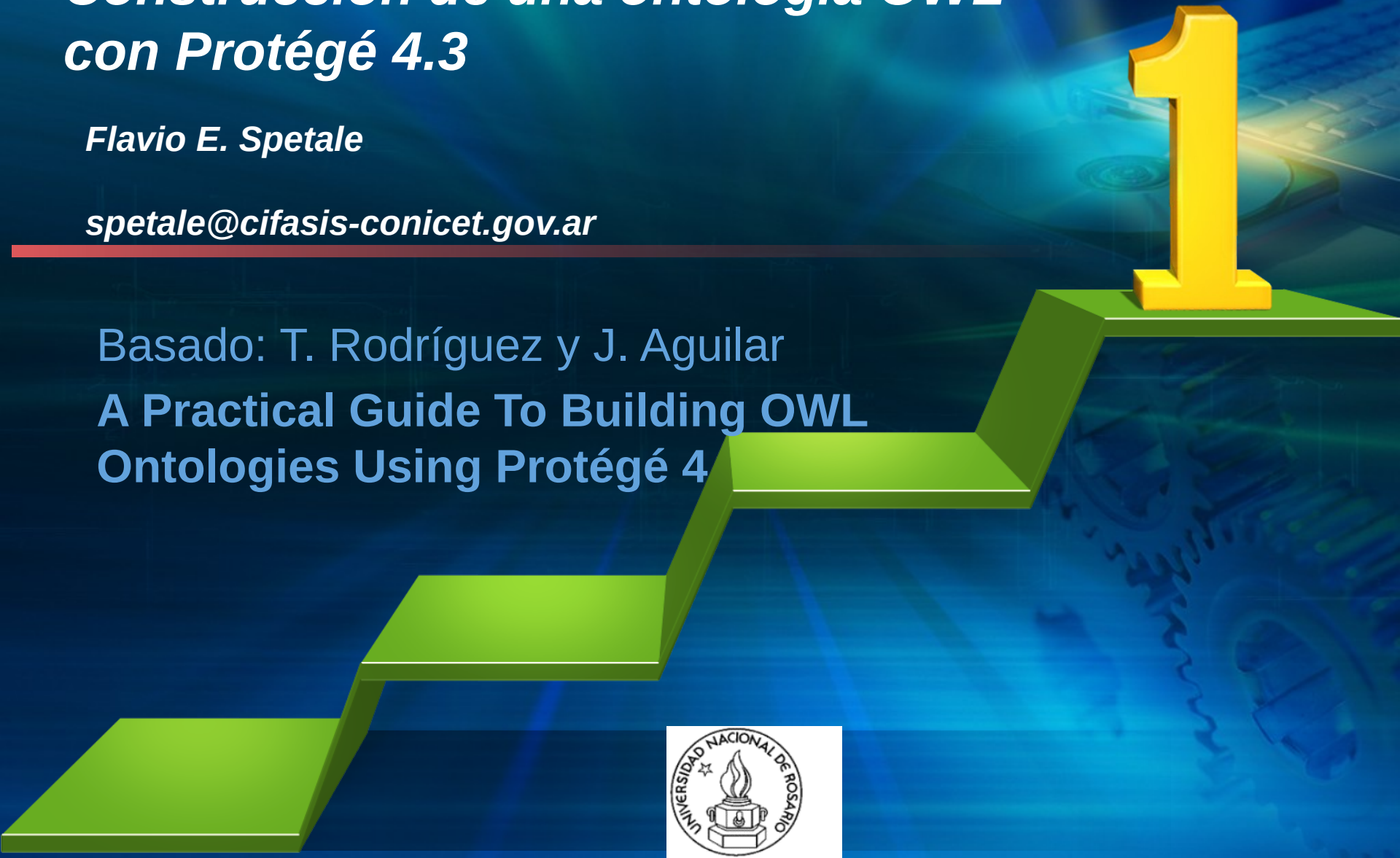
Construcción de una ontología OWL con Protégé 4.3

Flavio E. Spetale

spetale@cifasis-conicet.gov.ar



Basado: T. Rodríguez y J. Aguilar

A Practical Guide To Building OWL Ontologies Using Protégé 4

Construcción de ontología OWL

- Las ontologías son usadas para capturar el conocimiento sobre algún dominio de interés.
- Una ontología describe los conceptos dentro del dominio y la relación que tiene entre esos conceptos.
- Un lenguaje estándar para hacer ontologías es OWL desarrollado por W3C.
- OWL permite describir conceptos y además cuenta con un conjunto de operadores (intercesión, unión, y negación).
- OWL esta basado en lógica descriptiva que permite el uso de un razonador.

¿Que son las lógicas descriptivas?

- Son lógicas completas con semántica formal:

  Fragmentos decidibles de logica de primer orden

  Estrechamente relacionado con la lógica proposicional/ lógica dinámica

- Propiedades computacionales bien definidas (peor caso alta complejidad).

- Amppliamente utilizado para el lenguaje de ontologías.

Lógica descriptiva: Sintaxis

La sintaxis de un miembro de la familia de la lógica descriptiva se caracteriza por su definición recursiva, en la cual se establecen los constructores que se pueden usar para formar términos conceptuales.

Algunos constructores están relacionados con constructores lógicos en lógica de primer orden (FOL), como intersección o conjunción de conceptos, unión o disyunción de conceptos, negación o complemento de conceptos, restricción universal y restricción existencial.

Otros constructores no tienen una construcción correspondiente en FOL, incluidas las restricciones de funciones, por ejemplo, inversa, transitividad y funcionalidad.

Componentes de una ontología OWL

Conceptos (clases): Animal, Doctor, Perro
Equivalente a los predicados unarios en FOL

Relaciones (propiedades): Tiene.Padre, Ama, Ladra
Equivalente a los predicados binarios en FOL

Instancias (individuos): Shadow, Flavio, Pilar
Equivalente a constantes en FOL

Ontologías	OWL	PROTÉGÉ
Instancias	Individuos	Casos (instance)
Relaciones	Propiedades	Slots
Conceptos	Clases	Clases

Individuos de una ontología OWL

Representan objetos del dominio de interés y son también conocidos como instancias.

Propiedades de una ontología OWL

Son relaciones binarias sobre los individuos y pueden ser inversas, transitivas o simétricas.

Clases de una ontología OWL

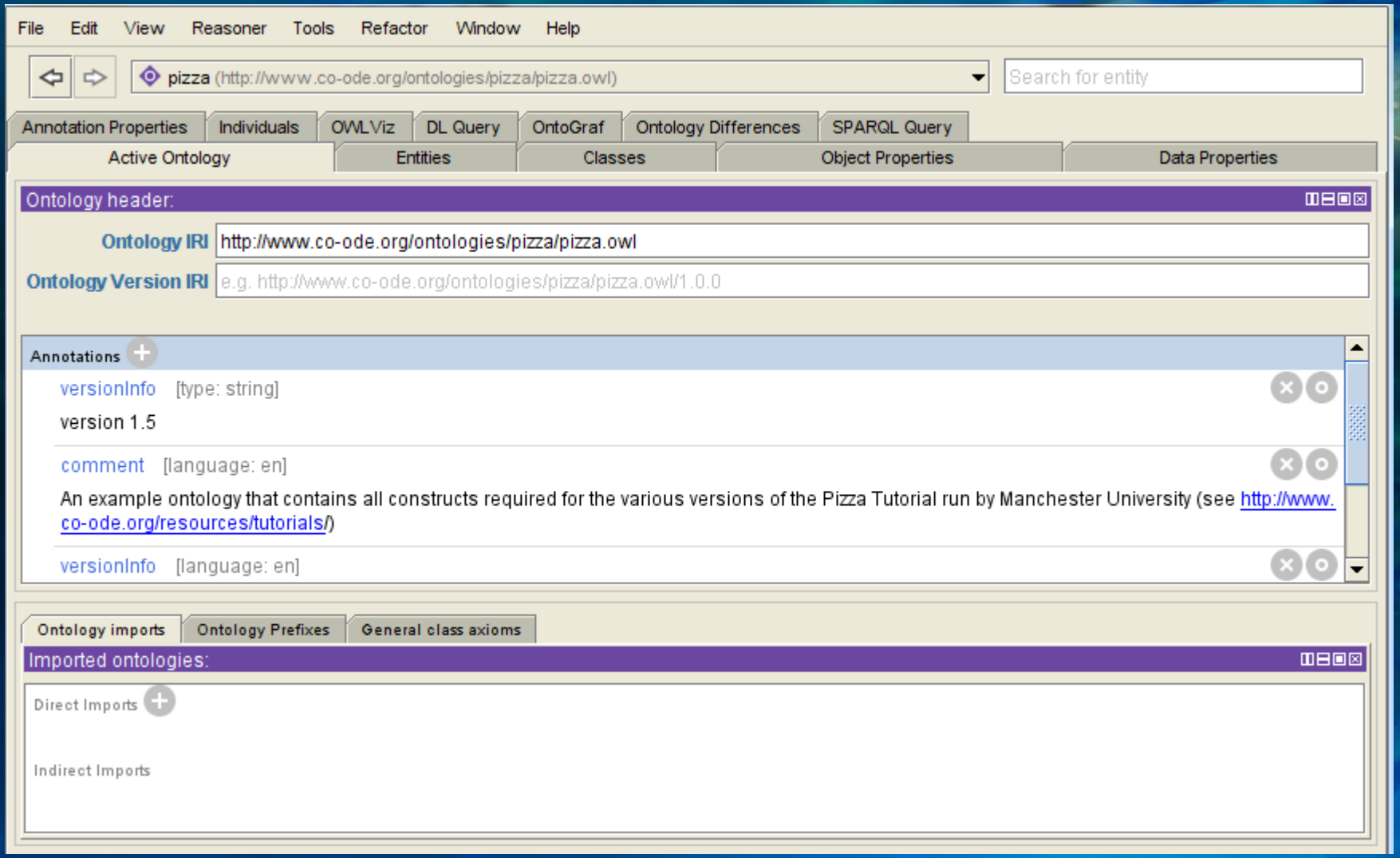
Se entienden como conjuntos que contienen individuos y pueden ser organizadas dentro de una jerarquía de clases y subclases conocida como taxonomía.

Correspondencia entre OWL y DL

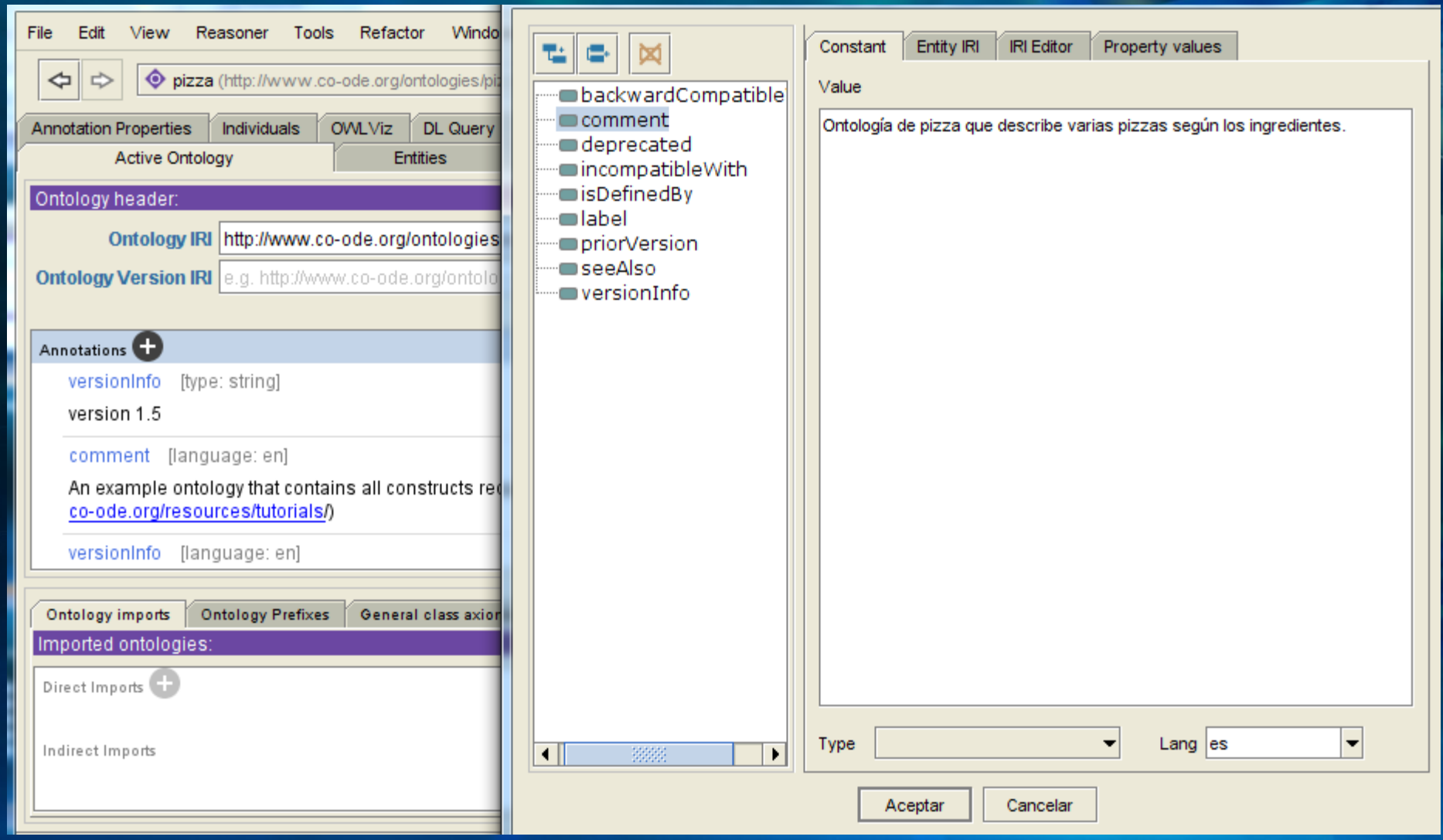
Constructor OWL	Representación DL	Ejemplo
owl:equivalentTo (C,D)	$C \equiv D (C \sqsubseteq D \; y \; D \sqsubseteq C)$	$Persona \equiv Humano$
rdfs:subClassOf (C,D)	$C \sqsubseteq D$	$Padres \sqsubseteq Persona$
owl:complementOf (C,D)	$C \equiv \neg D (negacion)$	$Varon \equiv \neg Mujer$
owl:disjointWith (C,D)	$C \sqsubseteq \neg D$	$Padre \sqsubseteq \neg Madre$
owl:intersectionOf (C,D)	$C \sqcap D (conjuncion)$	$Padres \sqcap Varon$
owl:unionOf (C,D)	$C \sqcup D (disjuncion)$	$Padre \sqcup Madre$
owl:oneOf (I1, I2)	$\{I_1\} \sqcup \{I_2\}$	$\{Juan\} \sqcup \{Maria\}$
owl:someValuesFrom(P,C)	$\exists P.C (existencial)$	$\exists tieneHijo.Hija$
owl:allValuesFrom(P,C)	$\forall P.C (universal)$	$\forall tieneHijo.Hijo$
owl:hasValue (P,I1)	$\exists P.\{I_1\}$	$\exists tieneHijo.\{Juan\}$
owl:cardinality(P,n)	$= n.P$	$= 2.tienePadres$
owl:minCardinality(P,n)	$\geq n.P$	$\geq 1.tieneHija$
owl:maxCardinality(P,n)	$\leq n.P$	$\leq 2.tieneHijos$

Un *concepto en DL se refiere a una clase en OWL.*
Un *rol en DL es una propiedad en OWL.*

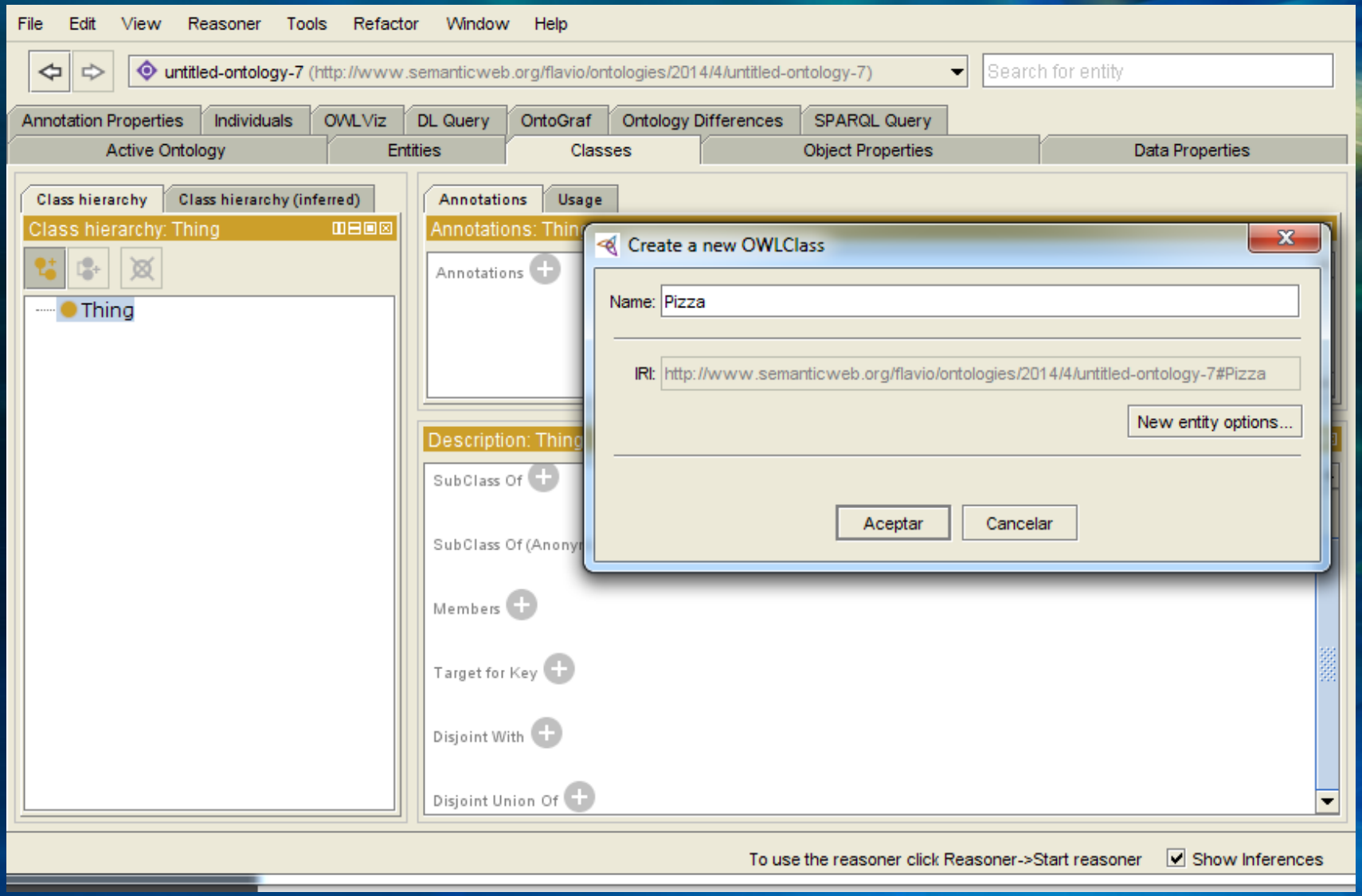
Interfaz de Protégé



Comentarios en Protégé



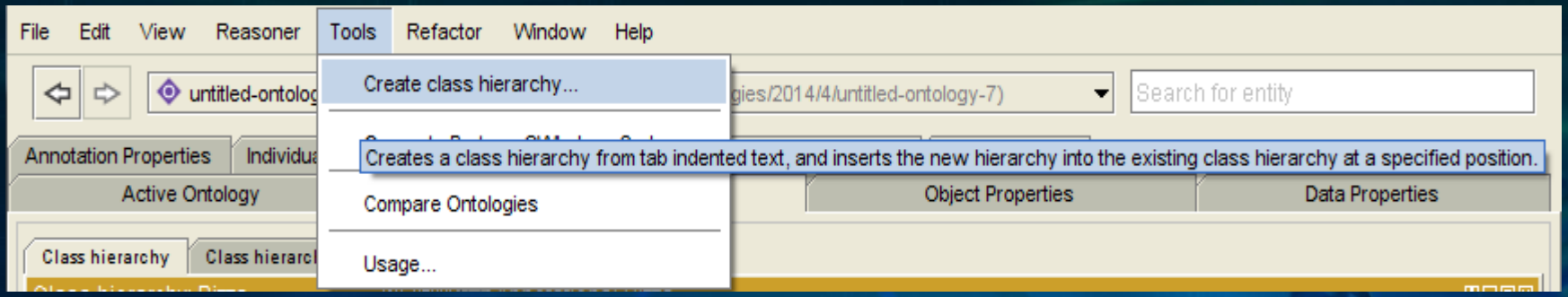
Clases en Protégé



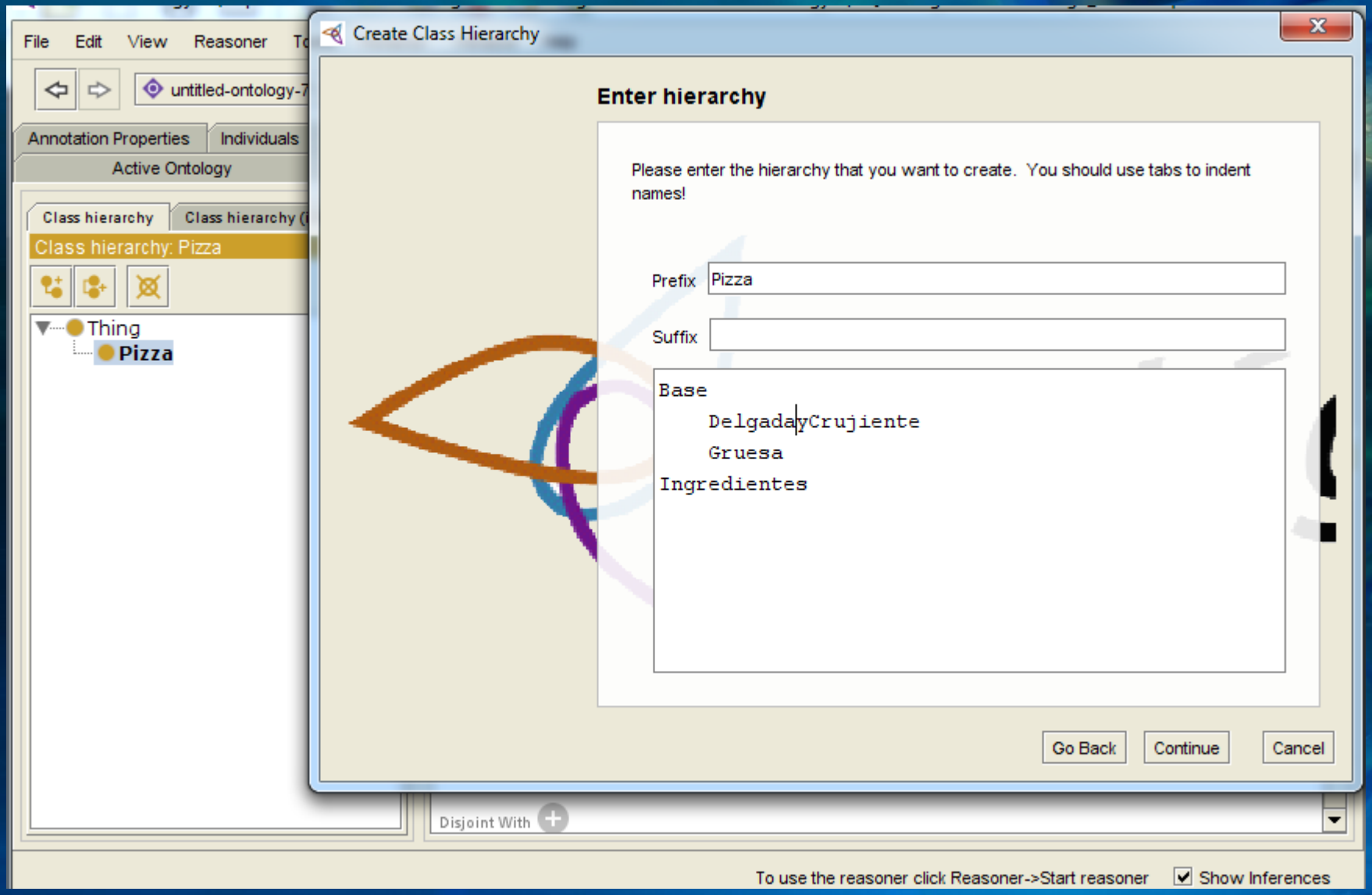
Clases en Protégé

Para generar clases y subclases en un solo paso se utiliza la herramienta:

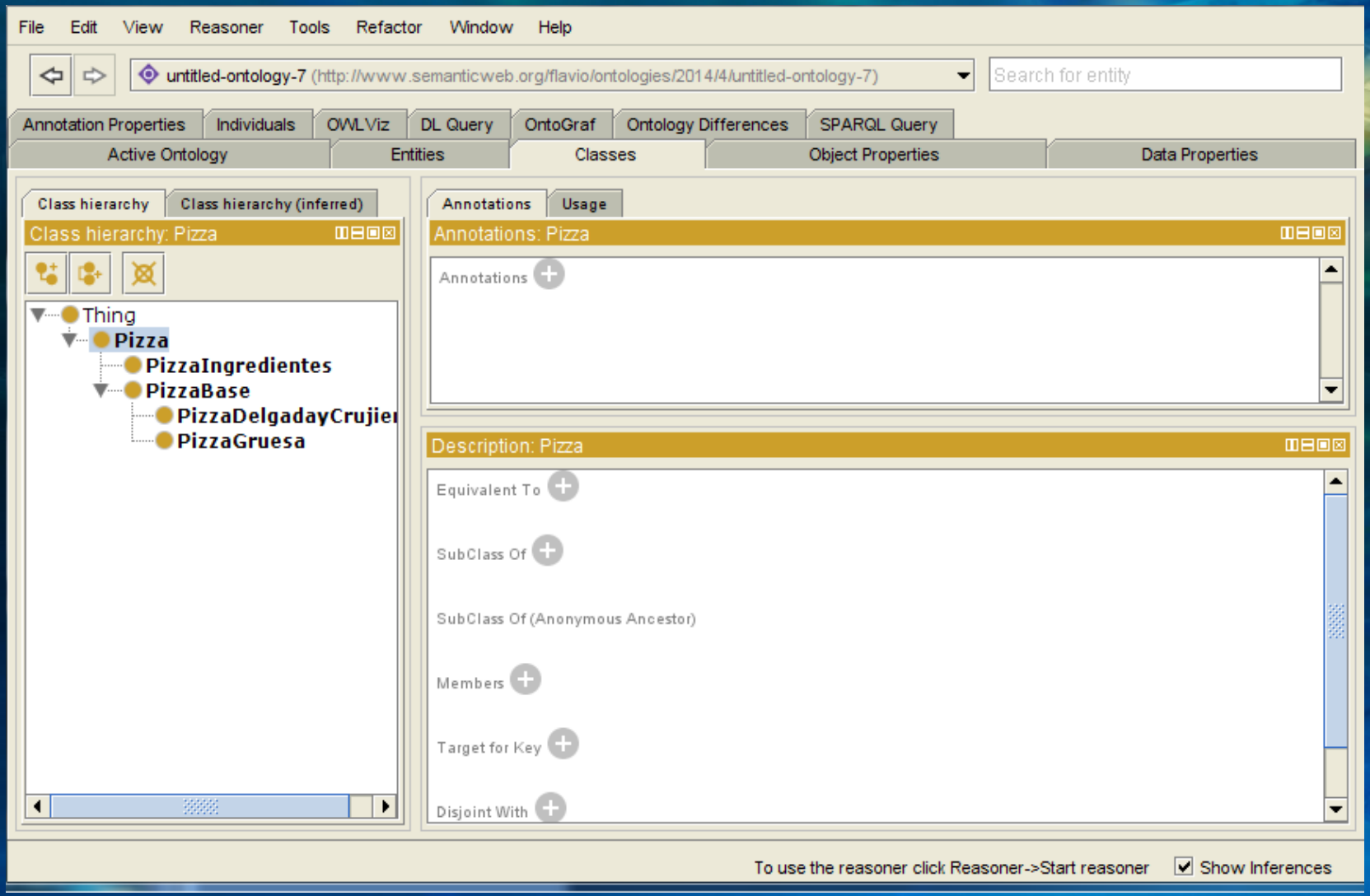
Create class hierarchy



Clases en Protégé

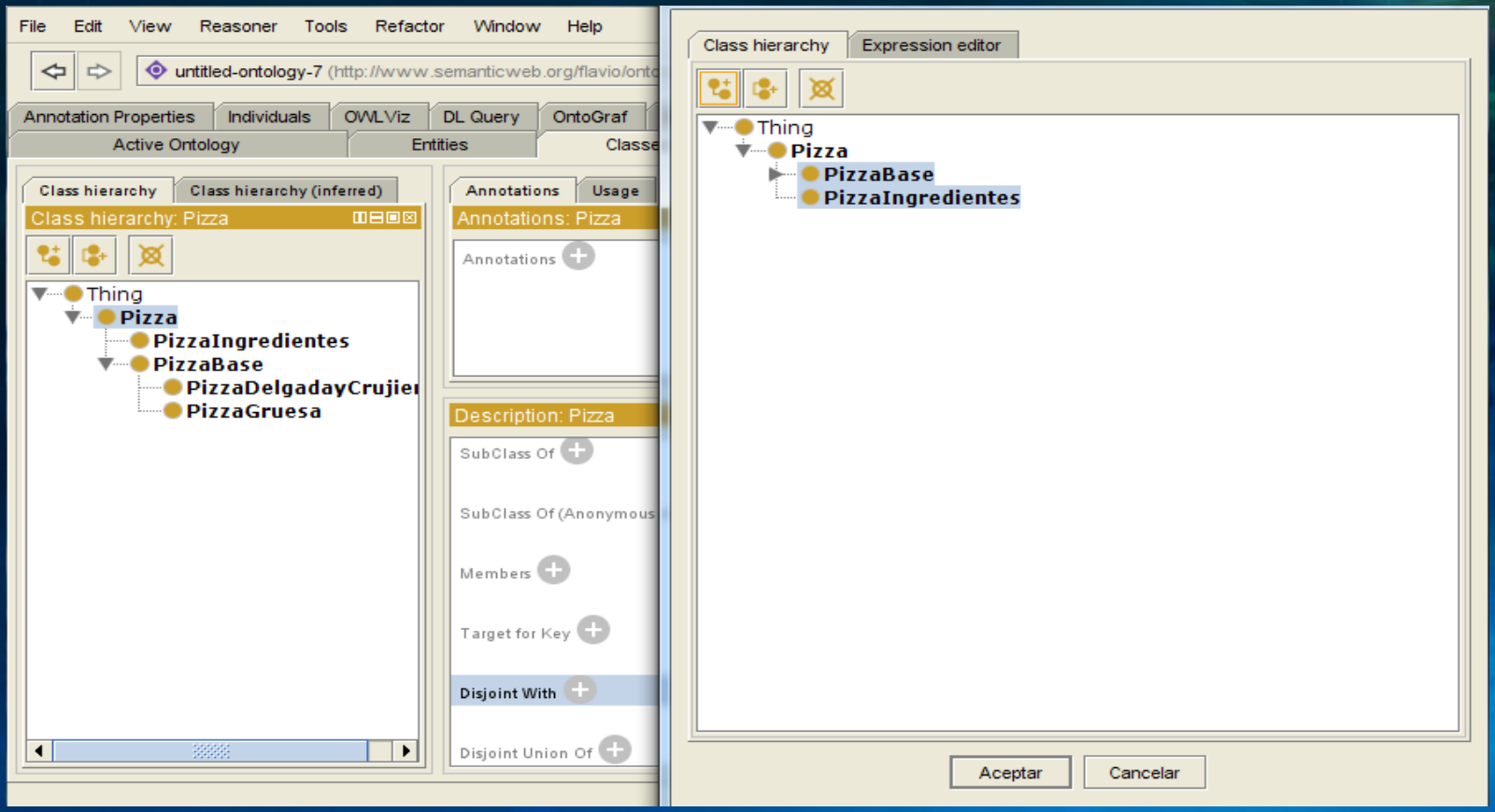


Clases en Protégé

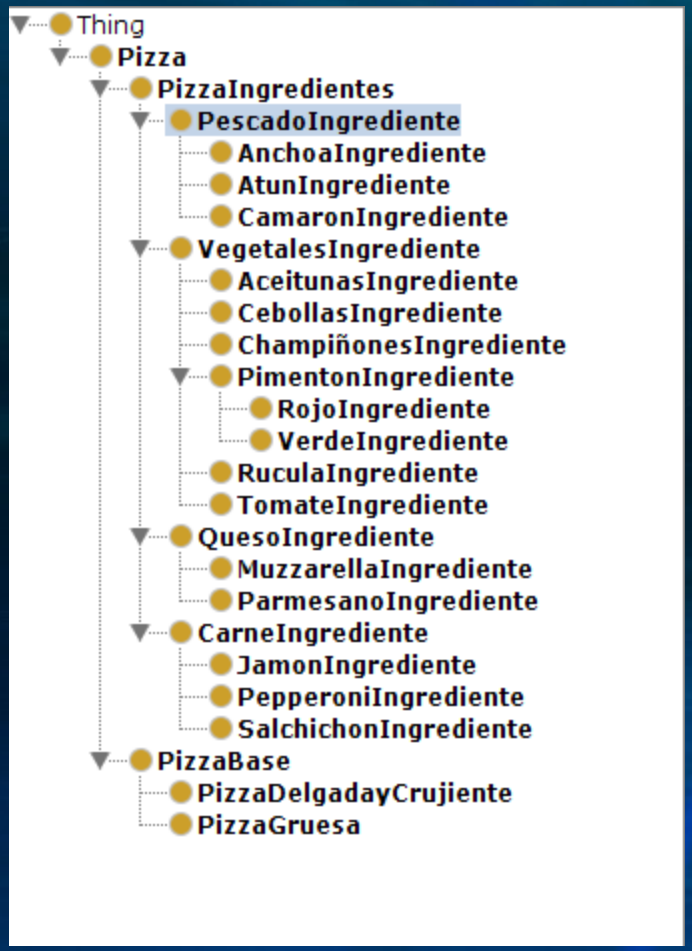


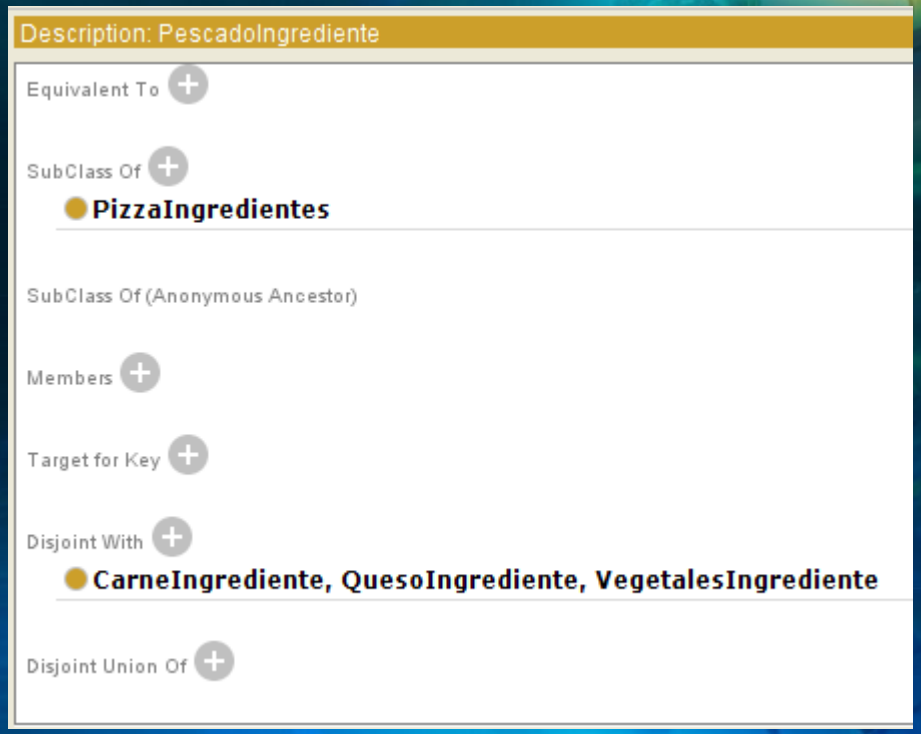
Clases Disjuntas en Protégé

Después de adicionar varias clases en la jerarquía de la ontología, se requiere establecer clases disjuntas, que indican que un objeto o individuo no puede ser instancia de más de una de estas clases que se establecieron disjuntas.



Clases Disjuntas en Protégé

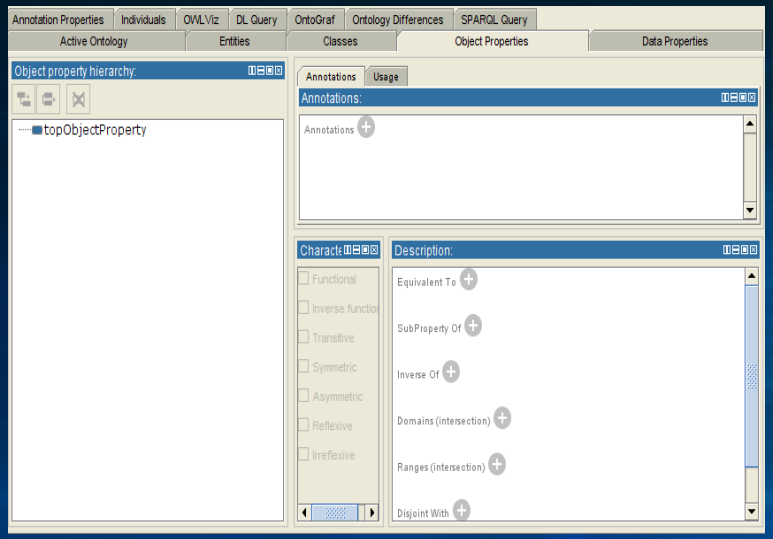


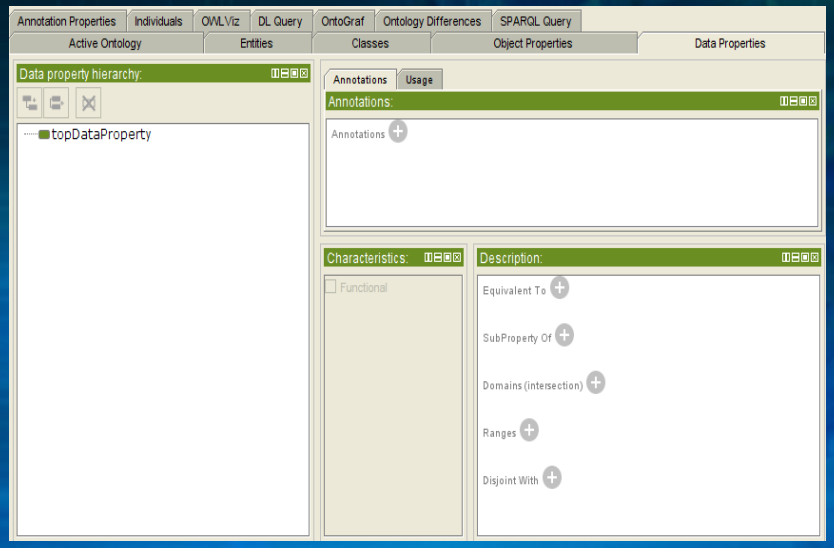


Propiedades OWL en Protégé

Existen dos tipos de propiedades en OWL:

- **“ObjectProperties”, que permite relacionar un individuo con otro**
- **“DatatypeProperties”, que relaciona un individuo con un XML Schema Datatype value o un literal RDF**





Propiedades OWL en Protégé

Los elementos que debe tener un ObjectProperty son:

Nombre

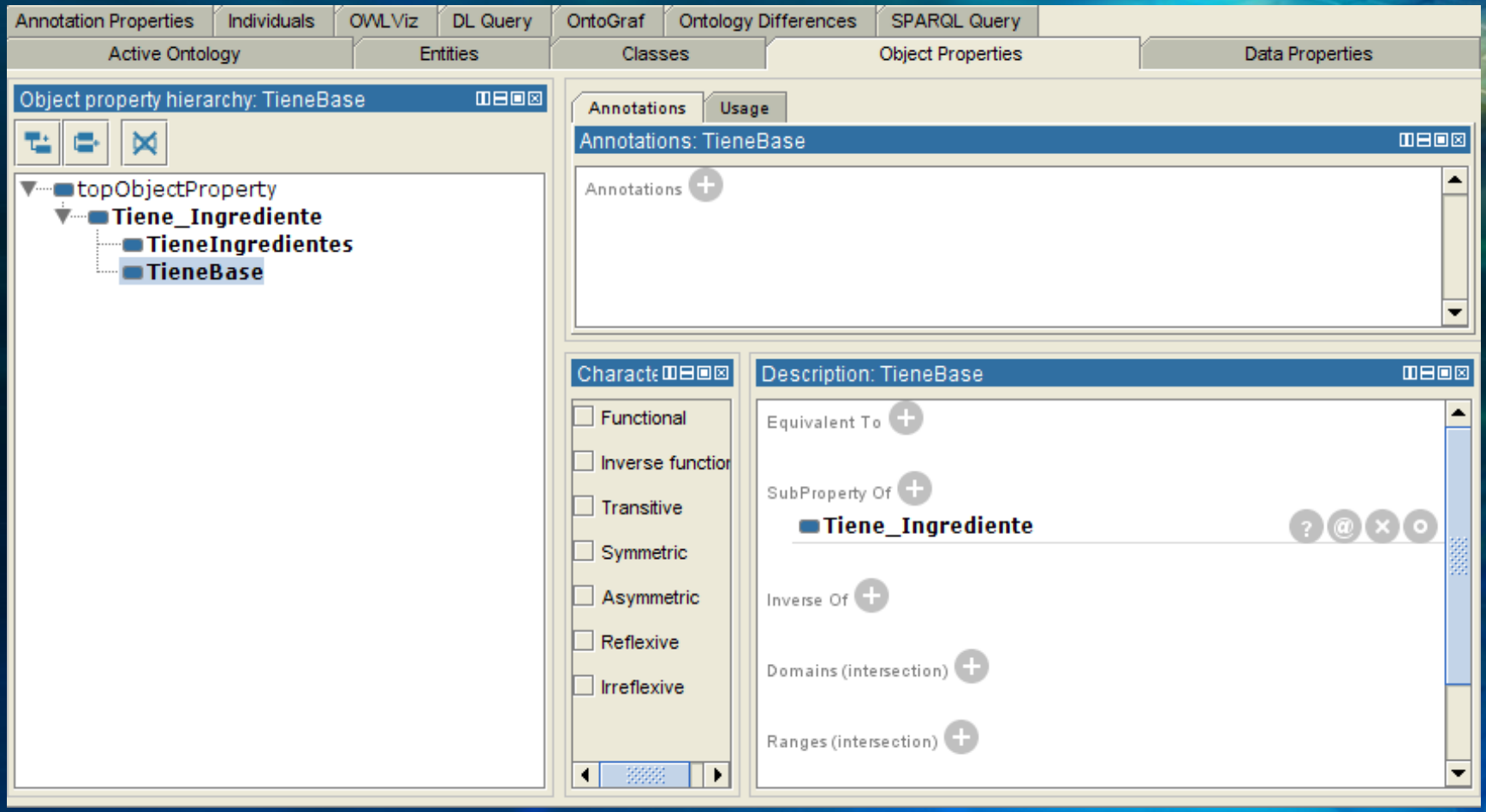
***Dominio*: hace referencia a la clase o clases iniciales**

***Rango*: hace referencia a la clase o clases finales.**

Ejemplo la relación: es profesor

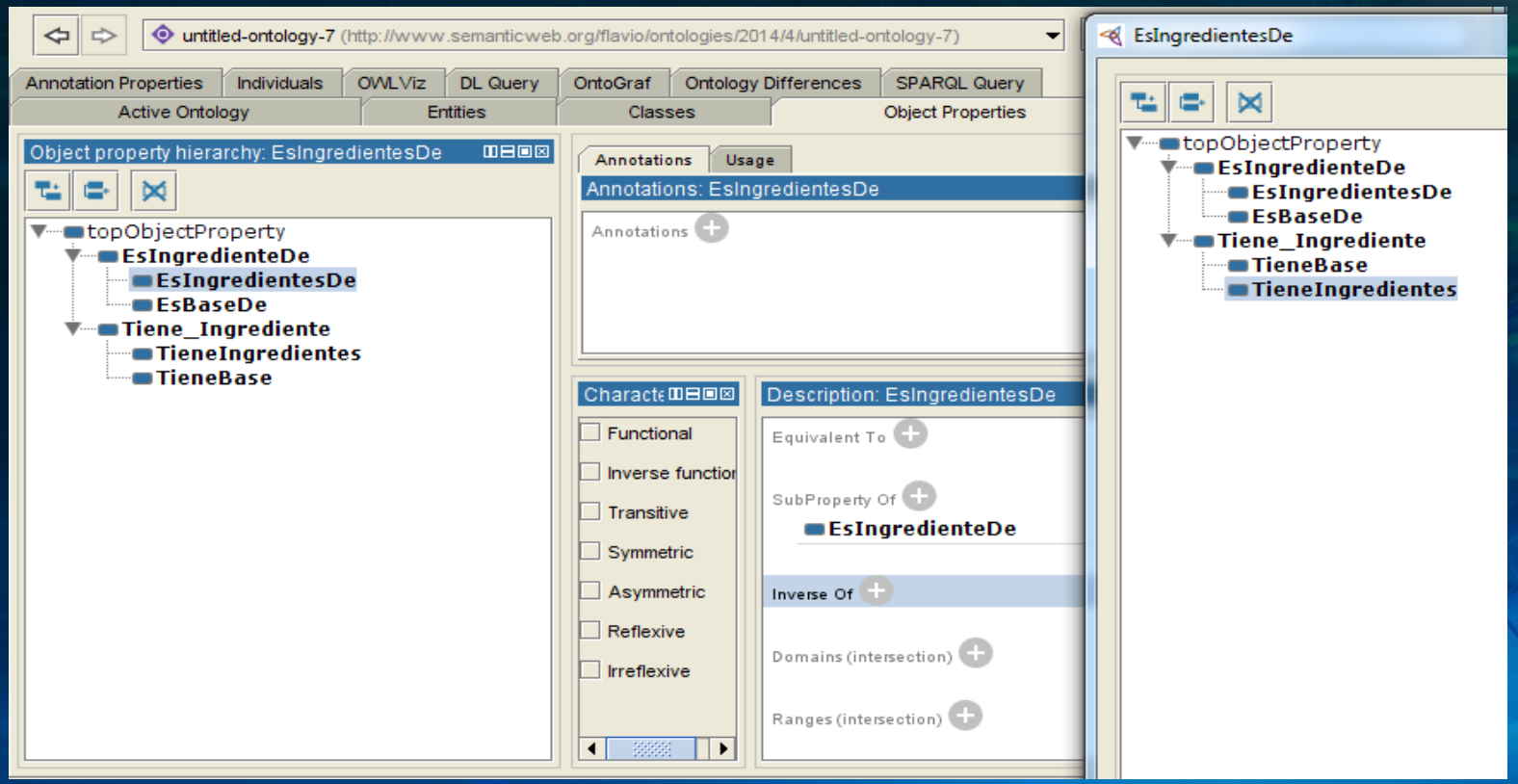
Nombre: es_profesor
Dominio: Docente
Rango: Estudiante

Propiedades OWL en Protégé



Propiedades Inversas en Protégé

Cada *ObjectProperty* debe tener su correspondiente propiedad inversa.
Si una propiedad enlaza un objeto A con otro B, entonces la propiedad inversa enlaza el objeto B con el A.



Características de las propiedades en Protégé

OWL permite que el significado de las propiedades sea enriquecido con características de las propiedades

- **Funcional**
- **Funcional Inversa**
- **Simétrica**
- **Transitiva**

Propiedad Funcional en Protégé

Define que a lo sumo un objeto puede estar relacionado con otro objeto.

Ejemplo, si se tienen tres objetos que son A, B y C y se tiene una propiedad funcional *tienePadre*, entonces se podrían asociar los objetos A y B por medio de la propiedad y daría como resultado A *tienePadre* B. Igualmente se podrían asociar los objetos A y C por medio de la propiedad y daría como resultado A *tienePadre* C. Como *tienePadre* es propiedad funcional, *se concluye que B y C son el mismo objeto*. En caso contrario estaríamos en una contradicción.

Propiedad Funcional Inversa en Protégé

Indica que puede estar a lo sumo un objeto relacionado con otro.

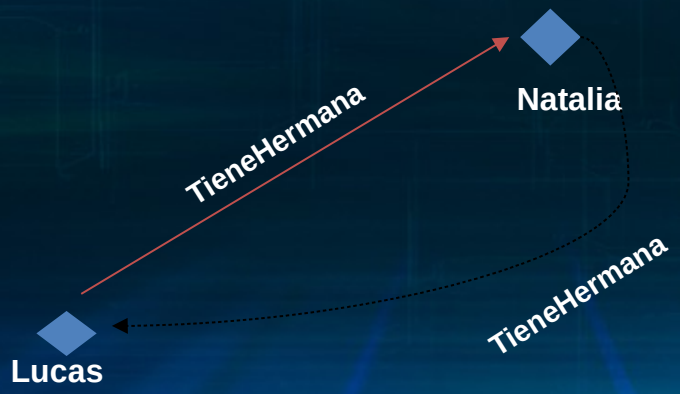
Ejemplo, si se tienen tres objetos que son A, B y C y se tiene una propiedad funcional *esPadreDe*, entonces se podría asociar el objeto B y A por medio de la propiedad y daría como resultado B *esPadreDe* A. Igualmente se podrían asociar los objetos C y A por medio de la propiedad y daría como resultado C *esPadreDe* A. Como *esPadreDe* es propiedad funcional inversa, se concluye que B y C son el mismo objeto.

Propiedad Transitiva en Protégé

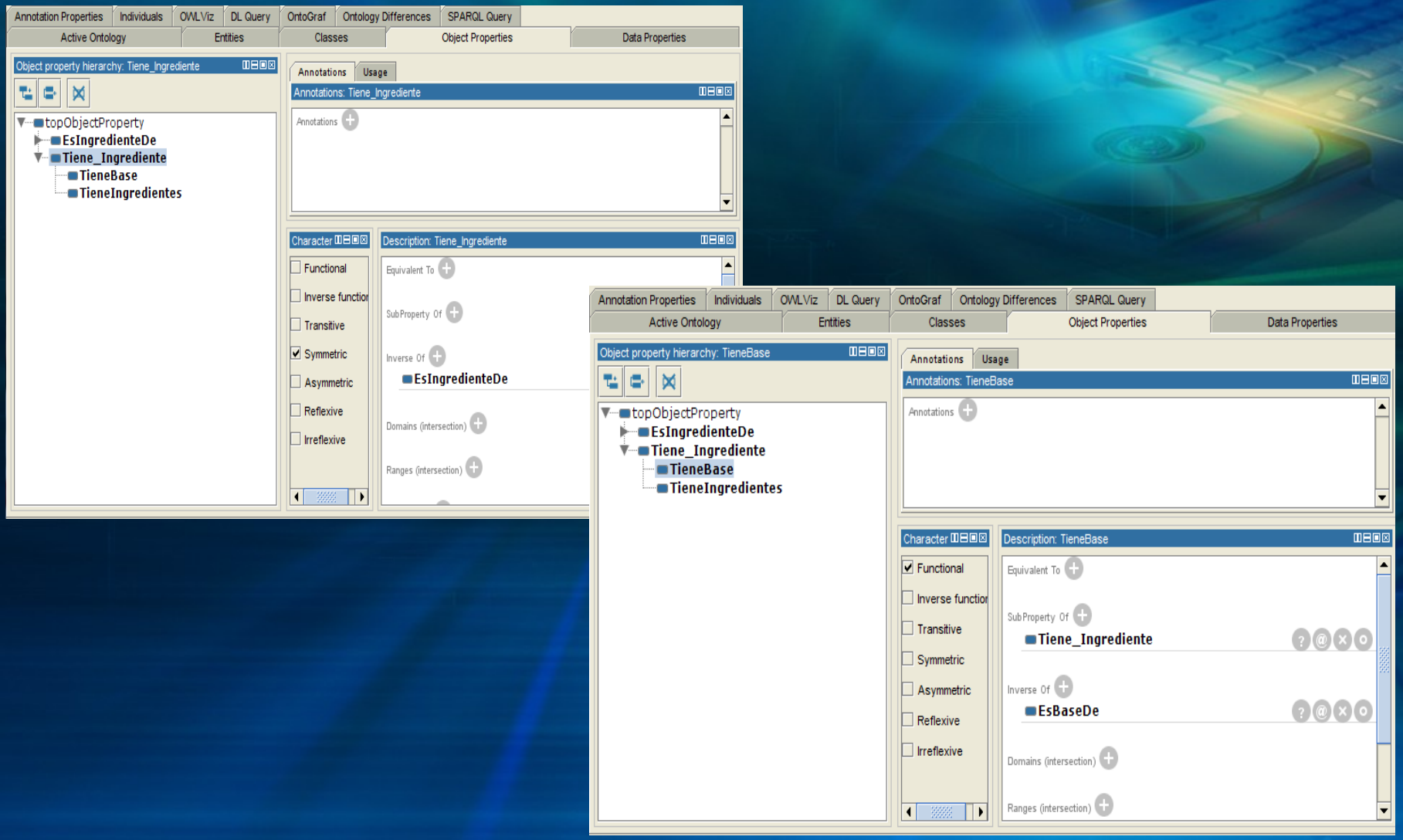
Relaciona dos objetos A y B, y además hay una propiedad que relaciona al objeto B con otro C, entonces se puede inferir que el objeto A está relacionado con el objeto C mediante la propiedad transitiva.

Propiedad Simétrica en Protégé

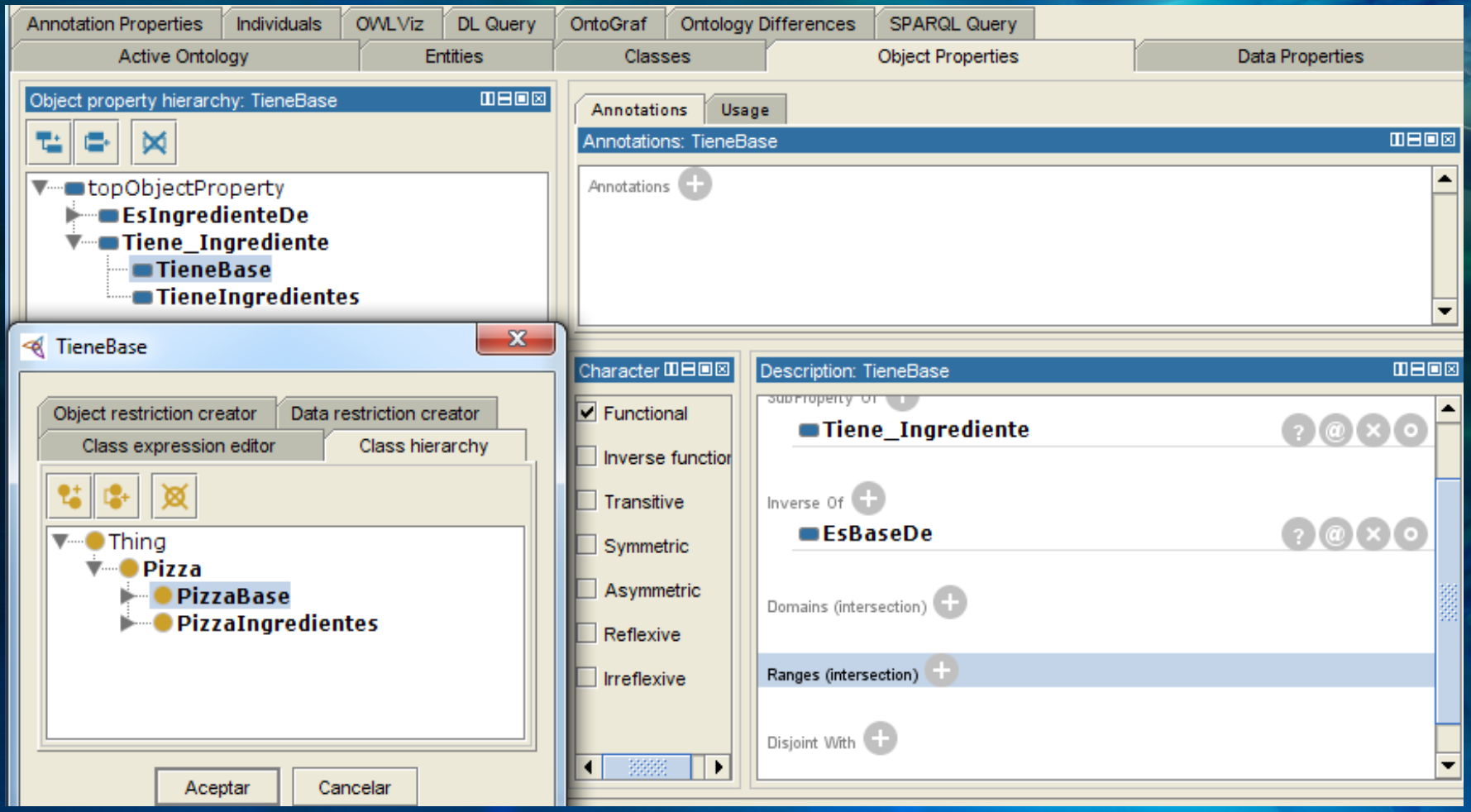
Relaciona a los objetos A y B, entonces el objeto B es relacionado por medio de la propiedad P con el objeto A.



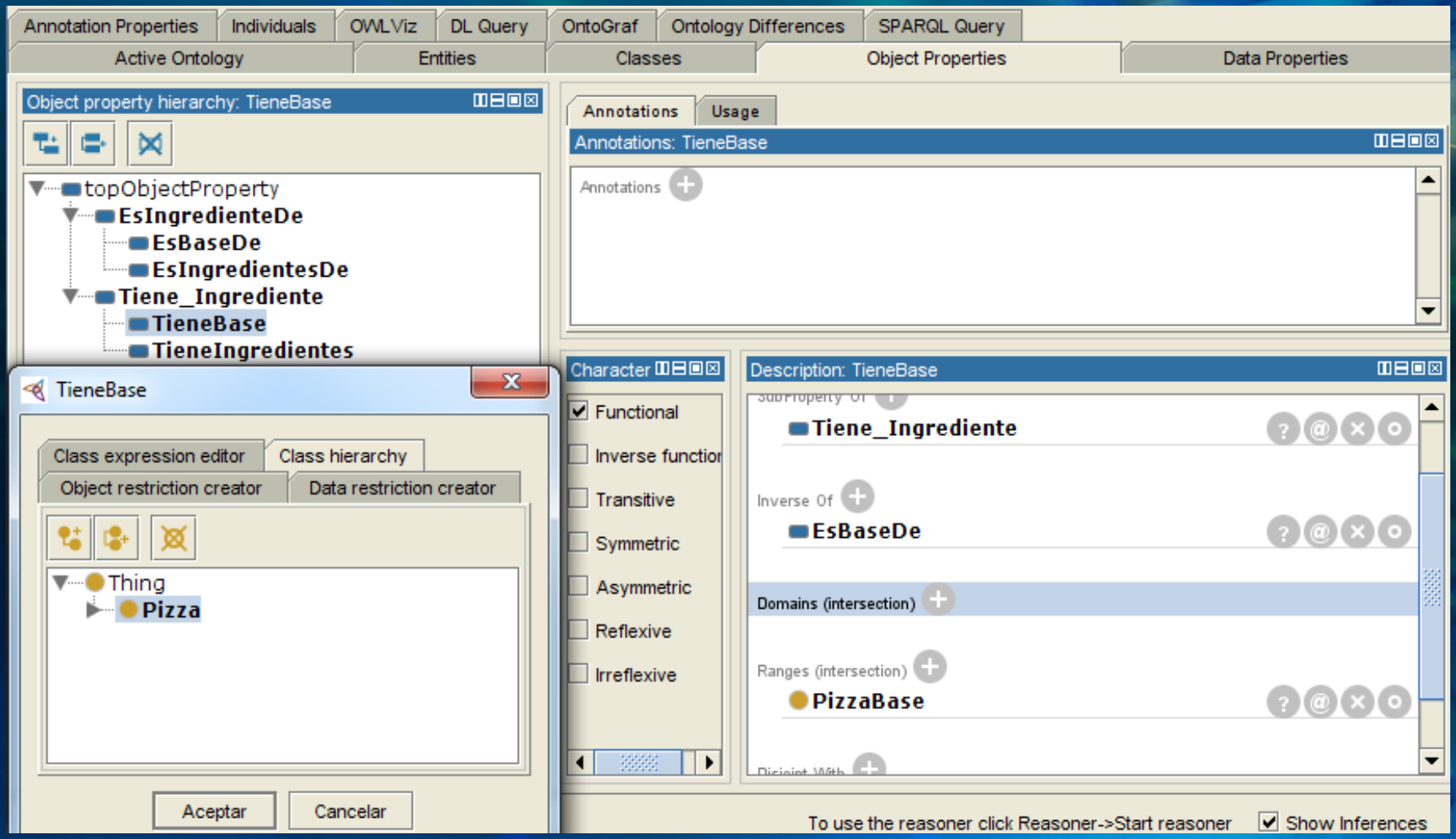
Propiedades en Protégé



Rango de una Propiedad en Protégé



Dominio de una Propiedad en Protégé



Restricciones de una Propiedad en Protégé

Las propiedades son utilizadas para crear restricciones en las clases en una ontología OWL.

Usualmente el nombre de la propiedad debería sugerir las restricciones impuestas a los objetos de la clase.

Las restricciones OWL se presentan en las siguientes tres categorías:

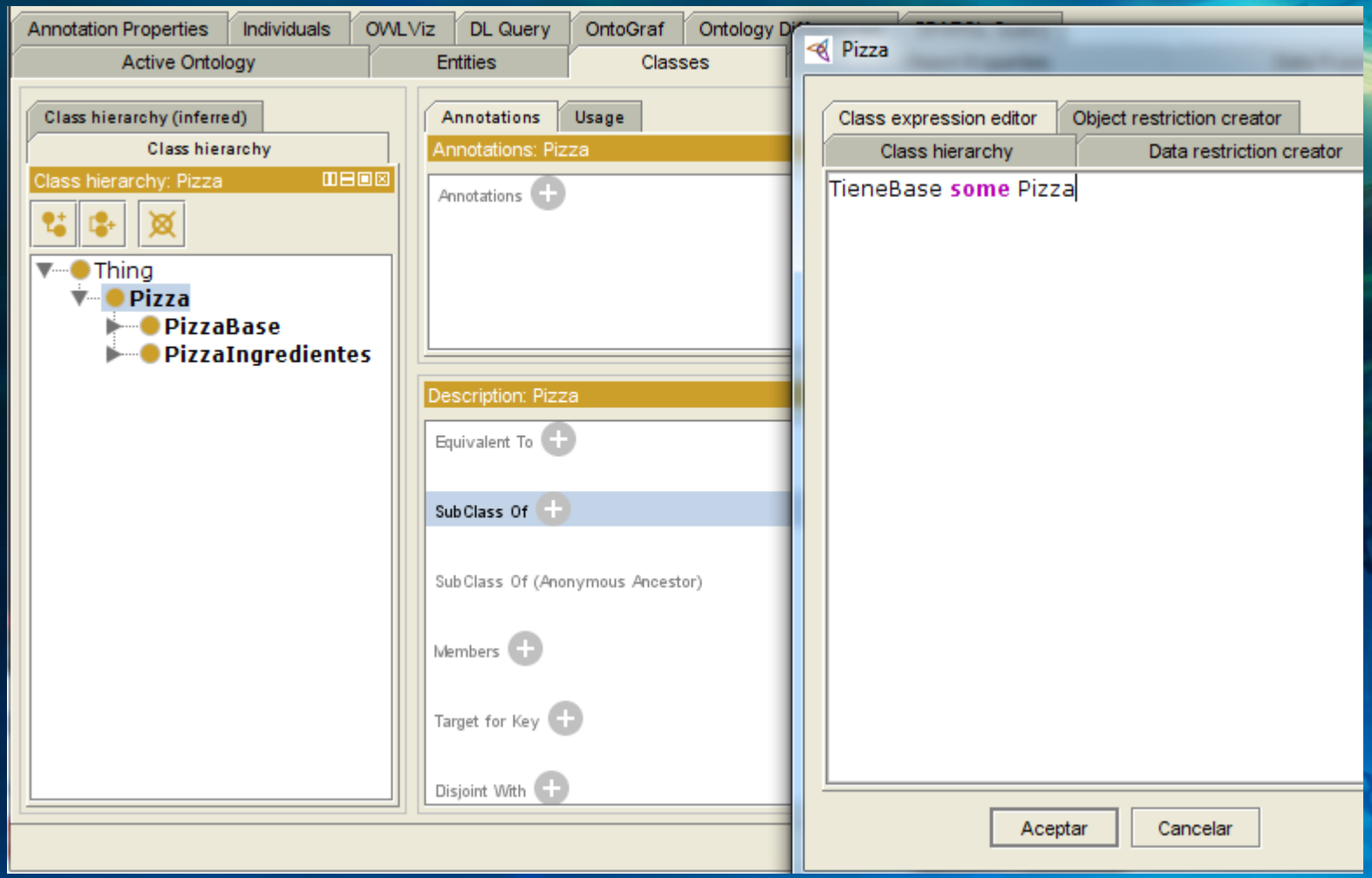
- Restricciones de cuantificación.
- Restricciones de cardinalidad.
- Restricciones de valor.

Restricciones de una Propiedad en Protégé

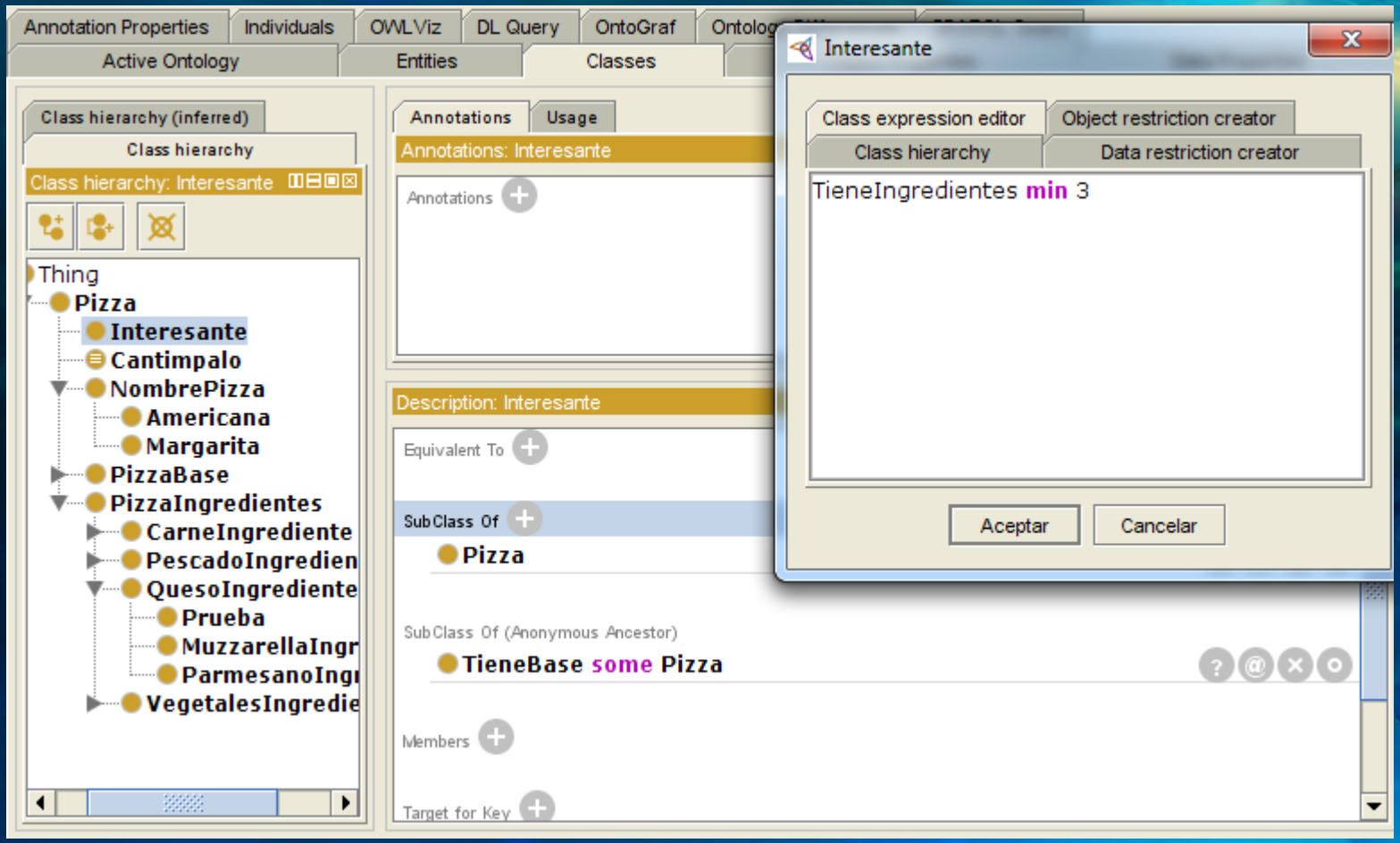
Cuantificador existencial (∃), el cual permite indicar la existencia de al menos un objeto. En Protégé 4 la palabra clave *some* es usado para denotar ∃.

Cuantificador universal (∀), el cual permite indicar la existencia de todos los objetos. En Protégé 4. la palabra clave es *only* es usado para denotar ∀.

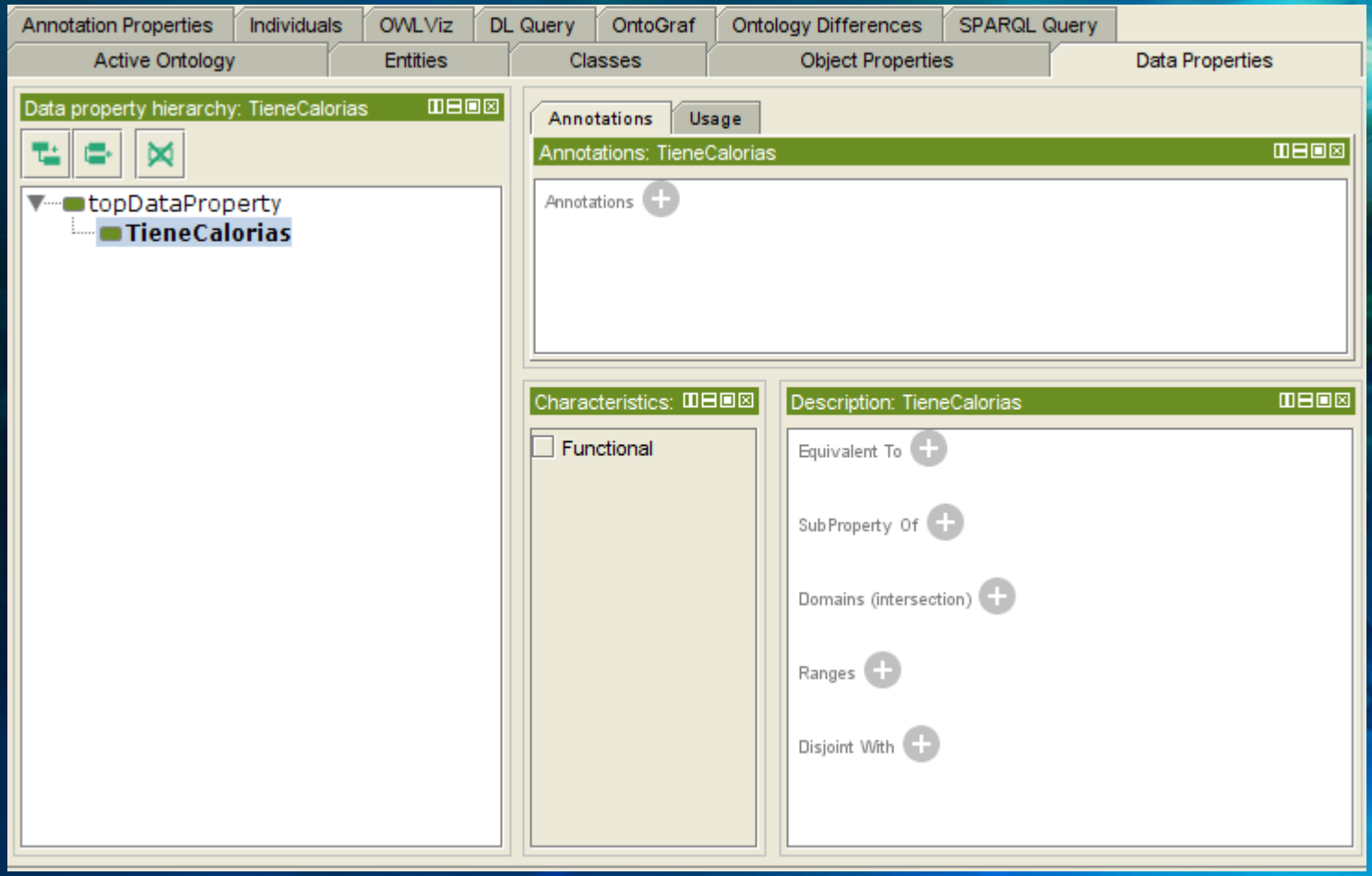
Restricciones de una Propiedad en Protégé



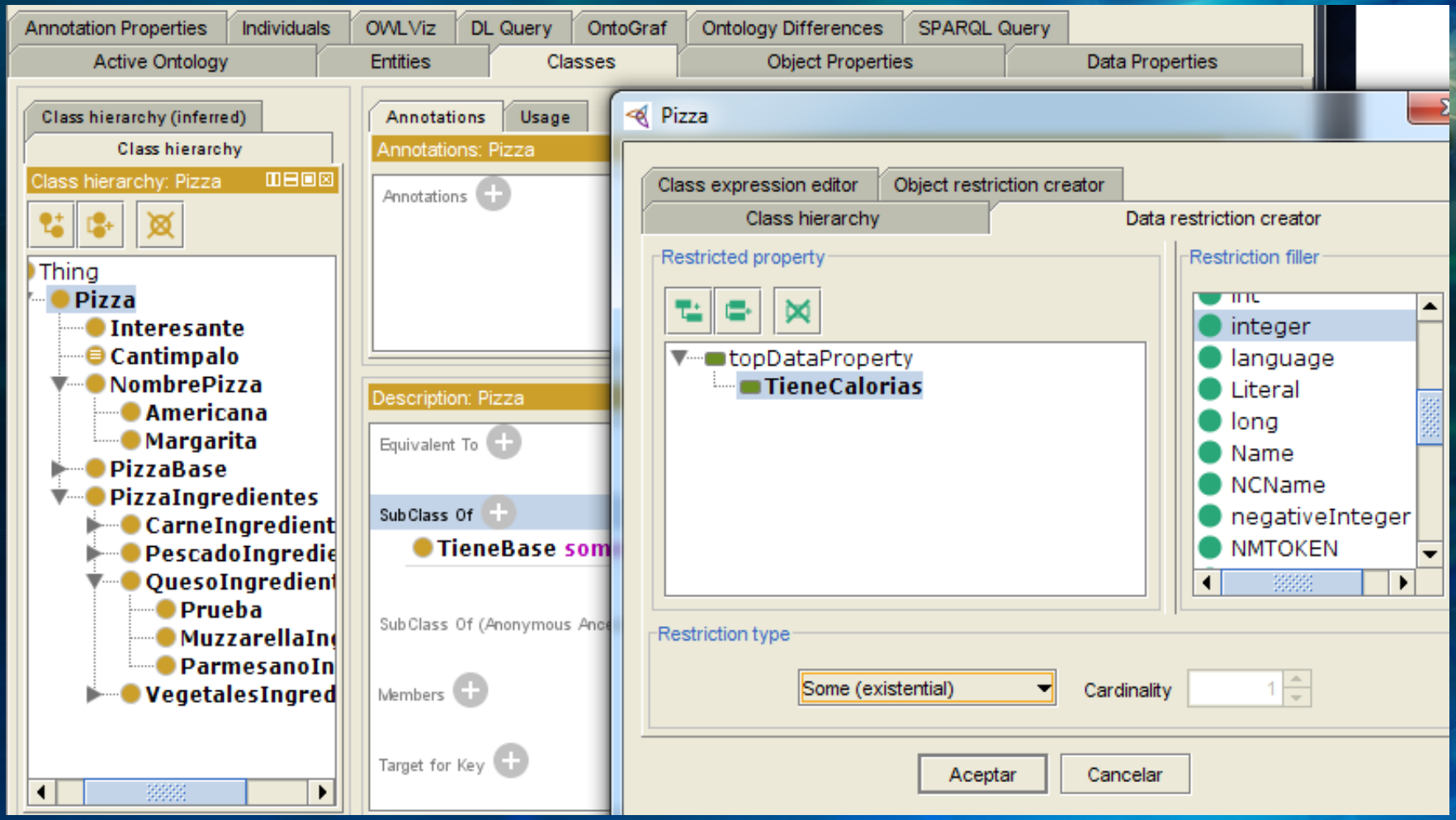
Restricción de Cardinalidad en Protégé



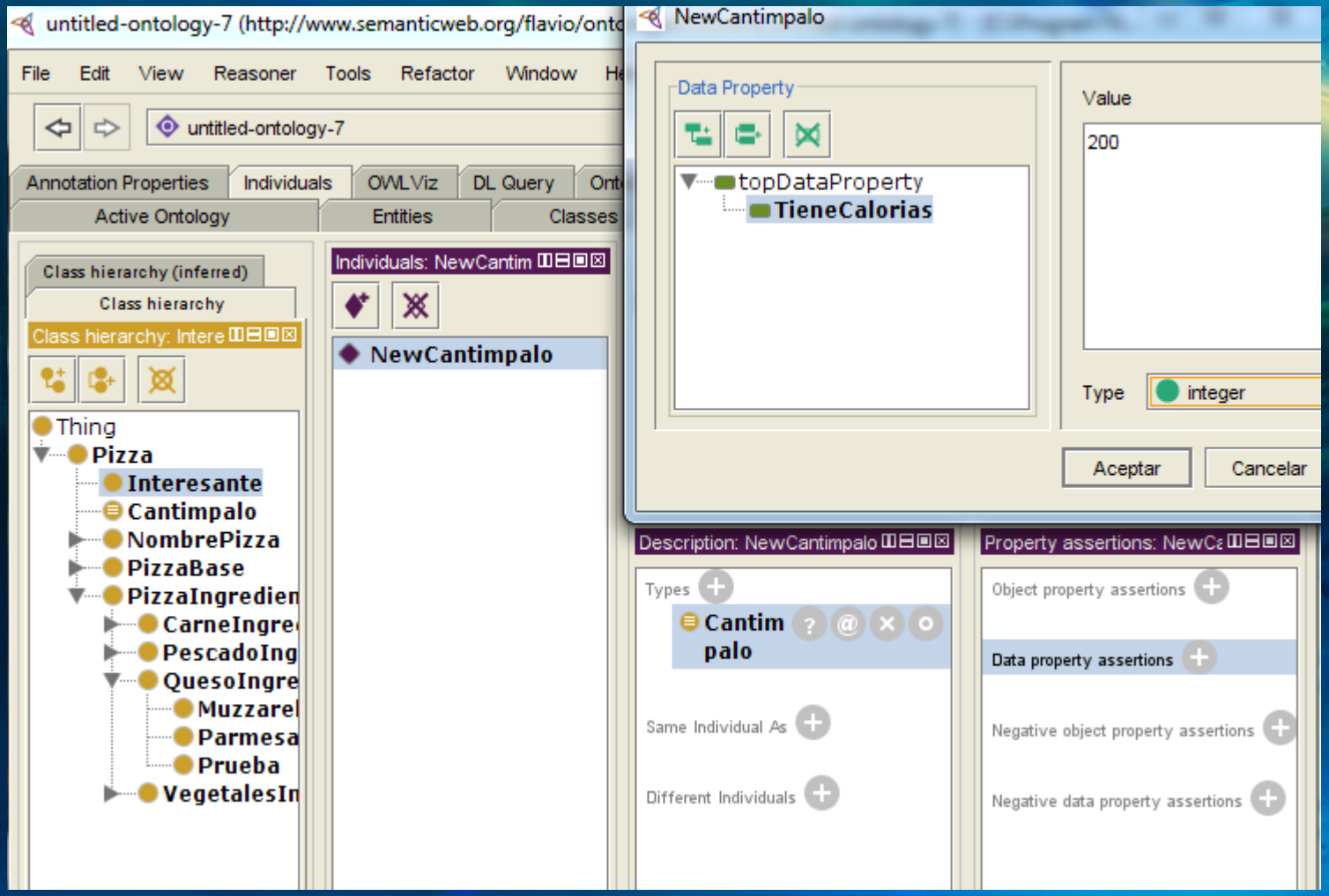
Propiedad Datatype en Protégé



Propiedad Datatype en Protégé



Propiedad Datatype en Protégé



Razonadores en Protégé

Una base de conocimiento (KB) se compone de un TBox y un ABox.

Un TBox describe conocimiento intencional en la forma de conceptos (clases) y definiciones de roles (propiedades).

Un ABox describe conocimiento por extensión y consiste de un conjunto finito de aserciones acerca de los individuos mientras utiliza los términos de la ontología. Conviene notar que un ABox representa un conocimiento incompleto acerca del mundo.

Las clases pueden ser organizadas en una jerarquía de superclases-subclases, conocido como *taxonomía*.

Estas relaciones pueden ser procesadas por un razonador en OWL-DL. Virtualmente toda consulta a una ontología OWL DL debe ser realizada utilizando un razonador que deduzca conocimiento implícito.

Los razonadores pueden ser agrupados en dos categorías: razonadores de lógica descriptiva y razonadores de programación lógica.

Razonadores en Protégé

Reasoning Characteristics

	CB	CEL	FaCT++	HermiT	Pellet	RP	SR	TrOWL (REL)
Methodology	consequence-based	completion rules	tableau-based	hypertableau	tableau-based	tableau-based	completion rules	approximation (completion rules)
Soundness	+	+	+	+	+	+	+	+ (+)
Completeness	+	+	+	+	+	+	+	- (+)
Expressivity	Horn $\mathcal{SHIF}$	$\mathcal{EL}^+$	$\mathcal{SROIQ}(\mathcal{D})$	$\mathcal{SROIQ}(\mathcal{D})$	$\mathcal{SROIQ}(\mathcal{D})$	$\mathcal{SHIQ}(\mathcal{D}-)$	$\mathcal{EL}^+$	third-party reasoner (approximating SROIQ; subset of $\mathcal{EL}^{++}$)
Incremental Classification (addition/removal)	-/-	+/-	-/-	-/-	+/+	-/-	+/-	-/-
Rule Support	-	-	-	+ (SWRL)	+ (SWRL)	+ (SWRL, nRQL)	-	-
Justifications	-	+	-	-	+	+	-	-
ABox Reasoning	-	+	+	+	+ (SPARQL)	+ (SPARQL, nRQL)	-	+ (SPARQL)

Razonadores lógica descriptiva

Los razonadores DL brindan los siguientes servicios de inferencia:

- Validación de la consistencia de una ontología: el razonador puede comprobar si una ontología no contiene hechos contradictorios
- Validación del cumplimiento de los conceptos de la ontología: el razonador determina si es posible que una clase tenga instancias. En el caso de que un concepto no sea satisfecho la ontología será inconsistente.
- Clasificación de la ontología: el razonador computa a partir de los axiomas declarados en el TBox, las relaciones de subclase entre todos los conceptos declarados explícitamente a fin de construir la jerarquía de clases.

Razonadores lógica descriptiva

Los razonadores DL brindan los siguientes servicios de inferencia:

- Posibilita la resolución de consultas durante la recuperación de información basada en ontologías: a partir de la jerarquía de clases se pueden formular consultas como conocer todas las subclases de un concepto, inferir nuevas subclases de un concepto, las superclases directas, etc.

- Precisiones sobre los conceptos de la jerarquía: el razonador puede inferir cuáles son las clases a las que directamente pertenece y mediante la jerarquía inferida obtener todas las clases a las cuales indirectamente pertenece una clase o individuo dentro de la ontología.

HermiT

HermiT es uno de los actuales proyectos de investigación del Laboratorio de Computación de la Universidad de Oxford.

Aunque puede ser empleado con cualquier ontología, los investigadores toman como punto de partida los requerimientos de ontologías médicas.

HermiT es un razonador para ontologías escritas empleando OWL y construido empleando cálculo hypertableau a fin de proveer razonamiento más eficiente.

Se anuncia como un razonador veloz y capaz de resolver ontologías complejas.

Pellet

Pellet fue desarrollado por el laboratorio Mindswap de la Universidad de Maryland (USA).

Entre sus características se destacan que admite la expresividad completa de OWL DL, razonamiento acerca de nominales (clases enumeradas o definidas por extensión), absorción, ramificación semántica y fue extendido para soportar las nuevas características propuestas en OWL 1.1 y OWL 2.

Pellet trata de un razonador DL basado sobre algoritmos tableaux. El núcleo del razonador, es el algoritmo tableaux (desarrollado para lógicas descriptivas potentes) el cual verifica la consistencia de la KB, es decir el par ABox y Tbox.

Pellet

Esencialmente un razonador tableaux posee sólo la funcionalidad de verificar la satisfactibilidad de un ABox con respecto a un TBox.

Este razonador implementa las mejores técnicas de optimización, lo que hace que su desempeño sea bueno, en especial cuando debe evaluar ontologías con mayor complejidad y expresividad; sin embargo no es tan eficiente como RacerPro o FACT++ en clasificaciones.

FaCT++

FaCT++, desarrollado por la Universidad de Manchester bajo el proyecto europeo “WonderWeb”, es un razonador DL basado en el algoritmo tableaux para lógica descriptiva. Cubre los lenguajes de ontología OWL y OWL2.

FaCT++ es un buen razonador para la TBox de una ontología, sin embargo, carece de soporte para otros tipos de datos que no sean string o integer (como si ocurre con Pellet) y tampoco posee soporte para razonamiento con la ABox de una ontología.

Entre sus características se destaca:

1) trabaja de forma eficiente con TBox de ontologías de tamaño grande y mediano

2) es posible utilizar el lenguaje de consultas para RDF.

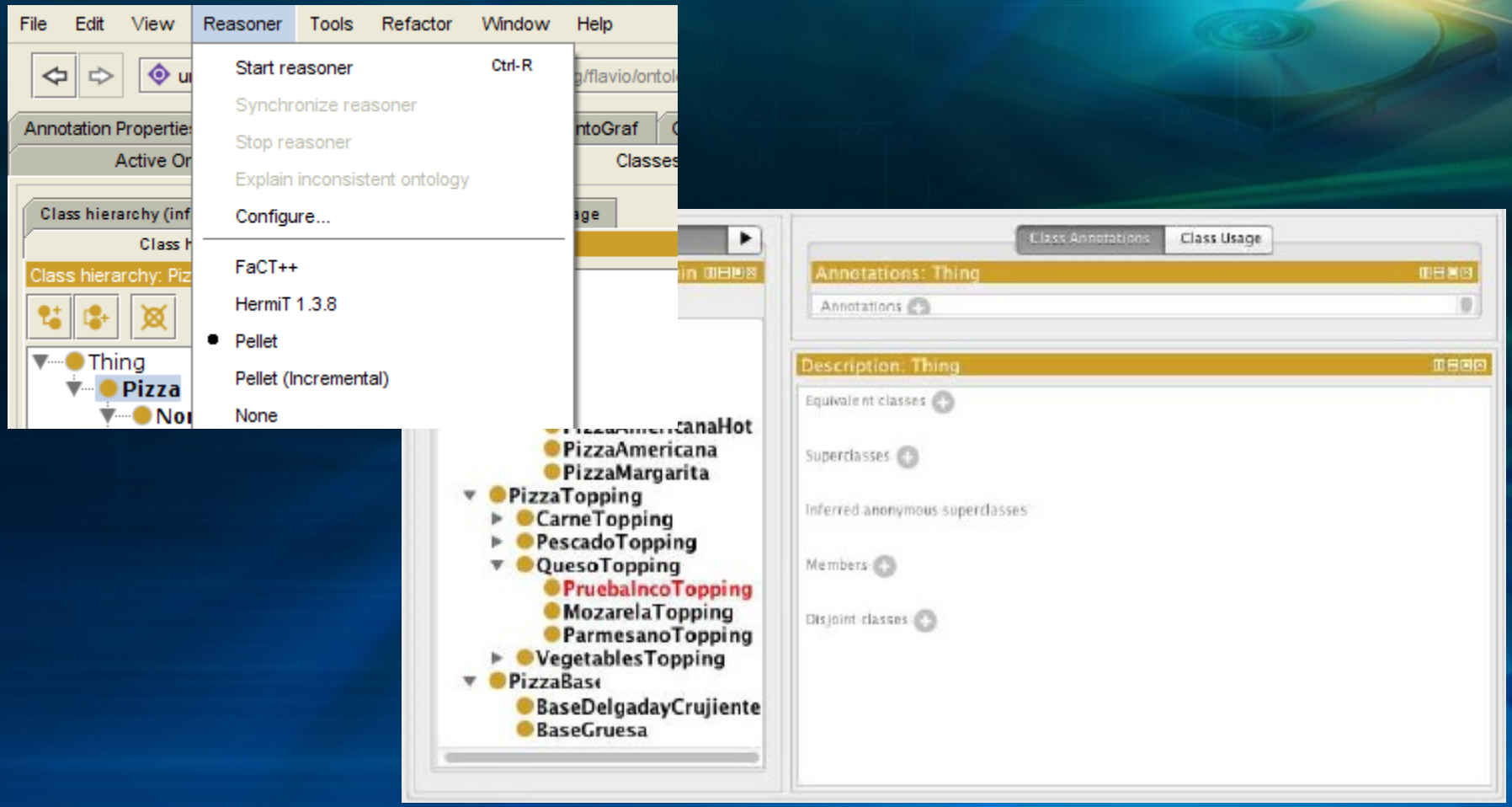
FaCT++

Mediante el algoritmo tableaux implementa un procedimiento de decisión para TBox y ABox.

También implementa nuevas características y optimizaciones, que permite personalizar para adicionar nuevas tácticas de razonamiento y la capacidad de razonar con lógicas descriptivas más potentes y cercanas a la expresividad de OWL-DL.

Clase inconsistente en Protégé

Clasificando con el razonador Pellet, Fact++ o HermiT



Condición necesaria y suficiente en Protégé