

Laboratorio de Ingeniería del Conocimiento con Owl/Protégé

Universidad de Valladolid
Escuela de Ingeniería Informática
Laboratorio de Ingeniería de Conocimiento
Curso 2022-2023

Organización de Grupos de Prácticas

- Grupos de 2, se mantienen las de las prácticas anteriores, salvo que alguien quiera cambiar.

PRIMERA SESION

Sesiones de Laboratorio:

- Sesión 1: 21 Noviembre (L) / 22 Noviembre (M)
- Sesión 2: 28 Noviembre (L) / 29 Noviembre (M)
- Sesión 3: 12 Diciembre(L) / 13 Diciembre (M)
- Sesión 4: 19 Diciembre (L) / 20 Diciembre (M)

Bibliografía de este Laboratorio

A Practical Guide to Building OWL Ontologies

Using Protégé 5.5 and Plugins

Edition 3.2

8 October 2021

Michael DeBellis

<https://www.michaeldebellis.com/post/new-protege-pizza-tutorial>

Evaluación de las prácticas

- Práctica 1: Ontología de Familias básica
- Práctica 2: Ontología de Familias avanzada
- Examen

Introducción: Casos de Uso de Ontologías

- Compartición de información científica
 - <https://www.nature.com/scitable/topicpage/ontologies-scientific-data-sharing-made-easy-77972/>
 - <http://geneontology.org/>
- Enriquecimiento de la gestión de inventario con ontologías
 - <https://www.oxfordsemantic.tech/blog/how-semantic-reasoning-can-enrich-the-features-offered-by-it-asset-management>
- Extracción de información automática y almacenamiento en formato orientado al almacenamiento de conocimiento
 - <https://es.dbpedia.org/>

Capturas de definición de ontología en el campo de la genética (protein information)

http://current.geneontology.org/ontology/subsets/goslim_pir.owl

The screenshot shows the Protégé interface with the 'goslim_pir' ontology loaded. The 'hasExactSynonym' property is selected in the left sidebar. The main pane displays the property's usage, showing 770 instances. The instances are listed in a table with columns for the property name, the domain, and the range. The domain is 'GO_0000003' and the range is 'reproductive physiological process'. The instances are: GO_0000003, GO_000128, GO_000178, GO_000307, GO_000346, and GO_000439. The description of the property is 'hasExactSynonym'. The domains and range are also shown at the bottom.

The screenshot shows the Protégé interface with the 'goslim_pir' ontology loaded. The 'RO_0002092' property is selected in the left sidebar. The main pane displays the property's usage, showing 11 instances. The instances are listed in a table with columns for the property name, the domain, and the range. The domain is 'BFO_0000050' and the range is 'BFO_0000051'. The instances are: BFO_0000050, BFO_0000051, BFO_0000066, RO_0002091, RO_0002093, RO_0002092, RO_0002211, RO_0002212, and RO_0002213. The description of the property is 'RO_0002092'. The domains and range are also shown at the bottom.

Modelado de estados de objetos en base a datos capturados por visión artificial

- Ejemplo de caso de uso que modela estado de objetos en un marco para detectar si están aislados, cercanos o solapándose.
- Se nutre de un sistema de aprendizaje automático que visualiza objetos en tiempo real con Yolo_v5 y que extrae información de localización y clase de objeto en un frame.
- La ontología, una vez diseñada, es alimentada con individuos y la información asociada a los mismos, de manera que deduce los estados.
- <https://www.oxfordsemantic.tech/blog/machine-learning-and-semantic-reasoning-the-perfect-union-using-object-detection-with-rdfox>

Introducción: Aplicación en empresas

Algunos puestos de trabajo en empresas van a manejar ontologías.
Ejemplos:

- Empleo en Danone: <https://www.linkedin.com/jobs/view/3320902110>
Data Engineering & Architecture Owner
 - API semantics and ontology management.
- Planet Pharma: <https://www.linkedin.com/jobs/view/3317614232>
Freelance Information Architect
 - Collaborate with the Data, Data Gov and **Ontology teams** on creating optimized models, complete with metadata and lineage information.

Sesion 1 - Agenda

- La herramienta Protégé
- Ontologías y Clases
- Propiedades

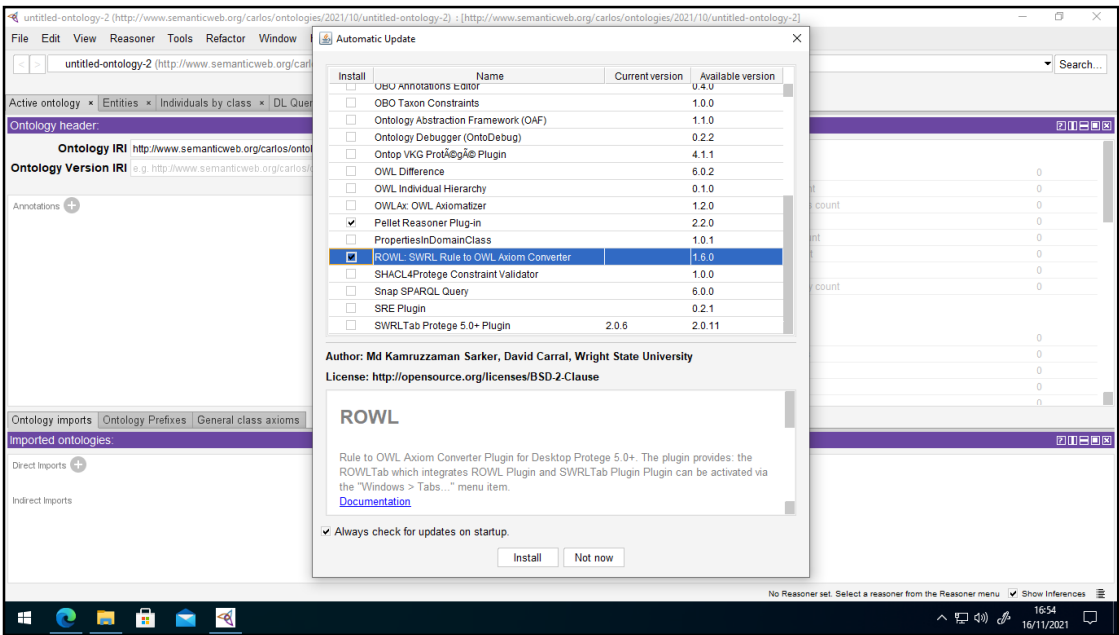
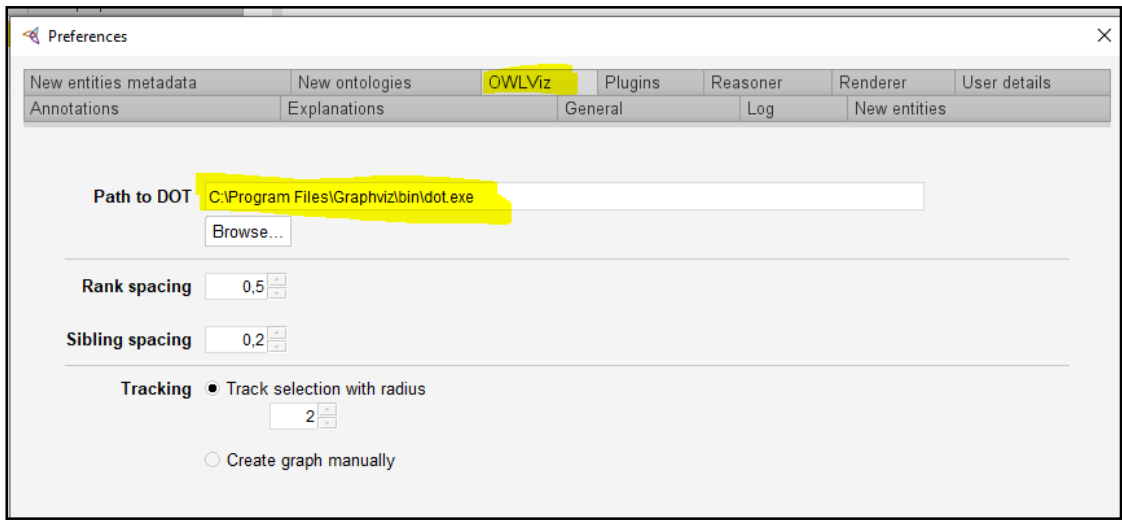
La Herramienta Protégé

Presentación de la Herramienta

- Protégé – desarrollado en la Universidad de Stanford para el Centro de Investigación de Informática Biomédica.
- Interfaz gráfico para el desarrollo de Ontologías.
- Permite trabajar con numerosos plugins para adaptarse a distintas necesidades, entre ellos razonadores con los que se pueden generar inferencias en base a las clases, propiedades y restricciones definidas por el usuario.

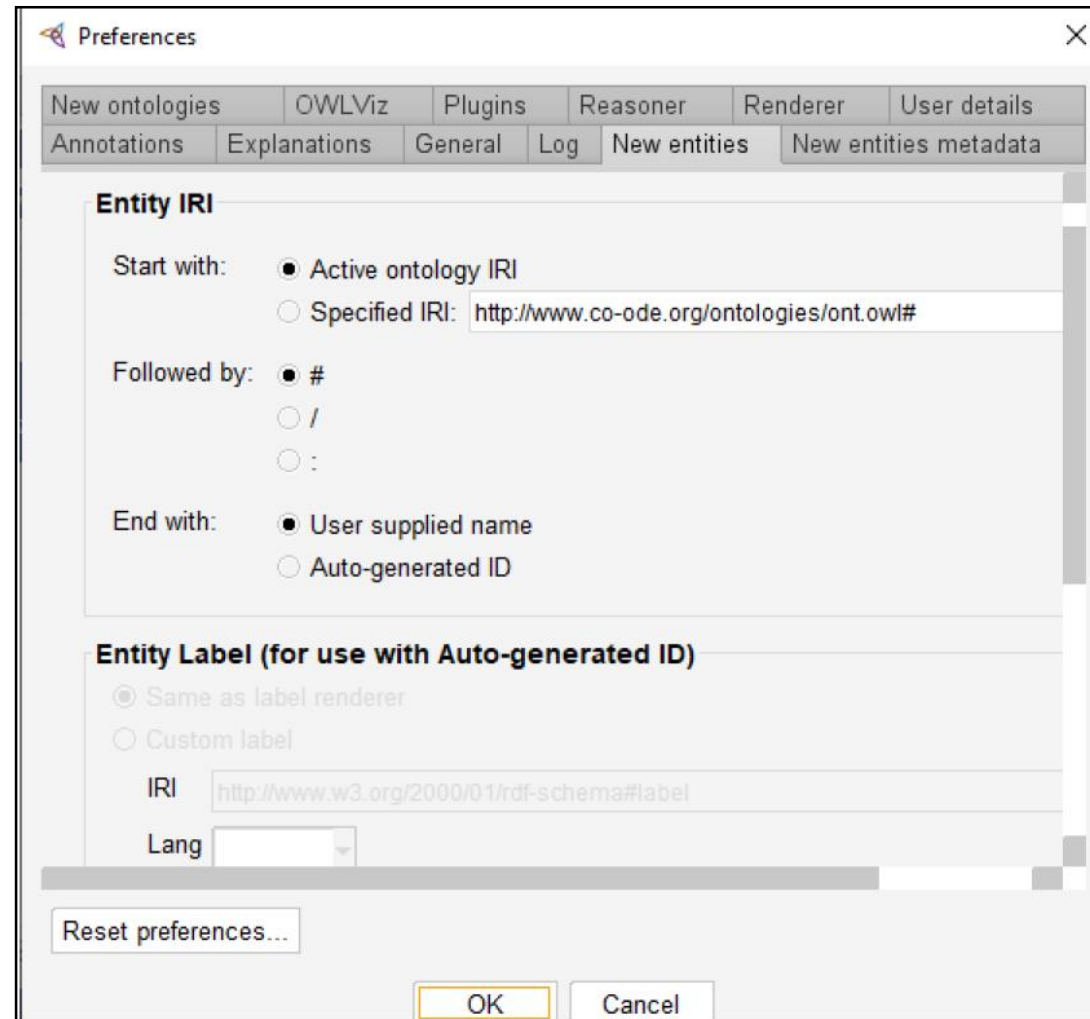
Configuración de la herramienta (I)

- Abrir Protégé. La aplicación va pedir actualización y plugins a instalar. Si aparecen como no instalados, seleccionar *Pellet Reasoner* y *ROWL: SWRL Rule to OWL Axiom Converter* (ver siguiente diapositiva)
- Configurar OWLViz si no está configurado
 - File → Preferences, en la pestaña OWLViz, hay que proporcionar Path to DOT, C:\Program Files\Graphviz\bin\dot.exe (Según instalación del software Graphviz)
- En las transparencias 15 a 18 se muestran el resto de elementos a configurar
- Si se ha hecho algún cambio, reiniciar Protégé
- En el menú Reasoner, seleccionar Pellet y activar el razonador tras el reinicio



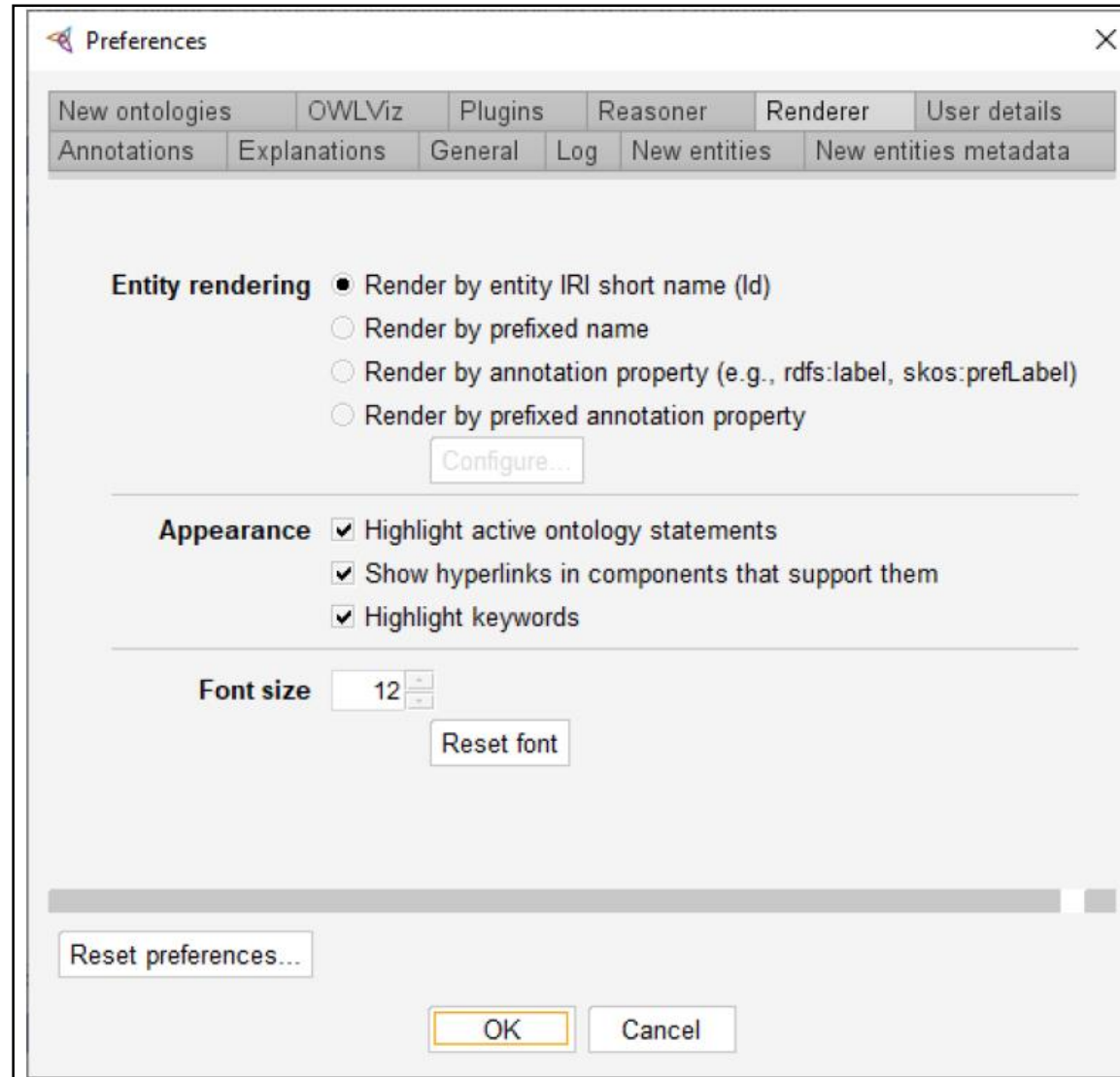
Configuración de la herramienta (II)

- File → Preferences



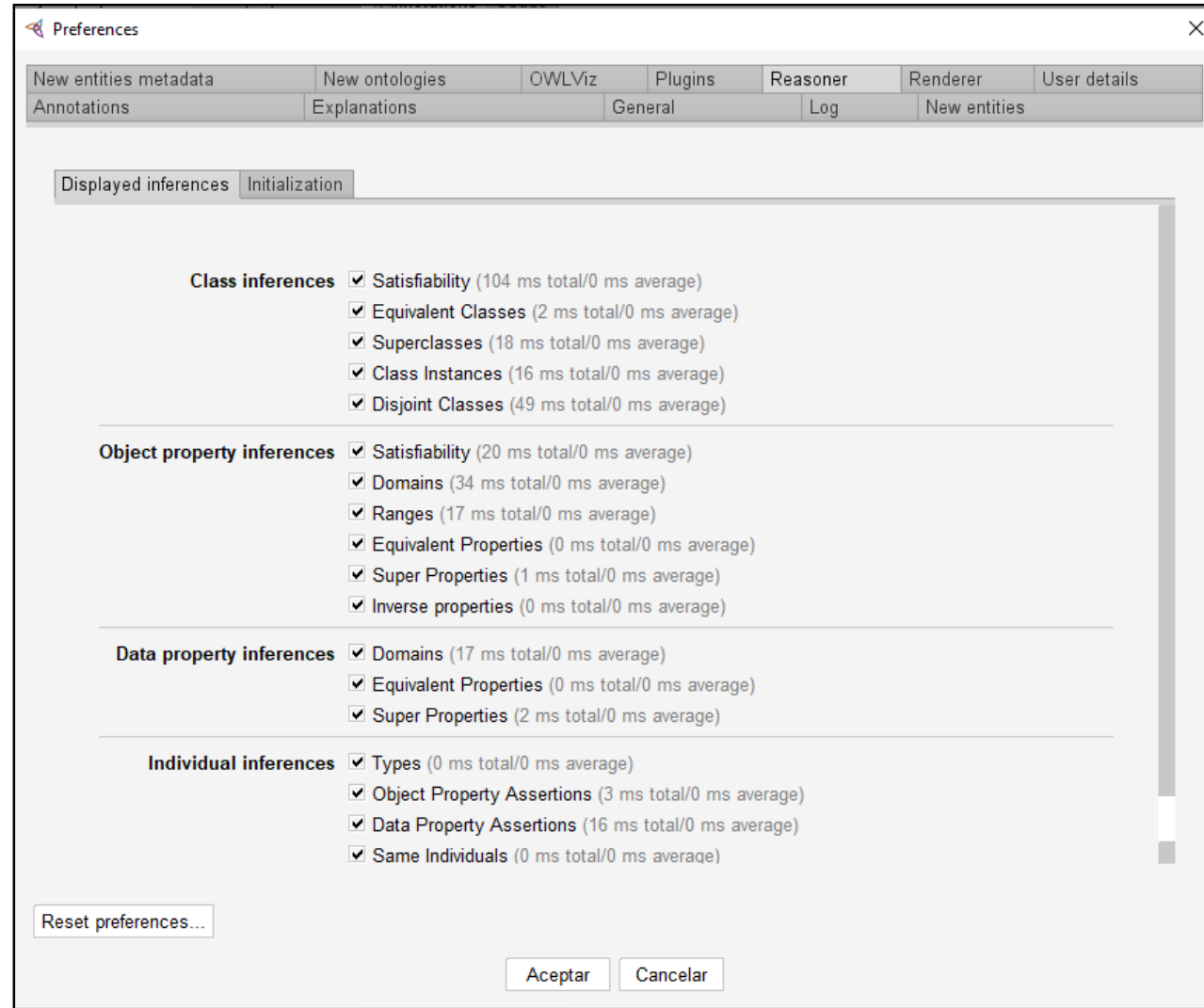
Configuración de la herramienta (III)

- File → Preferences



Configuración de la herramienta (IV)

- File → Preferences



Configuración de la herramienta (V)

- Podemos organizar las ventanas de la manera que más nos interese, activar o desactivar vistas y utilizar distintos tipos de solapado.
- Añadimos una pestaña principal (Individuals By class)
 - Seleccionamos **Windows > Tabs > Individuals by class**.
- Añadimos una ventana a una ventana principal para usarlas más adelante:
 - Nos situamos en la vista Entities y en la pestaña Classes. Seleccionamos **Window > Views > Class Views > Usage** (permite ver dónde se ha usado, seguir la traza, esa clase).
 - Aparecerá un **punto negro (ver siguientes dispositivas)** La ubicación de la vista seleccionada, se situará dependiendo de donde coloquemos el **punto negro**. Probar en distintas zonas de la ventana, porque se pueden solapar u organizar horizontal o verticalmente.
 - Seleccionamos **Window > Views > Individual Views > Individuals by type (inferred)** .
 - Ejemplo: añadirla junto con Annotations y Usage que se encuentran en la parte derecha.
- Para cerrar vistas, simplemente, hacer clic en la X de la esquina correspondiente.

pizza (http://www.semanticweb.org/sergg/ontologies/2021/10/pizza) : [E:\apache-tomcat-10.0.11\webapps\pizza-lab-checkpoint-3.owl]

File Edit View Reasoner Tools Refactor Window Help

pizza (http://www.semanticweb.org/sergg/ontologies/2021/10/pizza) Search...

Active ontology x Entities x Classes x Individuals by class x OWLViz x DL Query x OntoGraf x SWRLTab x SQWRLTab x

Classes Object properties Data properties Annotation properties Datatypes Individuals

Class hierarchy: owl:Thing

owl:Thing

- Pizza
- PizzaBase
- PizzaTopping

Annotations: owl:Thing

Annotations +

Description: owl:Thing

Equivalent To +

SubClass Of +

General class axioms +

SubClass Of (Anonymous Ancestor)

Instances +

Target for Key +

Disjoint With +

Disjoint Union Of +

To use the reasoner click Reasoner > Start reasoner ☒ Show Inferences ⓘ

pizza (http://www.semanticweb.org/sergg/ontologies/2021/10/pizza) : [E:\apache-tomcat-10.0.11\webapps\pizza-lab-checkpoint-3.owl]

File Edit View Reasoner Tools Refactor Window Help

Active ontology x Entities x Classes x Individuals by class x OWLViz x DL Query x OntoGraf x SWRLTab x SQWRLTab x

Classes | Object properties | Data properties | Annotation properties | Datatypes | Individuals

Class hierarchy: owl:Thing

- owl:Thing
 - Pizza
 - PizzaBase
 - PizzaTopping

Annotations: owl:Thing

Annotations +

Usage: owl:Thing

Show: ☒ this ☒ disjoints ☒ named sub/superclasses

Found 0 uses of owl:Thing

Description: owl:Thing

Equivalent To +

SubClass Of +

General class axioms +

SubClass Of (Anonymous Ancestor)

Instances +

Target for Key +

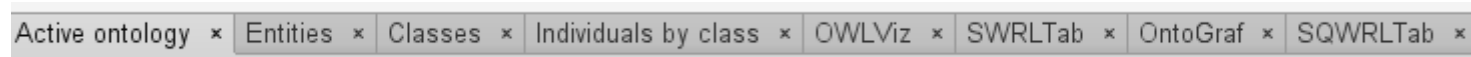
Disjoint With +

Disjoint Union Of +

To use the reasoner click Reasoner > Start reasoner ☒ Show Inferences

Vistas de Protégé

En la parte superior localizamos las vistas principales. Utilizaremos estas en el laboratorio.



- *Active Ontology*, el IRI de la ontología (que es el recurso al que hacer referencia desde otras Ontologías), muestra la descripción de la ontología, incluye métricas, prefijos para el uso de otras ontologías y recursos.
- *Entities* es la vista desde la que se accede de manera directa a las definiciones de clases, propiedades de objetos, propiedades de datos e individuos. Es la que más usaremos durante el laboratorio.

Paseo por la herramienta (Vistas útiles)

- Navegar desde la pestaña Entities (si alguna no aparece, activarla siguiendo los pasos descritos antes).
 - Classes – desde donde creamos la jerarquía de clases
 - Annotations
 - Equivalent To (nos servirá más adelante cuando hablemos de Open World Assumption vs Closed World Assumption)
 - SubClass Of, que define las características y restricciones de la clase
 - Instances
 - Disjoint With, para definir que si se es de esta clase, NO se puede ser de las otras
 - Object Properties – desde donde crearemos las relaciones entre clases
 - Annotations – importante documentar, porque en ocasiones la descripción podría ser ambigua sobre a qué clase se refiere
 - Characteristics – funcional, transitivas, simétricas, reflexivas, etc.
 - Inverse Of – para representar relaciones inversas, práctico para simplificar la compresión en ocasiones
 - Domain – La clase “origen” de la relación
 - Ranges – La clase “destino” de la relación
 - Data Properties – desde donde crearemos las relaciones entre clases y tipos de datos básicos (int, boolean, ...)
 - Individuals – desde donde podemos ver las instancias de cada clase y los valores que tienen asociados
 - Interesante ver Individuals by class para elegir con qué vista estamos más cómodos
- Navegar brevemente por OWLViz para ver jerarquía de clases, OntoGraph para ver relaciones de clases e incluso instancias/individuos, SWRLTab y SQWRLTab en las que se localizan las reglas que nos permitirán enriquecer el conocimiento que aporta la ontología.

Ontologías y Clases

Creamos nuestra primera Ontología

- Vista Active Ontology (ver siguiente diapositiva).
- Ponemos el nombre Ontology IRI: pizza.
- Añadimos una Anotación (Normalmente son metadatos, tales como comentarios o etiquetas, útiles para documentación y búsqueda).
- Revisamos, en las pestañas de abajo y observamos los prefijos, que ayudan a reducir el tamaño del XML que se genera por debajo.

pizza (http://www.semanticweb.org/sergg/ontologies/2021/10/pizza) : [http://www.semanticweb.org/sergg/ontologies/2021/10/untitled-ontology-23]

File Edit View Reasoner Tools Refactor Window Help

< > pizza (http://www.semanticweb.org/sergg/ontologies/2021/10/pizza)

Active ontology x Entities x Classes x Individuals by class x OWLViz x DL Query x OntoGraf x SWRLTab x SQWRLTab x

Ontology header:

Ontology IRI <http://www.semanticweb.org/sergg/ontologies/2021/10/pizza>

Ontology Version IRI e.g. <http://www.semanticweb.org/sergg/ontologies/2021/10/untitled-ontology-23/1.0.0>

Annotations +

[rdfs:comment](#)

Ontology to describe pizza world

Ontology imports Ontology Prefixes General class axioms

Ontology prefixes:

Prefix	Value
owl	http://www.w3.org/2002/07/owl#
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs	http://www.w3.org/2000/01/rdf-schema#
xml	http://www.w3.org/XML/1998/namespace
xsd	http://www.w3.org/2001/XMLSchema#

Nuestra Primera Clase

- Convención de Nombrado: Joroba de Camello (CamelBack)
 - Palabras empezando con mayúscula sin espacios
 - No usamos plurales para los nombres de las clases
- ¿Cómo cambiar el nombre de una clase, una propiedad, o una instancia?
 - Buscar las 3 líneas horizontales en las vistas
- Seleccionar la vista Entities y la pestaña Classes
- Creamos la clase *Pizza* seleccionando *Owl:Thing*, dando al botón derecho y seleccionando *Add subclass*
- RECOMENDACIÓN:
 - Grabamos en formato *Owl Functional Syntax* con frecuencia
 - Al hacer seleccionar File->Save (o Save As) la primera vez nos pide el formato. Seleccionar el indicado
 - Sincronizamos el razonador con mucha frecuencia para identificar cuando cometemos un error
 - Seleccionar Reasoner->Synchronize reasoner

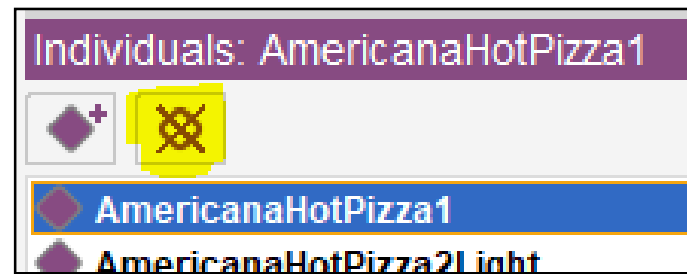
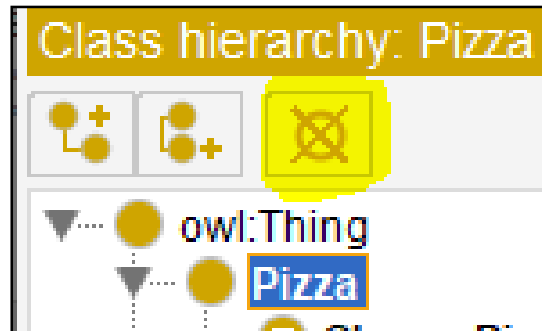
Atajo para crear jerarquías de clases

- Desde la vista Entities, pestaña Classes
 - Tools > Create Class Hierarchie
 - O sobre un objeto, clic derecho > Add Subclasses
- Desde la pestaña de Object/Data Properties
 - Tools > Create Object/Data Property hierarchie

Se definen usando una tabulación para indicar subclases.

Para eliminar clases o individuos, hay que buscar el icono con el aspa (ver siguiente dispositiva).

Eliminar clases, individuos...



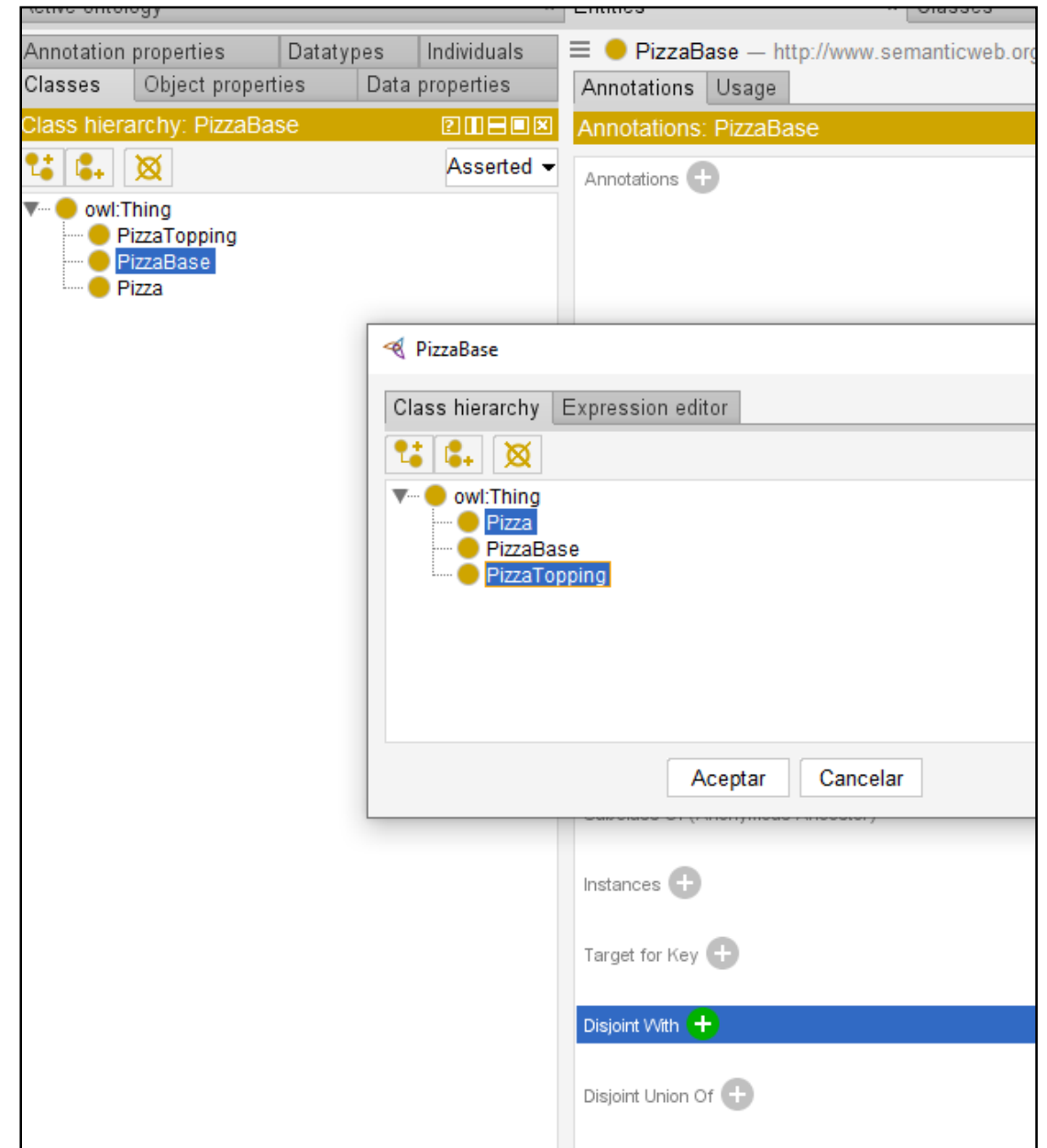
Añadimos dos clases más

Ambas subclases de owl:Thing

- PizzaBase
 - PizzaTopping
-
- En la siguiente página se muestra cómo convertirlas en disjuntas:
 - Clases disjuntas: no pueden tener un elemento que sea instancia de ambas clases. Dicho de otra manera, indicar que un individuo o extensión pertenece a ambas, hace la ontología inconsistente.

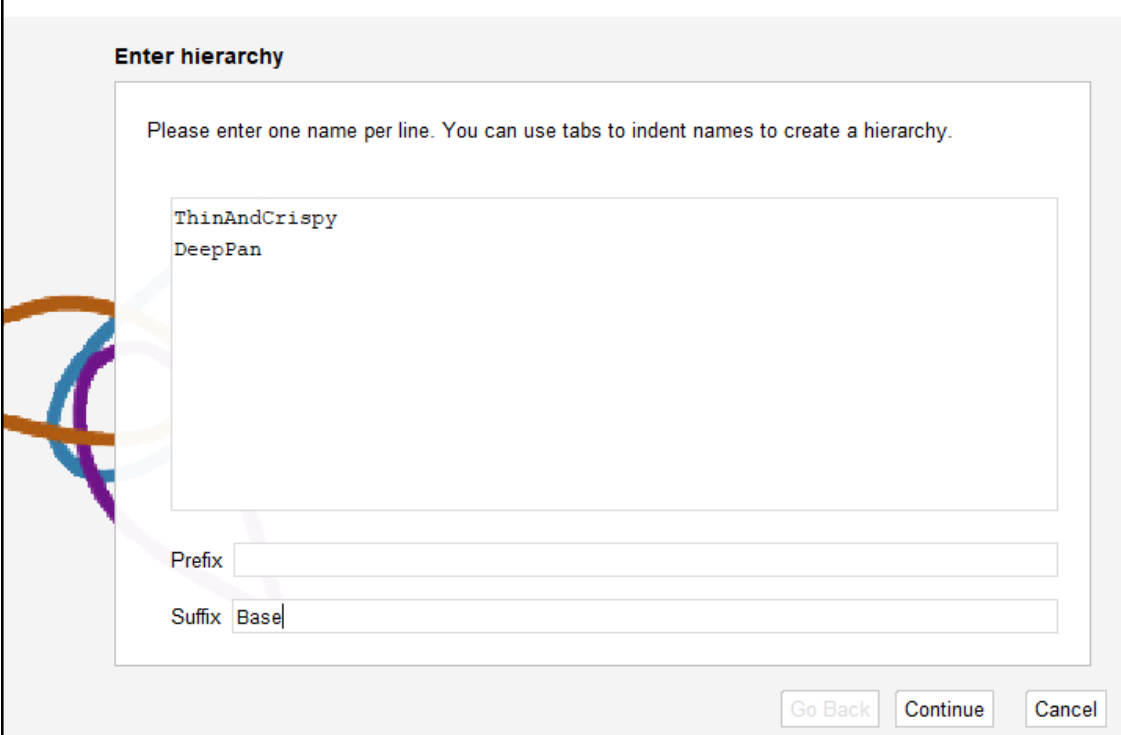
Clases disjuntas

- La manera de enriquecer una ontología es añadir toda la información posible para que la consistencia sea máxima
- Indicar que las clases son disjuntas
 - Pizza, PizzaBase, PizzaTopping
- Asegurarse primero de seleccionar una clase para tener la referencia
 - Seleccionar la clase PizzaBase
 - Seleccionar, a continuación, el signo + al lado de Disjoint with (ventana Description a la derecha abajo)
 - Al dar al +, aparece una nueva ventana con varios pestañas. Seleccionar *Class Hierarchie* y elegir las otras dos clases. La opción *Expression* editor la veremos más adelante



Crear Jerarquía de clases usando Sufijos

- Para agilizar la creación de clases y disponer de una estandarización de nombres, se recomienda usar prefijos y sufijos comunes (cuando el idioma y la circunstancia lo permita)
- Clase Padre: PizzaBase ➔ Add subclasses
 - ThinAndCrispy
 - DeepPan
 - Suffix: Base
- Tras dar a Continue, aceptaremos la opción que viene por defecto de hacer las clases disjuntas (porque una base de pizza no puede ser de los dos tipos a la vez)



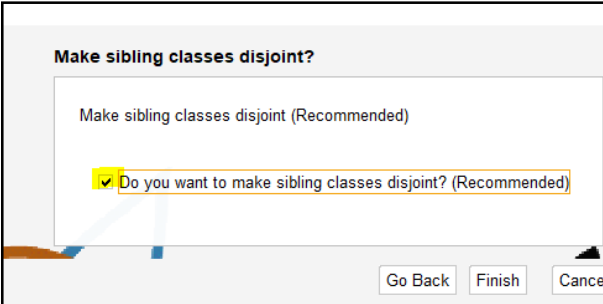
Enter hierarchy

Please enter one name per line. You can use tabs to indent names to create a hierarchy.

```
ThinAndCrispy
DeepPan
```

Prefix

Suffix

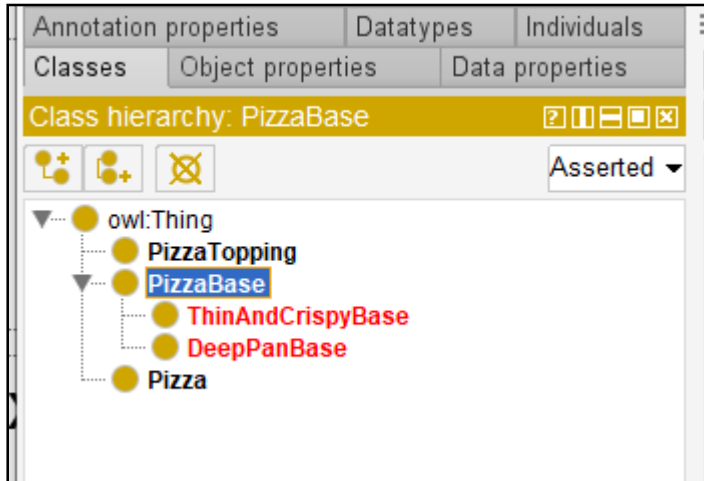


Make sibling classes disjoint?

Make sibling classes disjoint (Recommended)

☒ Do you want to make sibling classes disjoint? (Recommended)

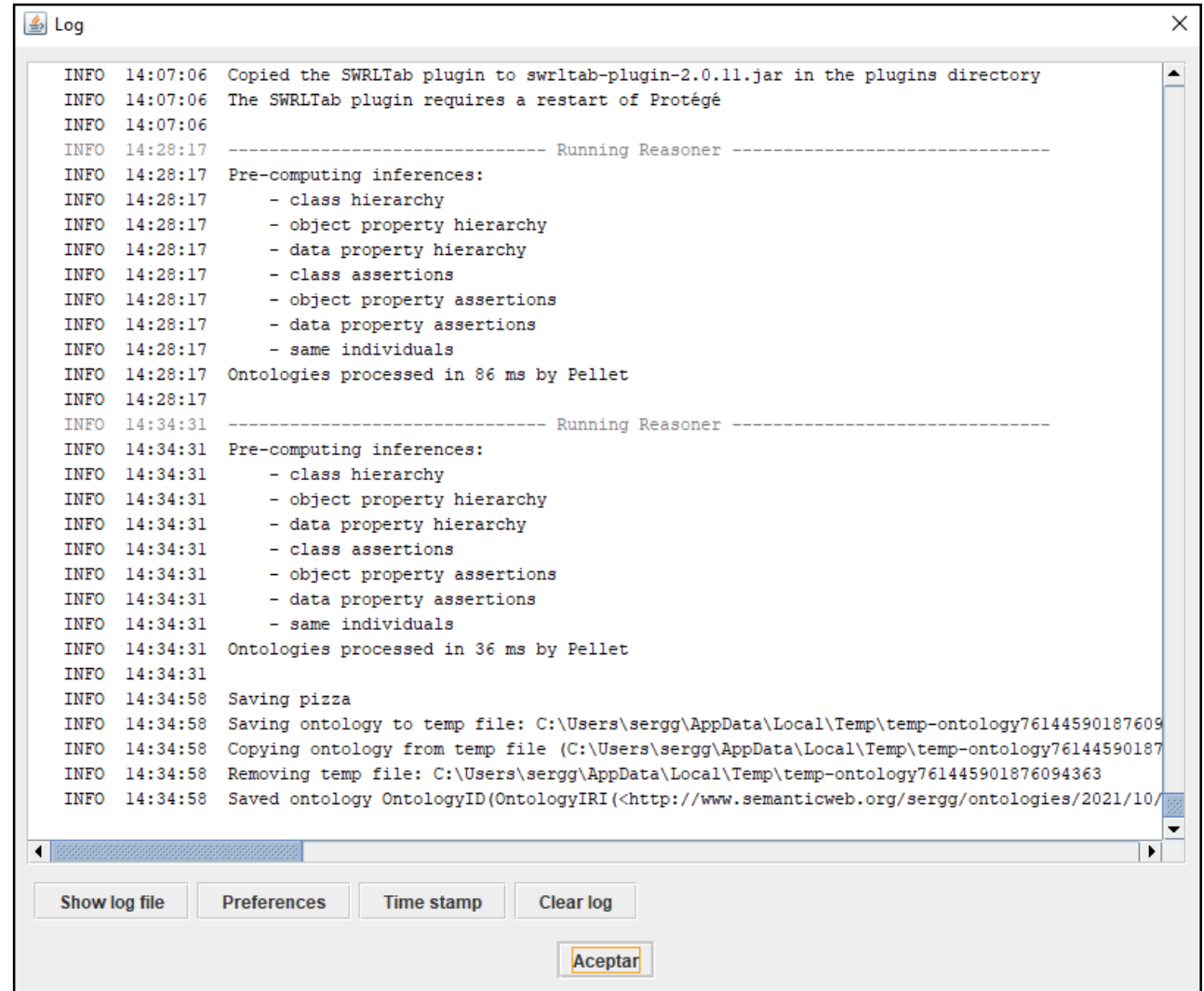
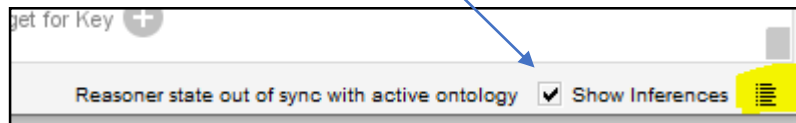
Jerarquía de clases



- Si las clases aparecen en rojo es necesario resincronizar el razonador para validarlas
- Si no está arrancado el razonador : Menu > Reasoner > Start reasoner
- Si ya está arrancado el razonador : Menu > Reasoner > Synchronize reasoner
- RECOMENDACIÓN:
 - Si el razonador detecta una inconsistencia en la ontología, detener el razonador (Menu > Reasoner > Stop reasoner) y refrescar el interfaz (Menu > Window > Refresh user Interface)
 - El log del razonador está en la parte inferior derecha de la ventana (ver transparencia siguiente)

Estado del razonador

- En el log del razonador podremos ver información relevante que nos ayudará a identificar donde hemos introducido una inconsistencia
- Aviso: Asegurarnos de que esté Activo el “Show Inferences”

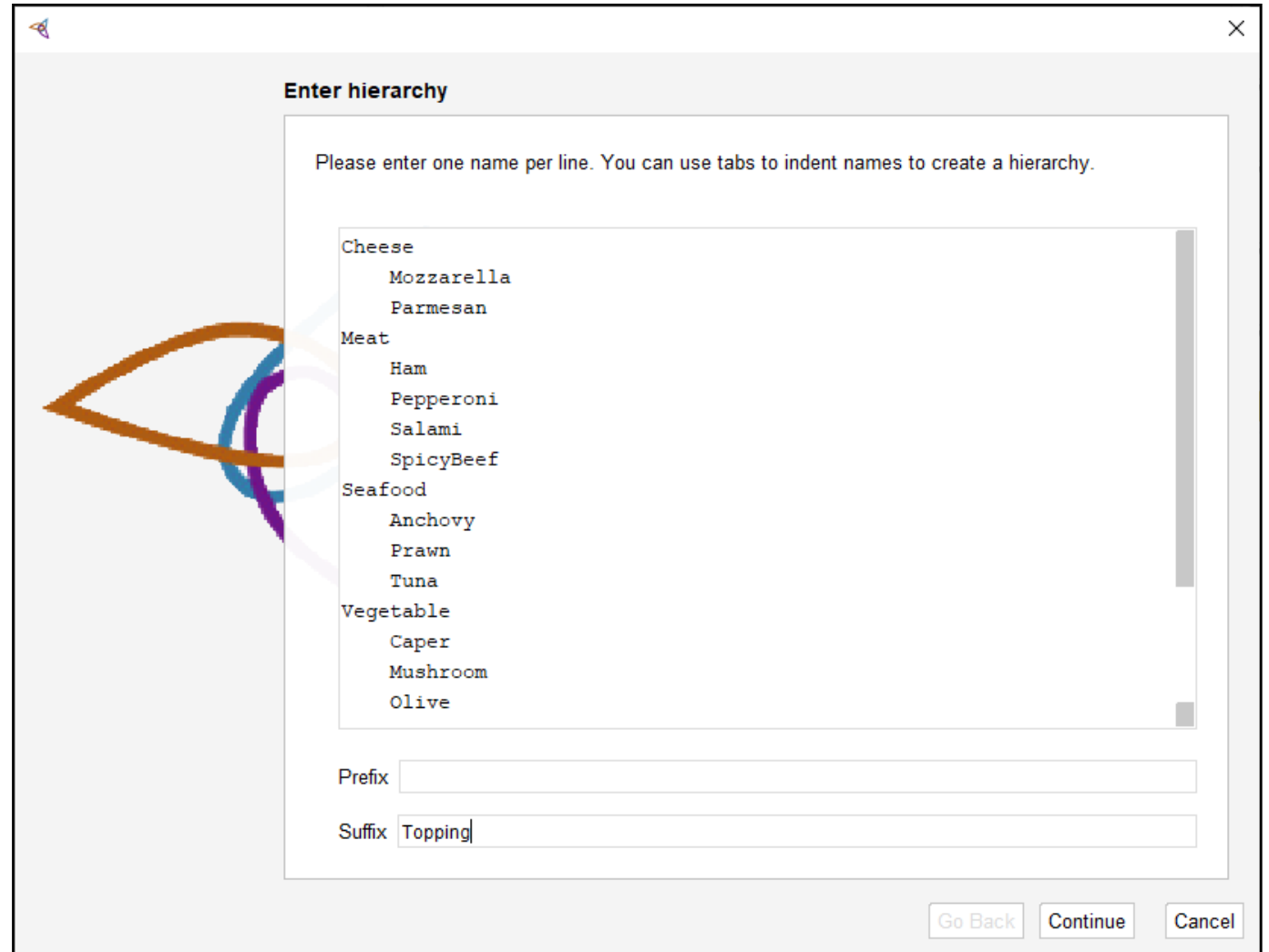


Crear Jerarquía de clases usando Sufijo

- Para agilizar la creación de jerarquía de clases, podemos hacerlo en una sola operación.
- Desde la vista Entities -> Classes, seleccionamos PizzaTopping clic derecho → Add subclasses
- Usamos el tabulador para indicar subclases de la anterior.

Cheese
 Mozzarella
 Parmesan
Meat
 Ham
 Pepperoni
 Salami
 SpicyBeef
Seafood
 Anchovy
 Prawn
 Tuna
Vegetable
 Caper
 Mushroom
 Olive
 Pepper
 RedPepper
 GreenPepper
 JalapenoPepper
Tomato

- Suffix: Topping
- Tras dar a Continue, aceptaremos la opción que viene por defecto de hacer las clases disjuntas **porque en este caso son así, pero podrían no serlo, en cuyo caso habría que desactivar la opción.**



The screenshot shows a dialog box titled "Enter hierarchy" with a close button (X) in the top right corner. Inside the dialog, there is a text instruction: "Please enter one name per line. You can use tabs to indent names to create a hierarchy." Below this is a text area containing a list of categories and their sub-items, indented with tabs: Cheese, Mozzarella, Parmesan, Meat, Ham, Pepperoni, Salami, SpicyBeef, Seafood, Anchovy, Prawn, Tuna, Vegetable, Caper, Mushroom, Olive. At the bottom of the dialog, there are two input fields: "Prefix" (empty) and "Suffix" (containing the text "Topping"). To the right of these fields are three buttons: "Go Back", "Continue", and "Cancel". The "Continue" button is highlighted with a blue border. A large, stylized eye icon is overlaid on the left side of the dialog box.

pizza (http://www.semanticweb.org/sergg/ontologies/2021/10/pizza) : [E:\apache-tomcat-10.0.11\webapps\pi...

File Edit View Reasoner Tools Refactor Window Help

< > pizza (http://www.semanticweb.org/sergg/ontologies/2021/10/pizza) Search...

PizzaTopping

Individuals by class x OWLViz x DL Query x OntoGraf x SWRLTab x SQWRLTab x

Active ontology x Entities x Classes x

Annotation properties Datatypes Individuals

Classes Object properties Data properties

Class hierarchy: PizzaTopping [?] [I] [E] [X]

Annotations Usage

Annotations: PizzaTopping [?] [I] [E] [X]

Annotations +

Description: PizzaTopping [?] [I] [E] [X]

Equivalent To +

SubClass Of +

General class axioms +

SubClass Of (Anonymous Ancestor)

Instances +

Target for Key +

Reasoner state out of sync with active ontology ☒ Show Inferences [?]

Class hierarchy: PizzaTopping

- owl:Thing
 - PizzaTopping
 - VegetableTopping
 - CaperTopping
 - MushroomTopping
 - OliveTopping
 - PepperTopping
 - TomatoTopping
 - CheeseTopping
 - MozzarellaTopping
 - ParmesanTopping
 - SeafoodTopping
 - AnchovyTopping
 - PrawnTopping
 - TunaTopping
 - MeatTopping
 - HamTopping
 - PepperoniTopping
 - SalamiTopping
 - SpicyBeefTopping
 - PizzaBase
 - ThinAndCrispyBase
 - DeepPanBase

Propiedades

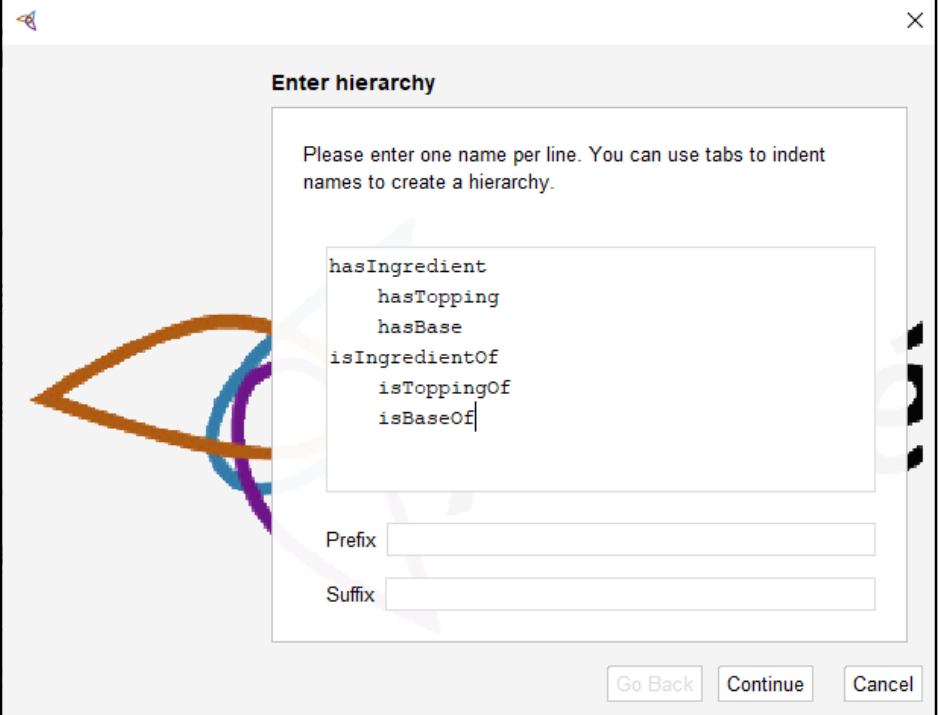
Propiedades en OWL

- Tres tipos de propiedades:
 - Object Properties: Se utilizan para relacionar objetos/individuos
 - Ej. Nick → Trabaja con → Matthew. *Nick* y *Matthew* son los objetos y *Trabaja con* la propiedad que los relaciona
 - Data Properties: Relacionan a los individuos con los valores de los datos (siendo los objetos más básicos)
 - Ej. Lemon Computer → Has color → Green. *Lemon Computer* es el objeto, *Green* es el dato y *Has color* la propiedad que los relaciona
 - Anotaciones (comentarios o etiquetas)
- Representados por debajo por tripletas
- Propiedad base: owl:topObjectProperty
- Existe jerarquía de propiedades. Ejemplo:
 - Property: tieneProgenitor
 - Sub Properties: tienePadre / tieneMadre

Creamos propiedades y sus correspondientes inversas

- Creamos la jerarquía de propiedades:
 - hasIngredient
 - hasTopping
 - hasBase
- Creamos la jerarquía inversa:
 - isIngredientOf
 - isToppingOf
 - isBaseOf

(Ver pestañas siguientes el paso a paso)



Enter hierarchy

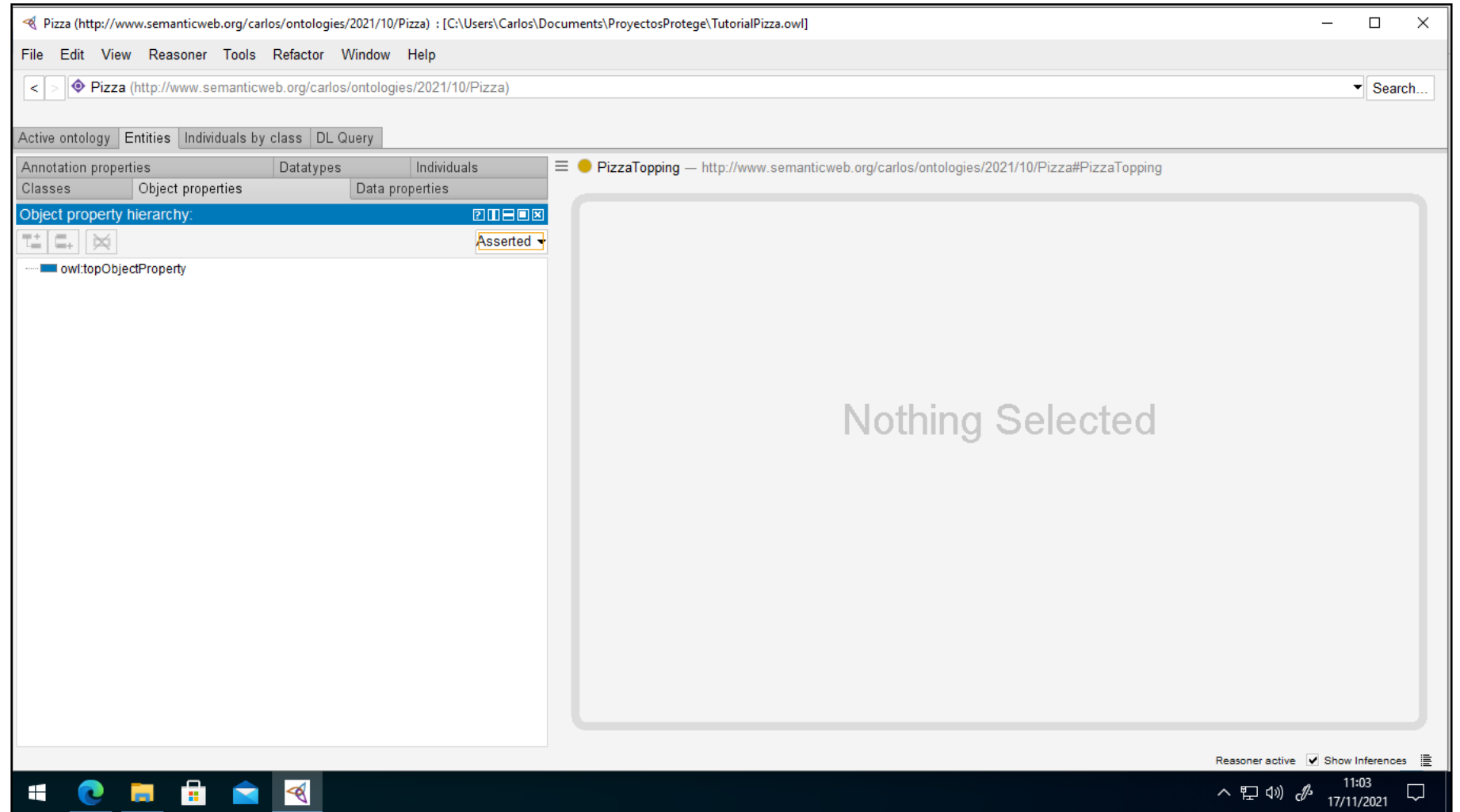
Please enter one name per line. You can use tabs to indent names to create a hierarchy.

```
hasIngredient
  hasTopping
  hasBase
isIngredientOf
  isToppingOf
  isBaseOf
```

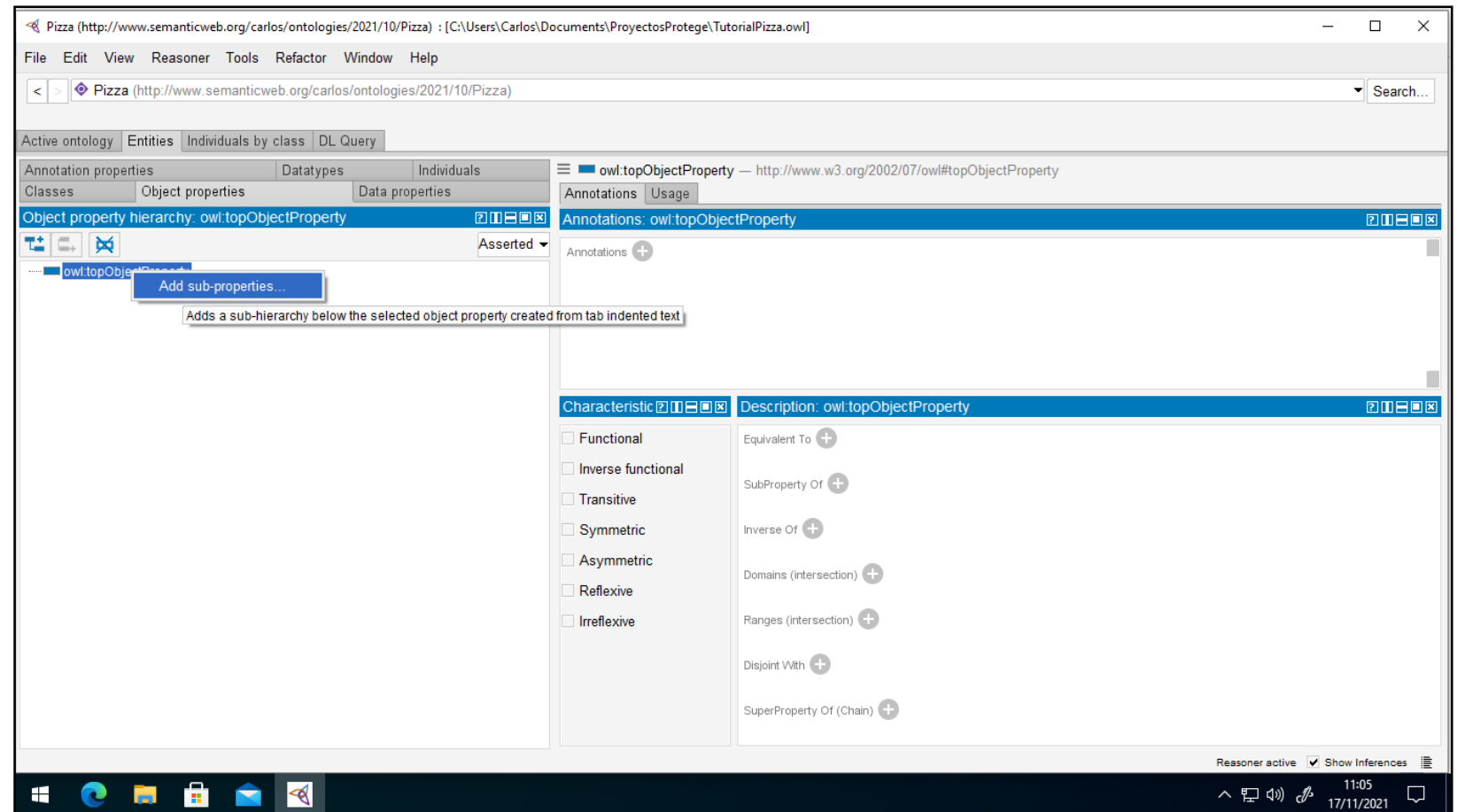
Prefix

Suffix

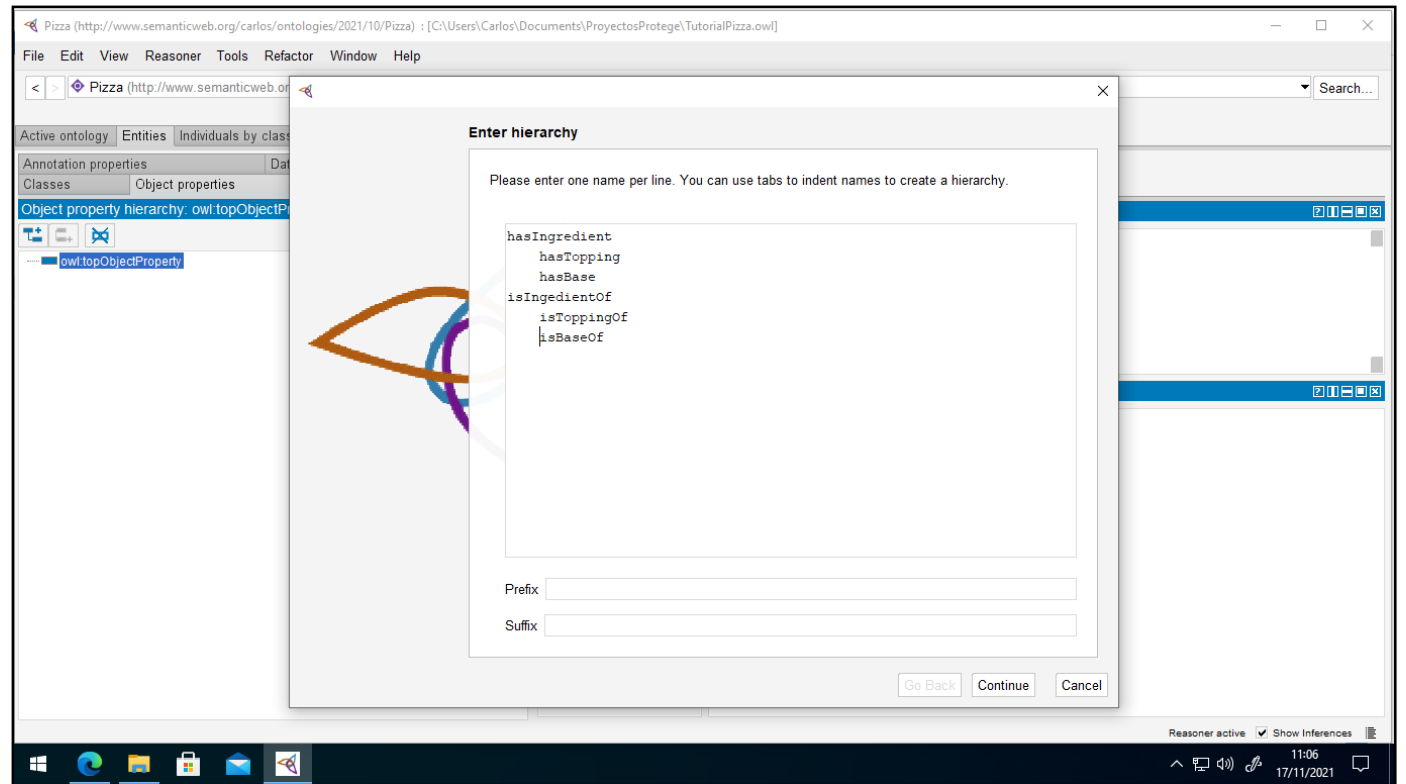
- Situarse en la vista Entities > Object Properties



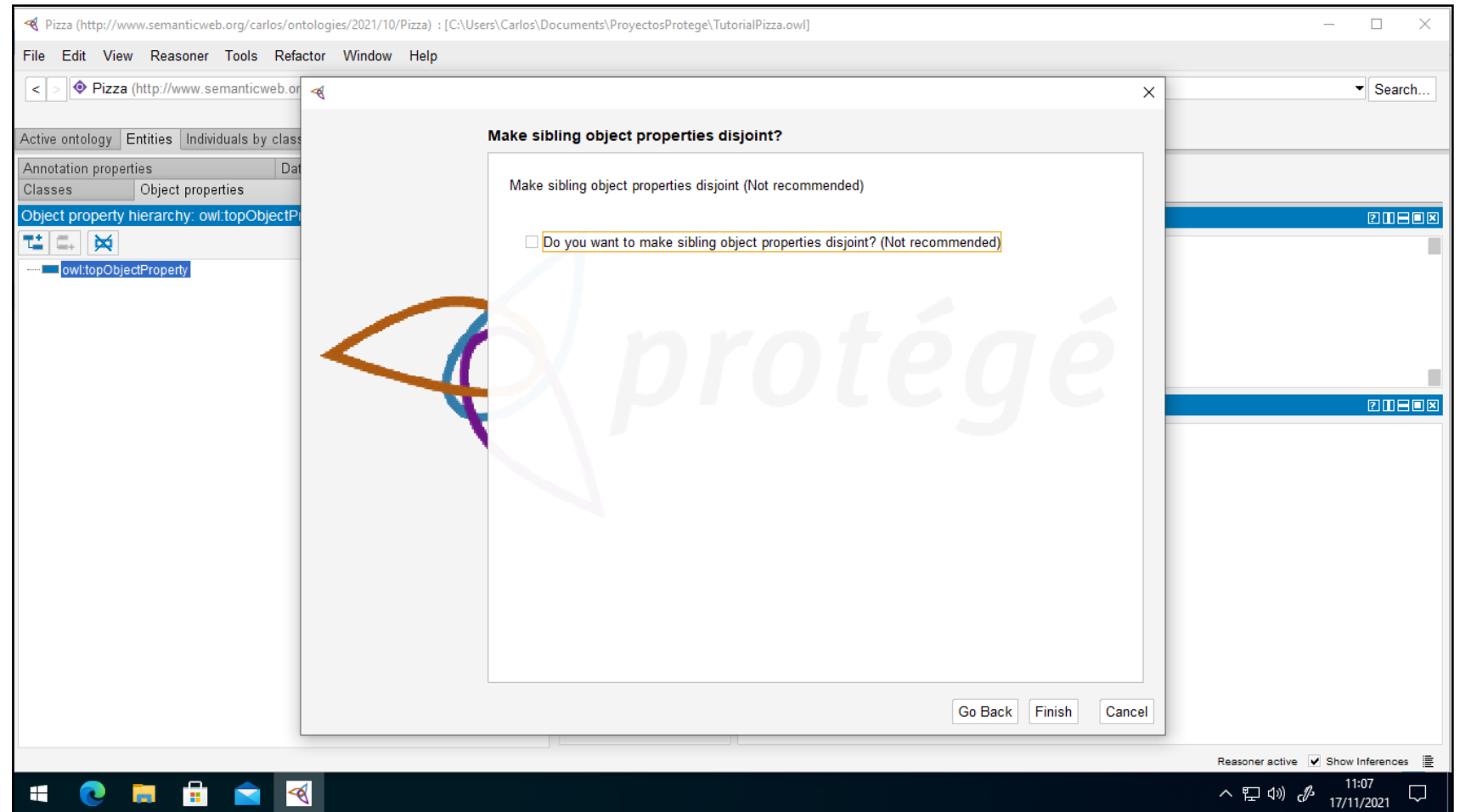
- Seleccionar el objeto `owl:topObjectProperty` y hacer clic derecho para seleccionar Add sub properties



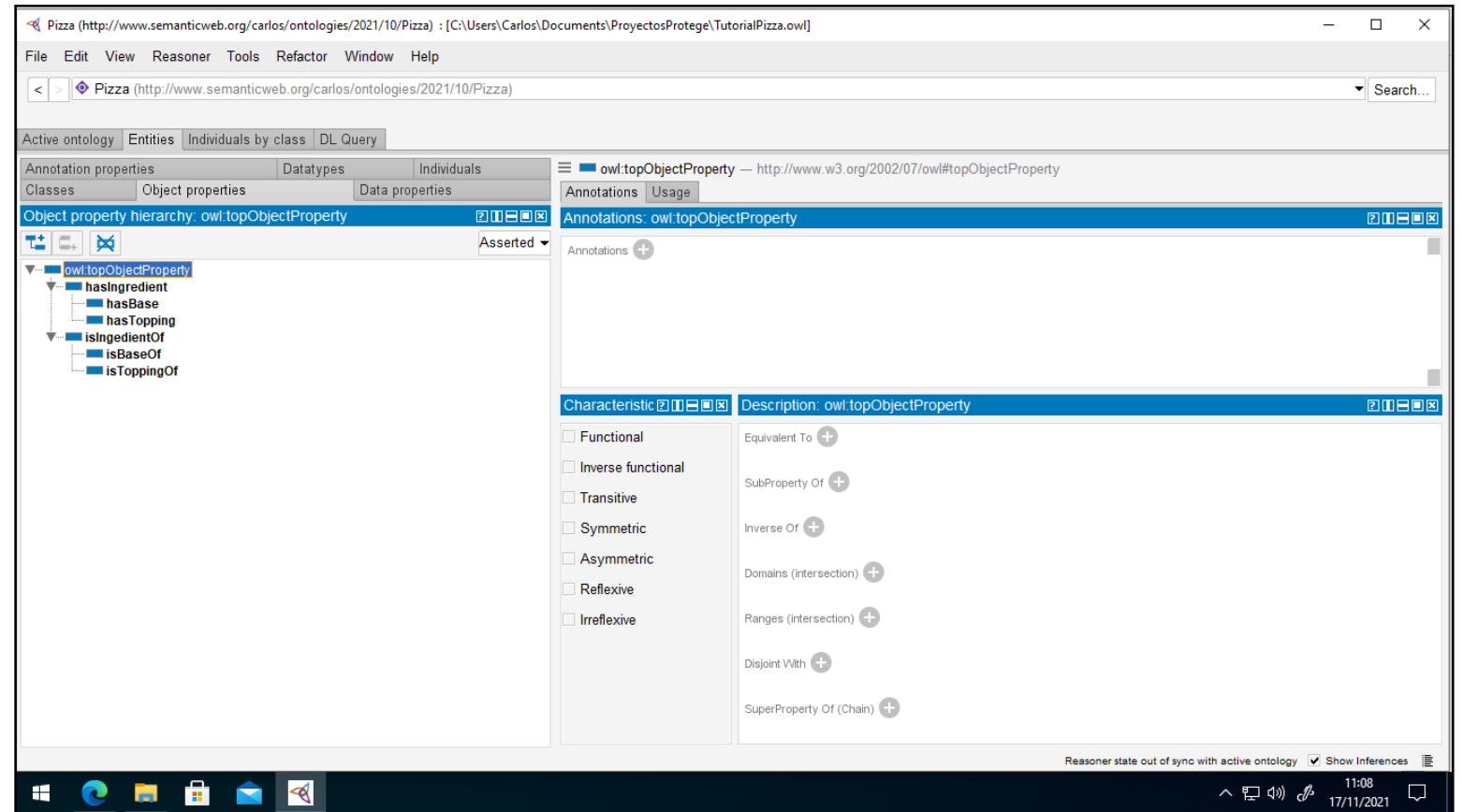
- Introducir la jerarquía, separada por tabuladores y hacer clic en Continue



- Dejar sin marcar la opción por defecto **porque las propiedades en este caso no son disjuntas.**

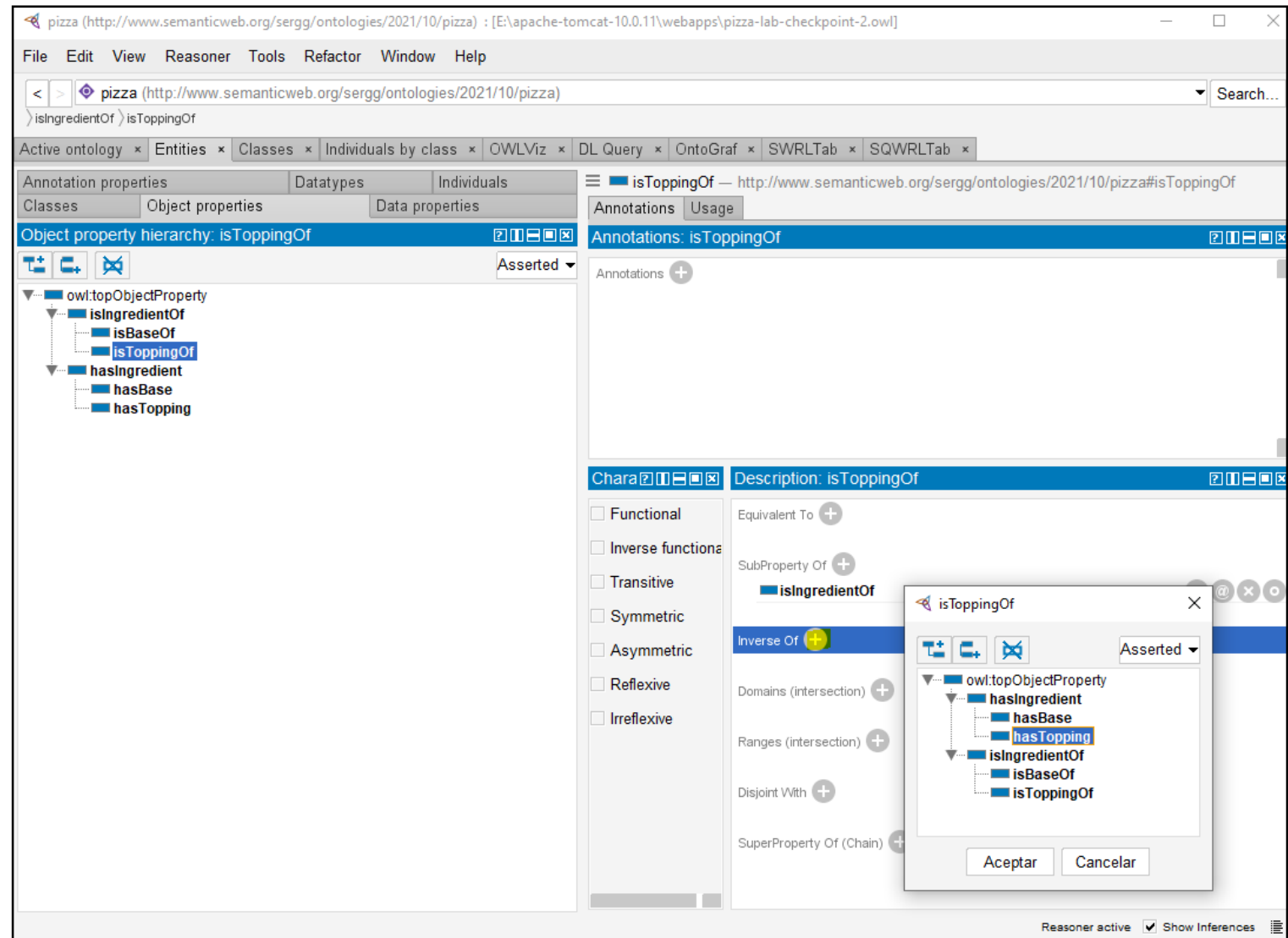


- Comprobar el resultado



Creamos propiedades y sus correspondientes inversas (II)

- Desde la vista de Entities, Pestaña Object Properties, siempre con la opción Asserted, establecemos una relación inversa entre las propiedades:
 - Para hacerlo, seleccionar el + al lado de InverseOf en la descripción (abajo a la derecha)
- isIngredientOf es inversa de hasIngredient
- isBaseOf es inversa de hasBase
- isToppingOf es inversa de hasTopping



Acciones sobre propiedades definidas por el usuario e inferidas por el razonador



- El contenido de esta imagen se observará más adelante. Ahora solamente ver el uso de los 4 símbolos de la derecha.

- La ? Localiza la explicación del razonador para llegar a esa conclusión
- La X elimina la restricción
- La @ muestra las anotaciones
- El círculo de la derecha permite la edición

Detectando clases que no pueden tener miembros

- Cuando el razonador detecta que una clase se ha definido de manera que no es posible crear ninguna instancia, el razonador, definirá una equivalencia a **owl:Nothing**
 - Normalmente, cuando esto pasa es que hay algún error. Es indicado porque la inferencia aparece en rojo. Si esto ocurre:
 - Sincronizar el razonador
 - Si el error persiste, parar y detectar el error antes de continuar. Para ello es útil ver el razonamiento seguido (clic en el símbolo “?” a la derecha de la inferencia), para obtener lo inferido.

Otras características de las propiedades

- Funcionales: aquella que tiene como mucho un valor para cualquier sujeto en particular (Ej., tieneFechaDeNacimiento)
- Funcional inversa: Ej., tieneISBN vs esISBNde
- Transitivas: Ej., esAntecesor
- Simétricas: Ej., esHermano
- Asimétricas: Ej., esMadre (nunca A puede ser madre de B y a la vez B de A)
- Reflexivas: Ej., investigaInstitución (una institución que se puede investigar a sí misma)
- Irreflexivas: Ej., esHermanaDe

El razonador, cuando está activo y sincronizado, SIEMPRE resolverá las propiedades según sus características y generará las inferencias marcándolas con fondo amarillo (**PizzaBase**)

La información introducida por el usuario, aparecerá en negrita con fondo neutro (**hasBase**)
➔ (ver página siguiente)

[pizza \(http://www.semanticweb.org/sergg/ontologies/2021/10/pizza\)](http://www.semanticweb.org/sergg/ontologies/2021/10/pizza) : [E:\apache-tomcat-10.0.11\webapps\pizza-lab-checkpoint-4...

File Edit View Reasoner Tools Refactor Window Help

[<](#) [>](#) [pizza \(http://www.semanticweb.org/sergg/ontologies/2021/10/pizza\)](#) Search...

Active ontology x Entities x Individuals by class x DL Query x

Individuals
 Annotation properties Datatypes
 Data properties
 Classes Object properties
 Object property hierarchy

hasBase — <http://www.semanticweb.org/sergg/ontologies/2021/10/pizza#hasBase>

Annotations Usage

Annotations: hasBase

Annotations +

owl:topObjectProperty
 hasIngredient
 hasBase
 hasTopping
 isIngredientOf
 isBaseOf
 isToppingOf

Charac Description: hasBase

☐ Functional
☐ Inverse functional
☐ Transitive
☐ Symmetric
☐ Asymmetric
☐ Reflexive
☐ Irreflexive

Equivalent To +

SubProperty Of +

hasIngredient

Inverse Of +

isBaseOf

Domains (intersection) +

Pizza

Ranges (intersection) +

PizzaBase

Disjoint With +

SuperProperty Of (Chain) +

Reasoner active ☒ Show Inferences

[pizza \(http://www.semanticweb.org/sergg/ontologies/2021/10/pizza\)](http://www.semanticweb.org/sergg/ontologies/2021/10/pizza) : [E:\apache-tomcat-10.0.11\webapps\pizza-lab-checkpoint-4...

File Edit View Reasoner Tools Refactor Window Help

[<](#) [>](#) [pizza \(http://www.semanticweb.org/sergg/ontologies/2021/10/pizza\)](#) Search...

Active ontology x Entities x Individuals by class x DL Query x

Individuals
 Annotation properties Datatypes
 Data properties
 Classes Object properties
 Object property hierarchy

isBaseOf — <http://www.semanticweb.org/sergg/ontologies/2021/10/pizza#isBaseOf>

Annotations Usage

Annotations: isBaseOf

Annotations +

owl:topObjectProperty
 hasIngredient
 hasBase
 hasTopping
 isIngredientOf
 isBaseOf
 isToppingOf

Charac Description: isBaseOf

☐ Functional
☐ Inverse functional
☐ Transitive
☐ Symmetric
☐ Asymmetric
☐ Reflexive
☐ Irreflexive

Equivalent To +

SubProperty Of +

isIngredientOf

Inverse Of +

hasBase

Domains (intersection) +

PizzaBase

Ranges (intersection) +

Pizza

Disjoint With +

SuperProperty Of (Chain) +

Reasoner active ☒ Show Inferences

Dominio y Rango en Propiedades (I)

- Las propiedades sirven para establecer relaciones entre las distintas clases a nivel de estructura de conocimiento y entre individuos a nivel de instanciación de la ontología a un caso real.
- Lo común en las propiedades es definir el Dominio y el Rango.
 - Dominio: conjunto origen (clases que pueden tener esa propiedad). En el caso hasTopping: solo la clase Pizza puede tener esta propiedad
 - Rango: conjunto destino (valores que puede tomar la propiedad). En el caso hasTopping: los valores asociados a esta propiedad serán los de la clase PizzaTopping

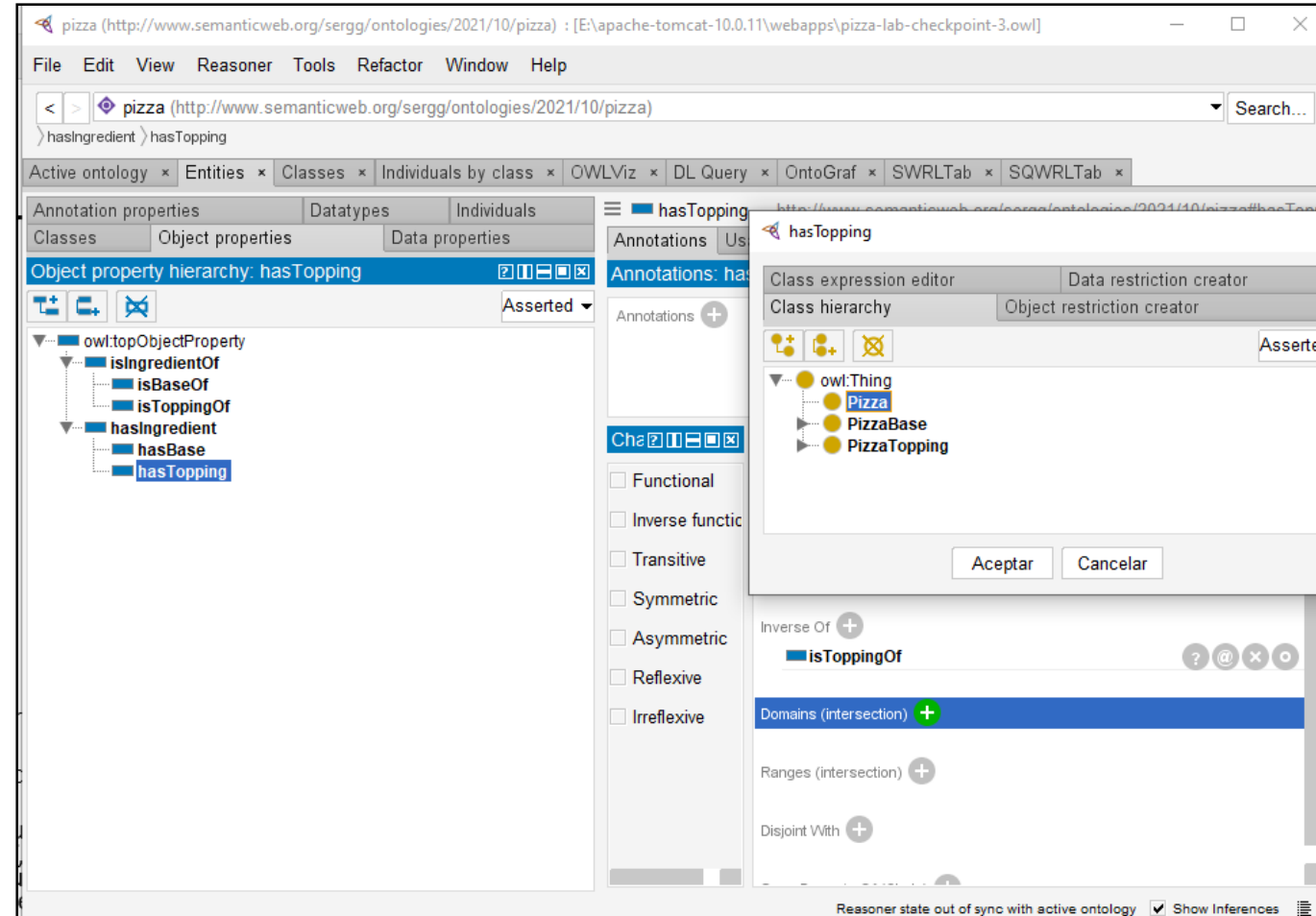
Ej. relación correcta: Pizza -> hasTopping -> MozzarellaTopping

Ej. relación que no podría ser por el rango: Pizza -> hasTopping ->

ThinAndCrispyBase

Dominio y Rango en Propiedades (II)

- Seleccionamos en la pestaña Object Properties: hasTopping y especificamos el Dominio y el Rango.
 - Domain + → Pizza
 - Range + → PizzaTopping
- Repetimos para “hasBase” (ojo, aquí el rango es PizzaBase)
- Sincronizar el razonador
 - Localizar las inferencias
- Dominio: importante fijarse en **“Intersection”**
 - Representa TODAS las condiciones que deben cumplirse.
 - Si ponemos en Dominio varias clases disjuntas, el razonador generará un error
 - Para definir varias clases que puedan usar una relación, se debe definir la unión usando “or”, de manera que, en lugar de *usar Class Hierarchy*, usaremos *ExpressionEditor* para elaborar esta unión de clases.



Describir y Definir Clases

Las Propiedades, nos permiten completar la definición de las clases:

- Clases **Primitivas**: Definen condiciones necesarias para formar parte de ella (Pizza)
- Clases **Definidas**: Definen condiciones necesarias y suficientes para confirmar que se forma parte de ella.
- Clases **Anónimas**: Son clases auxiliares que internamente genera el razonador en los casos que necesite para hacer inferencias.

Uso de restricciones en Propiedades

- Propiedades de Objetos: Dominio/Rango
- Propiedades de Datos: Tipo de Dato (xsd:string, ...)
- Se pueden usar las propiedades para definir nuevas clases, por ejemplo, la clase “PersonasQueVivenEnEspaña” se puede definir como la que representa a aquellas instancias de clase *Persona*, que tienen una relación *viveEn*, con la instancia “España” de la clase *Pais*
- Tipos de restricciones:
 - Cuantificación (algunas o todas) – combinación de algunos o todos con las siguientes palabras clave:
 - “some” – existencial: alguna de las propiedades cumple el requisito
 - Ej., CheesePizza: tiene topping (**hasTopping**) de algún (**some**) tipo de queso (**CheeseTopping**)
 - “only” – únicamente de un conjunto de un tipo
 - Ej., Restricción: todos los toppings (**hasTopping**) que tienen son de tipo solo (**only**) vegetal (**VegetableTopping**)
 - Cardinalidad, restringe el número de individuos relacionados con esa propiedad
 - Ej., PadresConUnSoloHijo – Restricción: 1 único hijo
 - Valor, permite especificar los valores que puede tener una propiedad (por ejemplo, valor menor de 100, o elementos de clases enumeradas con un subconjunto finito de elementos)
 - Ej. CaloricPizza: tiene calorías (hasCaloricContent) por encima de 400 (veremos como expresar esta restricción más adelante)
- Las restricciones permiten definir nuevas clases (definidas o anónimas)

Definición de Clases en base a restricciones

- Desde la aplicación, seleccionando una clase nos fijamos en la sección *Description* para ver toda la información de la clase, especificada por el usuario o inferida.
- Owl puede redefinir la estructura jerárquica de las clases, si detecta que hay una manera más general (cuando ocurre, se puede visualizar en la sección inferida en la vista *Class hierarchy* en la opción *Inferred*).

Añadir una restricción a una clase existente

- Seleccionar la clase Pizza
- En Class Description View (ver la siguiente página)
 - Subclassess Of +
 - Object restriction creator=>Restricted property => Restriction filler
 - Property to Restrict: hasBase
 - Restriction filler: PizzaBase
 - Restriction type : Some (Existential) (tipo que significa que esta restricción indica una condición necesaria de al menos 1 elemento)
 - Clic en OK
 - Repetir lo mismo con “hasTopping” (pero con PizzaTopping)
- Ahora la clase Pizza está en el conjunto de objetos que tienen una base (hasBase) del tipo PizzaBase (ver la siguiente página)
 - Esto es una condición necesaria.

pizza (http://www.semanticweb.org/sergg/ontologies/2021/10/pizza) : [E:\apache-tomcat-10.0.11\webapps\pizza-lab-checkpoint-4.owl]

File Edit View Reasoner Tools Refactor

< > pizza (http://www.semanticweb.org/sergg/ontologies/2021/10/pizza)

Active ontology x Entities x Classes x Individuals

Annotation properties Datatypes

Classes Object properties

Class hierarchy: Pizza

+ - ✕

owl:Thing

- PizzaTopping
- PizzaBase
 - ThinAndCrispyBase
 - DeepPanBase
 - Pizza

Data restriction creator Class expression editor Object restriction creator Class hierarchy

Restricted property

+ - ✕

owl:topObjectProperty

- hasIngredient
- hasBase
- hasTopping
- isIngredientOf

Assertion

+ - ✕

owl:Thing

- Pizza
- PizzaBase
- PizzaTopping

Restriction type

Some (existential) Cardinality 1

Acceptar Cancelar

SubClass Of +

General class axioms +

SubClass Of (Anonymous Ancestor)

Instances +

Target for Key +

Disjoint With +

PizzaBase, PizzaTopping

Disjoint Union Of +

Reasoner state out of sync with

pizza (http://www.semanticweb.org/sergg/ontologies/2021/10/pizza) : [E:\apache-tomcat-10.0.11\webapps\pizza-lab-checkpoint-4.owl]

File Edit View Reasoner Tools Refactor Window Help

< > pizza (http://www.semanticweb.org/sergg/ontologies/2021/10/pizza)

Active ontology x Entities x Classes x Individuals by class x OWLViz x DL Query x OntoGraf x SWRLTab x SQWRLTab

Annotation properties Datatypes Individuals

Classes Object properties Data properties

Class hierarchy: Pizza

+ - ✕

owl:Thing

- PizzaTopping
- PizzaBase
 - ThinAndCrispyBase
 - DeepPanBase
 - Pizza

Annotations: Pizza

Annotations +

Description: Pizza

Equivalent To +

SubClass Of +

hasBase some PizzaBase
 hasTopping some PizzaTopping

General class axioms +

SubClass Of (Anonymous Ancestor)

Instances +

Target for Key +

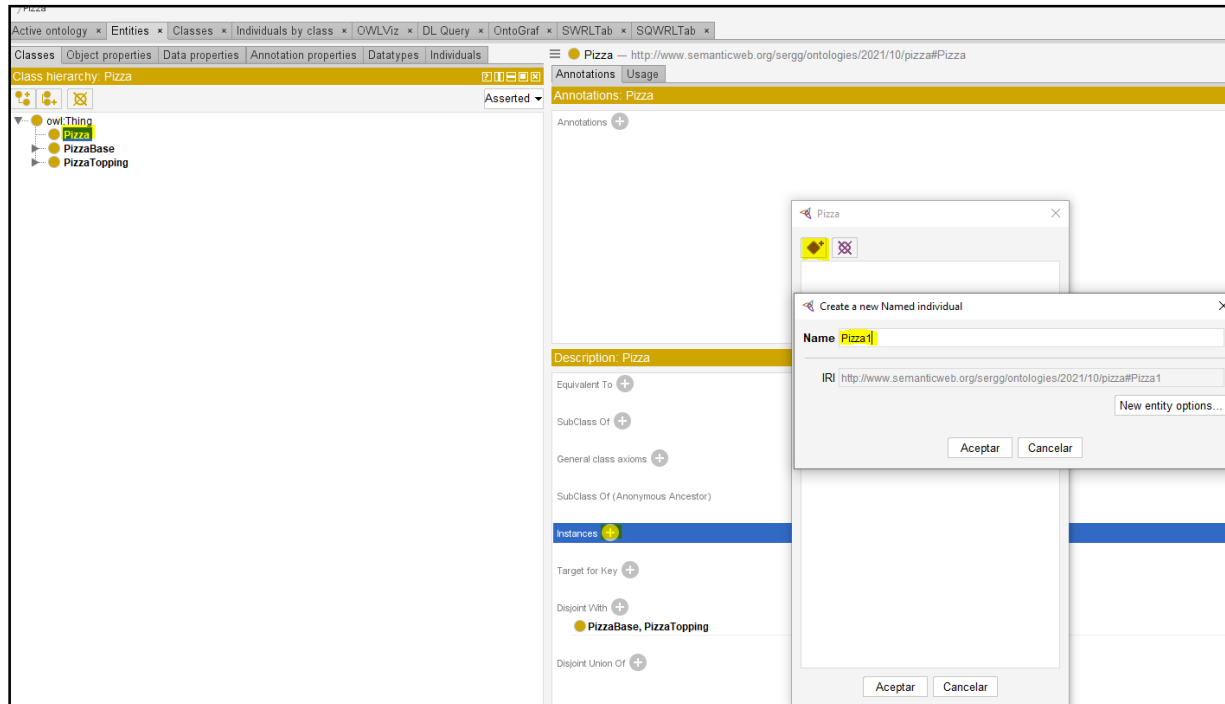
Disjoint With +

PizzaBase, PizzaTopping

Reasoner state

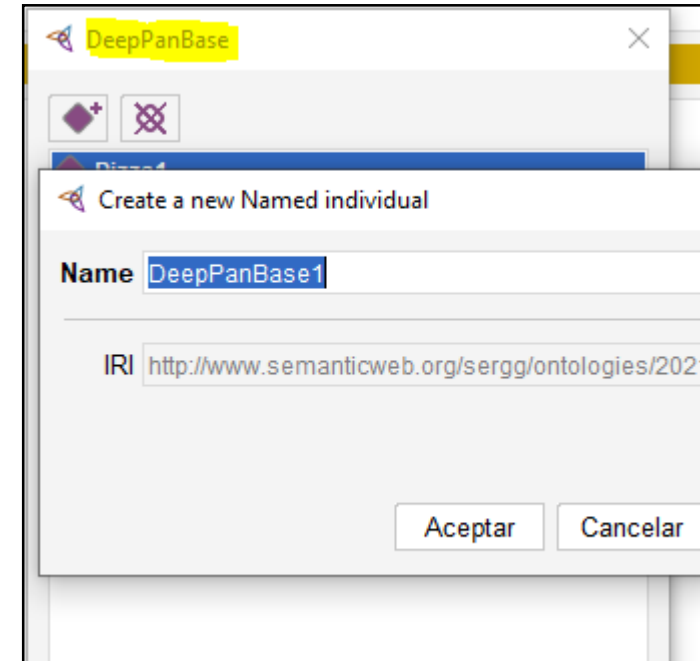
Crear Individuos

- Los individuos pueden crearse a cualquier nivel de la jerarquía. Siempre pertenecerán a owl:Thing
- Seleccionar la clase Pizza
 - En la vista Description hacer clic en + Instancias
 - Hacer clic en el rombo
 - Introducir el nombre



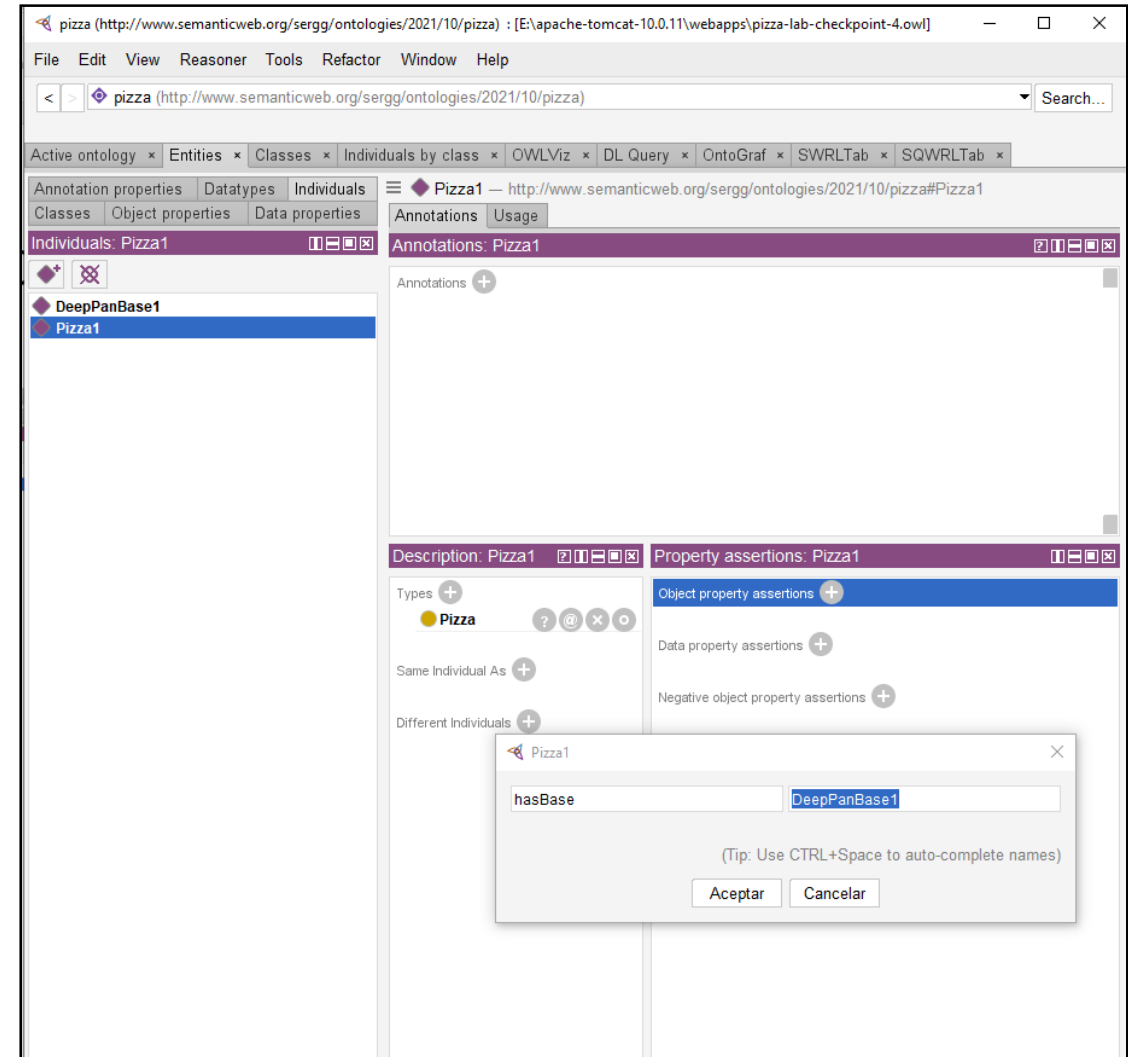
Crear Individuos

- Creamos una instancia de base de pizza



Definir Propiedades sobre individuos

- Desde la pestaña Individuals Seleccionar la instancia Pizza1
- En la vista Property assertions
- Clic en + Object Property assertions
- Introducir el nombre de la propiedad (hasBase) y el valor DeepPanBase1 (que es el otro individuo que hemos creado)
- Al hacer clic en aceptar, se añade la propiedad.
- Sincronizamos el razonador y observamos la relación inversa inferida en la instancia DeepPanBase1 (ver siguiente página)







[pizza](http://www.semanticweb.org/sergg/ontologies/2021/10/pizza) (http://www.semanticweb.org/sergg/ontologies/2021/10/pizza) : [E:\apache-tomcat-10.0.11\webapps\pizza-lab-checkpoint-3.owl]



File Edit View Reasoner Tools Refactor Window Help



< > [pizza](http://www.semanticweb.org/sergg/ontologies/2021/10/pizza) (http://www.semanticweb.org/sergg/ontologies/2021/10/pizza) Search...





Active ontology × Entities × Classes × Individuals by class × OWLViz × DL Query × OntoGraf × SWRLTab × SQWRLTab ×


Annotation properties Datatypes Individuals
 Classes Object properties Data properties





Individuals: DeepPanBase1    


 


 **DeepPanBase1**
 **Pizza1**


Annotations: DeepPanBase1    


Annotations 





Description: DeepPanBas    


Types 



 **PizzaBase** ? @ × ○


Same Individual As 


Different Individuals 


Property assertions: DeepPanBase1    


Object property assertions 

 **isBaseOf** Pizza1 ? @
 **isIngredientOf** Pizza1 ? @

Data property assertions 

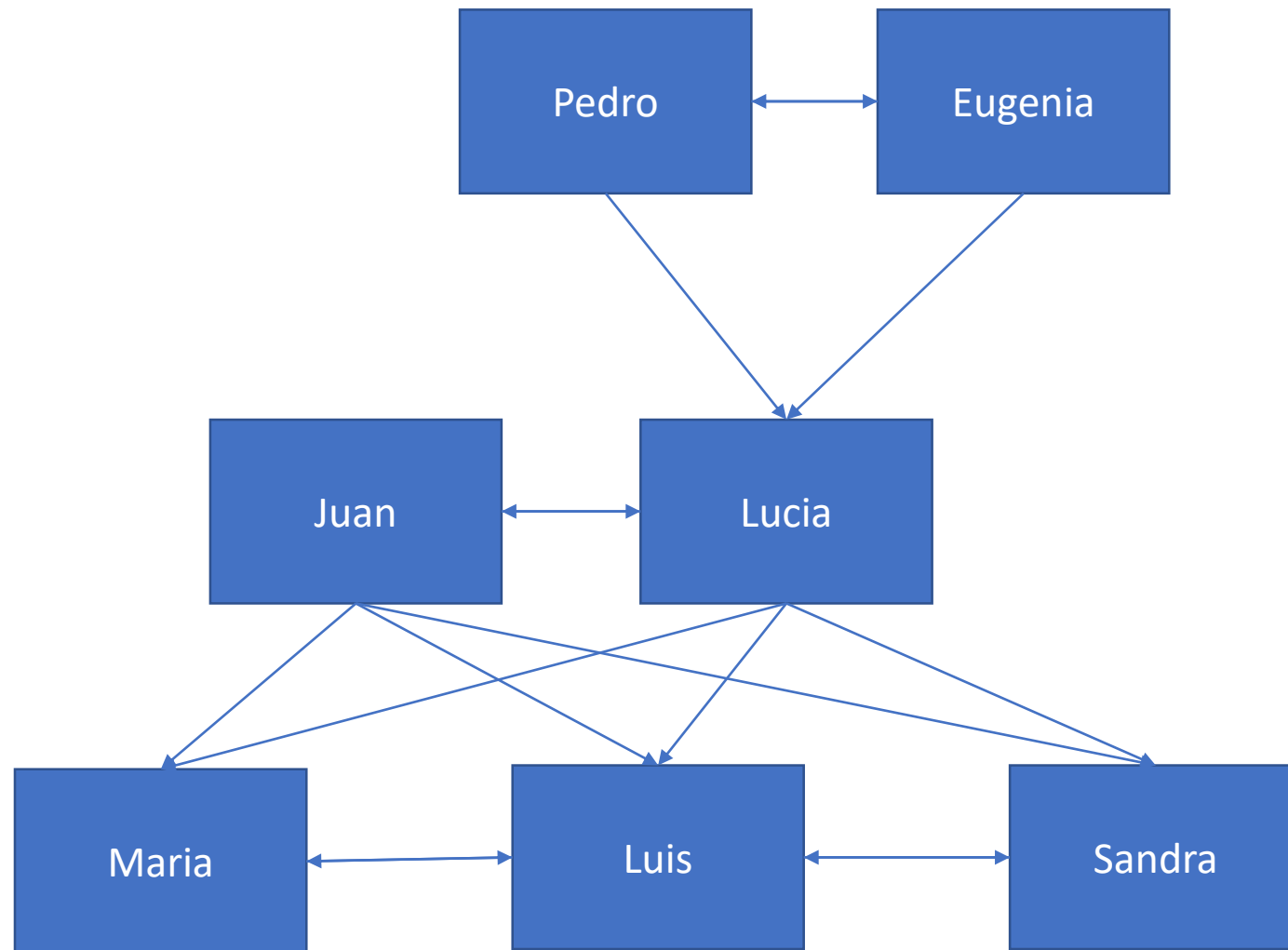
Negative object property assertions 

Negative data property assertions 

Reasoner active ☒ Show Inferences 

PRÁCTICA: PARTE 1

- Elaborar una ontología para las familias, considerando:
 - Una clase para representar a las personas
 - Una propiedad para definir hermanos
 - Propiedades de ascendencia y descendencia entre padres e hijos
 - Una propiedad que establezca que dos personas son pareja
 - Una clase que identifique a las personas que tienen pareja
 - Una clase que identifique a las personas que son antecesoras de otra persona
- Instanciar la ontología, con los individuos de la siguiente página.
- Obligatoria: Lo revisaremos en la próxima sesión de prácticas.



FIN DE LA PRIMERA SESION