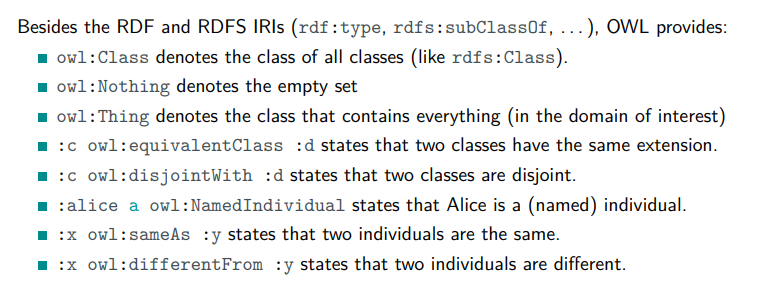
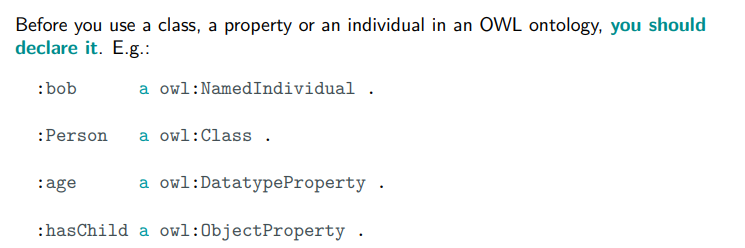


OWL semantic can be described as translation to description logic

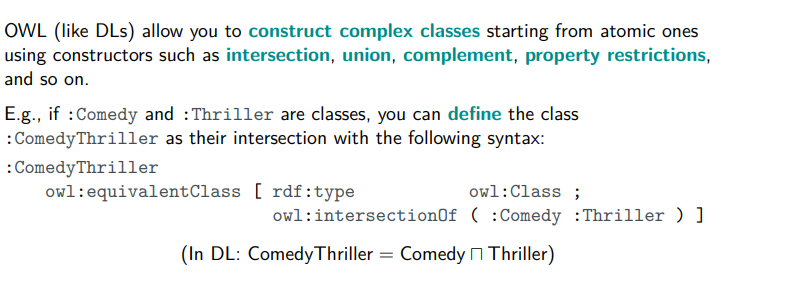
OWL vocabulary:



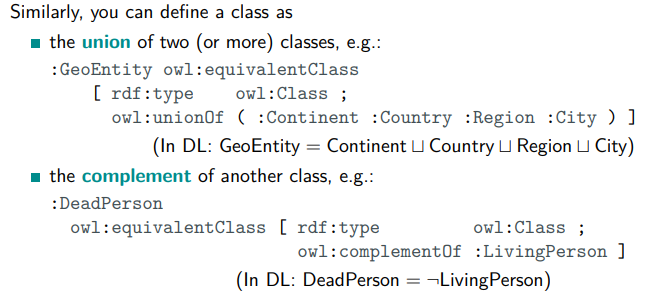
by default OWL assume that two individuals with different name are separated

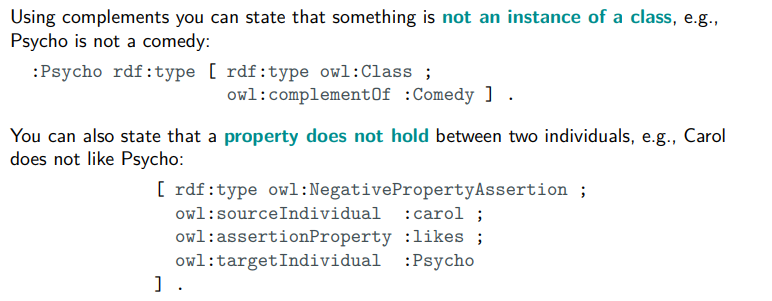


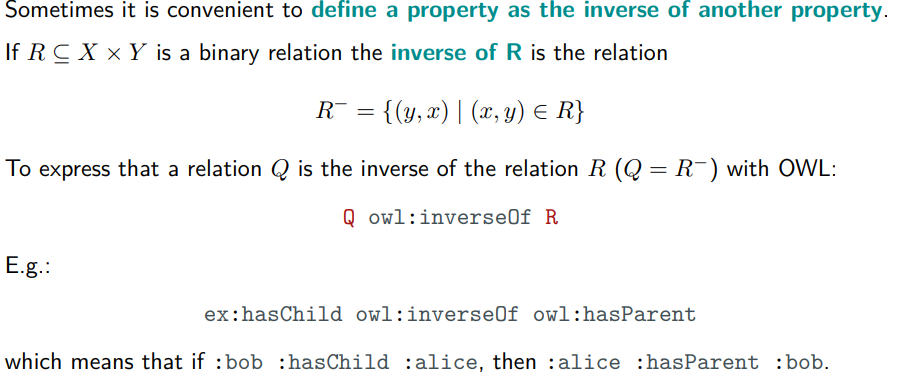
specify if what you introduce is an individual, a class…

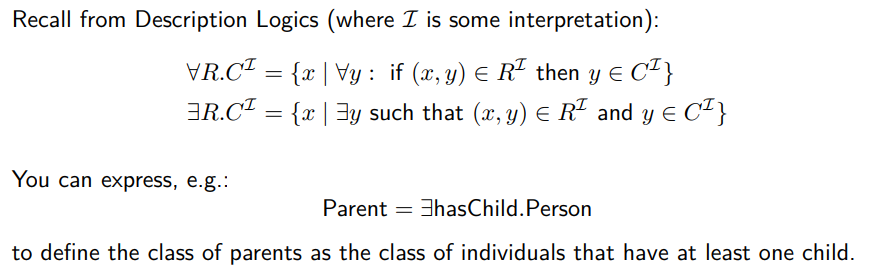


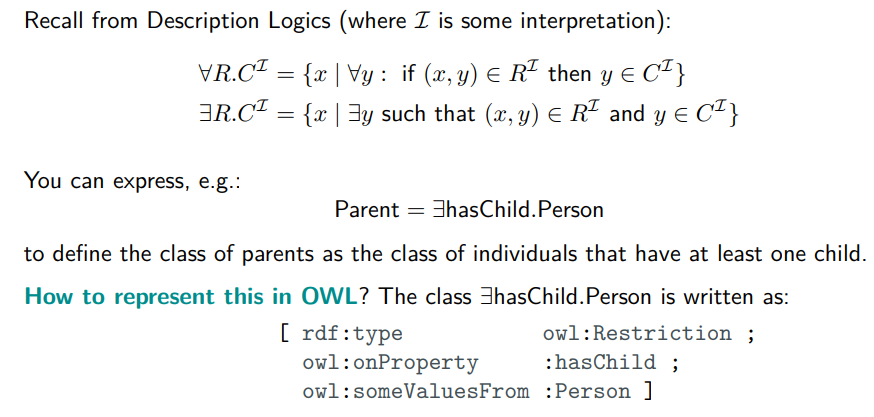
∏ = and











**There are people, movies, actors, comedy actors, etc. (i.e., they are classes: the first time you use a class, you should state that it is an owl:Class)**

:people a owl:Class .

:movie a owl:Class .

:actors a owl:Class .

:comedyactors a owl:Class .

**• Every movie is either a horror movie, a comedy or a drama. Hint. Movie = Horror ⊔ Comedy ⊔ Drama**

:horror a owl:Class .

:comedy a owl:Class .

:drama a owl:Class .

:movie owl:equivalentClass [rdf:type owl:Class; owl:unionOf ( :horromovie :comedymovie :dramamovie ) ]

**Comedy horror movies are movies that are both comedies and horrors. Hint. ComedyHorror = Comedy ⊔ Horror**

(:comedyhorror a owl:Class . )

:comedyhorror owl:equivalentClass [ rdf:type owl:Class ;

owl:intersectionOf ( :Comedy :Thriller )]

**There is nothing that is both a comedy and a drama**

:comedy owl:disjointWith :drama .

**Directed by, director of, starring and stars in are properties (also write down which classes they relate, and which properties are inverses of which)**

rivedere

ex:directedby a owl:ObjectProperty .

ex:directorof a owl:ObjectProperty .

ex:starring a owl:ObjectProperty .

ex:starsin a owl:ObjectProperty .

ex:director a rdfs:Class .

ex:movie a rdfs:Class .

ex:actor a rdfs:Class .

ex:directedby rdfs:domain ex:movie; rdfs:range ex:director .

ex:directorof rdfs:domain ex:director; rdfs:range ex:movie .

ex:starring rdfs:domain ex:movie; rdfs:range ex:actor .

ex:starsin rdfs:domain ex:actor; rdfs:range ex:movie .

ex:directorof owl:inverseOf ex:directedby

ex:starring owl:inverseOf ex:starin

**Jane Campion has directed The Power of the Dog.**

ex:JaneCampion rdf:hasDirected ex:TPOTD

TPOTD rdf:directedby :JaneCampion

**Charlie Chaplin does not star in The Power of the Dog.**

[rdf:type owl:NegativePropertyAssertion ;

owl:sourceIndividual :CharlieChaplin ;

owl:assertionProperty :starsIn ;

owl:targetIndividual :ThePowerOfTheDog ]

**Actors are defined as entities who have starred in at least one movie. Hint. Actor = ∃starsIn.Movie**

:Actor owl:equivalentClass [rdf:type owl:Restriction ;

owl:onProperty :starsIn ;

owl:someValuesFrom :Movie ] .

**Comedy actors are defined as actors who only star in comedy movies. Hint.**

**ComedyActor = Actor u ∀starsIn.ComedyMovie**

ex:comedyActory a owl: Class ;

owl: equivalentClass [ a owl:Class ; owl:intersectionOf ( ex:actor

**Every movie has at least one director and one actor. Hint. Movie ⊑ ∃directedBy.Director ⊔ ∃starring.Actor**

:Movie rdfs:subClassOf [ owl:intersectionOf ( [ rdf:type owl:Restriction ;

owl:onProperty :directedBy ;

owl:someValuesFrom :Director ]

[ rdf:type owl:Restriction ;

owl:onProperty :starring ;

owl:someValuesFrom :Actor ] ) ]

# parte 2

[SPARQL playground](https://sparql-playground.sib.swiss/)

(optional) Translate the ontology into Description Logic Syntax (just the complex class definitions and the assertions).

HappyPerson ⊆ Person

BlueItalianCheese = BlueCheese ∩ Italiancheese

GoodCheese = BlueCheese ∪ ItalianCheese

:bob a :Person

:bob :hasChild \_:x

:alice :hasParent :bob