OWL Web Ontology Language

Contenido

- OWL2: Introducción
- Conceptos Básicos de OWL
- Sintaxis – DL, OWL, Manchester
- Lenguaje OWL
  - Clases
  - Propiedades: : Object Properties, Data Properties
  - Individuos

Bases OWL: Sintaxis

- Toda la información en lenguajes ontológicos se expresa mediante un **tripleta**.
- Un tripleta consta de un sujeto, un predicado y un objeto.

Ejemplos:

subject predicate object
Azuay tieneCapital Cuenca
Cuenca tieneAlcalde Pedro_Palacios
Pedro_Palacios fechaNacimiento 1976

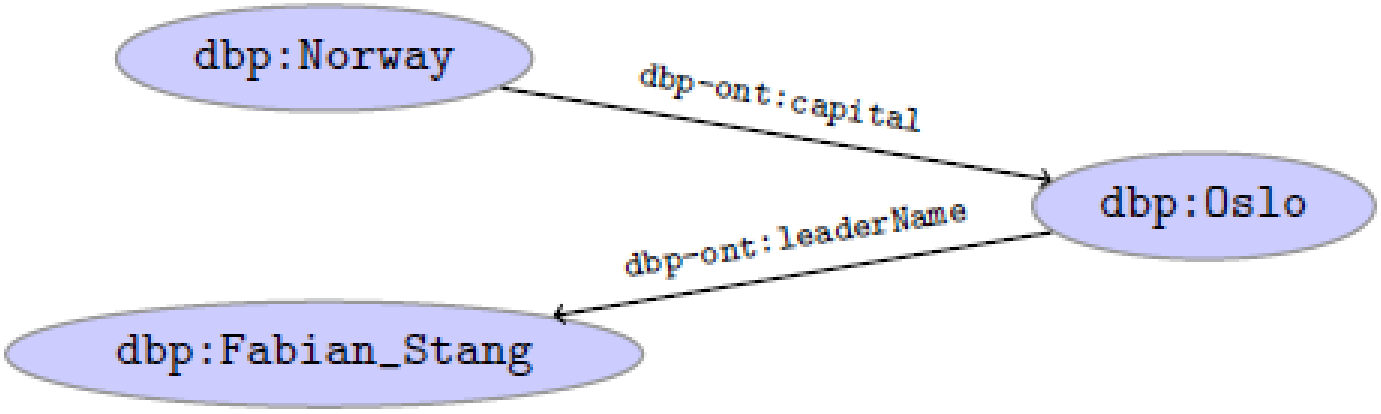
- Otra palabra para un tripleta es una **declaración** o un **hecho**.

Bases OWL: Sintaxis

- Un grafo es un conjunto de tripletas. P.ej.,

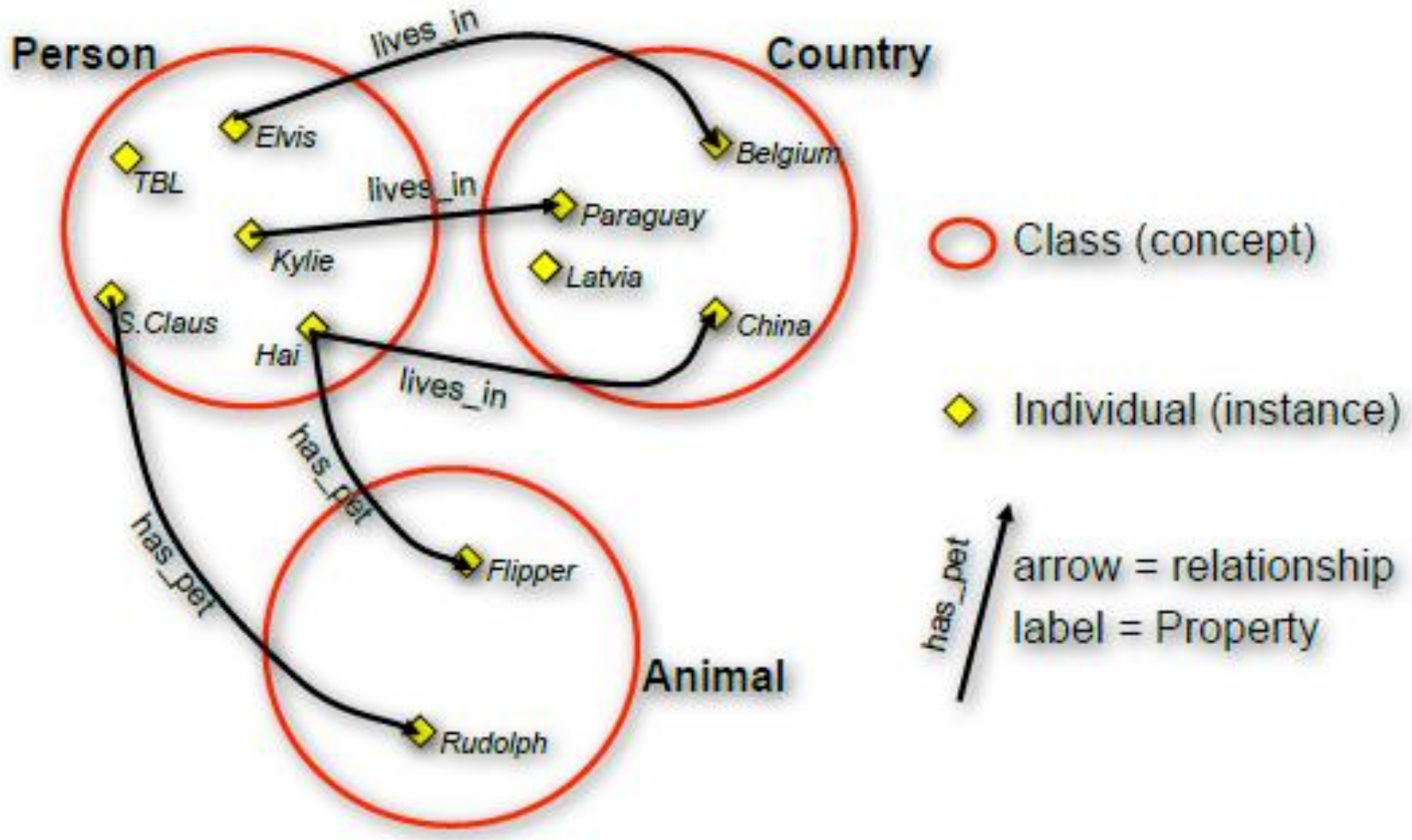
  dbp:Noruega dbp-ont:capital dbp:Oslo .
  dbp:Oslo dbp-ont:leaderName dbp:Fabian_Stang .

- es un grafo que contiene dos tripletas.
- Los grafos a menudo se representan como un gráfico etiquetado dirigido:



Elementos Básicos de Ontologías OWL

Resumen de construcciones OWL

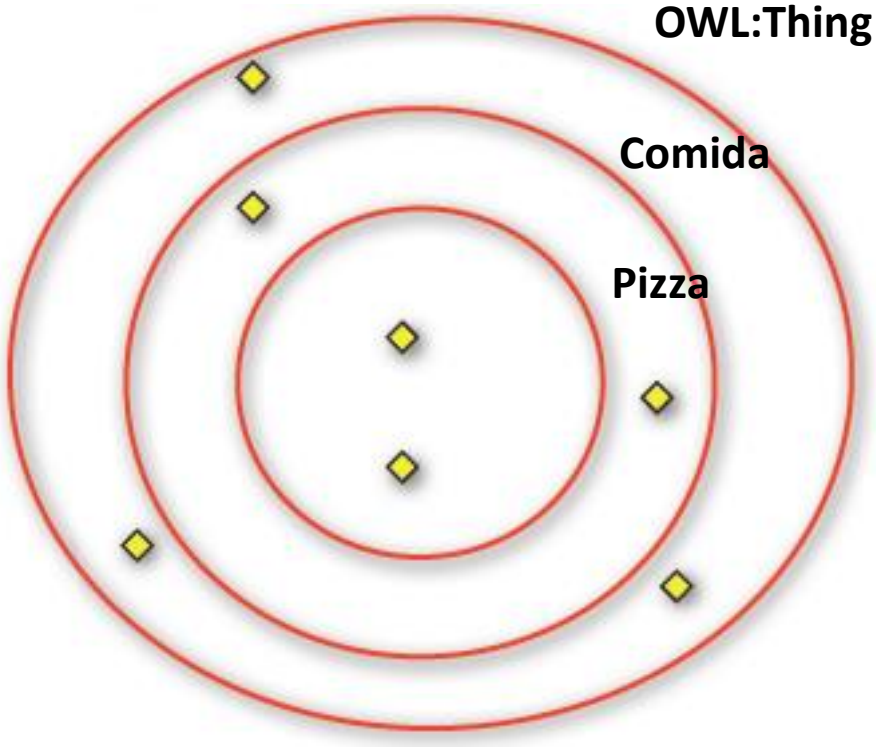


Clases: Declaración

- Hay dos clases predefinidas
  - *owl:Thing* (clase que contiene todos los clases)
  - *owl:Nothing* (clase vacía)
- Cada clase (*class) en* OWL es miembro de la clase *owl:Thing*
- Clases son definidas usando **owl:Class**
- Definición de una clase
  :Persona a owl:Class .

Clases: Inclusión

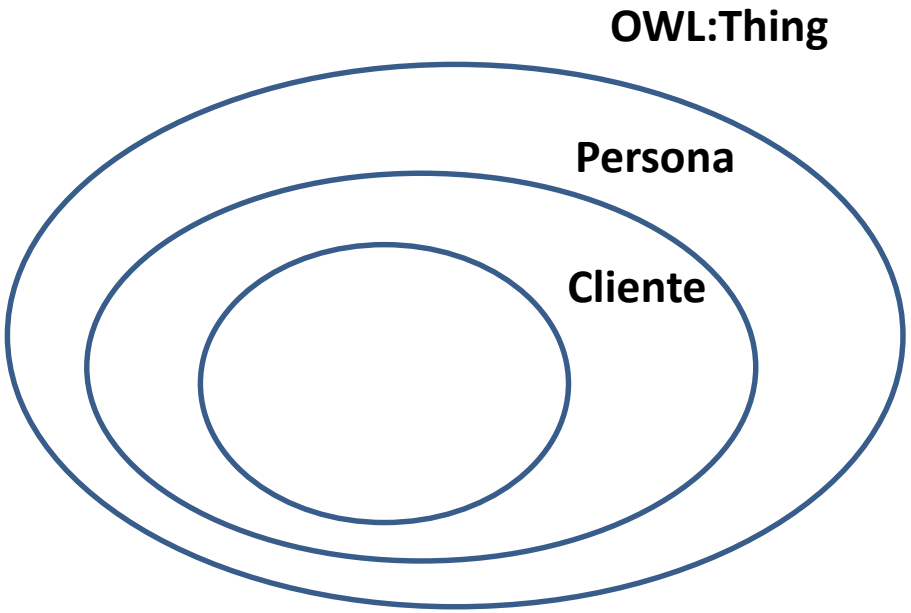
- Superclase/subclase son relaciones de tipo “es un(a)”
- **Todos** los miembros de una subclase son miembros de su superclase



- Comida subsume Pizza
- Comida es una superclase de pizza.
- Pizza es una subclase de Comida
- Todos los miembros de Pizza son también miembros de Comida
- Todo es miembro de OWL:Thing

Clases: Inclusión

- Superclase/subclase son relaciones de tipo “es un(a)”
- **Todos** los miembros de una subclase son miembros de su superclase

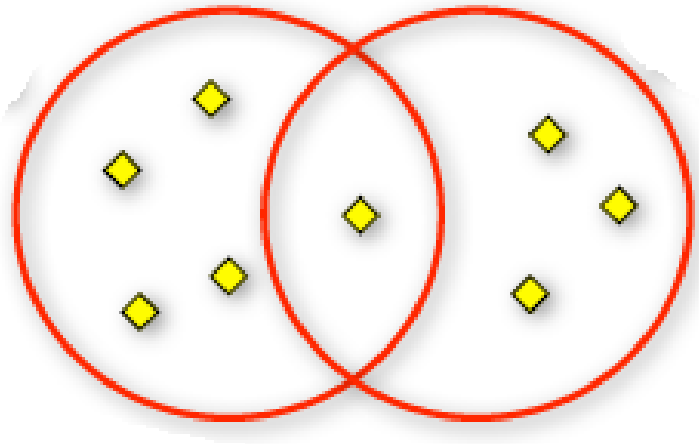


Cliente $\subseteq$ Persona

Persona $\subseteq$ OWL:Thing

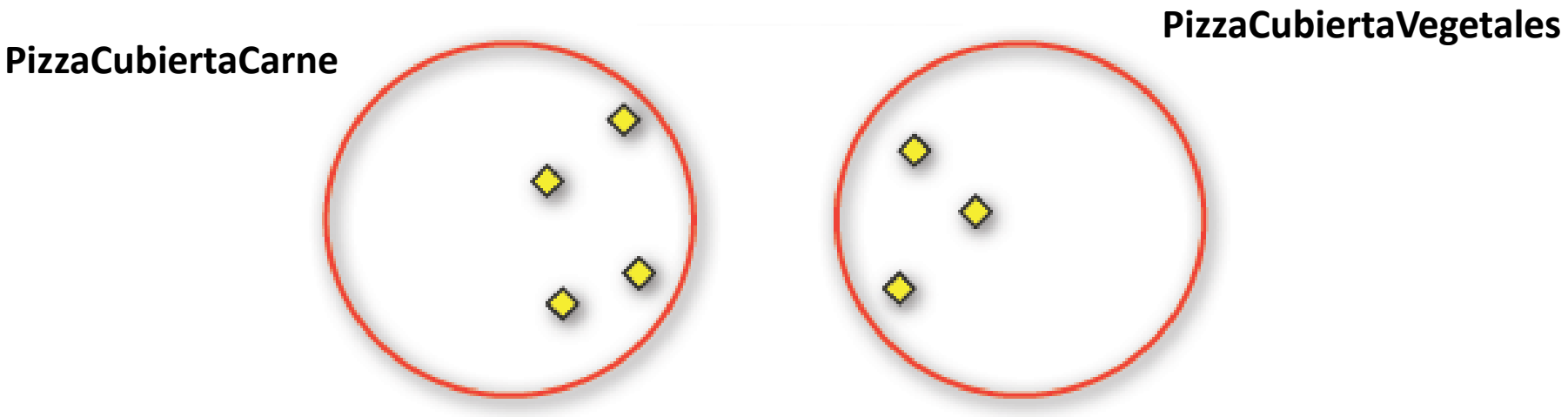
Clases: Disjuntas

- Las clases en OWL se asume que se "superponen"
  - Los individuos de una clase A también pueden ser individuos de la clase B
  - **No se puede suponer que un individuo no es miembro de una clase en particular simplemente porque no se ha declarado que sea miembro de esa clase**.



Clases: Disjuntas

- Indicar que dos clases son disjuntos significa que por ejemplo PizzaCubiertaCarne y PizzaCubiertaVegetales no pueden ser lo mismo al mismo tiempo.

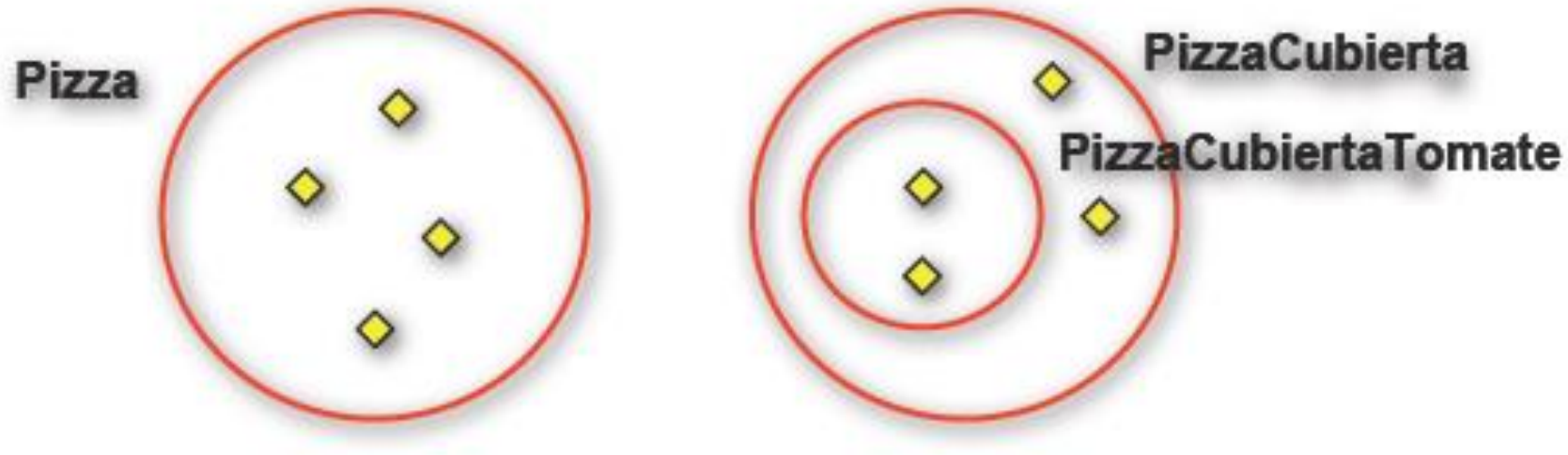


PizzaCubiertaCarne no puede nunca ser una subclase de **PizzaCubiertaVegetales** (y vice-versa)

Esto ayuda ha encontrar errores

Clases: Disjuntas

Las clases disjuntas se heredan de la jerarquía de inclusión.



$$\text{Pizza} \sqcap \text{PizzaCubierta} \sqsubseteq \bot$$

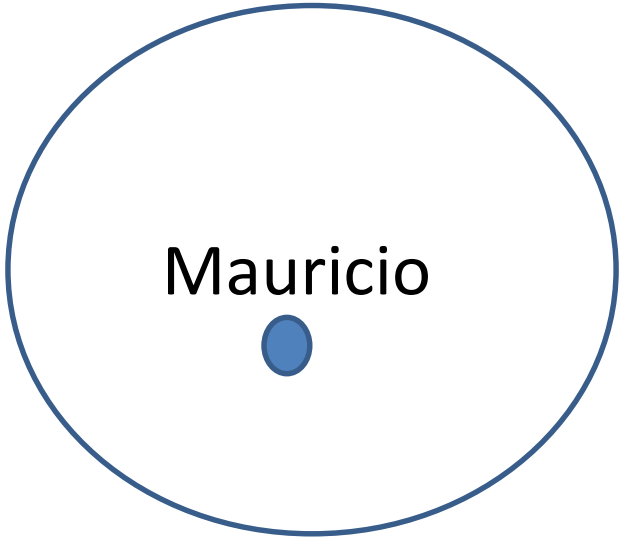
Algo que es un ***PizzaCubiertaTomate*** no puede ser una ***Pizza*** porque su superclase, ***PizzaCubierta***, es disjunto a ***Pizza***

Miembros de una Clase: Instancias

- rdf:type, que relaciona el recurso con su clase (el recurso se declara como una instancia de esa clase)

- ex:mauricio rdf:type ex:Persona .
- ex:mauricio a ex:Persona .

- Persona(Mauricio)



Inicio Tutorial 1 –OWL
Agregando Clases

Ontología sobre Juegos de Video

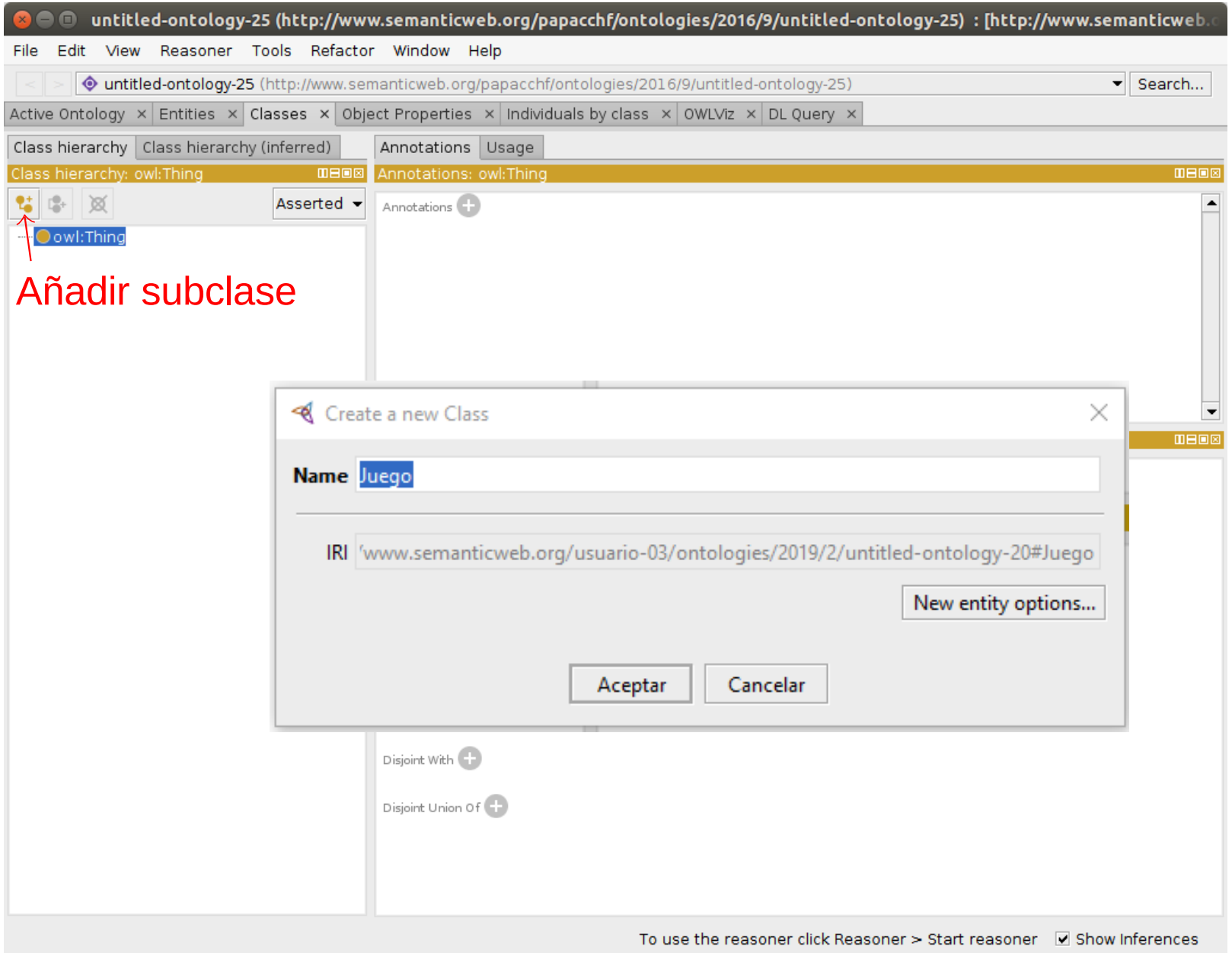
Supongamos que queremos construir una ontología sobre videojuegos de la siguiente manera.

Clases	Clases	Propiedades	definiciones

- Juego
  - JuegoFamoso
  - LoL
  - Ajedrez
  - Sudoku
- Plataforma
  - Windows
  - MacOSX
  - Linux
	- TipoJuego
	  - UnSoloJugador
	  - MultiJugador
	  - DeRoles
	  - Enlinea
	- DificultadJuego
	  - Difícil
	  - Normal
	  - Fácil
		tieneDificultad
		tienePlataforma
		tieneTipo
			JuegoMultiPlataforma
			JuegoDificil
			JuegoNormal
			JuegoFacil
			JuegoParaLinux
			JuegoParaWindows
			JuegoParaMacOSX
			. . .

Agregando clases

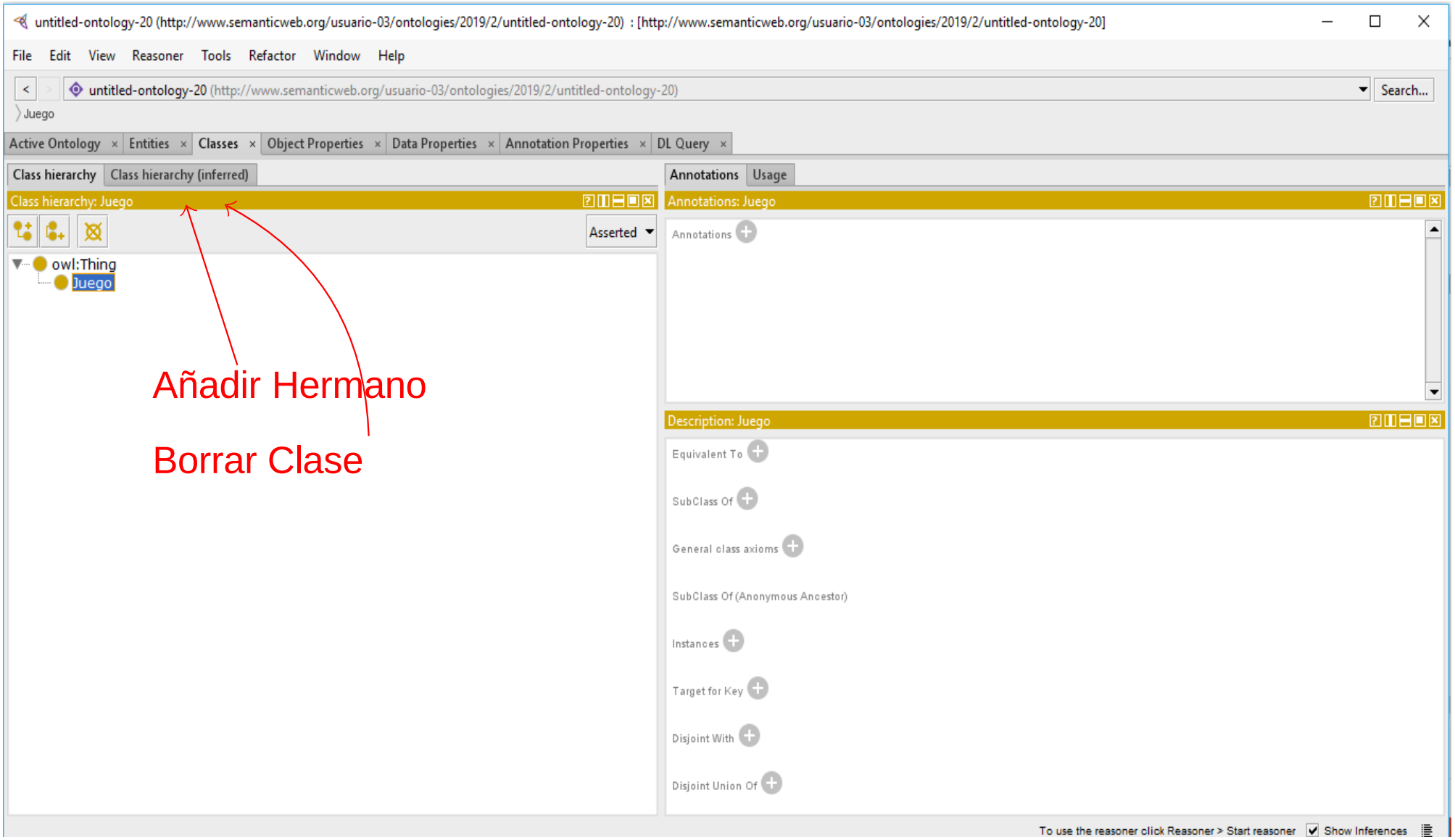
Asegúrese de tener abierta la pestaña "Classes" Window → Tabs → Classes



Agregando clases

Asegúrese de tener abierta la pestaña "Classes“

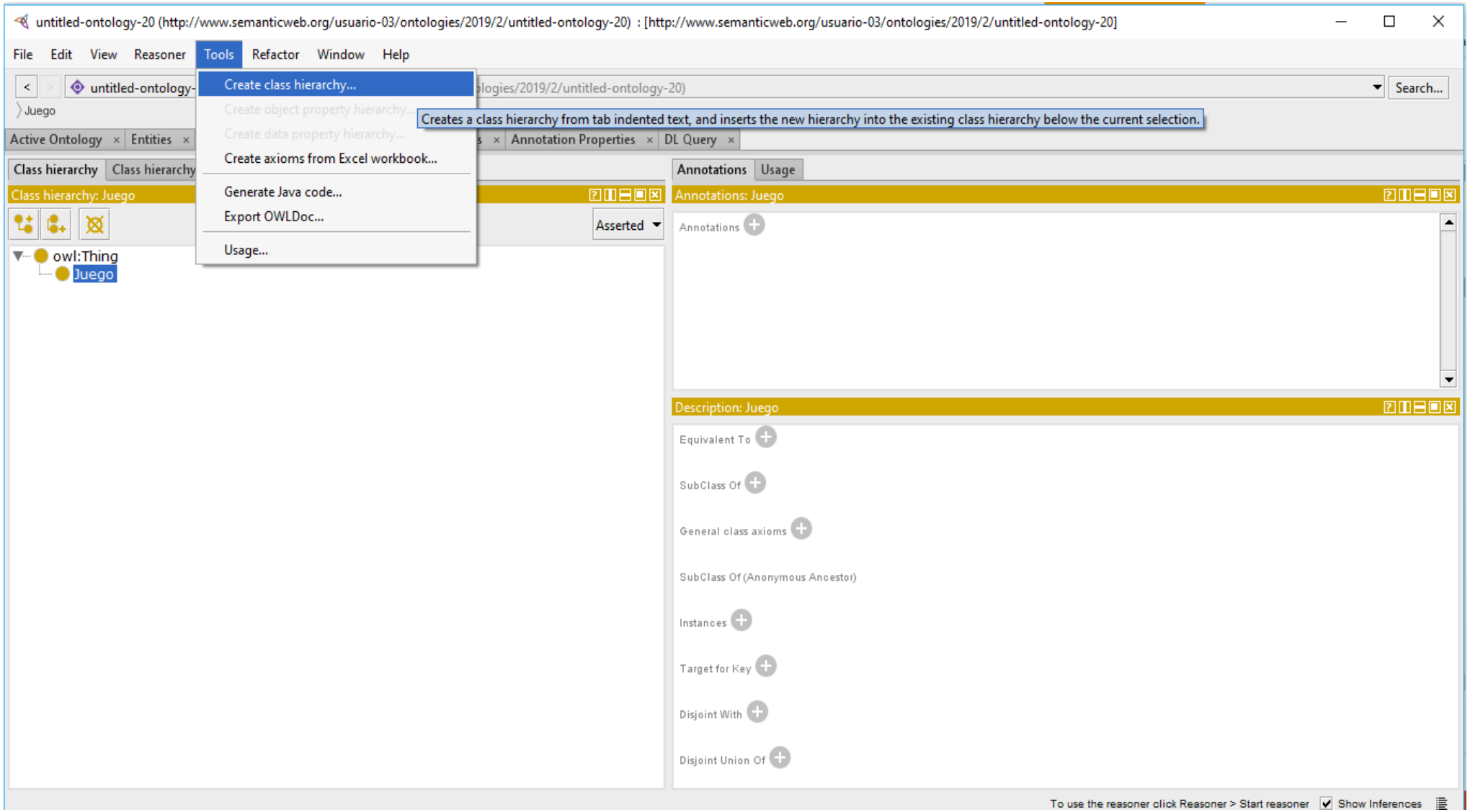
Window → Tabs → Classes



Agregando Jerarquía de Clases

Permite acelerar el proceso de añadir clases.

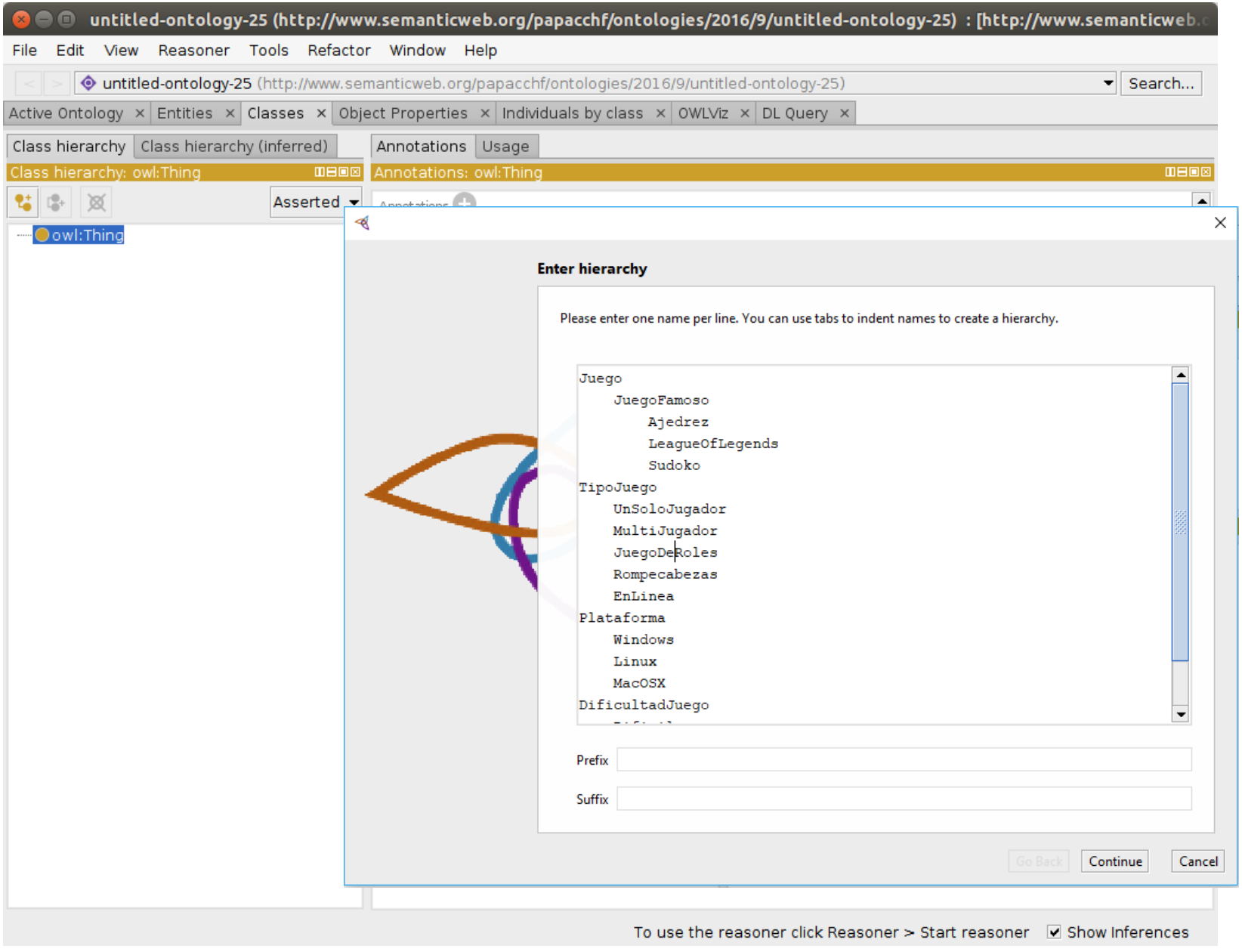
Tools → Create class hierarchy. . .



Agregando Jerarquía de Clases

Permite acelerar el proceso de añadir clases.

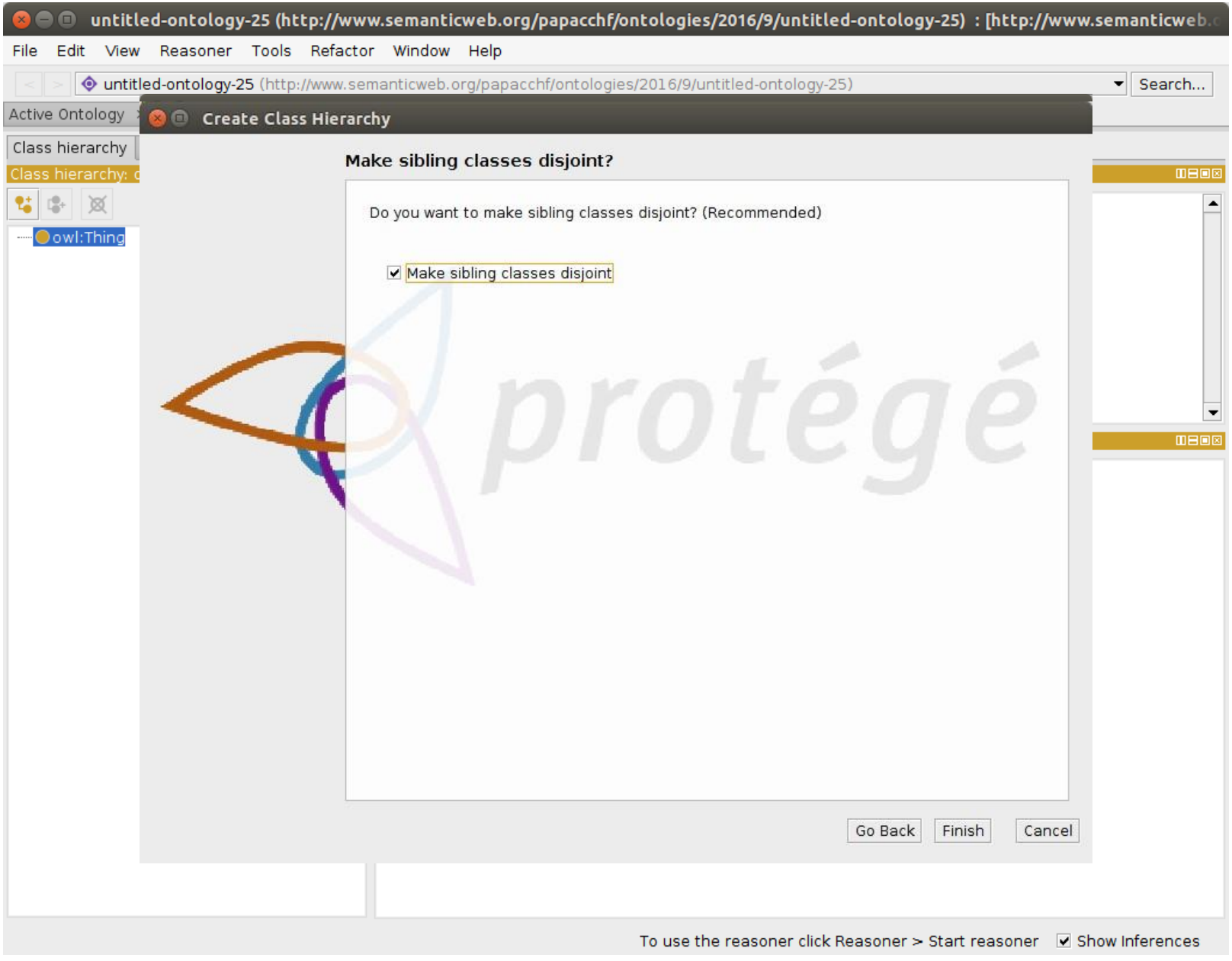
Tools → Create class hierarchy. . .



Agregando Jerarquía de Clases

Permite acelerar el proceso de añadir clases.

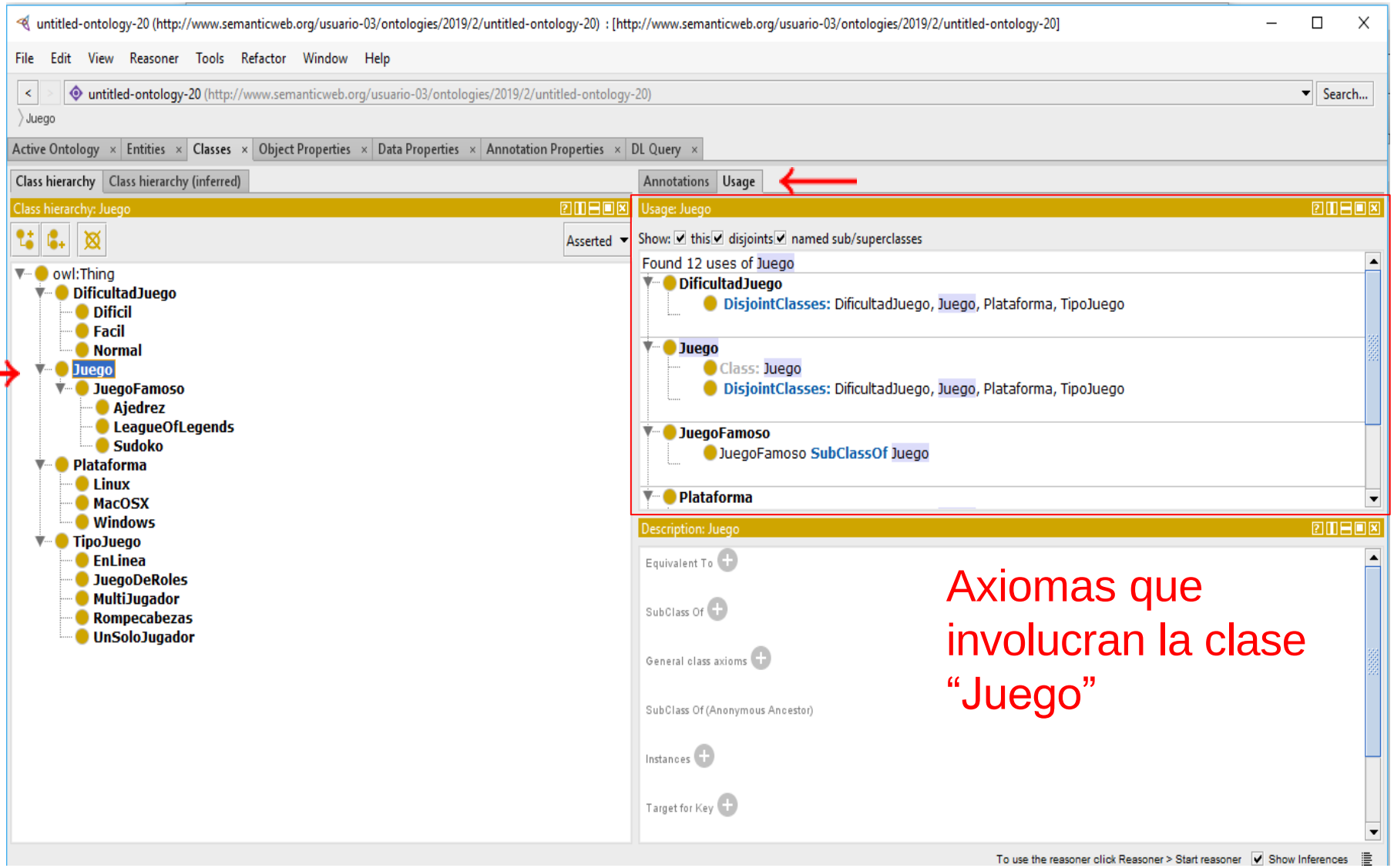
Tools → Create class hierarchy. . .



Agregando Jerarquía de Clases

Esto permite observar el uso de una clase particular.

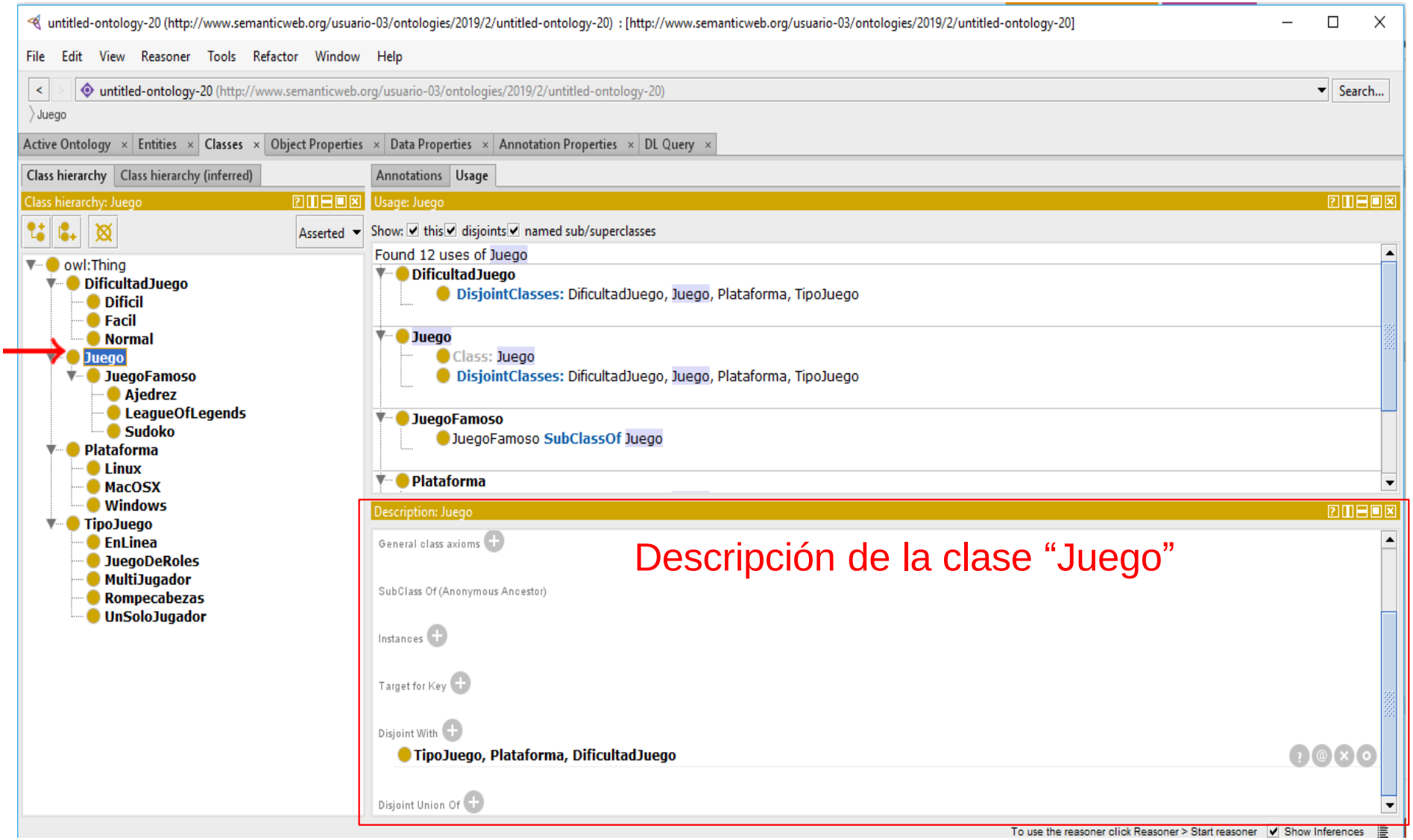
Class hierarchy → Usage . . .



Agregando Jerarquía de Clases

Es posible observar la descripción de una clase.

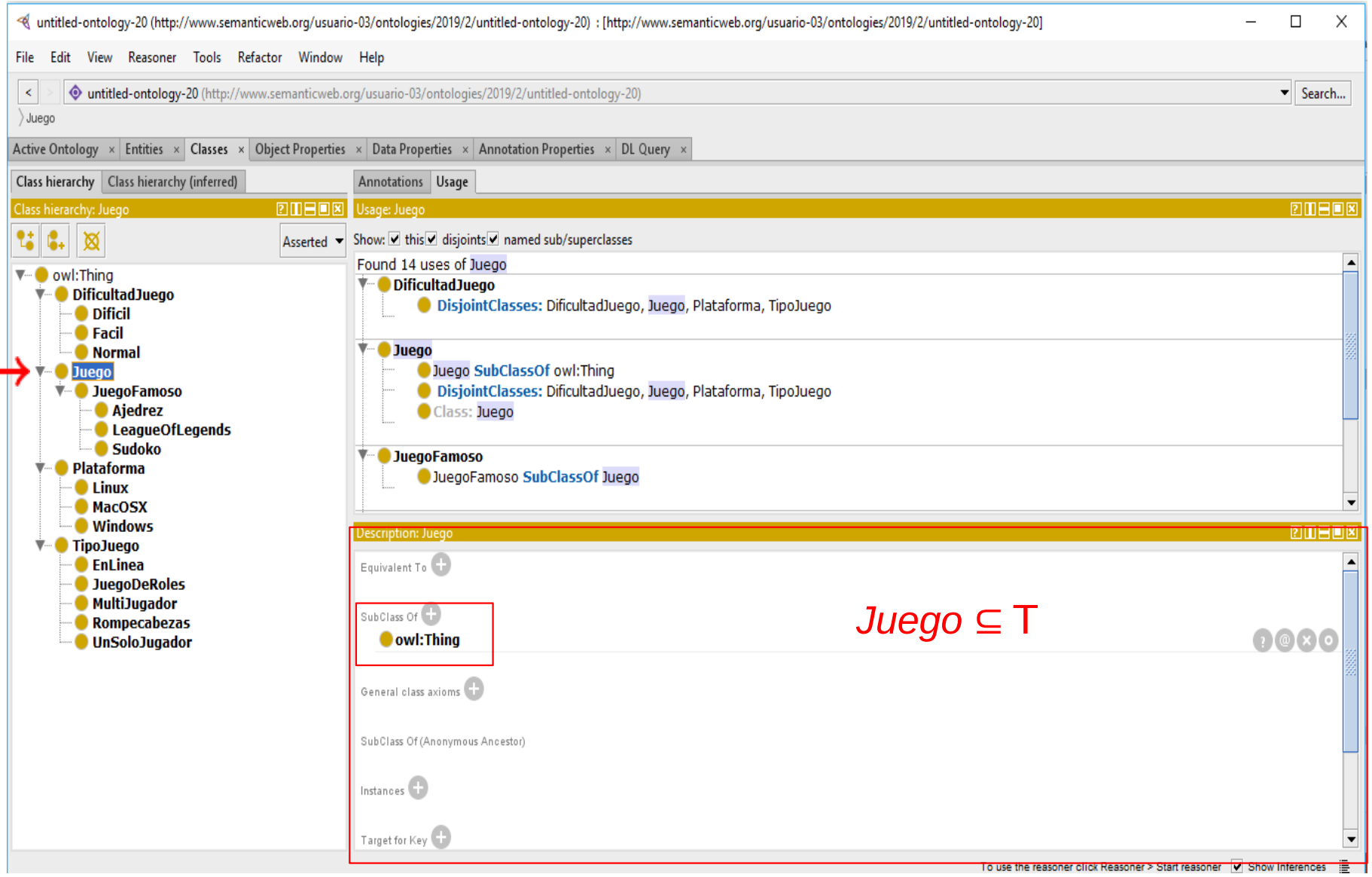
Class hierarchy → Usage . . .



Agregando Jerarquía de Clases

Es posible observar la descripción de una clase.

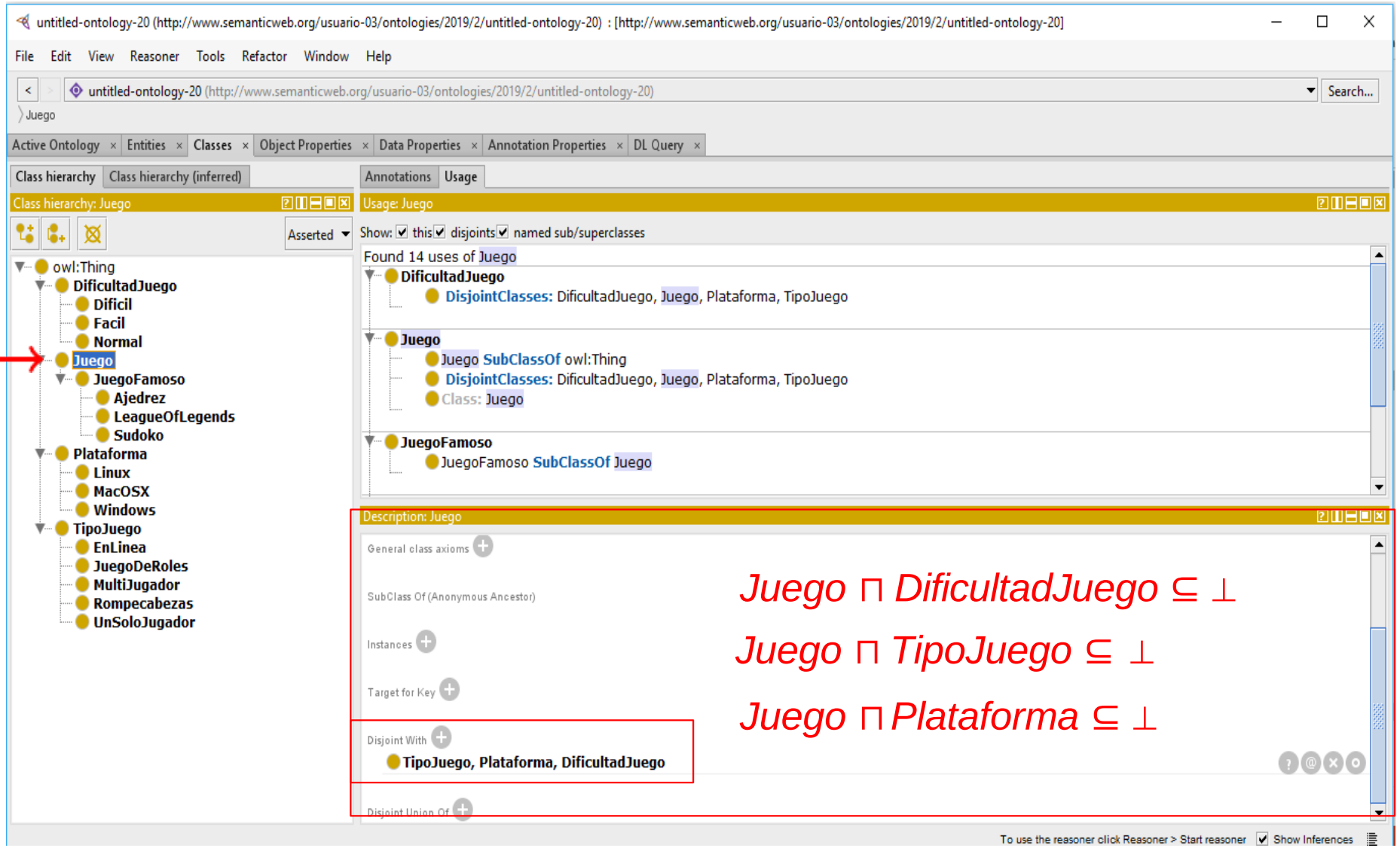
Class hierarchy → Usage . . .



Agregando Jerarquía de Clases

Es posible observar la descripción de una clase.

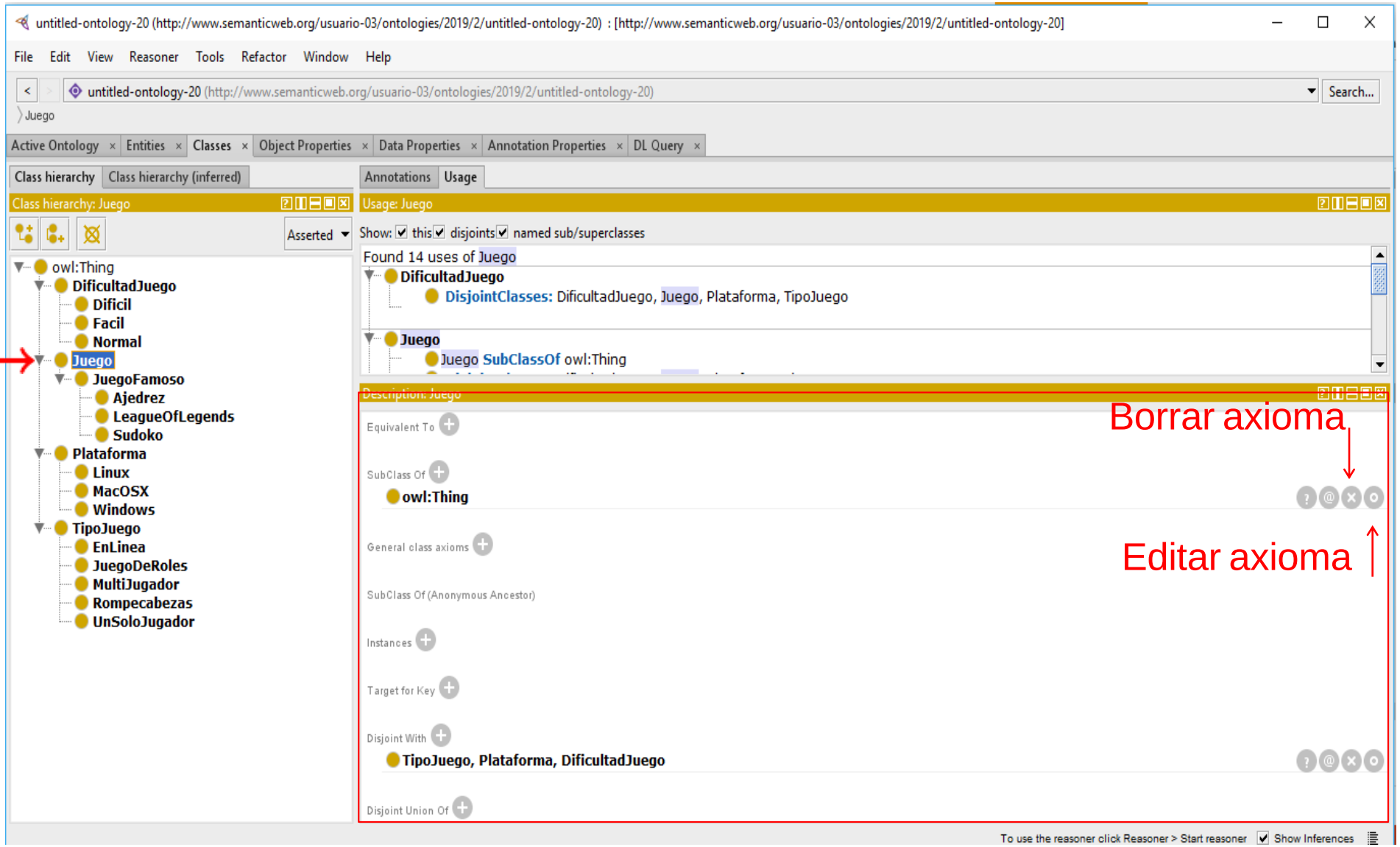
Class hierarchy → Usage . . .



Agregando Jerarquía de Clases

Es posible observar la descripción de una clase.

Class hierarchy → Usage . . .



Fin Tutorial 1 –OWL Agregando Clases

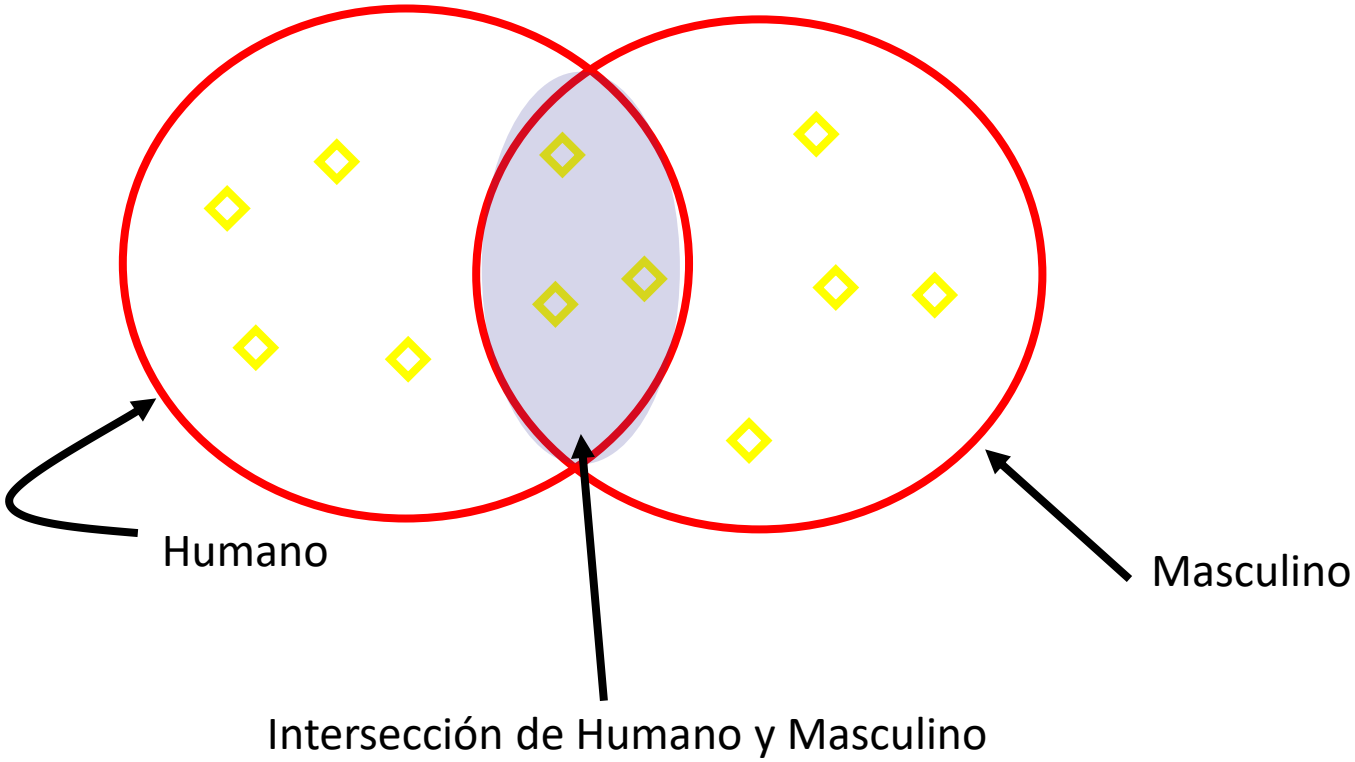
Clases Complejas en OWL

Clases Complejas

- (OWL DL) provee constructores con los cuales se pueden formar clases basadas en operaciones básicas de conjuntos:
  - Intersección
  - Union
  - Complemento
- Clases Disjuntas
- ….

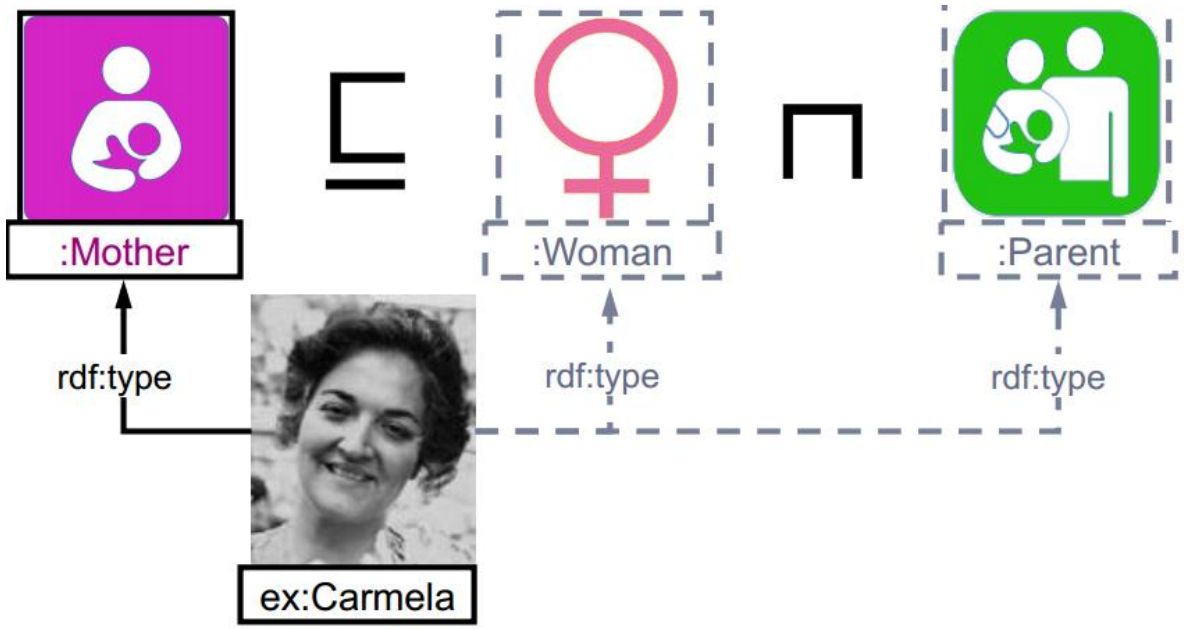
Clases Complejas: Intersección de Clases

- Las instancias/individuos de la intersección de dos clases son simultáneamente instancias de ambas clases
- owl:intersectionOf



Clases Complejas: Intersección de Clases

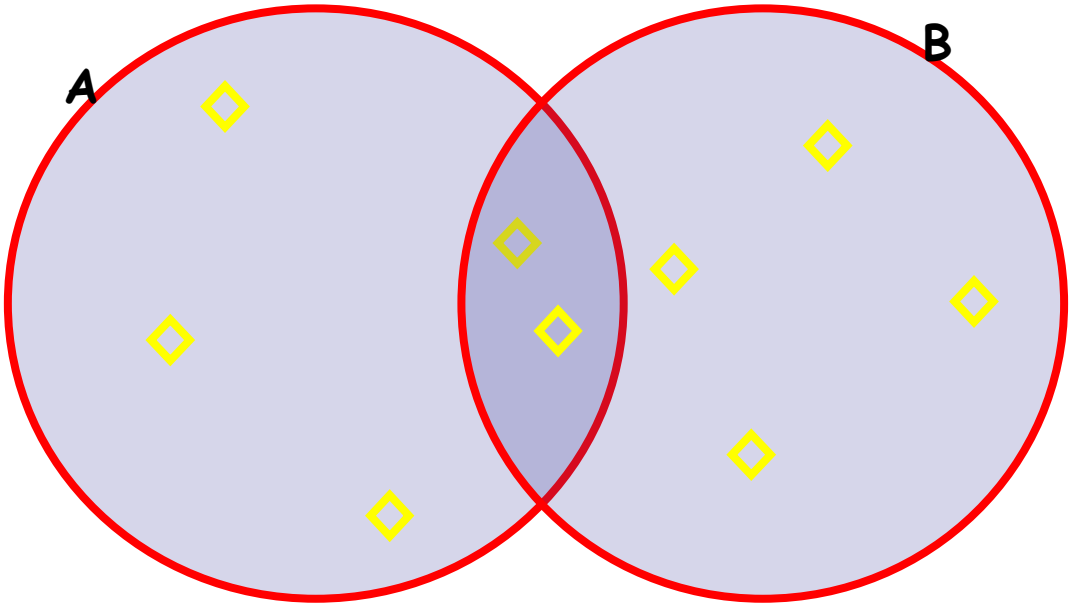
- Ejemplo: owl:intersectionOf



ex:Carmela rdf:type :Mother .
:Mother rdfs:subClassOf [owl:intersectionOf (:Woman :Parent)]
ex:Carmela rdf:type :Woman , :Parent .

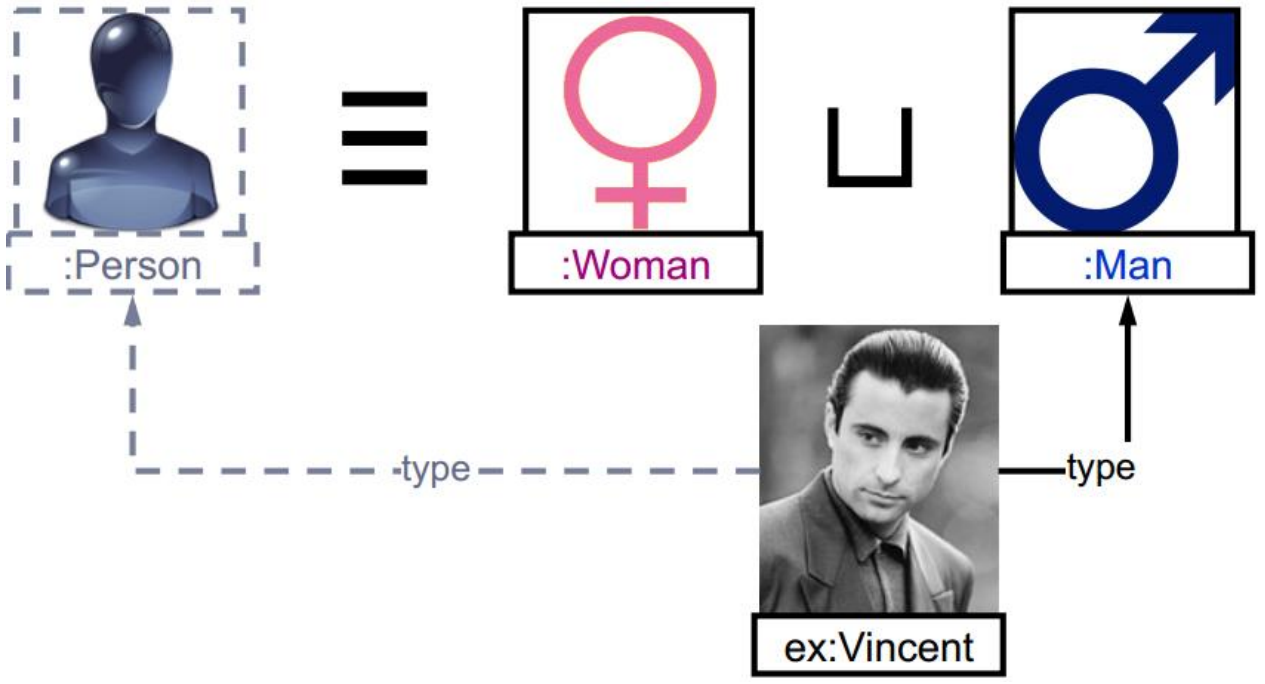
Clases Complejas: Unión de Clases

- Instancias/Individuos de la unión de dos clases son la instancia de una o ambas clases.
- owl:unionOf



Clases Complejas: Unión de Clases

- Ejemplo: owl:unionOf



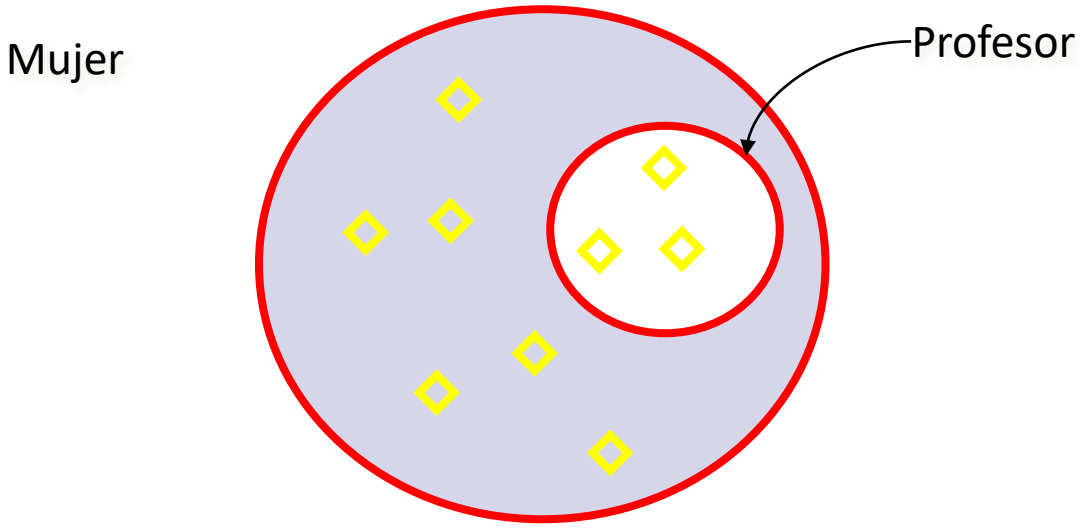
ex:Vincent rdf:type :Man .

:Person owl:equivalentClass [owl:unionOf (:Woman :Man)]

ex:Vincent rdf:type :Person .

Clases Complejas: Complemento

Una clase complemento se especifica negando otra clase. Contendrá los individuos que no están en la clase negada.



Mujer and (not Profesor)

Clases Complejas: Clases Enumeradas

- OWL provee la forma de especificar una clase vía una enumeración directa de sus miembros
  - **`owl:oneOf`**.
- Ejemplo:

```
<owl:Class rdf:ID="ColorVino">
        <rdfs:subClassOf rdf:resource="#DescripcionVino"/>
        <owl:oneOf rdf:parseType="Collection">
                <WineColor rdf:about="#Blanco" />
                <WineColor rdf:about="#Rosa" />
                <WineColor rdf:about="#Rojo" />
        </owl:oneOf>
</owl:Class>
```

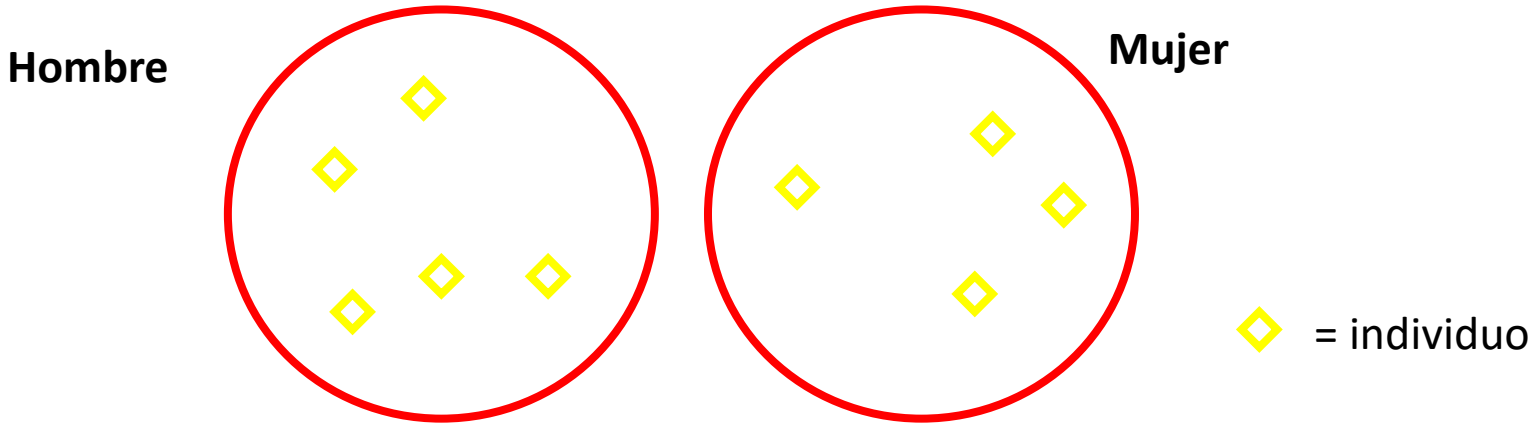
Clases Complejas: Clases Enumeradas

- **`Ejemplo: owl:oneOf`**

:DonCorleone owl:equivalentClass
[owl:oneOf (ex:Vito ex:Michael ex:Vincent)]
ex:Vito rdf:type :DonCorleone .
ex:Michael rdf:type :DonCorleone .
ex:Vincent rdf:type :DonCorleone .

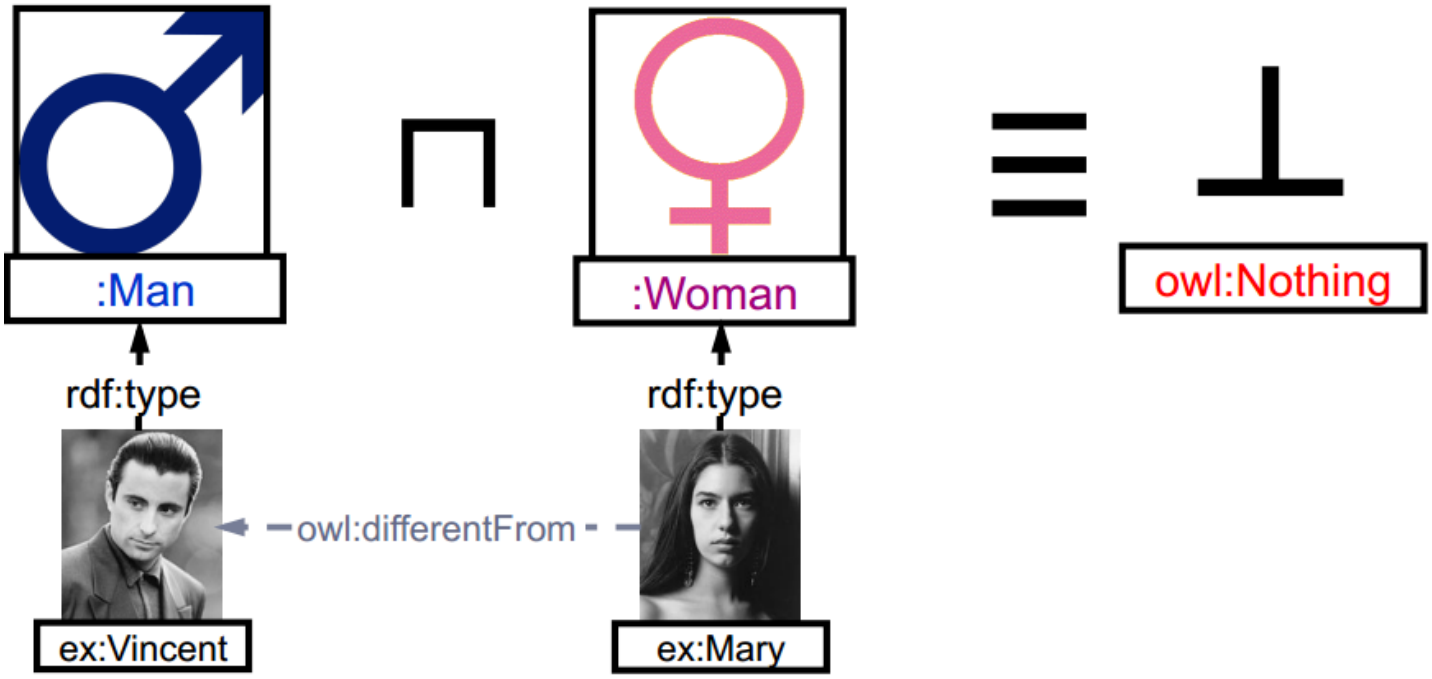
Clases Complejas: Clases Disjuntas

- OWL asume que las clases se superponen
- La desunión de un conjunto de clases puede ser expresado usando **`owl:disjointWith`**
  - Un individuo que es miembro de una clase no puede ser simultáneamente una instancia de otra.



Clases Complejas: Clases Disjuntas

- Ejemplo **`owl:disjointWith`**



ex:Vincent rdf:type :Man . ex:Mary rdf:type :Woman .
:Man owl:disjointWith :Woman .
ex:Mary owl:differentFrom ex:Vincent .