OWL Web Ontology Language

Contenido

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

Propiedades

- Dos tipos de propiedades:
  - Propiedad Objeto: `objeto propiedad objeto`

    `owl:ObjectProperty`
  - Propiedad TipoDato: `objeto propiedad valor`

    `owl:DatatypeProperty`

- Estos se hicieron por separado para facilitar la creación de razonamientos sólidos y completos.

- Una propiedad tiene algunas "características" que ayudan al razonamiento:
  - **`subPropertyOf:`** **Define jerarquía de propiedades**
  - **`domain:`** **Define el sujeto de la tripleta**
  - **`range:`** **Define el objeto de la tripleta**

Propiedades

- ObjectProperty: Relaciona dos objetos

```
ex:dicta rdf:type owl:ObjectProperty .
ex:dicta rdfs:domain ex:Profesor .
ex:dicta rdfs:range ex:exCurso .
```

- Datatype Properties: Relaciona un objeto con un valor

```
ex:edad rdf:type owl:DatatypeProperty .
ex:edad rdfs:domain ex:Persona .
ex:edad rdfs:range xsd:nonNegativeInteger .
```

Propiedades

- Las propiedades ObjectProperty y Datatype pueden declararse de manera opcional con dominio y rango

Sin dominio o sin declaración de rango:

- si no se ha hecho ninguna declaración de rango para la propiedad P, entonces no se ha dicho nada sobre los valores de esta propiedad. De manera similar para ninguna declaración de dominio.

!No es posible inferir nada!

Propiedades

- Ejemplo

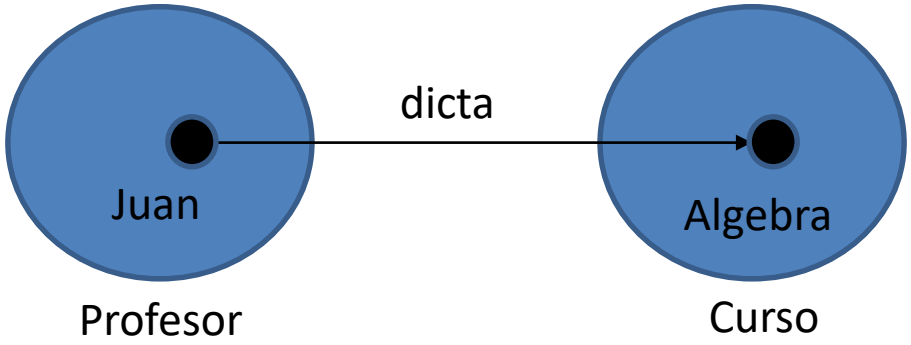
```
ex:dicta rdf:type owl:ObjectProperty .
ex:dicta rdfs:domain ex:Profesor .
ex:dicta rdfs:range ex:exCurso .
ex:Juan ex:dicta ex:Algebra .
```

Intencional

Extensional

- Infiere

```
ex:Juan rdf:type ex:Profesor .
ex:Algebra rdf:type ex:Curso .
```



Propiedades

- Ejemplo

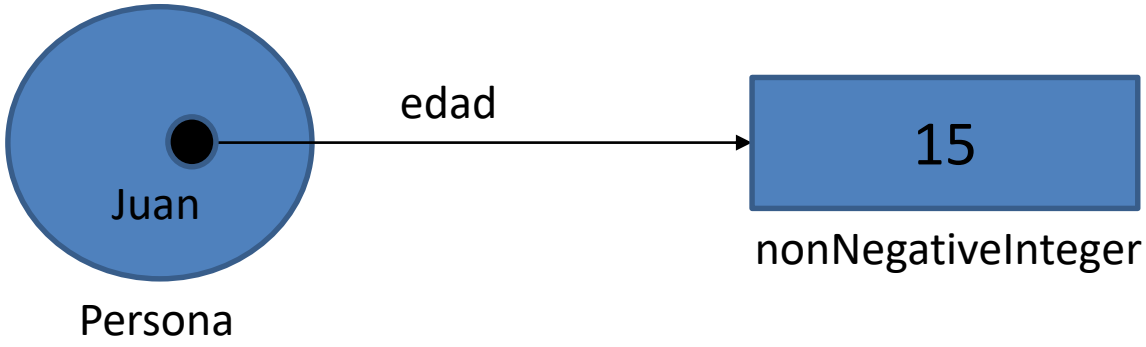
```
ex:edad rdf:type owl:DatatypeProperty .
ex:edad rdfs:domain ex:Persona .
ex:edad rdfs:range xsd:nonNegativeInteger .
ex:Juan ex:edad 15 .
```

Intencional

Extensional

- Infiere

```
ex:Juan rdf:type ex:Persona .
```



Jerarquía de Propiedades: Subpropiedades

- Qué significa una subpropiedad?

- Ejemplo

```
ex:tieneGato rdf:type owl:ObjectProperty .
ex:tieneGato rdfs:domain :Persona .
ex:tieneGato rdfs:range :Animal .
ex:tieneGato rdfs:subPropertyOf ex:tieneMascota .
```

Inicio Tutorial 1 – OWL
Creación de Propiedades

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
			. . .

Ahora qué?

Hasta el momento se ha creado

- Una jerarquía inicial de clases
- Axiomas de desunión (disjoint) básicos (entre hermanos)

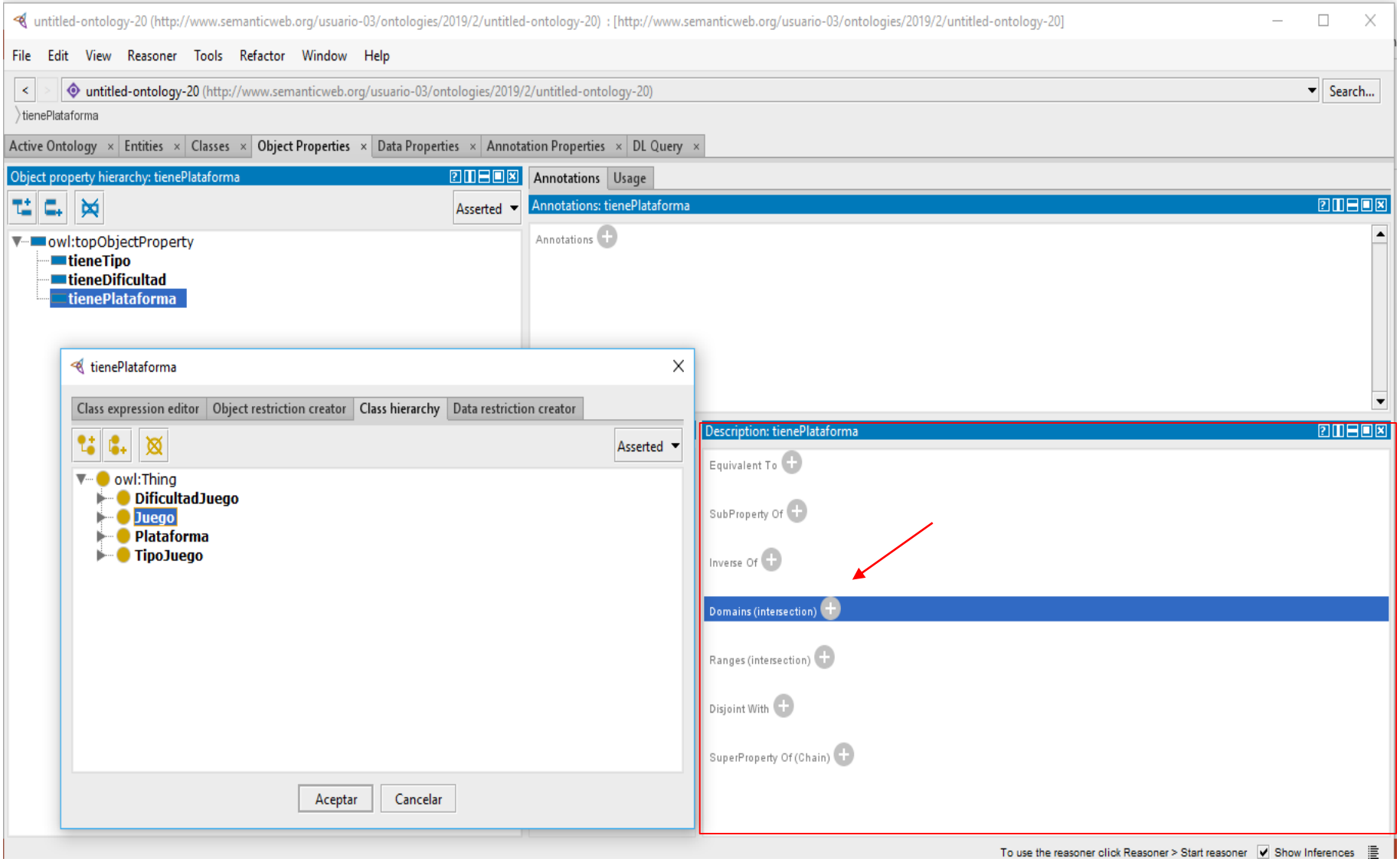
Qué se requiere agregar….

- **relaciones entre clases (propiedades)**

Object Properties (Domain and Range)

Asegúrese de tener la pestaña “Object Properties” abierta

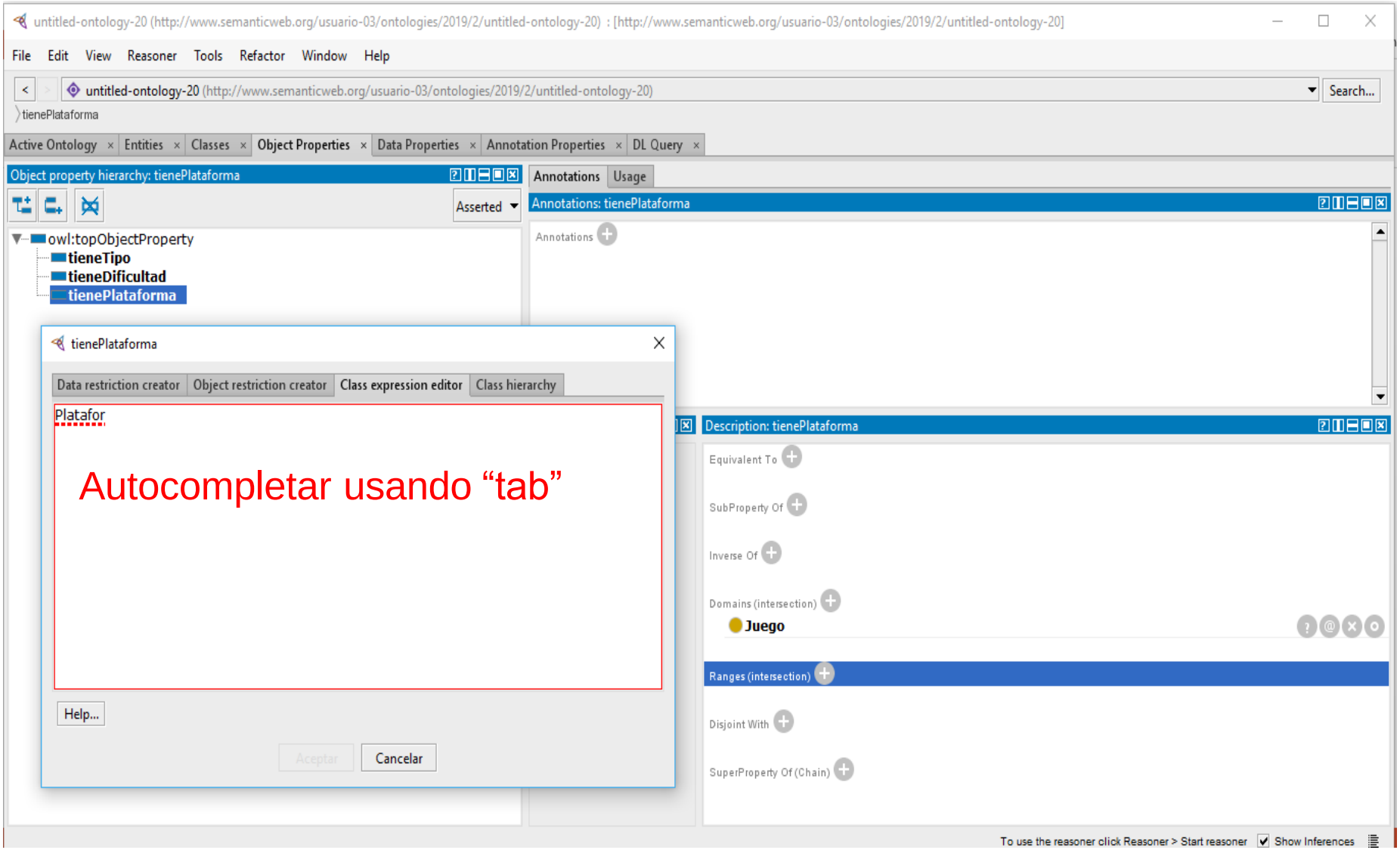
Window → Tabs → Object Properties



Object Properties (Domain and Range)

Asegúrese de tener la pestaña “Object Properties” abierta

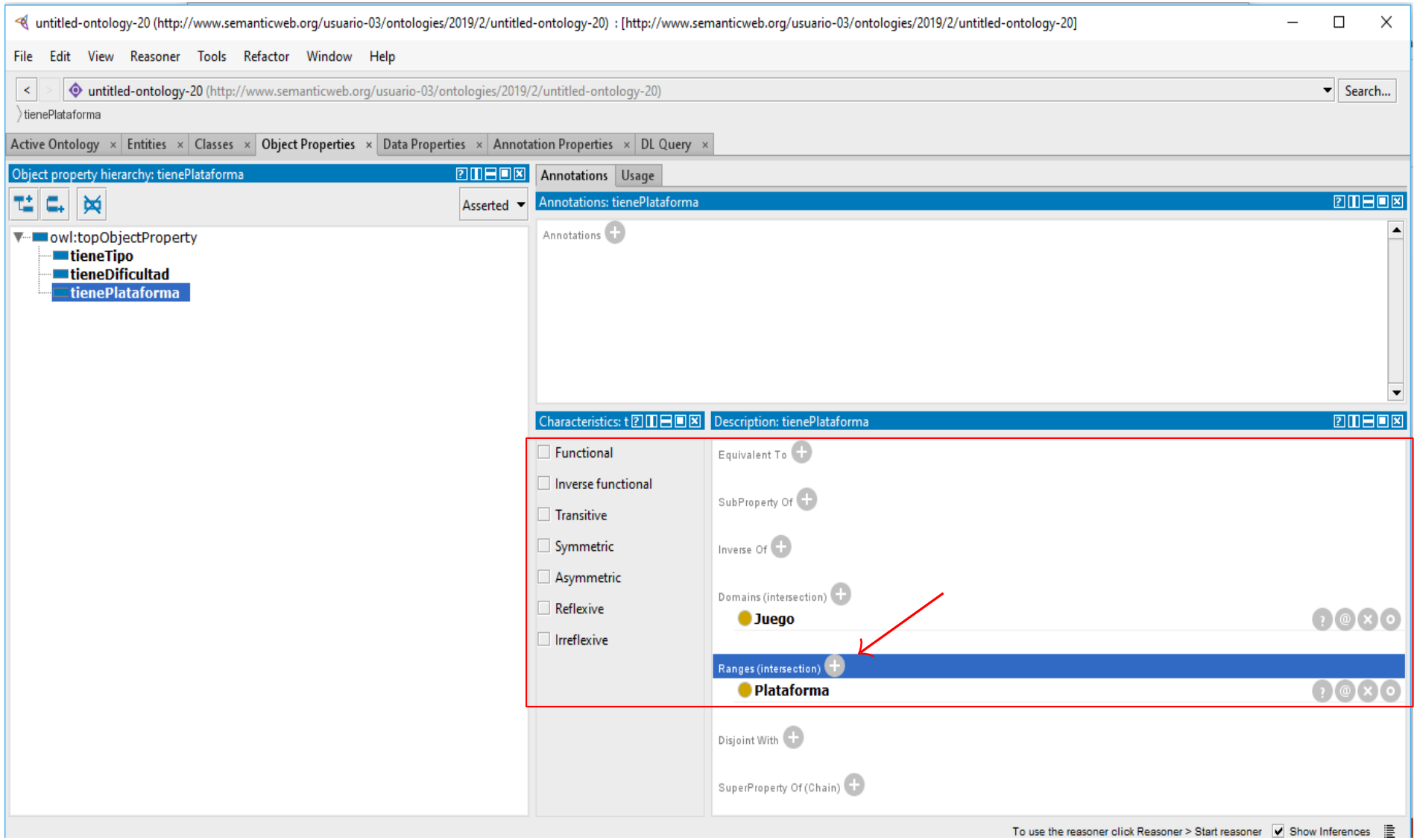
Window → Tabs → Object Properties



Object Properties (Domain and Range)

Asegúrese de tener la pestaña “Object Properties” abierta

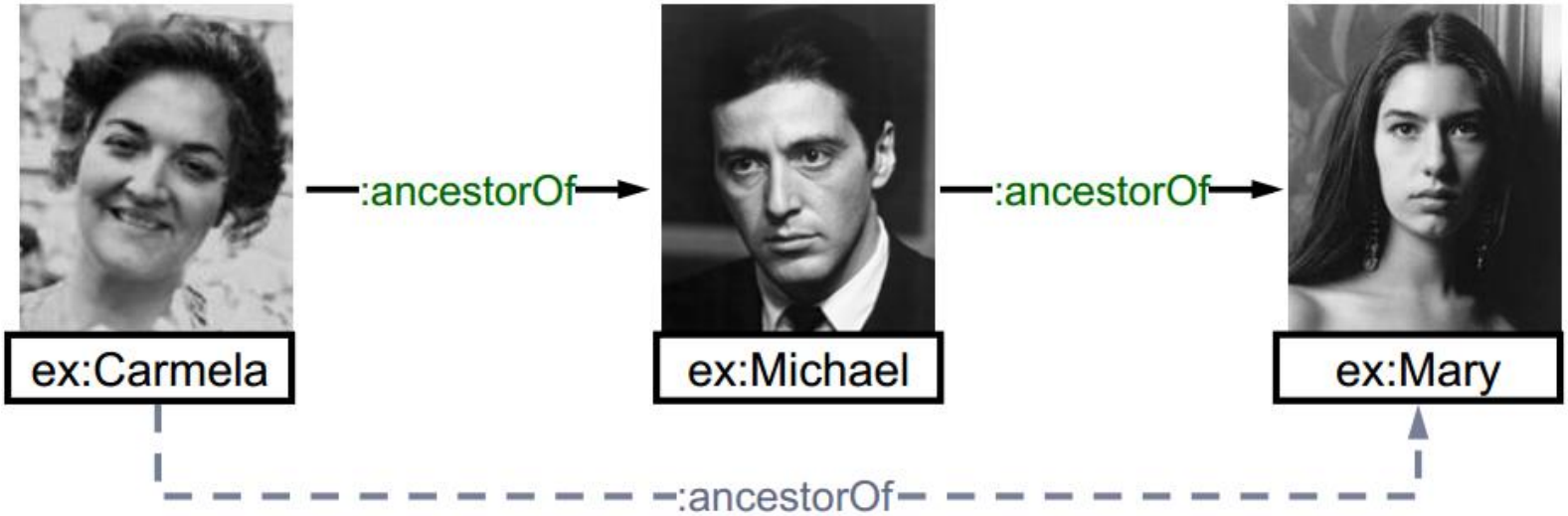
Window → Tabs → Object Properties



Fin Tutorial 1 – OWL
Creación de Propiedades

Características Propiedades Objeto (1/5)

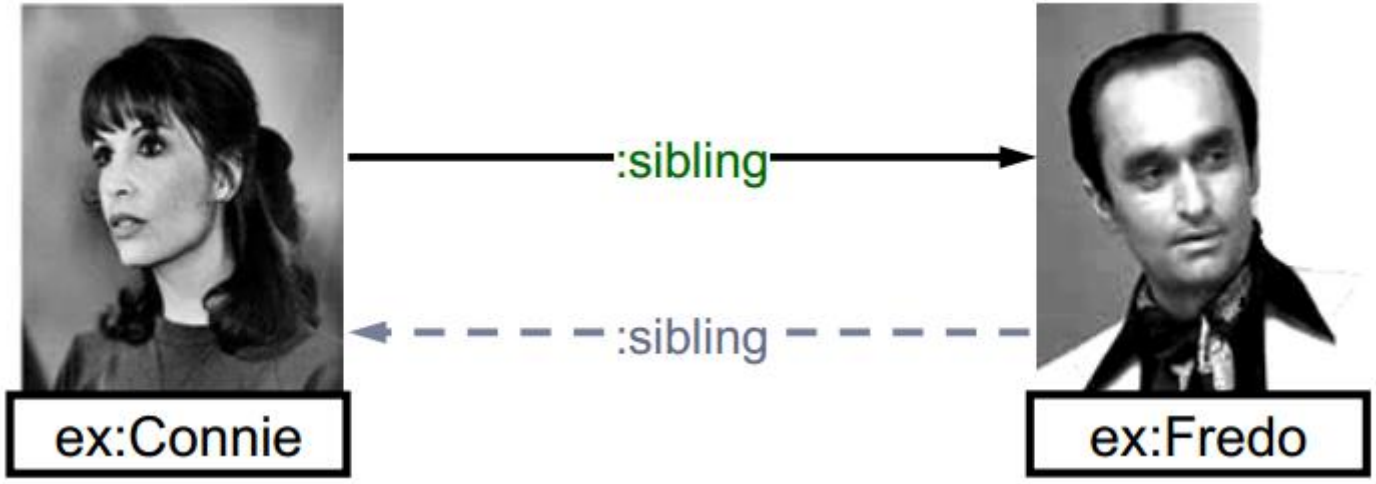
- Propiedad Transitiva
  - **P(x,y) y P(y,z) → P(x,z)**
- **owl:TransitiveProperty**
  - e.j. “tiene major grado que”, “es ancestro de”




```
ex:Carmela :ancestorOf ex:Michael .
ex:Michael :ancestorOf ex:Mary .
:ancestorOf rdf:type owl:TransitiveProperty .
   ex:Carmela :ancestorOf ex:Mary .
```

Características Propiedades Objeto (2/5)

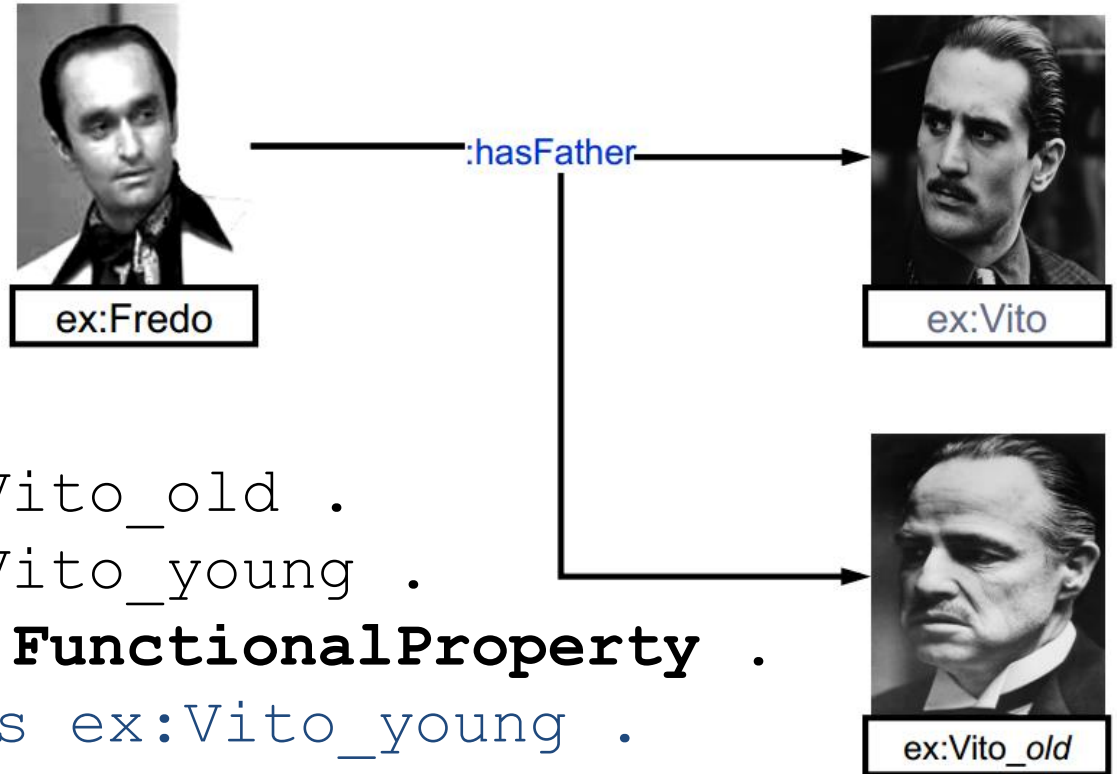
- Propiedad simétrica
  - **`P(x,y) if and only if P(y,x)`**
- **`owl:SymmetricProperty`**
  - e.j. “tiene el mismo grado que”, “es hermano de”




```
ex:Connie :sibling ex:Fredo .
:sibling rdf:type owl:SymmetricProperty .
   ex:Fredo :sibling ex:Connie .
```

Características Propiedades Objeto (3/5)

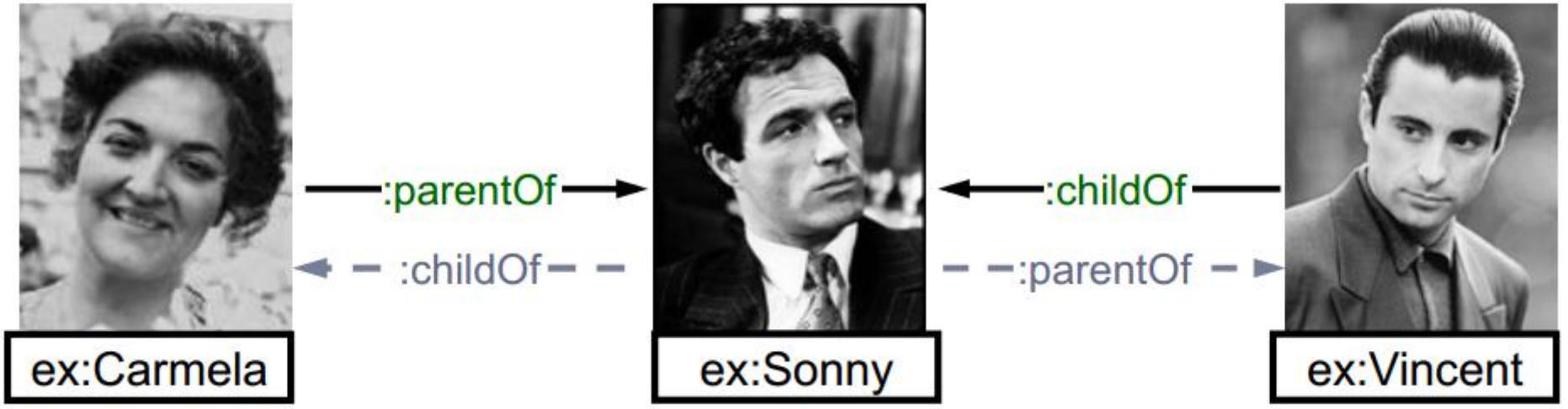
- Propiedad funcional
  - **P(x,y) y P(x,z) implica y = z**
- **owl:FunctionalProperty** define una propiedad que tiene al menos un valor por cada objeto
  - e.j. “edad”, “altura”, “SupervisorDirecto”




```
ex:Fredo :hasFather ex:Vito_old .
ex:Fredo :hasFather ex:Vito_young .
:hasFather rdf:type owl:FunctionalProperty .
   ex:Vito_old owl:sameAs ex:Vito_young .
```

Características Propiedades Objeto (4/5)

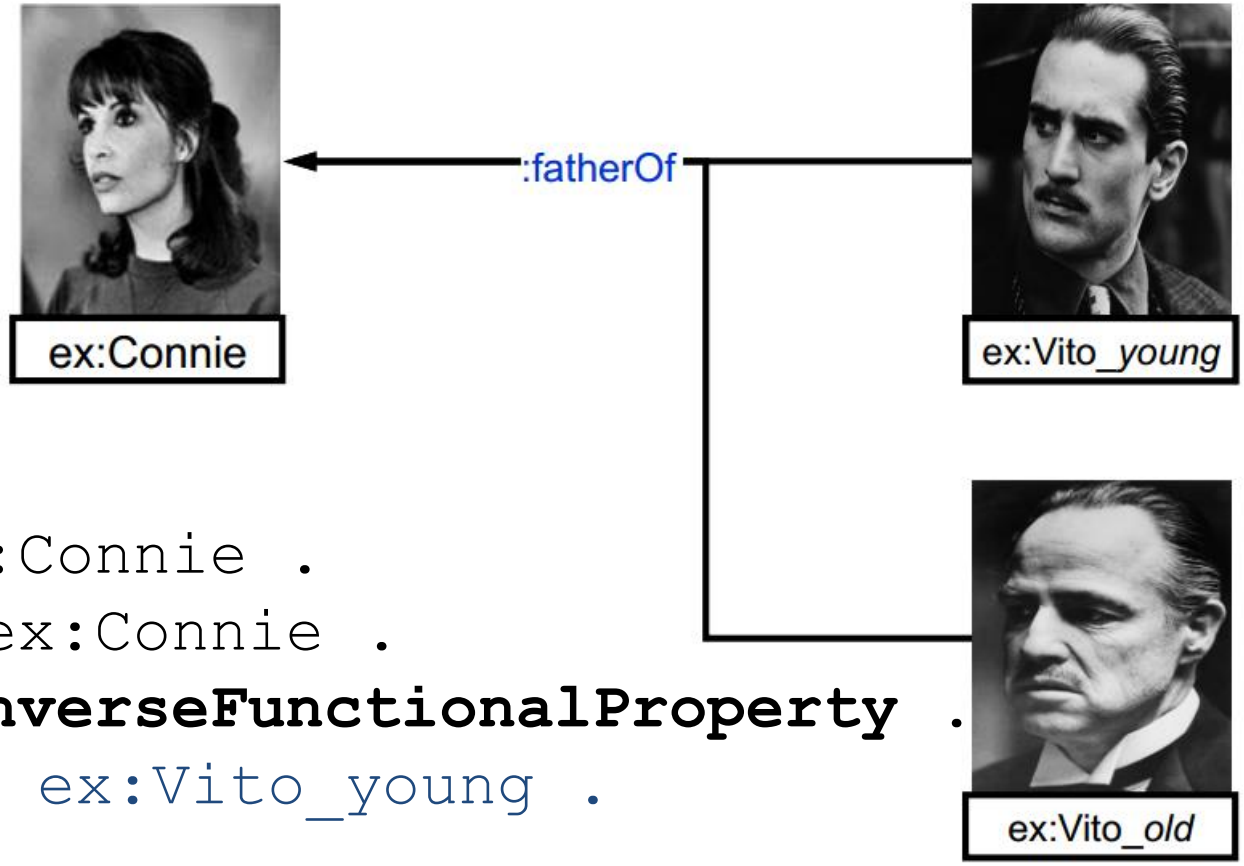
- Propiedad Inversa: **owl:inverseOf**
  - **P1(x,y) iff P2(y,x)**




```
ex:Carmela :parentOf ex:Sonny .
ex:Vincent :childOf ex:Sonny .
:parentOf owl:inverseOf :childOf .
   ex:Sonny :parentOf ex:Vincent .
   ex:Sonny :childOf ex:Carmela .
```

Características Propiedades Objeto (5/5)

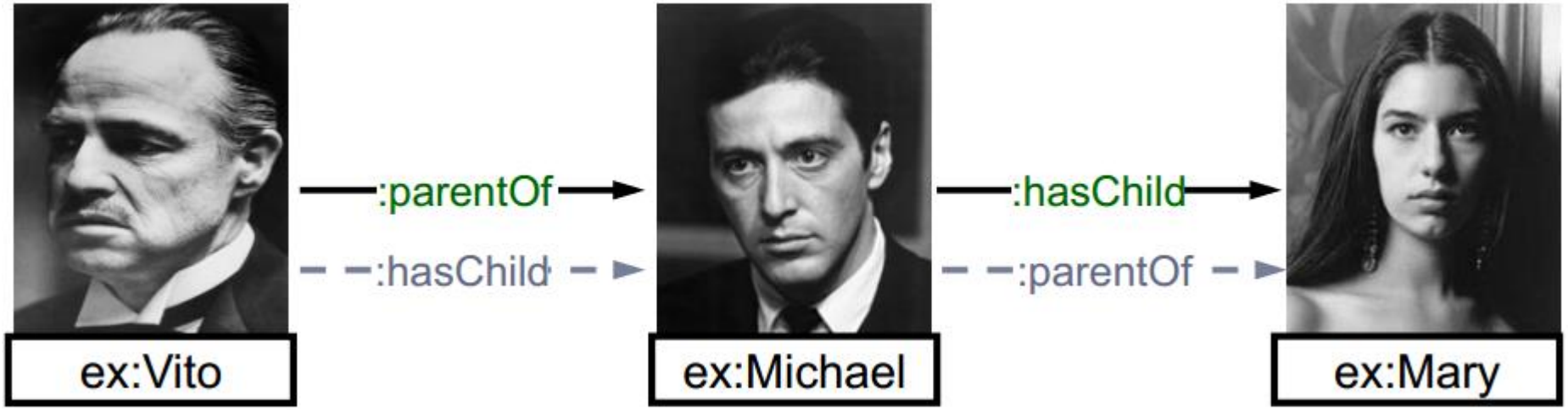
- Propiedad Inversa Funcional
  - **P(y,x) and P(z,x) implica y = z**
- owl:InverseFunctionalProperty define una propiedad para los cuales dos objetos diferentes no pueden tener el mismo valor




```
ex:Vito_old :fatherOf ex:Connie .
ex:Vito_young :fatherOf ex:Connie .
:fatherOf rdf:type owl:InverseFunctionalProperty .
   ex:Vito_old owl:sameAs ex:Vito_young .
```

Características Propiedades Objeto (5/5)

- Propiedad Equivalente: owl:EquivalentProperty




```
ex:Vito :parentOf ex:Michael .
ex:Michael :hasChild ex:Mary .
:parentOf owl:equivalentProperty :hasChild .
   ex:Vito :hasChild ex:Michael .
   ex:Michael :parentOf ex:Mary .
```