

Ontologías, Protégé y OWL

Ulises Cortés, Sergio Álvarez, Ignasi Gómez-Sebastià,
Luis Oliva

{ia,salvarez,igomez,loliva}@cs.upc.edu

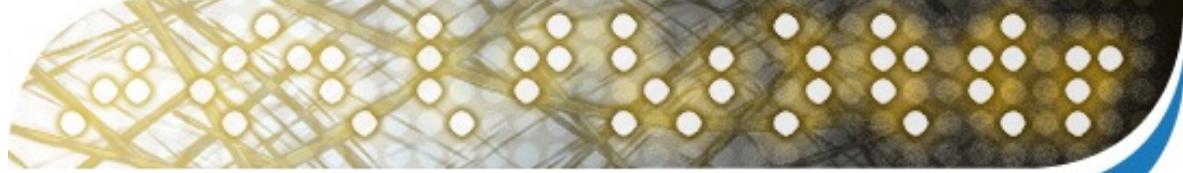
<http://www.cs.upc.edu/~{ia,salvarez,salvarez,loliva}>

SID2022

<https://kemlg.upc.edu>



Knowledge Engineering and Machine Learning Group
UNIVERSITAT POLITÈCNICA DE CATALUNYA

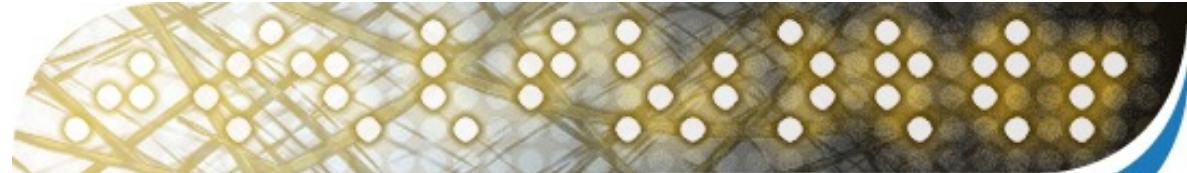


Teoría básica sobre Ontologías



Knowledge Engineering and Machine Learning Group
UNIVERSITAT POLITÈCNICA DE CATALUNYA

<https://kemlg.upc.edu>





Filosofía (Aristóteles y Platón)

- Filosóficamente: Parte de la metafísica que estudia lo que hay
 - Lo que existe. Entidades o **conjuntos** de entidades
 - Formas en que se relacionan las entidades que existen
 - ◆ Formas en que se relacionan los universales (Químico) y los particulares a los que se aplica ese universal (Arquímedes, El mago Merlín, Manel Poch)
- Término moderno: Leibniz ("Introductio ad Encyclopaediam arcanam")
 - ¿Qué hay?
 - El "Todo"



Informática

- Esquema conceptual (detallado) de un domino (o varios)
- Facilita la comunicación (intercambio de información) entre sistemas heterogéneos
- Representación del conocimiento. Estructura que contiene todas las entidades y relaciones relevantes para el dominio tratado
 - Cyc (Ontología con conocimiento genérico)
- Uso en Inteligencia Artificial:
 - **Razonamiento**
 - **Clasificación**



Base tecnológica

- DAML
 - Fuertemente asociado a WebServices
 - DARPA
- OIL
 - Comisión Europea
 - Primer lenguaje de intercambio de Ontologías
 - Anticuado y poco expresivo
- RDF
 - Resource description framework
 - Esquema general de representación de información
 - Sin semántica asociada
- OWL
 - Inspirado en DAML y OIL
 - Construido sobre RDF + XML





RDF Schema

<https://www.w3.org/TR/rdf-schema>

- Classes
 - rdfs:Resource
 - rdfs:Class
 - rdfs:Literal
 - rdfs:Datatype
 - rdf:langString
 - rdf:HTML
 - rdf:XMLLiteral
 - rdf:Property
- Properties
 - rdfs:range
 - rdfs:domain
 - rdf:type
 - rdfs:subClassOf
 - rdfs:subPropertyOf
 - rdfs:label
 - rdfs:comment

Ontology Web Language (OWL)

<https://kemlg.upc.edu>



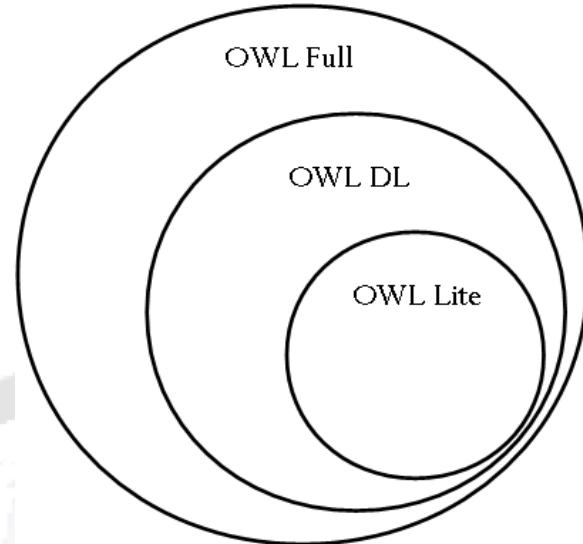
Knowledge Engineering and Machine Learning Group
UNIVERSITAT POLITÈCNICA DE CATALUNYA





OWL es como una patata frita

- OWL-FULL
 - Sin restricciones
 - Permite bucles en las relaciones
 - Algunas sentencias pueden llevar tiempo infinito en resolverse
- OWL-DL
 - Término medio
 - Basado en lógica descriptiva
 - Contiene algunas restricciones
- OWL-LITE
 - Básico
 - Taxonomía con restricciones simples





OWL Full

- **owl:Class = rdfs:Class**
 - Todos los documentos RDF son documentos OWL-FULL
- **owl:Thing = rdfs:Resource**
- **owl:ObjectProperty = rdf:Property**
 - **owl:DatatypeProperty** hereda de **owl:ObjectProperty**
- Un recurso puede ser a la vez clase e instancia
 - Boeing-747 is an instance of the class AirplaneType
 - *airplane1* is an instance of the class Boeing-747
- Permite el uso de todas las restricciones definidas en OWL
- Sin restricciones sobre los axiomas (pueden ser inconsistentes)



OWL Full

<https://www.w3.org/TR/owl-ref>

owl:allValuesFrom
owl:SomeValuesFrom
owl:hasValue
owl:maxCardinality
owl:minCardinality
owl:cardinality
owl:intersectionOf
owl:unionOf
owl:complementOf
rdfs:subClassOf
owl:equivalentClass
owl:disjointWith

rdfs:subPropertyOf
rdfs:domain
rdfs:range
owl:equivalentProperty
owl:inverseOf
owl:FunctionalProperty
owl:InverseFunctionalProperty
owl:TransitiveProperty
owl:SymmetricProperty
owl:sameAs
owl:differentFrom
owl:AllDifferent



OWL DL

- Permite todos los tipos de restricciones, con limitaciones
 - Restricciones numéricas no soportadas en propiedades transitivas
 - owl:DatatypeProperty no hereda de owl:ObjectProperty
 - ◆ Propiedades como inverseOf, SymmetricProperty o TransitiveProperty no se pueden aplicar a DatatypeProperties
 - Restricciones en las anotaciones (annotation property)
 - ◆ Sólo soporta datos, literales, URI o Instancias
 - ◆ Una annotation property no puede tener sub-propiedades
- Los axiomas definidos han de ser jerárquicos, correctos y consistentes (e.g. no depender de clases no declaradas)
 - Los axiomas sobre igualdad o diferencia han de referirse a las instancias (*individuals*)



OWL Lite

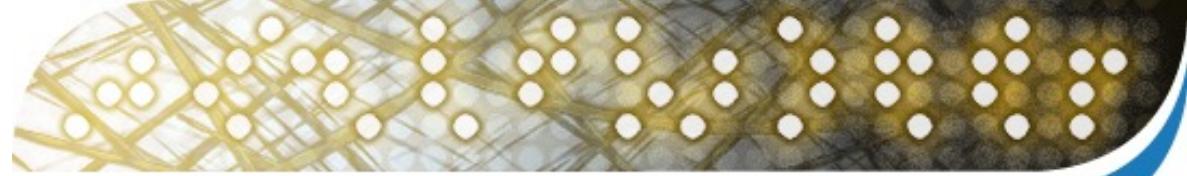
- Restricciones de cardinalidad 0..1
- No permite algunas restricciones importantes como `disjointWith`
- Casi tan complejo como OWL-DL de modo que apenas se usa

Ciclo de desarrollo de una Ontología

<https://kemlg.upc.edu>



Knowledge Engineering and Machine Learning Group
UNIVERSITAT POLITÈCNICA DE CATALUNYA



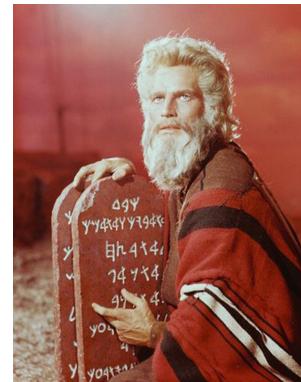


Ejemplo: PizzaOntology

- Descargad la ontología:
 - <https://raw.githubusercontent.com/owlcs/pizza-ontology/master/pizza.owl>
- Cargad la ontología en Protégé
- Añadid algún comentario (Annotations)

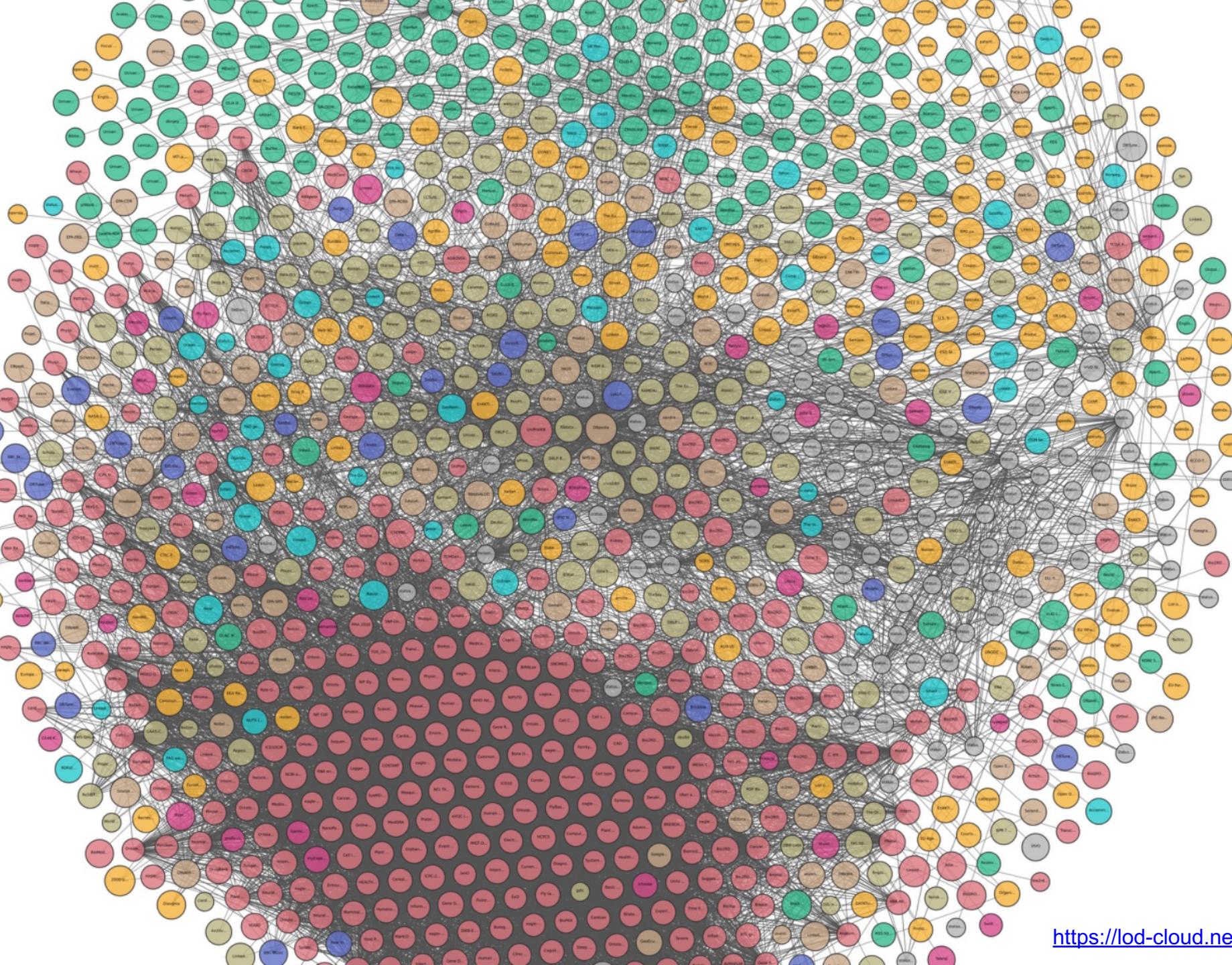


- Cual es el dominio que intentamos cubrir con la Ontología
 - Problema de la granularidad
 - Problema de la omnisciencia
- Para que vamos a usar la Ontología
- Identificar las *Competency Questions*
 - Para qué tipos de preguntas debe darnos respuesta la información que contiene la Ontología
- Las decisiones no son finales, pueden cambiar durante el ciclo de desarrollo de la Ontología





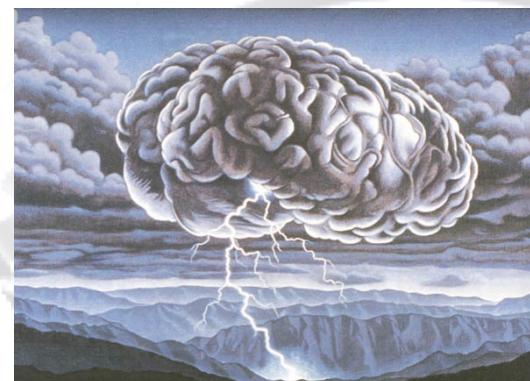
- ¿Por qué?
 - Eficiencia, menor coste de desarrollo
 - Integración directa con sistemas que usen esa Ontología
 - Uso de Ontologías que han sido validadas en casos de uso prácticos (aplicaciones)





- Cuáles son los términos sobre los que vamos a hablar
- Cuáles son las propiedades de esos términos
- Qué queremos decir sobre esos términos

- Primer paso:
 - No organizar los términos, hacer una lista con lo que queremos incluir en la Ontología





- Definir las clases y la taxonomía
 - Una clase es un concepto del dominio, no un objeto
 - ◆ ¡No sólo entidades, pueden ser propiedades!
 - Una clase es un conjunto de elementos con propiedades similares
 - ◆ ¿Y qué es cada elemento entonces?
- La taxonomía es la jerarquía de clases
 - ¿Cuándo agrupamos dos clases en la misma superclase?
 - ◆ La respuesta la tenéis en esta transparencia



determine scope

consider reuse

enumerate terms

define classes

define properties

define constraints

create instances





- Definir las clases y la taxonomía
 - Top-Down: Definir primero los conceptos más generales, y luego especializar
 - ◆ ¿Y cuál es el concepto más general?
 - Bottom-up: Definir los conceptos más específicos y luego agruparlos en clases más generales
 - ¿Y cuándo los agrupamos?
 - Combinación: Definir los conceptos más importantes y luego generalizarlos y especificarlos en paralelo
 - Útil si aplicamos otra de las dos técnicas y nos quedamos atascados
- Clases disjuntas
 - Una instancia no puede pertenecer a ambas clases a la vez



Ejemplo: PizzaOntology

- Cread una subclase de Thing (Entities)
- Cread un hermano y una subclase de esta clase
- Cread otro hermano
- Borrad el segundo hermano
- Haced que la clase y su hermano sean disjuntas (Disjoint With)
 - ¿Hace falta hacerlo para las dos?



Tipos de clases: Primitiva

- *Necessary conditions*

- Si algo reúne las condiciones no es necesariamente obligatorio que sea un miembro de la clase
- **PERO:** Un elemento escogido al azar que sabemos que es miembro de la clase, sabemos que reúne las condiciones
- ¿Qué ocurre con un elemento escogido al azar que sabemos que reúne las condiciones?

Description: CheeseyPizza

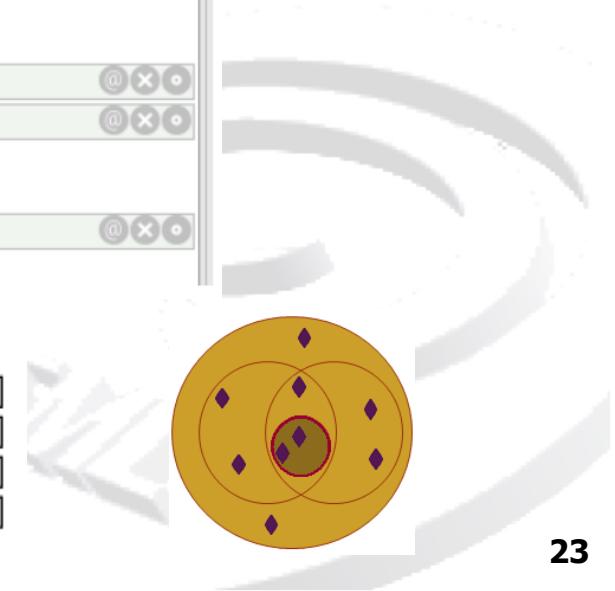
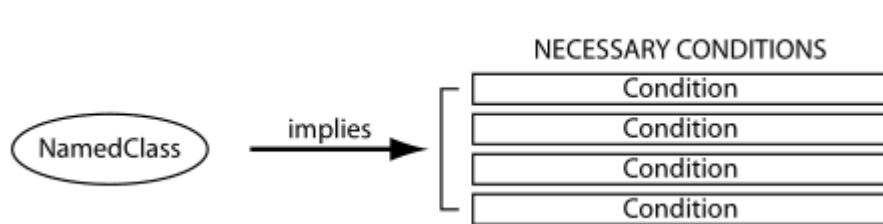
Equivalent classes +

Superclasses +

- Pizza
- hasTopping some CheeseTopping

Inherited anonymous classes

- hasBase some PizzaBase





Tipos de clases: Equivalente

- *Necessary conditions and sufficient conditions*
 - *Si algo reúne las condiciones es suficiente para decir que es un miembro de la clase*
 - *Un elemento escogido al azar que sabemos que es miembro de la clase, sabemos que reúne las condiciones*
 - *¿Qué ocurre con un elemento escogido al azar que sabemos que reúne las condiciones?*

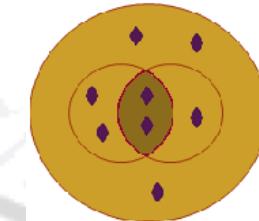
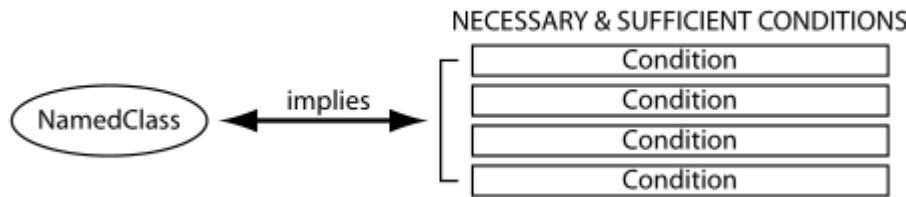
Description: RealItalianPizza

Equivalent To +
Pizza and (hasCountryOfOrigin value Italy)

SubClass Of +
hasBase only ThinAndCrispyBase

General class axioms +

SubClass Of (Anonymous Ancestor)
hasBase some PizzaBase





Ejemplo: PizzaOntology

- Observad la diferencia entre NamedPizza y RealItalianPizza
 - ¿Cuál es primitiva y cuál equivalente?
 - ¿Cuáles son las condiciones suficientes para que una pizza sea RealItalianPizza?
 - ¿Qué condiciones añade a las suficientes?



- Asociadas a la clases (Dominio-Rango):

- Rango no es una clase
 - ◆ Data Properties
- Rango es una clase
 - ◆ Relaciones a otras instancias de la clase
 - ◆ Object Properties



- Las restricciones definen el conjunto de valores posibles para una propiedad
- Las restricciones más comunes son:
 - Dominio
 - Rango
- No son restricciones a comprobar, son axiomas



determine scope

consider reuse

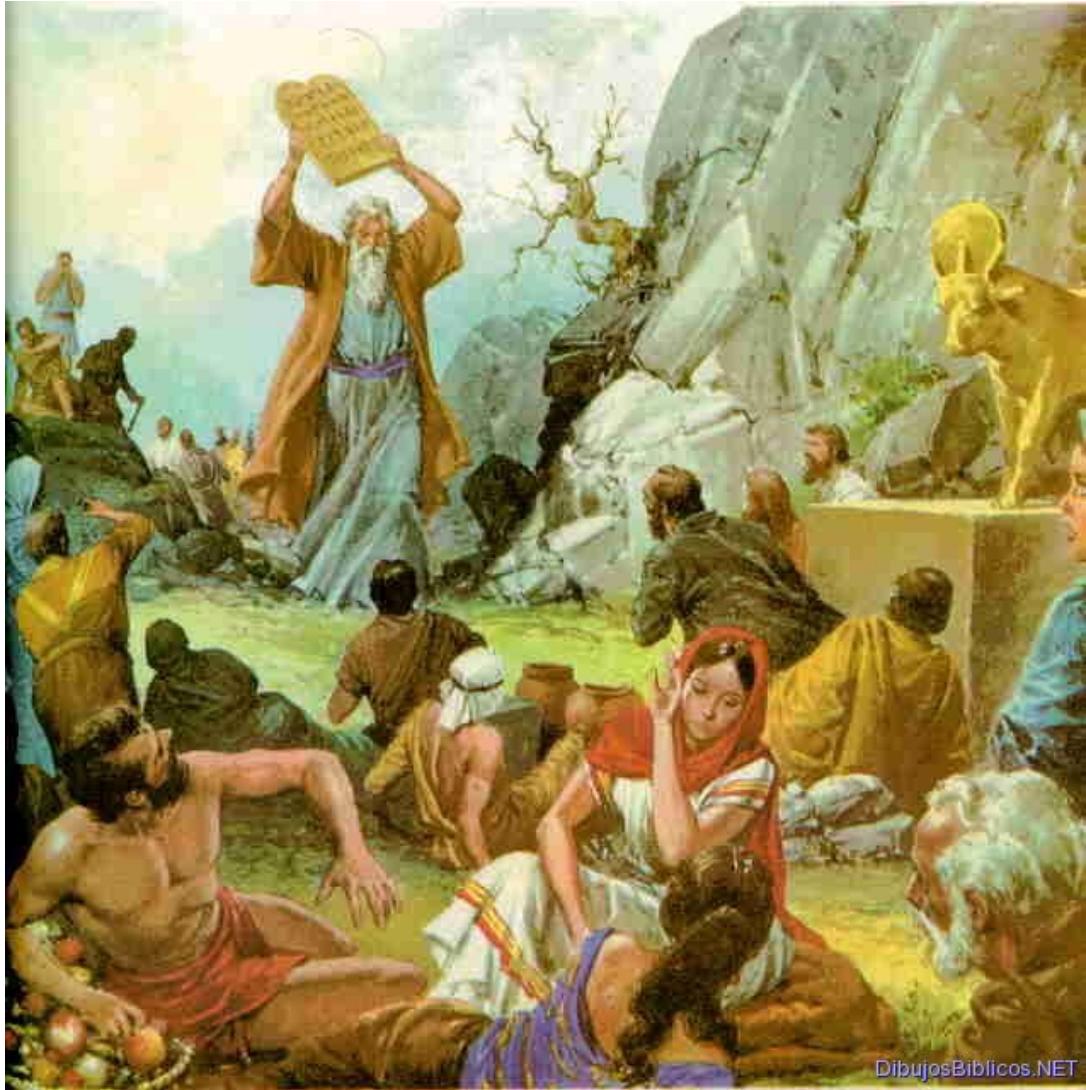
enumerate terms

define classes

define properties

define constraints

create instances



- Moisés y los creadores de Ontologías inconsistentes





Ejemplo: PizzaOntology

- Arrancad el Reasoner incluído en Protégé
 - Se puede configurar para ampliar/acotar el ámbito de lógica
- Analizad inconsistencias
 - Buscad las inconsistencias (en rojo)
 - ¿A qué se deben estas inconsistencias?
 - Usad el símbolo de interrogación para obtener explicaciones
- Analizad la clasificación del razonador
 - Buscad las inferencias de clasificación (en amarillo)
 - Usad el símbolo de interrogación para obtener explicaciones
- Parad el Reasoner



Ejemplo: PizzaOntology

- Cread una ObjectProperty para expresar en qué país se vende
 - Asignad dominio y rango
- Cread una subpropiedad de hasIngredient para poder representar el relleno del borde
 - Asignad dominio y rango
 - Asignad alguna restricción, como por ejemplo *disjoint with* o *inverse of*
- Cread una DataProperty para poder representar el precio
 - Asignad dominio y rango



- Crear instancias de las clases
 - La clase se convierte en un tipo directo de la instancia
 - Las superclases del tipo directo son tipos de la instancias
- Asignar valores a las propiedades
 - Los valores asignados deben cumplir las restricciones impuestas
 - Se pueden usar razonadores para comprobar que las restricciones se cumplan



Ejemplo: PizzaOntology

- Cread una instancia de DeepPanBase (Individuals)
- Cread una instancia de Pizza con las siguientes propiedades:
 - Tiene como país de origen Italia
 - Tiene como base la instancia de DeepPanBase que habéis creado
- En el menú, arrancad el Reasoner
 - ¿Es inconsistente? ¿Por qué?
- Borrad estas instancias y volved a sincronizar el Reasoner



Ejemplo: PizzaOntology

- Cread un topping TurtleTopping
- Cread una Pizza llamada SuperMarioPizza con condiciones necesarias y suficientes: tener como ingredientes champiñones y tortugas
- Cread una instancia de una pizza de tipo Pizza con ingredientes instancias de champiñones y tortugas
- Sincronizad el Razonador y observad cómo se clasifica
- Cambiad SuperMarioPizza para que las condiciones sean sólo necesarias y observad la diferencia tras sincronizar

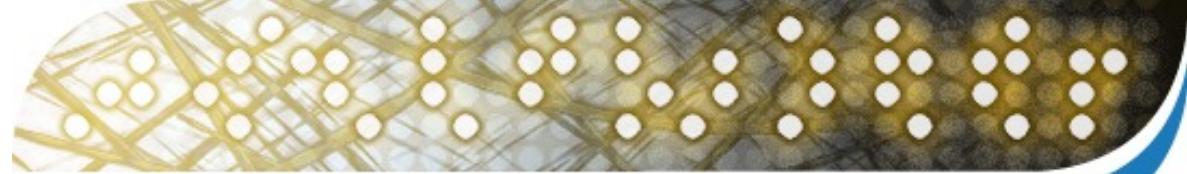


Restricciones

<https://kemlg.upc.edu>



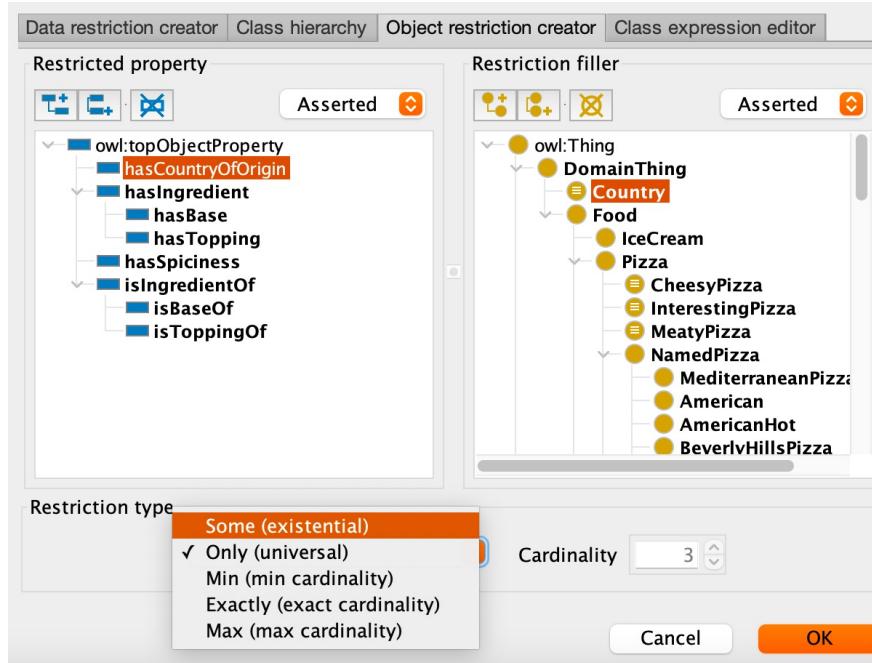
Knowledge Engineering and Machine Learning Group
UNIVERSITAT POLITÈCNICA DE CATALUNYA





Restricciones

- Algunas restricciones están disponibles en el editor de Protégé



- Guía para editar en texto libre:
 - <http://protegeproject.github.io/protege/class-expression-syntax/>



Restricciones: axiomas de clase

- Enumeraciones de individuos: `oneOf`
 - Permite describir una clase por sus posibles instancias
- Clases excluyentes entre si: `disjointWith`
 - Una instancia no puede pertenecer a las dos clases a la vez
- Clases equivalentes entre si: `sameClassAs`
 - No confundir con `equivalentClass`
 - Todas las instancias de una clase han de ser instancias de la otra clase
 - No todos los razonadores realizan la inferencia (ineficiente)
 - En cualquier caso es útil para enlazar ontologías



Restricciones: Quantifier restrictions

- Existencial
 - Una pizza picante es una pizza que contiene al menos un ingrediente picante
 - Some
- Universal
 - Una pizza vegetariana es una pizza donde todos los ingredientes son de origen no animal
 - Only



Restricciones: Cardinality restrictions

- Mínimo número de relaciones: `minCardinality`
- Máximo número de relaciones: `maxCardinality`
- Número exacto de relaciones: `cardinality`
- Restricciones cualificadas
 - Limitan el rango de la relación



Restricciones: por valor

- hasValue restrictions
 - Equivalente a enumeraciones de instancias
 - Usa el símbolo \exists .

Pizza Escandinava =

Pizza.tienePaisOrigen \exists . {Dinamarca, Noruega, Suecia}



Restricciones: operaciones de conjuntos

- Intersección de dos clases:
 - `intersectionOf`
 - En Protégé: `classA and ClassB`
- Unión de dos clases:
 - `unionOf`
 - En Protégé: `classA or ClassB`
- Complemento de una clase (todas las instancias que no pertenecen a esa clase):
 - `complementOf`
 - En Protégé: `not classA`



Ejemplo: PizzaOntology

- Cread las siguientes pizzas:
 - Pizza con marisco: contiene como mínimo marisco
 - Pizza de marisco: todos los ingredientes son de marisco
 - Pizza ecléctica: mínimo 10 ingredientes
 - Pizza de oferta: máximo 2 ingredientes
 - Pizza binaria: exactamente 2 ingredientes
 - Pizza triqueso: exactamente 3 ingredientes, todos de queso
 - Pizza escandinava
 - ◆ Tendréis que editar en texto libre (Class Expression Editor) y también editar la clase Country
 - ◆ Utilizad or y value (tenéis ejemplos en American y en la guía)
 - Pizza aburrida especial: pizzas que no sean InterestingPizza, pero que estén en la unión entre las MeatyPizza y las CheesyPizza

Propiedades

<https://kemlg.upc.edu>



Knowledge Engineering and Machine Learning Group
UNIVERSITAT POLITÈCNICA DE CATALUNYA





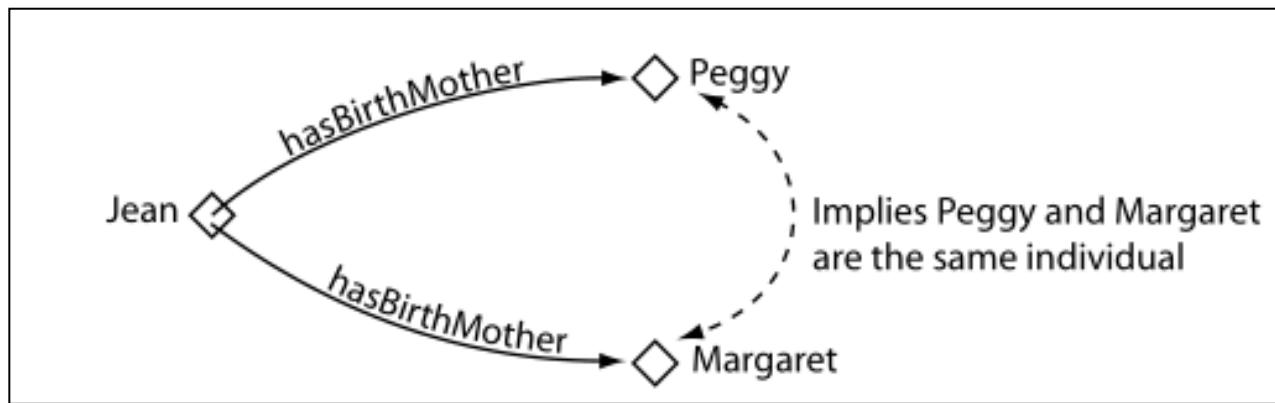
Propiedades

- Propiedades y subpropiedades
 - ¿Cuándo agrupamos propiedades en subpropiedades?
- Domino y rango de la propiedad
 - ¡Recordad que son tratadas como axiomas!
- Propiedad Inversa
 - Del tipo hasComponent vs isComponentOf
 - Donde el dominio y el rango, se intercambian



Propiedades

- Propiedad funcional
 - Cuando A y B están relacionados mediante una propiedad funcional sólo una instancia de B puede estar relacionada con A
 - ◆ ¿Qué ocurre si más de una instancia de B está relacionada con A?

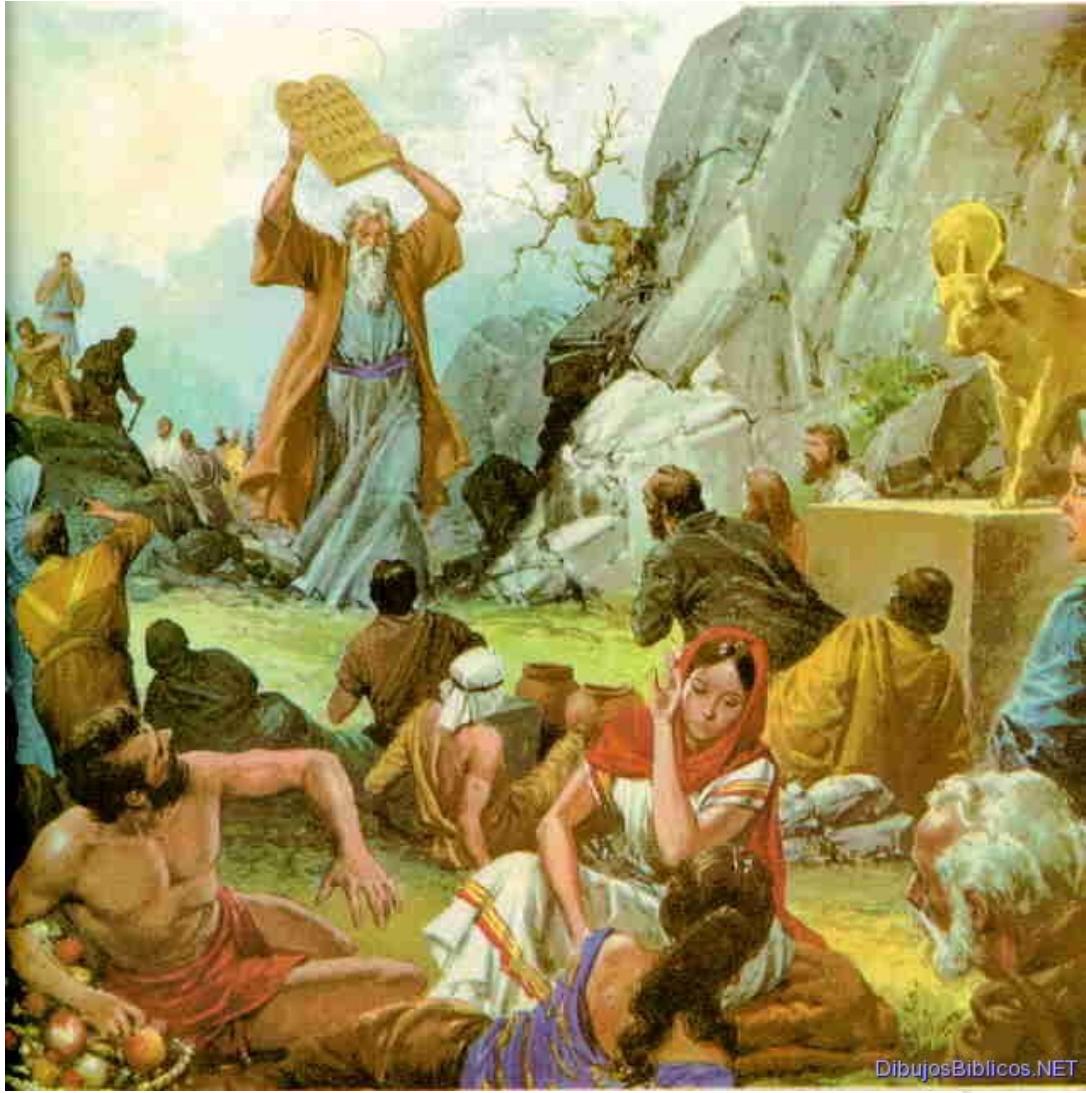


- Propiedad funcional inversa (e.g. número de serie)



Propiedades

- Propiedad transitiva
 - Si una instancia de A se relaciona con una de B y una de B con C, la instancia de A se relaciona con la de C
- Propiedad simétrica
 - Si una instancia de A se relaciona con una de B, la de B se relaciona con la de A
- Propiedad asimétrica (antisimétrica)
 - Si una instancia de A se relaciona con una de B, la de B no se puede relacionar con la de A
 - ◆ ¿Y qué pasa si se relaciona?



DibujosBiblicos.NET

- Moisés y los creadores de Ontologías inconsistentes



Propiedades

- Propiedad reflexiva
 - Si P es reflexiva y una instancia A tiene P, P se aplica a la instancia de A
- Propiedad irreflexiva
 - Si P es reflexiva y una instancia A tiene P, P no se puede aplicar a la instancia de A
 - ◆ ¿Y qué pasa si se aplica?



Ejemplo: PizzaOntology

- Cread una propiedad funcional que asigne un creador a una NamedPizza (tendréis que crear clases)
 - Asignad dos creadores a una instancia de NamedPizza
 - Sincronizad el Razonador
 - Qué inferencia ha hecho?
 - Identificad los creadores como Different Individuals y resincronizad
- Cread una propiedad transitiva que permita representar que la creación de una NamedPizza está influenciada por otra
- Cread una propiedad simétrica que permita expresar que dos ingredientes combinan bien
- Observad las características disponibles en las data properties
 - ¿Tiene sentido?



Ejercicio

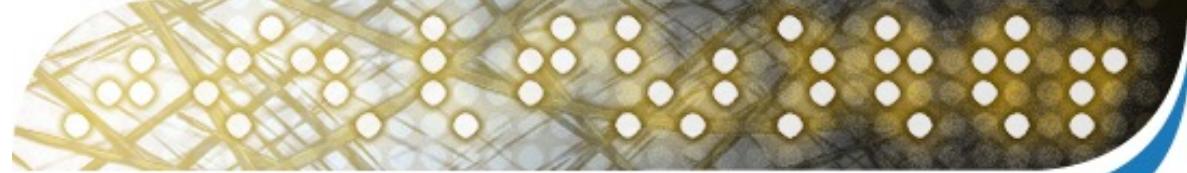
- Cread una ontología que permita:
 - Distinguir entre diferentes tipos de agente:
 - ◆ Explorador
 - ◆ Recolector
 - ◆ Almacenamiento
 - Representar un grafo mediante nodos y relaciones entre los mismos
 - Expresar la relación de que dos agentes están a una distancia de 1 nodo (nodos contiguos) en base a las relaciones entre nodos
 - Expresar la relación de que un agente Recolector puede transferir recursos a un agente Almacenamiento en base a la relación de estar en nodos contiguos
- Opcional: poder expresar distancias arbitrarias entre nodos

Referencias

<https://kemlg.upc.edu>



Knowledge Engineering and Machine Learning Group
UNIVERSITAT POLITÈCNICA DE CATALUNYA



Referencias



- Cyc
 - <http://www.cyc.com/>
 - <http://www.opencyc.org/>
- DAML
 - <http://www.daml.org/>
- OIL
 - <http://www.cs.vu.nl/~frankh/postscript/IEEE-IS01.pdf>
- RDF
 - <http://www.xml.com/pub/a/2001/01/24/rdf.html>
- OWL
 - <http://owl.cs.manchester.ac.uk/tutorials/protegeowltutorial>

Referencias



- Protégé
 - <http://protege.stanford.edu>
- Ontología de pizzas
 - www.co-ode.org/ontologies/pizza/



**Knowledge Engineering and Machine Learning Group
UNIVERSITAT POLITÈCNICA DE CATALUNYA**

Ulises Cortés, Sergio Álvarez, Ignasi Gómez-Sebastià,
Luis Oliva

{ia,salvarez,igomez,loliva}@cs.upc.edu

<http://www.cs.upc.edu/~{ia,salvarez,igomez,loliva}>

SID2022