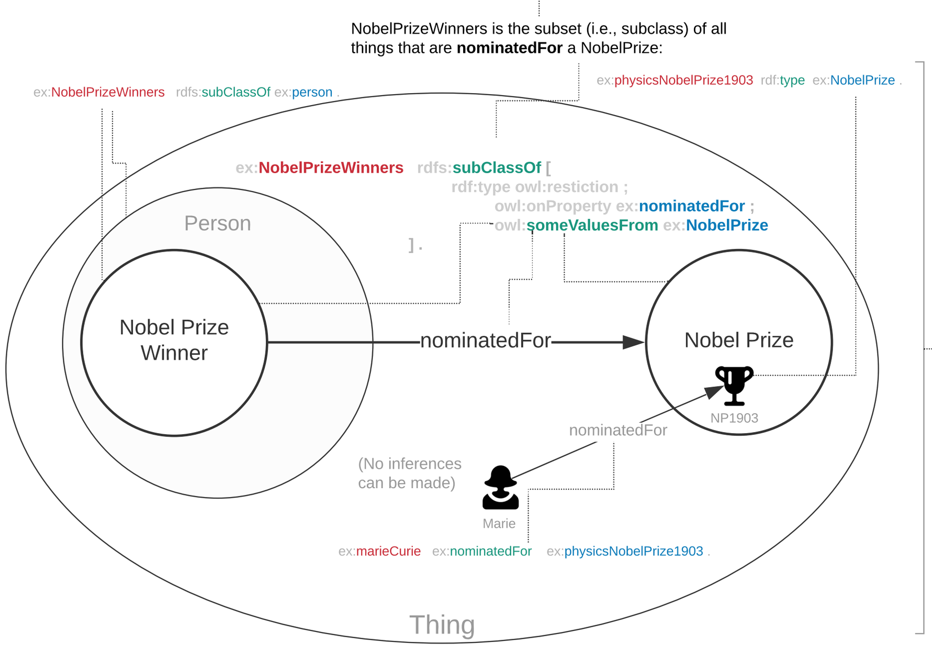
*rdfs:**subclassOf*** | *owl:**someValuesFrom***

Being a Nobel Prize Nominee is a ***necessity*** to be classified as a Nobel Prize Winner, but it is not ***sufficient*** to be classified as one.



In more plain words: Being a Nobel Prize Nominee is a condition for being a Nobel Prize Winner.

NobelPrizeWinners is the subset (i.e., subclass) of all things that are **nominatedFor** a NobelPrize:



To be considered a Nobel Prize winner — it is **necessary** — to be nominated for a Nobel Prize (among other possible prizes)

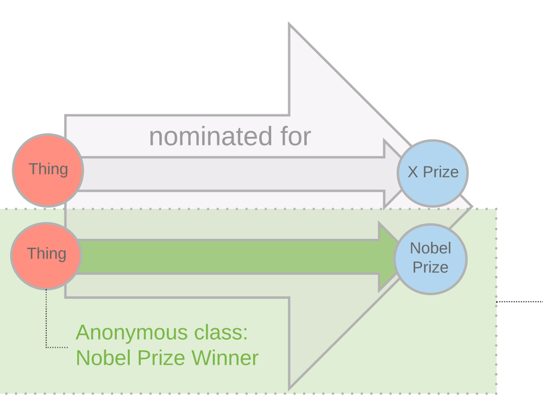
```
ex:NobelPrizeWinners    rdfs:subClassOf      ex:Person ;
                        rdfs:subClassOf      [
                                 rdf:type            owl:Restriction ;
                                 owl:onProperty      ex:nominatedFor ;
                                 owl:someValuesFrom  ex:NobelPrize ] .
ex:marieCurie             ex:nominatedFor  ex:physicsNobelPrize1903 .
ex:physicsNobelPrize1903  rdf:type         ex:NobelPrize .
-----------------------------------------------------------------
… nothing is inferred …
```

So, given this reasoning...
IF
Marie Curie is nominated for something named "Physics Nobel Prize 1903"
AND if this thing is of type "Nobel Prize",
THEN
We cannot infer anything.

This is because even though Marie Curie's nomination meets the condition of ***necessity*** (i.e., makes her eligible), it does not meet the condition of ***sufficiency*** for membership to Nobel Prize Winners.

Indeed, in reality, being nominated does not mean having won a prize.

This relates to the open world assumption: We know that Marie Curie is a nominee, but this is all we know in this particular ontology. In an open world (i.e., a world in which more information than we have in the ontology may exist), it could be perfectly possible that she is a nominee, but not a winner. Having no evidence that she is not a winner does not guarantee that she is a winner.



Exclusive Necessity: Subclass & Only

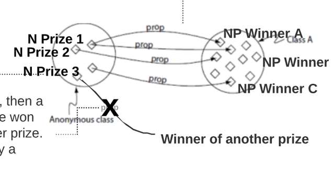
Forcing a property to connect two classes exclusively using a necessity rule

*rdfs:**subclassOf*** | *owl:**allValuesFrom***

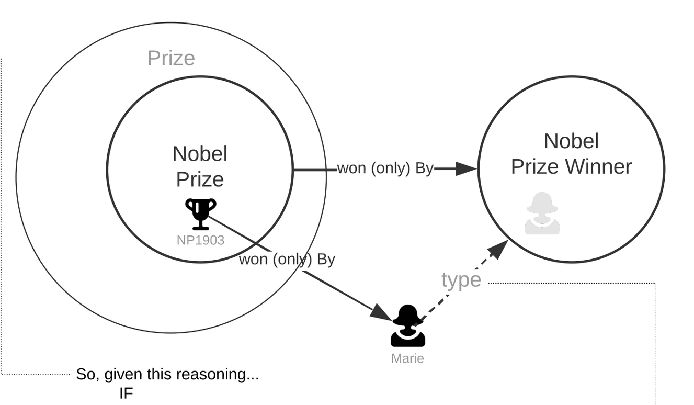
A Nobel Prize can be won only by a Nobel Prize winner, and having won by a Nobel Prize winner is ***necessary*** to be categorized as a Nobel Prize:

For a Nobel Prize — it is ***necessary*** — to be won ***only*** by a NobelPrizeWinner

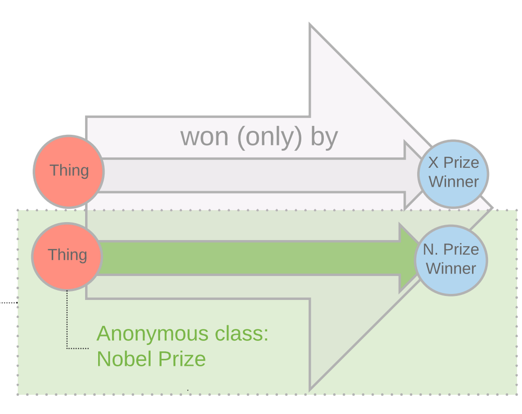
```
ex:NobelPrize    rdfs:subClassOf    ex:Prize ;
                 rdfs:subClassOf    [ rdf:type  owl:Restriction ;
                                      owl:onProperty   ex:wonBy ;
                                      owl:allValuesFrom  ex:NobelPrizeWinner ] .
ex:physicsNobelPrize1903  ex:wonBy   ex:marieCurie .
ex:physicsNobelPrize1903  rdf:type   ex:NobelPrize .
-----------------------------------------------------------------
ex:marieCurie             rdf:type   ex:NobelPrizeWinner .
```



If it is 'allValuesFrom', then a Nobel Prize cannot be won by a winner of another prize. (It can only be won by a 'Nobel Prize Winner')



So, given this reasoning...
IF
Marie Curie has won something named "Physics Nobel Prize 1903"
AND if this thing is of type "Nobel Prize",
THEN
Marie Curie must be a "Nobel Prize Winner".



Exclusive Sufficiency: Equivalent & Only

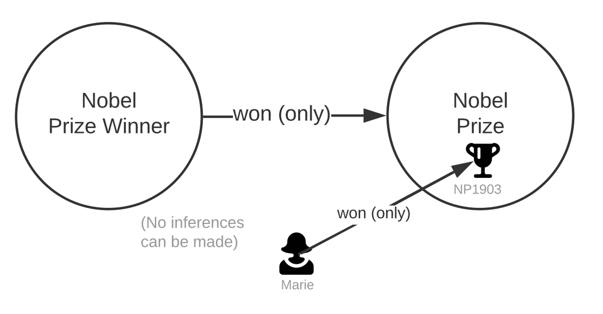
Forcing a property to connect two classes exclusively using a sufficiency rule (example #1)

*owl:**equivalentClass*** | *owl:**allValuesFrom***

Every Nobel Prize Winner only won Nobel Prizes, and having won only a Nobel Prize is ***sufficient*** to be classified as a Nobel Prize winner.

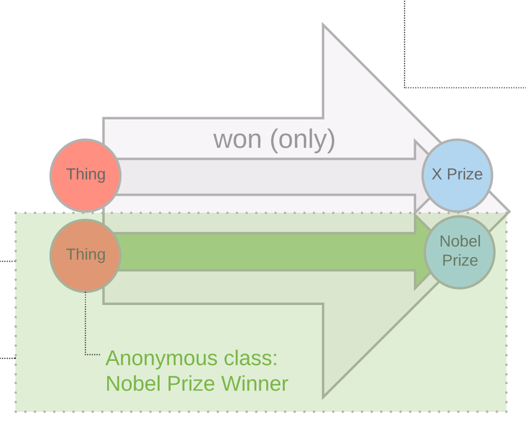
To be considered a Nobel Prize Winner — It is ***sufficient*** to — have won ***only*** a Nobel Prize (and to have won nothing else)

```
ex:NobelPrizeWinner   owl:equivalentClass   [ rdf:type  owl:Restriction ;
                                              owl:onProperty   ex:won ;
                                              owl:allValuesFrom  ex:NobelPrize ] .
ex:marieCurie             ex:won     ex:physicsNobelPrize1903 .
ex:physicsNobelPrize1903  rdf:type   ex:NobelPrize .
-----------------------------------------------------------------
… nothing is inferred …
```



So, given this reasoning...
IF
Marie Curie has won something named "Physics Nobel Prize 1903"
AND if this thing is of type "Nobel Prize",
THEN
We cannot infer anything.

We can't infer anything, because we cannot know (at least within the confines of an ontology) whether Marie Curie has not won any other prizes than Nobel Prize. This is the **Open World Assumption in OWL**.



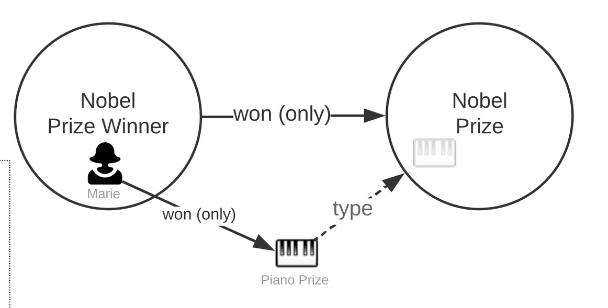
Forcing a property to connect two classes exclusively using a sufficiency rule (example #2)

*owl:**equivalentClass*** | *owl:**allValuesFrom***

Every Nobel Prize Winner only won Nobel Prizes, and having won only a Nobel Pize is ***sufficient*** to be classified as a Nobel Prize winner. Or, having won by a Nobel Prize winner [whom by definition has only one prize, which is a Nobel prize] is ***sufficient*** to be classified as a Nobel Prize.

To be considered a Nobel Prize Winner — It is sufficient to — have won **only** a Nobel Prize (and to have won nothing else)

```
ex:NobelPrizeWinner   owl:equivalentClass   [ rdf:type  owl:Restriction ;
                                              owl:onProperty   ex:won ;
                                              owl:allValuesFrom  ex:NobelPrize ] .
ex:marieCurie   ex:won     ex:pianoCompetition1901 .
ex:marieCurie   rdf:type   ex:NobelPrizeWinner.
-----------------------------------------------------------------
ex:pianoCompetition1901   rdf:type   ex:NobelPrize
```



So, given this reasoning...
IF
Marie Curie has won something named "Piano Competition Prize 1901"
AND if Marie Curie is a (i.e., has type) "Nobel Prize Winner",
THEN
"Piano Competition Prize 1901" must be of class "Nobel Prize".

We infer this faulty conclusion, because if as a Nobel Prize Winner, Marie Curie is allowed to only win one prize (that is the Nobel-), and if she has won a prize (i.e., Piano Competition Prize), then the price she has won must be a Nobel Prize. This is a result of the faulty class restriction we made.

Value

Forcing a property to connect a class and a value

*owl:**hasValue***

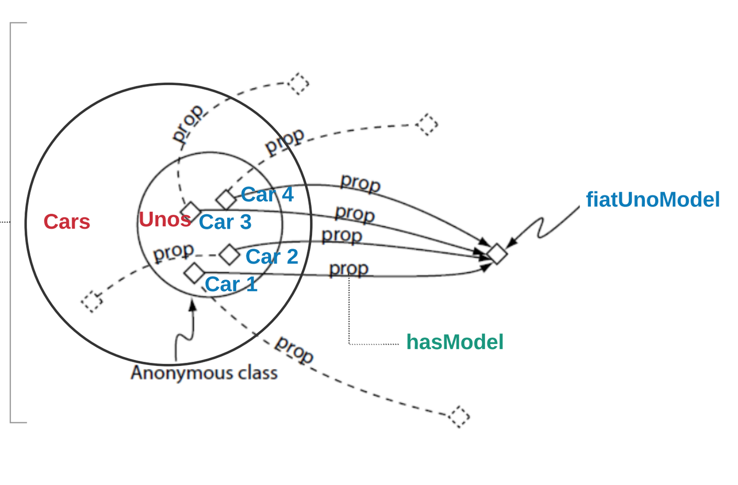
All members of a class have this instance as value; and depending on the statement, this is either ***necessary*** or ***sufficient*** to classify something as a class member.

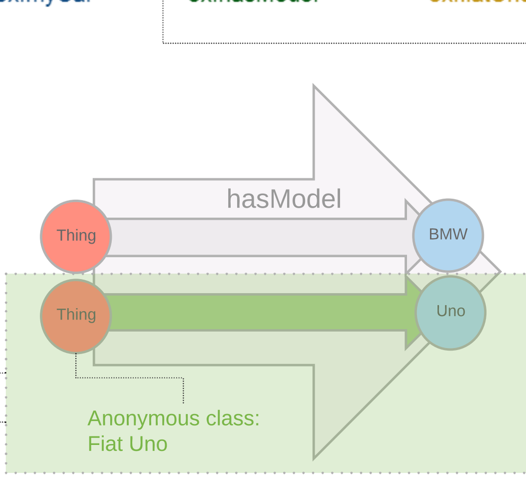
To be considered a Fiat Uno — a thing must have the 'model' value: Fiat Uno — and this alone is ***sufficient*** to prove that something is a Fiat Uno

```
ex:FiatUno   rdfs:subClassOf      ex:Car ;
             owl:equivalentClass  [ rdf:type  owl:Restriction ;
                                    owl:onProperty  ex:hasModel ;
                                    owl:hasValue    ex:fiatUnoModel ] .
ex:myCar     ex:hasModel          ex:fiatUnoModel .
-----------------------------------------------------------------
ex:myCar     rdf:type             ex:FiatUno .
```

To be considered a Fiat Uno — a thing must have the 'model' value: Fiat Uno — but this alone is not enough to prove that something is a Fiat Uno

```
ex:FiatUno   rdfs:subClassOf      ex:Car ;
             rdfs:subClassOf      [ rdf:type  owl:Restriction ;
                                    owl:onProperty  ex:hasModel ;
                                    owl:hasValue    ex:fiatUnoModel ] .
ex:myCar     rdf:type             ex:FiatUno .
-----------------------------------------------------------------
ex:myCar     ex:hasModel          ex:fiatUnoModel .
```





Cardinality

*owl:**minQualifiedCardinality***

Forcing a property to connect a class and a specified number of instances

If someting is a Table — it has a minimum of — four legs

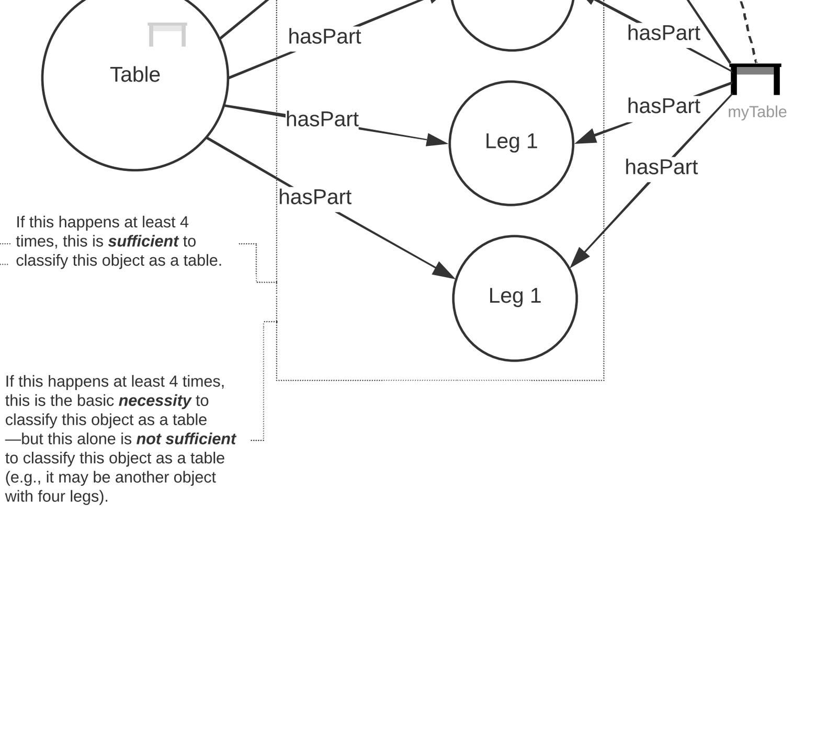
```
ex:Table  owl:equivalentClass [ rdf:type  owl:Restriction ;
                  owl:onProperty              ex:hasPart ;
                  owl:minQualifiedCardinality "4"^^xsd:integer ;
                  owl:onClass                 ex:Leg ] .
ex:myTable ex:hasPart   ex:l1, ex:l2, ex:l3, ex:l4 . # all in ex:Leg, and all different
-----------------------------------------------------------------
ex:myTable rdf:type     ex:Table .
```

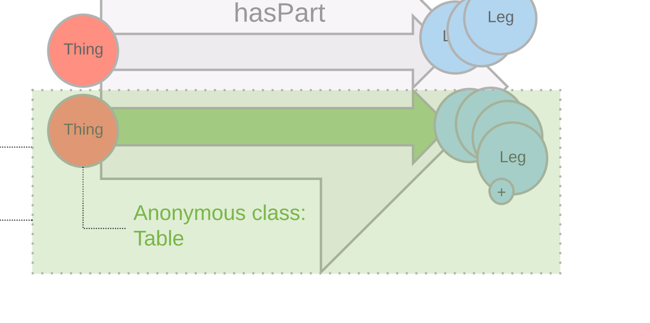
If this happens at least 4 times, this is ***sufficient*** to classify this object as a table.

To be considered a Table — a thing must have minimum — four legs

```
ex:Table  owl:subClassOf [ rdf:type  owl:Restriction ;
                  owl:onProperty              ex:hasPart ;
                  owl:minQualifiedCardinality "4"^^xsd:integer ;
                  owl:onClass                 ex:Leg ] .
ex:myTable ex:hasPart   ex:l1, ex:l2, ex:l3, ex:l4 . # all in ex:Leg, and all different
-----------------------------------------------------------------
… nothing is inferred …
```

If this happens at least 4 times, this is the basic ***necessity*** to classify this object as a table —but this alone is ***not sufficient*** to classify this object as a table (e.g., it may be another object with four legs).





Local Reflexivity

Selecting a self-projecting class

*owl:**hasSelf*** | *rdfs:**subClassOf***

To be considered a Narcissist — a thing must at least have a — "loves" relation to itself

```
ex:Narcissist  rdfs:subClassOf [ rdf:type  owl:Restriction ;
                   owl:onProperty  ex:loves ;
                   owl:hasSelf     "true"^^xsd:boolean ] .
ex:Narcissus   rdf:type   ex:Narcissist .
-----------------------------------------------------------------
ex:Narcissus   ex:loves   ex:Narcissus
```