

Ontologías

IIA-LCC

2019

bulacio@cifasis-conicet.gov.ar

Transparencias base, Ian Horrocks

<http://www.cs.man.ac.uk/~horrocks/Slides/>



Organización

- Parte teórica
- Práctica en papel (parcial)
- Práctica en Labs
- Trabajo Práctico: en Labs + informe
 - Diseño de Ontología de Deportes.
 - Clases
 - Propiedades
 - Restricciones
 - Esquema gráfico.



Bibliografía

- Thomas R. Gruber (1993). Toward principles for the design of ontologies used for knowledge sharing. Int. Journal of Human-Computer Studies, 43(5-6) 907-928, 1995.
- OWL: a Description Logic Based Ontology Language for the Semantic Web
- NaB Chandrasekaran (1999). What Are Ontologies, And Why Do We Need Them? , IEEE Intelligent Systems. <http://www.cse.ohio-state.edu/~chandra/What-are-ontologies-and-why-we-need-them.pdf>
- **Natalya F. Noy and Deborah L. McGuinness (2005). Desarrollo de Ontologías 101: Guía para crear tu primera Ontología. Stanford University, Stanford, CA.**
<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.135.6625&rep=rep1&type=pdf>
- Matthew Horridge. (2011). A Practical Guide To Building OWL Ontologies Using Protégé 4 and CO-ODE Tools. The University Of Manchester.
- Suárez-Figueroa, M.; Gómez-Pérez, A.; Motta, E.; Gangemi, A. (Eds.). Ontology Engineering in a Networked World. Springer, 2012



Bibliografía

- RDF: <http://www.w3.org/RDF/>
- RDF Schema: <http://www.w3.org/TR/rdf-schema/>
- OWL: <http://www.w3.org/TR/owl-features/>
- Protégé: <http://protege.stanford.edu/download/download.html>
- OBO Edit: <http://www.oboedit.org/>

Lógica, repaso:

- <http://dit.upm.es/~gfer/ssii/rcsi/rcsich4.html#rcsise23.html>
- <https://www.cs.us.es/~jalonso/cursos/li-03/temas/tema-6.pdf>
- <http://slideplayer.es/slide/2444305/#>



Bibliografía (adicional)

Spetale F., Tapia E., Murillo J., Krsticevic F, Ponce S., Bulacio P. Proper integration of feature subsets boosts GO subcellular localization predictions. *Revista Argentina de Bioingeniería*, Vol 22, Nro 1, 2018.

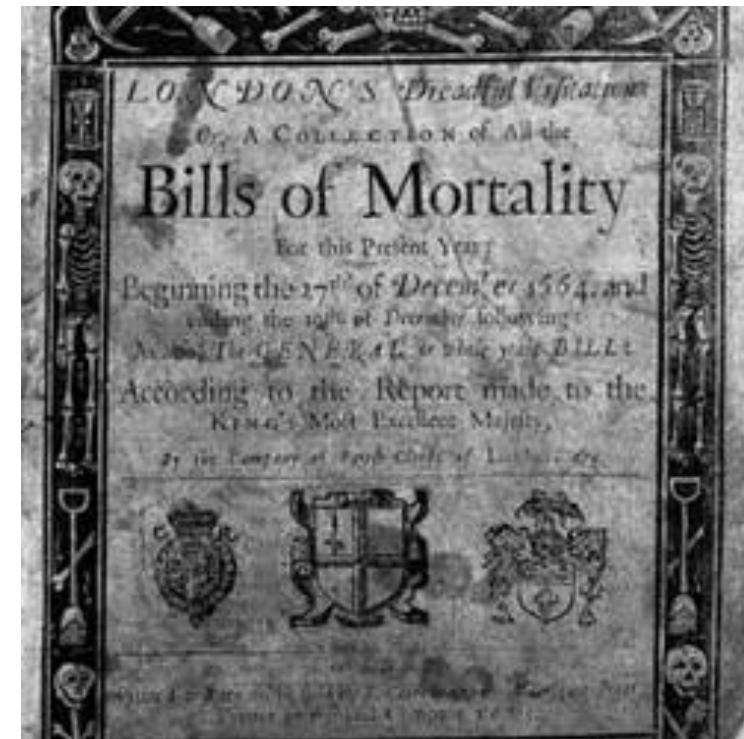
Spetale F., Bulacio P., Krsticevic F., Ponce S., and Tapia, E. Formalization of Gene Ontology relationships with factor graph towards Biological Process prediction. IFMBE Proceeding, Springer Nature Singapur, pp 58-61, 2017.

Spetale FE, Tapia E, Krsticevic F, Roda F, Bulacio P. *A Factor Graph Approach to Automated GO Annotation*. PLoS ONE 11(1): e0146986. doi: 10.1371/journal.pone.0146986, 2016.

Etiquetar, un viejo problema: 1662, Londres

<http://www.ncbi.nlm.nih.gov/pmc/articles/PMC200893/pdf/mlab00259-0008.pdf>

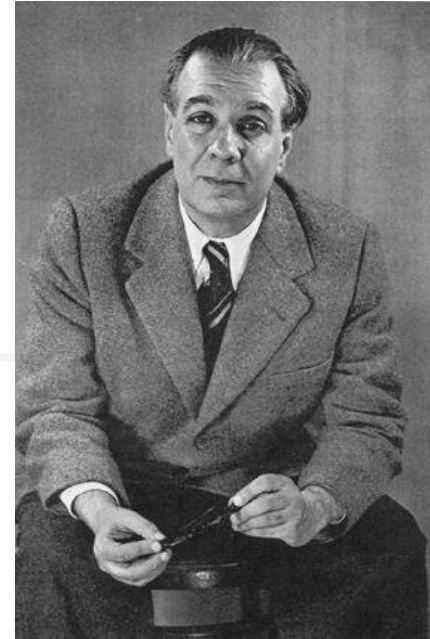
- Harold W. Jones. "It is interesting to examine the tables of "notorious diseases." It can hardly fail to remind us of the general run of death certificates in the more backward of our American communities of forty years ago. Here we discover that apoplex, falling sickness, dead in the streets, headache, lethargy, lunatique, overlaid and starved, palsy, sodainly, -frighted, hanged themselves, smothered, and vomiting were admitted as sufficient explanation of the cause of death."



***National Center for Biotechnology Information

Etiquetar: los animales...

- (a) pertenecientes al emperador,
- (b) embalsamados,
- (c) amaestrados,
- (d) lechones,
- (e) sirenas,
- (f) fabulosos,
- (g) perros sueltos,
- (h) incluidos en esta clasificación,
- (i) que tiemblan como enojados,
- (j) innumerables,
- (k) dibujados con un pincel finísimo de pelo de camello,
- (l) etcétera,
- (m) que acaban de romper un jarrón,
- (n) que de lejos parecen moscas.



(...)notoriamente no hay clasificación del universo que no sea arbitraria y conjetural. La razón es muy simple: no sabemos qué cosa es el universo".



Clasificar

Dado un grupo de datos $P = \{p_1, \dots, p_n\}$ \in un espacio de interés del Universo descrito por un conjunto de clases $C = \{c_1, \dots, c_c\}$,



- *Cómo caracterizo a los **datos**?*
- *Cómo defino las **clases**? Están **relacionadas**?*
- *Qué y Cómo **infiero**?*



Ontología

disciplina que trata con la naturaleza y organización de la realidad...

- **Explicitar** las suposiciones de un dominio
 - Vocabulario común
 - Restricciones
- **Comunicación:**
 - Entre personas/aplicaciones de SW (protocolos)
- **Reusar** el KW del dominio
 - Facilita la modificación/actualización



Ontología: empecemos...

Definiciones:

- Elefante: miembro del reino animal;
- Herbívoro: animal que se alimenta en base a plantas;
- Elefante adulto: elefante de más de 20 años.

Restricciones:

- Todos los elefantes son de tipo «africano» o «asiático»;
- Su peso es a lo sumo 2.000 kg;
- ...



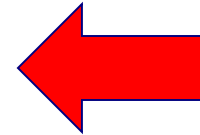
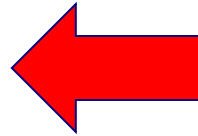
Qué necesito

- Representar el conocimiento
 - Formalismo de representación
 - Lenguaje
- Framework
 - Protégé
 - OBO Edit
 - ...
- Razonador



Qué necesito

- Representar el conocimiento
 - Formalismo de representación
 - Lenguaje
- Framework
 - Protégé
 - OBO Edit
 - ...
- Razonador





Representación de KW: Lógicas descriptivas (DL)

- **Descripciones de conceptos** para describir un dominio;
- +
- La semántica que establece una equivalencia entre las fórmulas de lógicas de descripción y expresiones en **lógica de predicados de primer orden**.

DL: extensión de *frames* (marcos) y redes semánticas, los cuales no representaban semántica basada en la lógica.

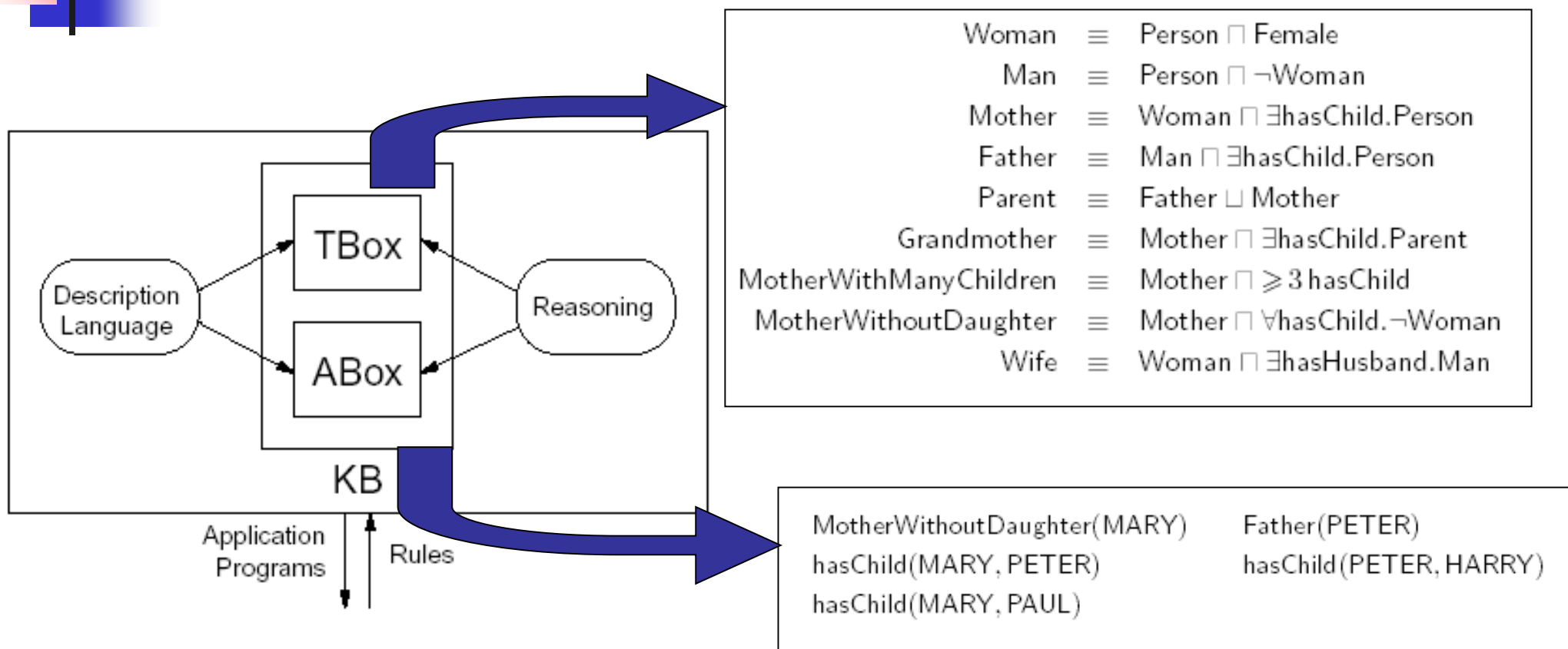


Lógica de representación: Lógica descriptiva

- La LD tiene una semántica formal basada en **expresiones lógicas**;
- Un **formalismo descriptivo**: **conceptos** (clases), **roles** (relaciones), **individuos** y **constructores**;
 - Un **formalismo terminológico**: declaración de términos generales, conceptos y propiedades (roles) de la *terminología descriptiva*.
 - Un **formalismo asertivo**: que introduce instancias sobre elementos y relaciones concretas del dominio. Instancias de conceptos e instancias de axiomas.
- Son capaces de inferir nuevo conocimiento a partir del conocimiento dado: algoritmos de razonamiento ***decidibles***.

Lógica Descriptiva

Familia



TBox (caja terminológica) contiene sentencias describiendo conceptos generales

ABox (caja de aserciones) contiene sentencias asociadas a los individuos/roles



Base de Conocimiento en LD

Una base de conocimiento DL **K** = **<T, A>**

- T (Tbox) es un conjunto de axiomas de la forma:
 - $C \sqsubseteq D$ (inclusión de concepto) C, D conceptos
 - $C \equiv D$ (equivalencia de concepto)
 - $R \sqsubseteq S$ (inclusión de rol) R, S roles
 - $R \equiv S$ (equivalencia de rol)
- A (Abox) es un conjunto de axiomas de la forma:
 - $x \in D$ (instanciación de concepto) x, y instancias
 - $\langle x, y \rangle \in R$ (instanciación de rol)

LD: Lenguajes

- **Constructores** para generar *conceptos y roles complejos* a partir de otros más simples (atómicos).
- Conjunto de **axiomas** para dar *aserciones* (propiedades) acerca de conceptos, roles e individuos.
- El **ALC** (Attributive Concept Language with Complements) es el DL más simple
 - Constructores incluyen booleanos: and \sqcap , or \sqcup , not \neg
 - Restricciones en los roles usando: \exists , \forall

E.g., Persona que *todos* sus hijos son *cualquiera* de los dos: Doctores o tienen un hijo Doctor

Person $\sqcap \forall \text{hasChild} . (\text{Doctor} \sqcup \exists \text{hasChild} . \text{Doctor})$



DL KB

- A **TBox** is a set of “schema” axioms (sentences), e.g.:

$\{\text{Doctor} \sqsubseteq \text{Person},$
 $\text{HappyParent} \equiv \text{Person} \sqcap \forall \text{hasChild}.(\text{Doctor} \sqcup \exists \text{hasChild}.\text{Doctor})\}$

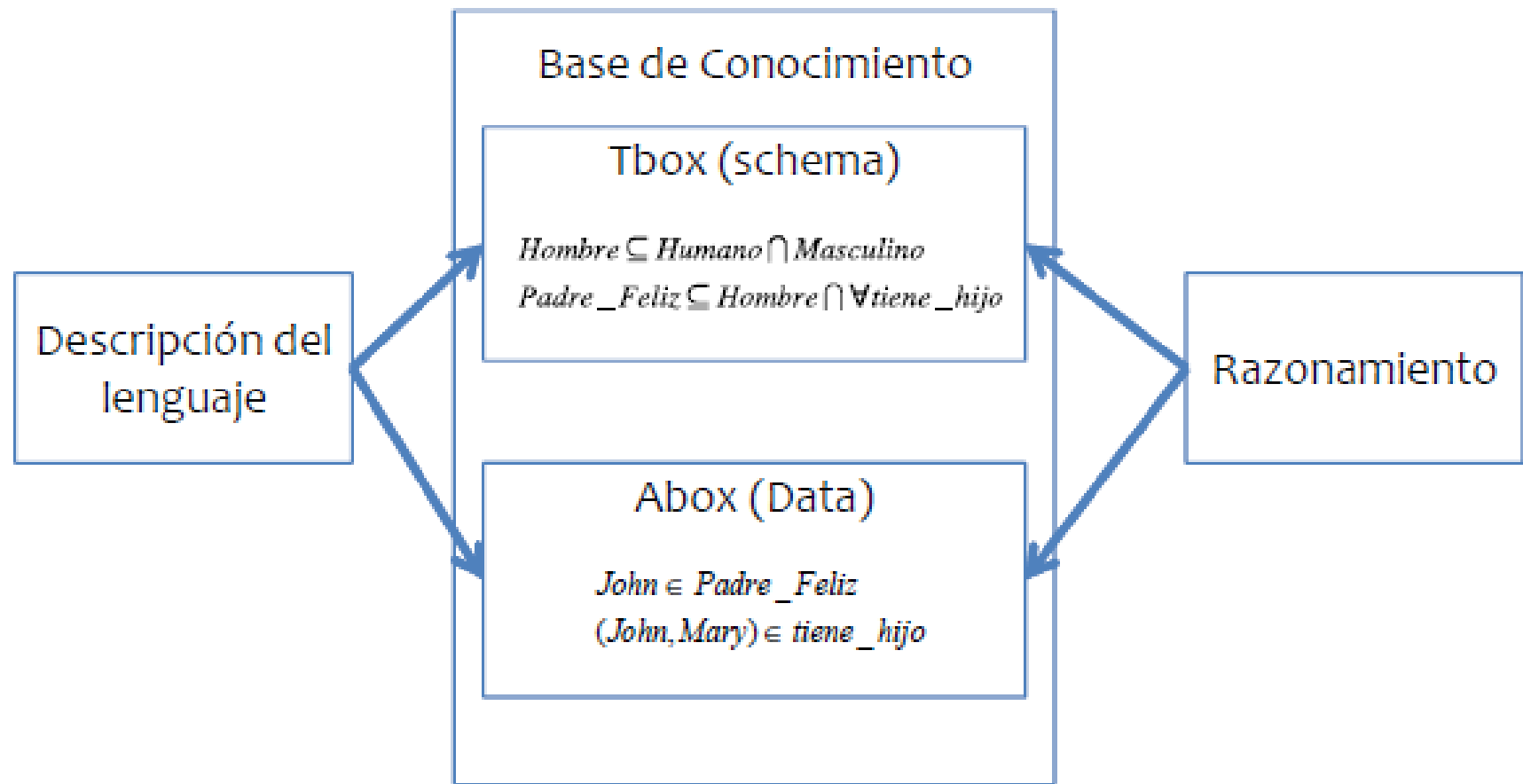
– i.e., a **background theory** (a set of non-logical axioms)

- An **ABox** is a set of “data” axioms (ground facts), e.g.:

$\{\text{John}:\text{HappyParent},$
 $\text{John hasChild Mary}\}$

– i.e., non-logical axioms including (restricted) use of nominals

DL KB: E.g.





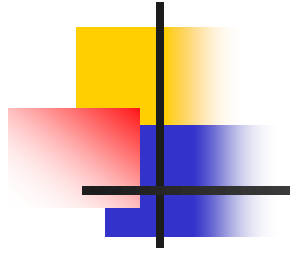
DL Basics

- **Concepts (formulae)**
 - E.g., Person, Doctor, HappyParent, (Doctor \sqcup Lawyer)
- **Roles (modalities)**
 - E.g., hasChild, loves
- **Individuals (nominals)**
 - E.g., John, Mary, Italy
- **Operators** (para formar conceptos y roles):
 - Computables (decidable) y si es posible, de baja complejidad



DL: Ej. de **Constructores** de conceptos y roles

- Restricciones numéricas de **cardinalidad** sobre roles, e.g., ≥ 3 hasChild, ≤ 1 hasMother
- **Nominales** (conceptos *singleton*), e.g., {Italy}
- Dominios concretos (tipos de datos), e.g., hasAge.(≥ 21)
- Roles **Inversos**, e.g., hasChild- (hasParent)
- Roles **Transitivos**, e.g., hasChild* (descendant)
- Composición de roles, e.g., hasParent o hasBrother (uncle)



ONTOLOGÍAS



Qué es una Ontología

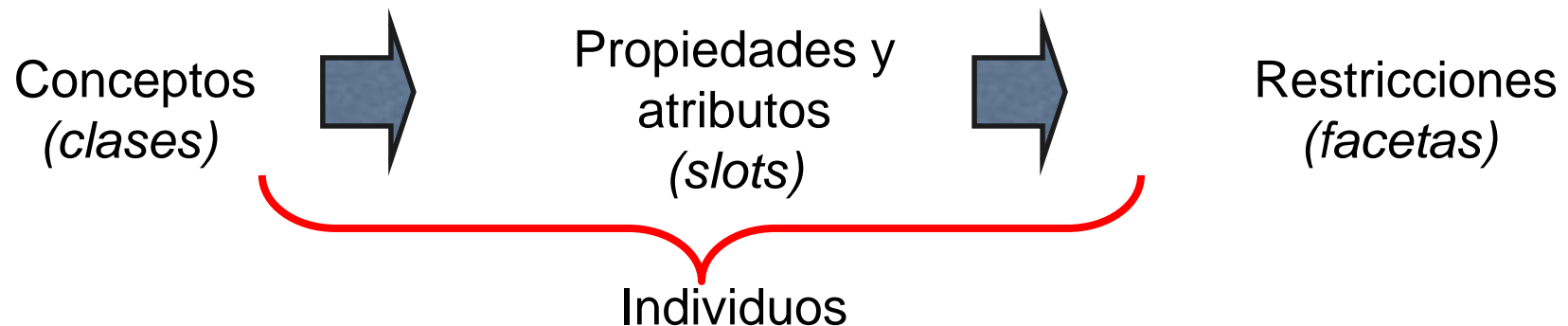
Def. formal de Gruber:

Una **ontología** define un conjunto de **primitivas representativas** con las que modelar un **dominio** de conocimiento o **discurso** (*meaning-constraints*).

Las **primitivas representativas** son típicamente **clases** (*sets*), **atributos** (*properties*), y **relaciones** (*class members relationships*).

Qué es una Ontología

- **Ontología:** descripción explícita de un dominio





Ontologías computacionales

- Describe formalmente un sistema conceptual
- Estructura: grafo
 - Cada *nodo* es un **concepto**
 - Los nodos se *unen* por **conectores tipados**
 - El conector “is-a” generalmente es un **grafo aciclico dirigido** (DAG):
 - Rooted: tiene una raíz (o varias como GO)
 - Directed: conectores con un sentido
 - Acyclic: no hay referencias circulares



Qué es una ontología

■ B. Chandrasekaran et al. 1999. What Are Ontologies, and Why Do We Need Them?. IEEE Intelligent Systems 14, 1, 20-26.

Cómo?

Con qué?

THEORIES IN AI FALL INTO TWO broad categories: ***mechanism*** theories and ***content*** theories.

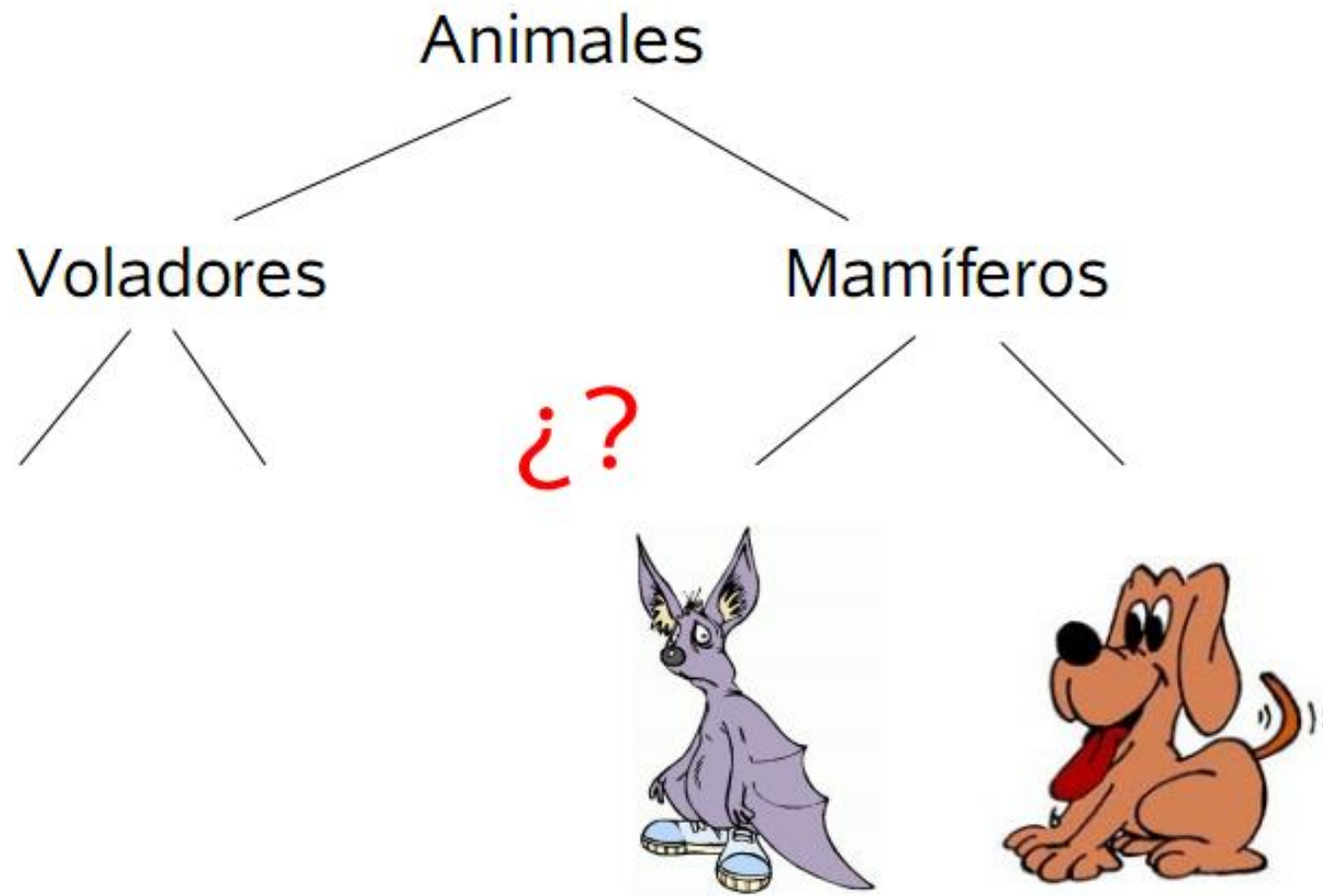
- Ontologies are **content** theories about the sorts of objects, properties of objects, and relations between objects that are possible in a specified domain of knowledge.
- They provide potential terms for describing our knowledge about the domain.



De ontología a KB...

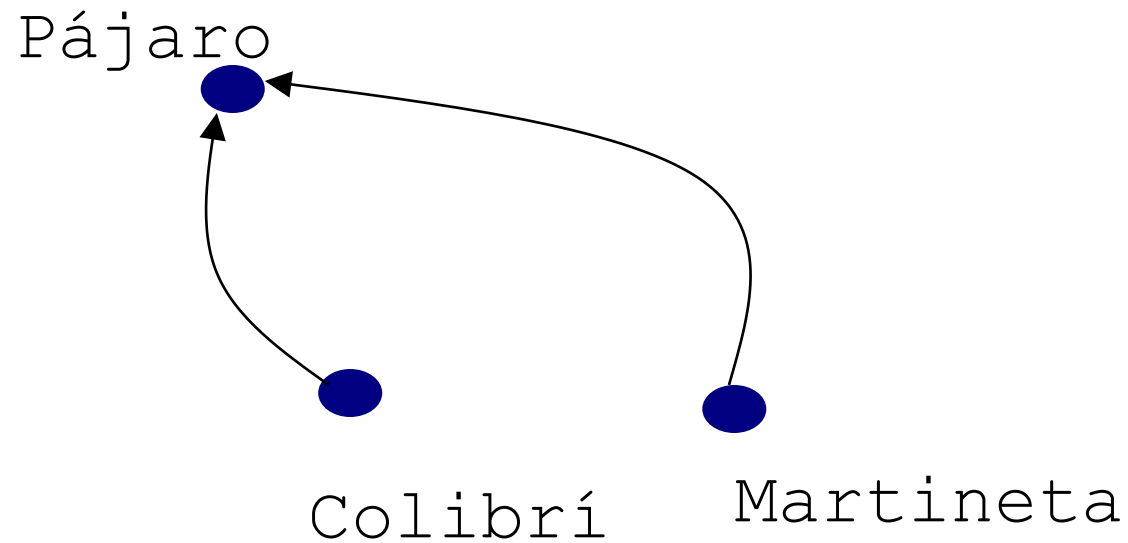
- Una ontología provee una **estructura** para describir un dominio de la cual puede construirse una KB.
 - Conjunto de conceptos
 - Relaciones...
- La KB usa estos términos para representar lo que es verdadero sobre algún caso particular.
- Ej.: Una ontología puede describir el dominio de medicina; una KB afirmaciones sobre cierto paciente que tiene una enfermedad...

Estructura



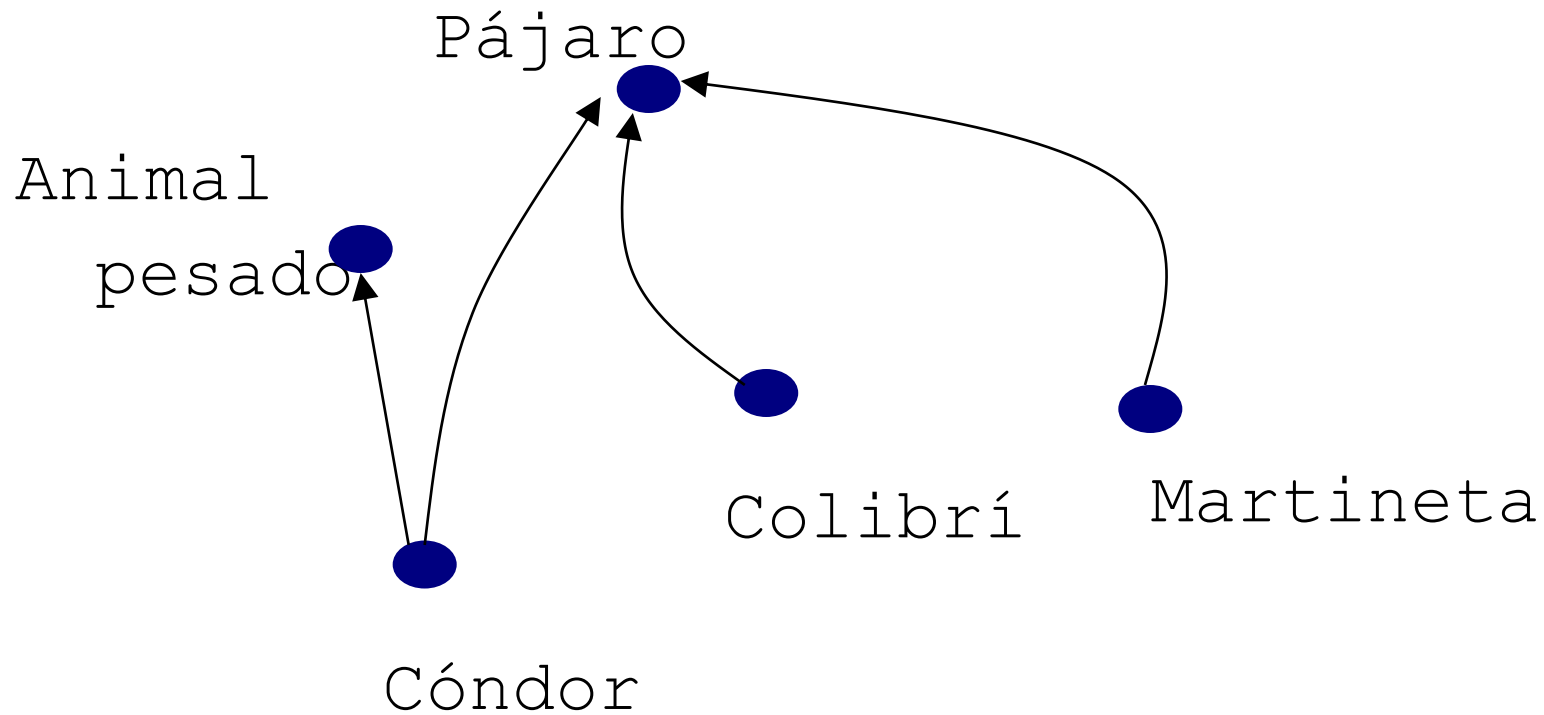


Ej. 1: cómo empezar?



Ej. 1

Apunto a conceptos más generales



Herencia múltiple...



Ej. 2 cómo representar?

**Luis tiene un perro
llamado Fido**

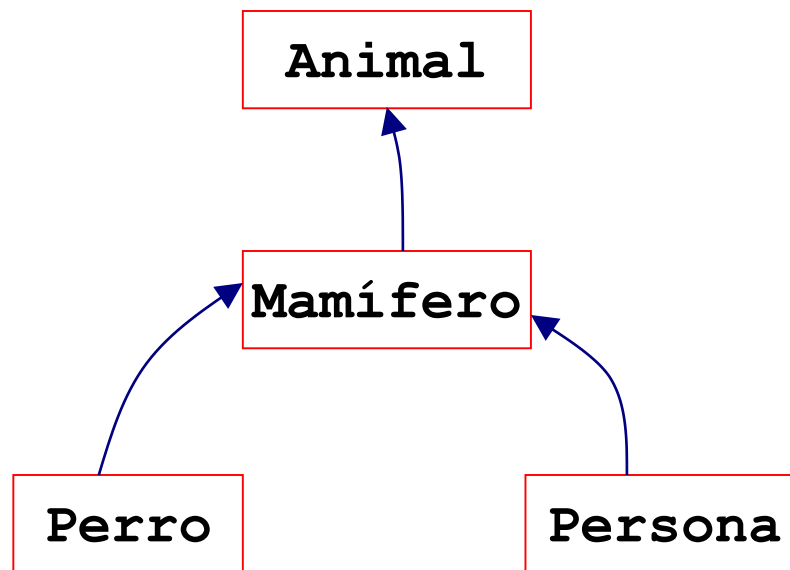


Ej. 2

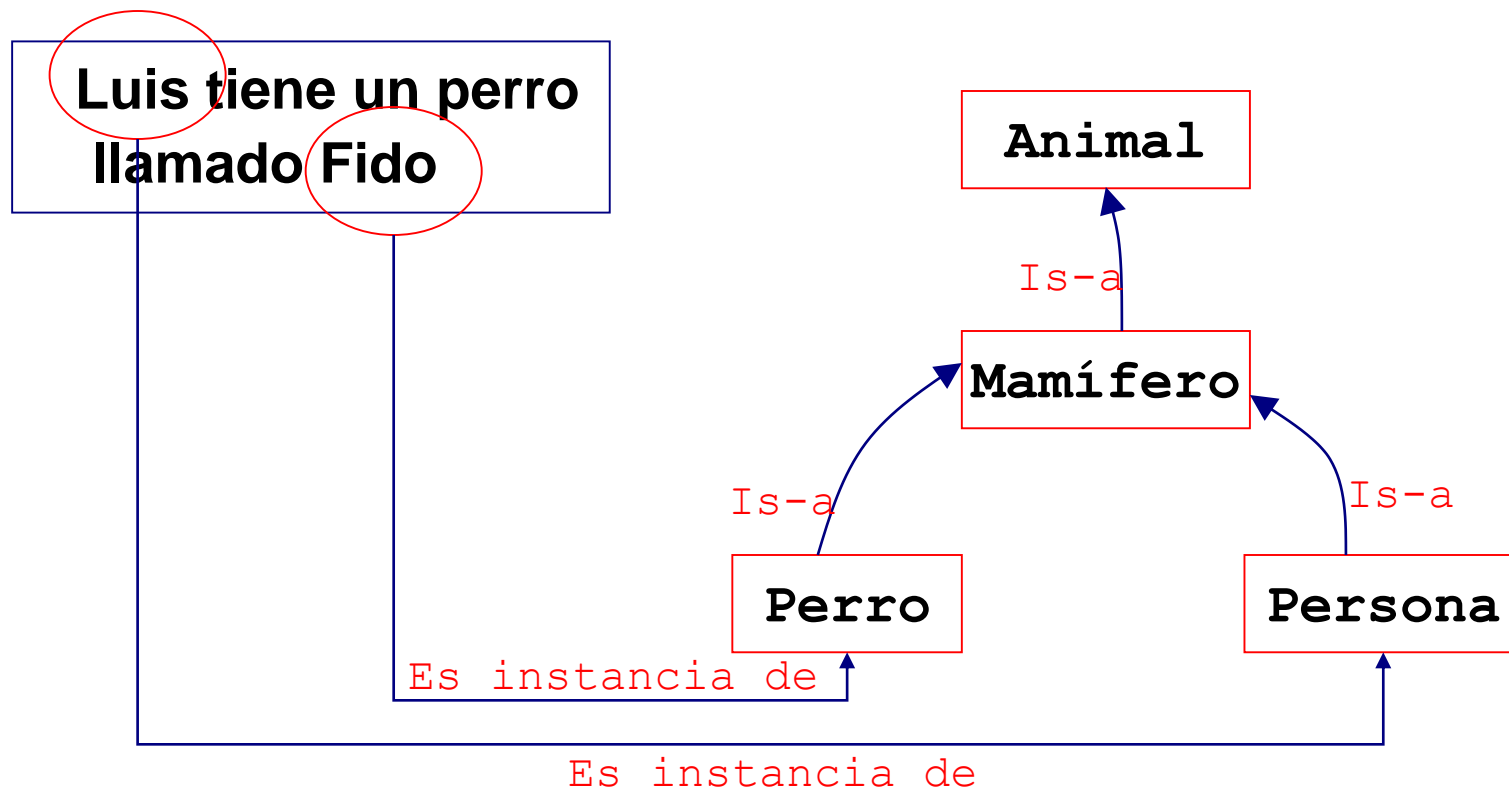
Luis tiene un perro
llamado Fido

Ej. 2

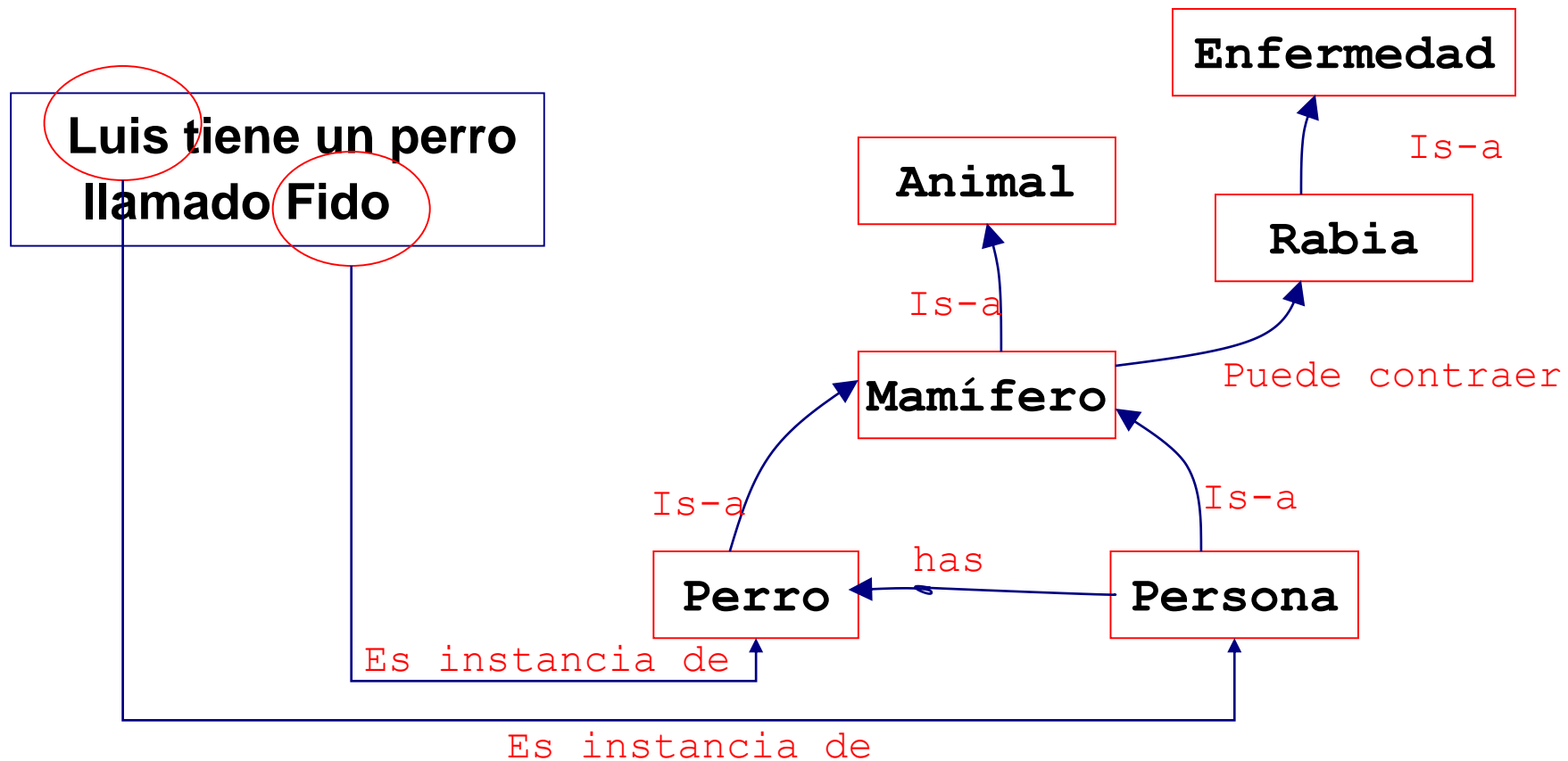
Luis tiene un perro
llamado Fido

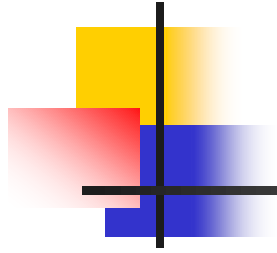


Ej. 2



Ej. 2





LENGUAJES DE ONTOLOGÍAS



Lenguajes de Ontologías

Un lenguaje de ontología

– usualmente introduce

- **Conceptos**: clases, entidades
- **Propiedades** de los conceptos: atributos, slots, roles.
- **Relaciones** entre conceptos.
- **Restricciones** de integridad

– pueden ser

- **simples**: solo conceptos.
- **frame-based**: solo conceptos y propiedades.
- **logic-based**: conceptos, propiedades y restricciones

– expresados a traves de diagramas.



Diferencias-Ontologías, Taxonomías-Modelos de Datos

Fuerte Semántica

(Logic-Based)

Logica Modal

Logica de Primer Orden

Teoría Local de Dominios

Description Logic

DAM+OIL, OWL

UML

Modelos Conceptuales

(Frame-Based)

RDF/S, F-logic

ERE

Thesaurus (conceptos tienen un significado)

ER

Taxonomías (es subclasificación de)

Modelo Relacional

Débil Semántica

(Simple)



Lenguaje de Ontologías: RDF a OWL

Dos lenguajes para tratar las deficiencias de RDF (Resource Description Framework)

- **OIL:** desarrollado en Europa
- **DAML-ONT:** desarrollado en USA (programa DARPA DAML)

Los esfuerzos se juntaron para producir **DAML+OIL**

- Realizado por “Joint EU/US Committee on Agent MarkupLanguages”
- Extiende (“un subconjunto de LD”) RDF

DAML+OIL subscribe a W3C como base para estandarización y se forma el grupo de trabajo Web-Ontology (**WebOnt**)

- WebOnt desarrolla el lenguaje **OWL** basado en DAML+OIL
- OIL, DAML+OIL and OWL son basados en **Description Logics**

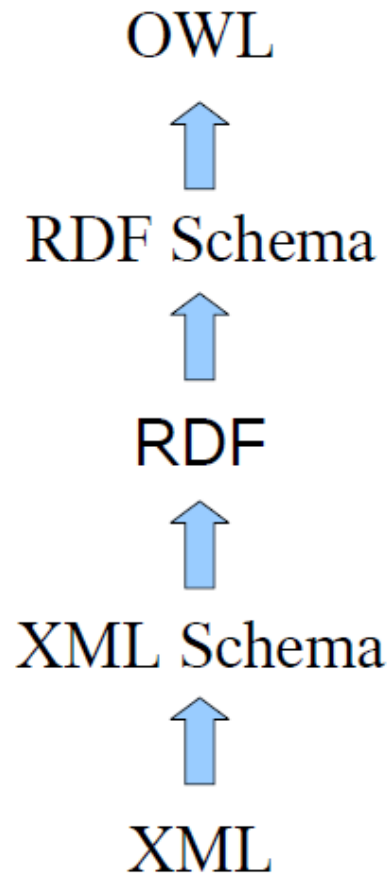
<http://www.w3.org/TR/owl-ref/#ref-owl-guide>



Lenguaje de Ontologías: RDF a OWL

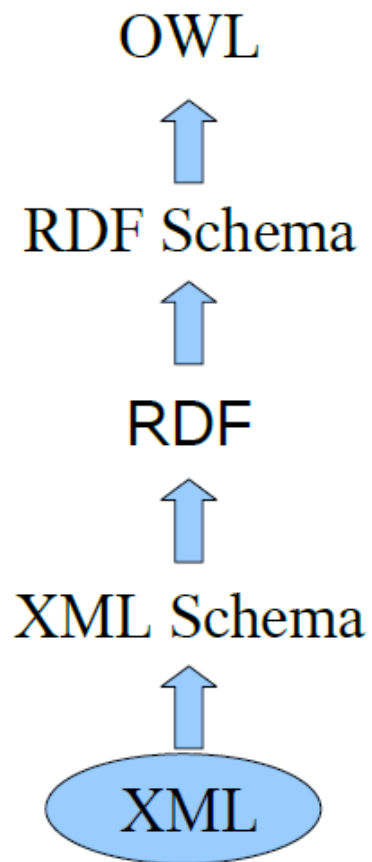
- Lenguajes basados en XML
 - RDF → RDF Schema (RDFS)
- RDFS se reconoce como un Lenguaje de Ontologías
 - Classes y properties
 - Sub/super-classes (y properties)
 - Range y domain (de properties)
- Pero RDFS débil Para describir los recursos en detalle, e.g.:
 - No hay restricciones de rango y dominio, de participación/cardinalidad
 - No hay propiedades transitiva, inversa o simétrica

Lenguaje de Ontologías





Lenguaje de Ontologías

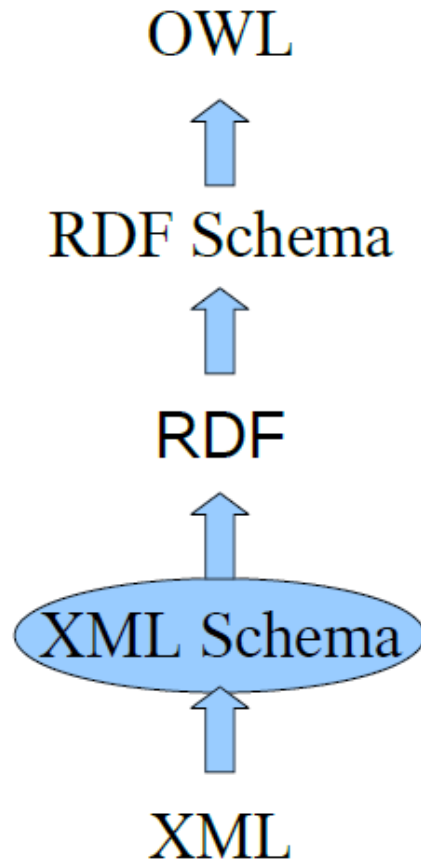


Es una sintaxis para documentos semi-estructurados.

No proporciona información semántica

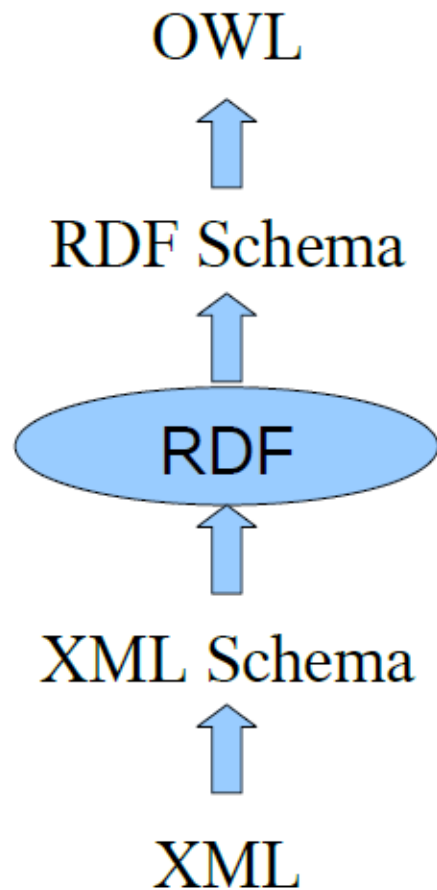


Lenguaje de Ontologías



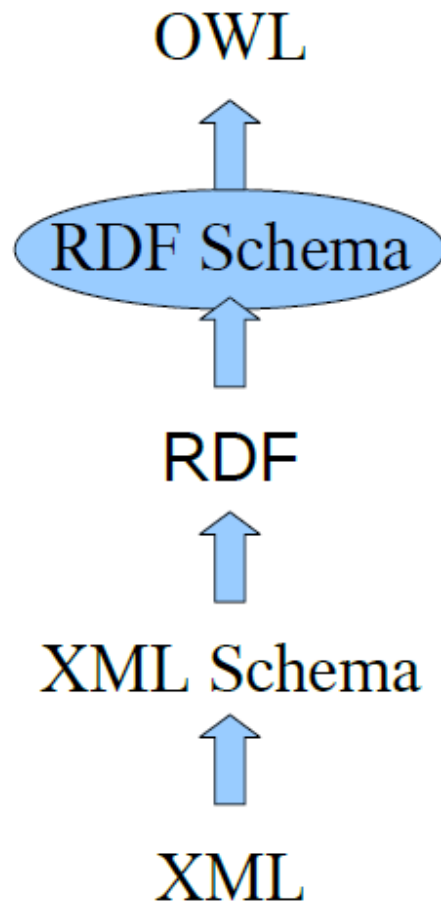
Lenguaje que restringe la estructura de XML. Además, le proporciona la capacidad de manejar tipos de datos

Lenguaje de Ontologías



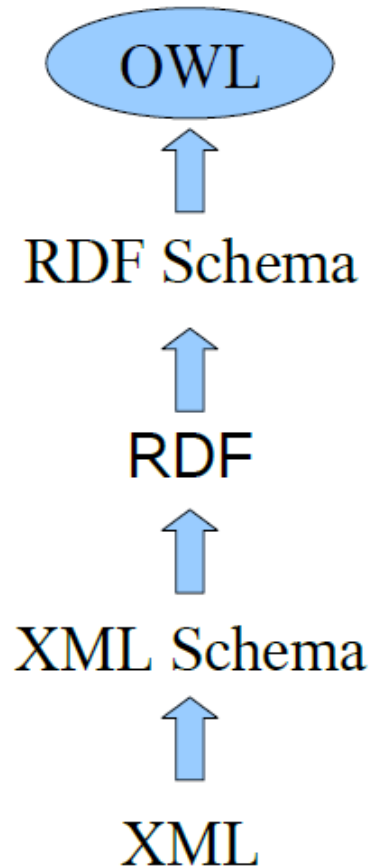
Modelo de datos para objetos (recursos) y las relaciones entre ellos. Ya tiene la capacidad de expresar cierta semántica

Lenguaje de Ontologías



**Vocabulario para la descripción
de propiedades y clases
de recursos RDF. Cuenta con
semántica para la generalización
de jerarquías de las
propiedades de las clases**

Lenguaje de Ontologías



**Provee de más vocabulario
para la descripción de
propiedades y clases, por
ejemplo:**

- relaciones entre clases
- cardinalidad
- equivalencia
- características de las propiedades

Class/Concept Constructors OWL

Constructor	DL Syntax	Example	FOL Syntax
intersectionOf	$C_1 \sqcap \dots \sqcap C_n$	Human \sqcap Male	$C_1(x) \wedge \dots \wedge C_n(x)$
unionOf	$C_1 \sqcup \dots \sqcup C_n$	Doctor \sqcup Lawyer	$C_1(x) \vee \dots \vee C_n(x)$
complementOf	$\neg C$	\neg Male	$\neg C(x)$
oneOf	$\{x_1\} \sqcup \dots \sqcup \{x_n\}$	{john} \sqcup {mary}	$x = x_1 \vee \dots \vee x = x_n$
allValuesFrom	$\forall P.C$	\forall hasChild.Doctor	$\forall y.P(x, y) \rightarrow C(y)$
someValuesFrom	$\exists P.C$	\exists hasChild.Lawyer	$\exists y.P(x, y) \wedge C(y)$
maxCardinality	$\leq nP$	≤ 1 hasChild	$\exists^{\leq n} y.P(x, y)$
minCardinality	$\geq nP$	≥ 2 hasChild	$\exists^{\geq n} y.P(x, y)$

- C is a concept (class); P is a role (property); x_i is an individual/nominal

Ontology Axioms

OWL Syntax	DL Syntax	Example
subClassOf	$C_1 \sqsubseteq C_2$	Human \sqsubseteq Animal \sqcap Biped
equivalentClass	$C_1 \equiv C_2$	Man \equiv Human \sqcap Male
subPropertyOf	$P_1 \sqsubseteq P_2$	hasDaughter \sqsubseteq hasChild
equivalentProperty	$P_1 \equiv P_2$	cost \equiv price
transitiveProperty	$P^+ \sqsubseteq P$	ancestor ⁺ \sqsubseteq ancestor

OWL Syntax	DL Syntax	Example
type	$a : C$	John : Happy-Father
property	$\langle a, b \rangle : R$	\langle John, Mary \rangle : has-child

- **OWL ontology** equivalent to **DL KB** (Tbox + Abox)

OWL RDF/XML Exchange Syntax

Person \sqcap \forall hasChild.(Doctor \sqcup \exists hasChild.Doctor)

```
<owl:Class>
  <owl:intersectionOf rdf:parseType=" collection">
    <owl:Class rdf:about="#Person"/>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasChild"/>
      <owl:allValuesFrom>
        <owl:unionOf rdf:parseType=" collection">
          <owl:Class rdf:about="#Doctor"/>
          <owl:Restriction>
            <owl:onProperty rdf:resource="#hasChild"/>
            <owl:someValuesFrom rdf:resource="#Doctor"/>
          </owl:Restriction>
        </owl:unionOf>
      </owl:allValuesFrom>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```




Ejemplo...



Where are ontologies used?

- e-Science, e.g., Bioinformatics
 - The Gene Ontology
 - The Protein Ontology (MGED)
 - “in silico” investigations relating theory and data
- Medicine
 - Terminologies
- Databases
 - Integration
 - Query answering
- User interfaces
- Linguistics
- The Semantic Web



the Gene Ontology

Search

gene or protein name

[Downloads](#)[Tools](#)[Documentation](#)[Projects](#)[About](#)[Contact](#)

[GO Helpdesk](#)[Mailing lists](#)[GO on SourceForge](#)

Quick Links

- Tools
- AmiGO browser
- OBO-Edit ontology editor
- Ontology downloads
- Annotation downloads
- Database downloads
- Documentation
- GO FAQ
- GO on SourceForge
- Contact GO

News

- GO on Twitter
- New GO annotation

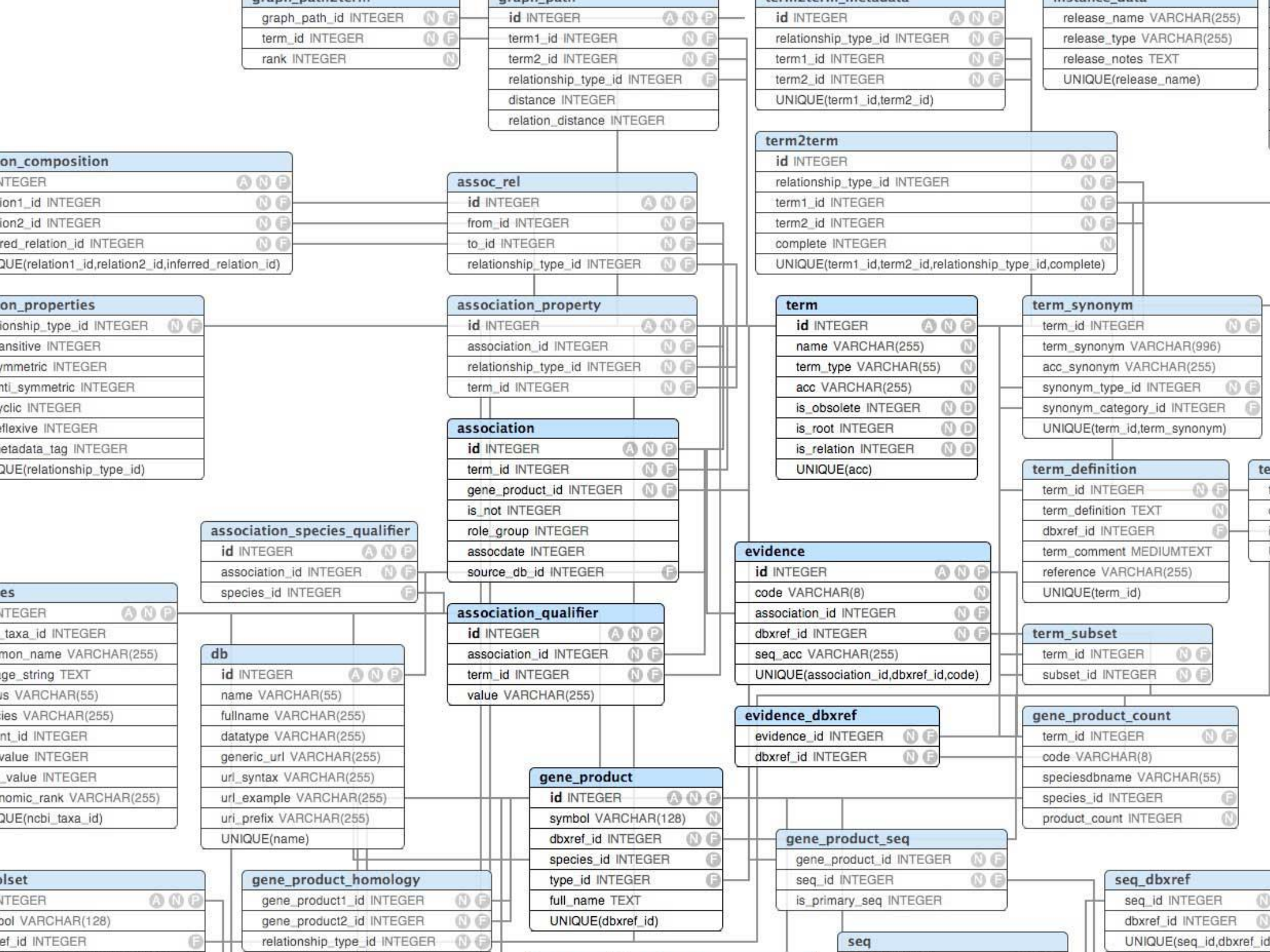
Welcome to the Gene Ontology website!

The Gene Ontology project is a major bioinformatics initiative with the aim of standardizing the representation of gene and gene product attributes across species and databases. The project provides [a controlled vocabulary of terms](#) for describing gene product characteristics and [gene product annotation data](#) from GO Consortium members, as well as [tools to access and process this data](#). [Read more about the Gene Ontology...](#)

Search the Gene Ontology Database

Search for genes, proteins or GO terms using [AmiGO](#):

☒ gene or protein name ☐ GO term or ID



OWL: Web Ontology Language

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE go:go PUBLIC "-//Gene Ontology//Custom XML/EDF version 1.0//EN"
"http://www.geneontology.org/dtd/go.dtd">
<go:go xmlns:go="http://www.geneontology.org/dtd/go.dtd#"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <rdf:RDF>
    <go:term rdf:about="http://www.geneontology.org/go#GO:0000001">
      <go:accession>GO:0000001</go:accession>
      <go:name>mitochondrion inheritance</go:name>
      <go:synonym>mitochondrial inheritance</go:synonym>
      <go:definition>The distribution of mitochondria, including the
mitochondrial genome, into daughter cells after mitosis or meiosis,
mediated by interactions between mitochondria and the cytoskeleton.
</go:definition>
      <go:is_a rdf:resource="http://www.geneontology.org/go#GO:0048308" />
      <go:part_of rdf:resource="http://www.geneontology.org/go#GO:0009530"
/>
      <go:negatively_regulates
        rdf:resource="http://www.geneontology.org/go#GO:0006312" />
    </go:term>
```



Ingeniería de Ontologías



Qué es “Ingeniería de Ontología”?

- **Ingeniería de Ontología:** Define **términos** en el dominio y **relaciones** entre ellos
 - Define **conceptos** en el dominio (**classes**)
 - Ordena los conceptos en una **jerarquía** (**subclass-superclass hierarchy**)
 - Define **atributos** y **propiedades** (**slots**) de las clases y las restricciones en sus valores
 - Define **individuos** y completa sus valores de atributos-propiedades

Pipeline: Paso a paso

1. Determinar el dominio y alcance de la ontología
2. Considerar la reutilización de ontologías existentes
3. Enumerar términos importantes
4. Definir las clases – jerarquías
5. Definir las propiedades de las clases: slots
6. Definir “constraints”/restricciones
7. Crear Instancias



1. Determinar Dominio y Alcance



determine
scope

consider
reuse

enumerate
terms

define
classes

define
properties

define
constraints

create
instances

- ¿Cuál es el dominio que cubrirá la ontología?
- ¿Para qué será usada?
- Para qué tipo de preguntas debe proveer respuestas (**competency questions**)?



1. Competency Questions: wine

- ¿Qué **características** del vino deberían considerarse al elegir uno?
- ¿Bordeaux es un vino **tinto** o **blanco**?
- ¿Cabernet Sauvignon va bien con *mariscos*?
- ¿Cuál es la mejor opción de vino para la *carne asada*?
- ¿Qué características de un vino afectan su adecuación para un *plato*?
- ¿El **sabor** o el **cuerpo** de un vino específico cambian con el **año de vendimia**?
- ¿Qué **vendimias** fueron buenas para **Napa Zinfandel**?

2. Analizar Reuso



- ¿Para qué reusar otras ontologías?
 - Para ahorrar **esfuerzo**
 - Para **interactuar** con herramientas que usan otras ontologías
 - Para usar ontologías que **han sido validadas**



2. Ej. Qué Reusar?

- Ontology libraries

- Protégé ontology library (protege.stanford.edu/ontologies.html)
- DAML ontology library (www.daml.org/ontologies)
- Ontolingua ontology library
(www.ksl.stanford.edu/software/ontolingua/)

- Upper ontologies

- IEEE Standard Upper Ontology (suo.ieee.org)
- Cyc (www.cyc.com)

3. Enumerar Términos Importantes



- De **que** *términos* necesitamos hablar?
- Cuales son las **propiedades** de esos *términos*?
- Qué queremos **decir** acerca de esos *términos*?



3. Enumerando Términos: wine

- *vino, uva, bodega, ubicación...*
- *color, cuerpo, sabor, contenido de azúcar...*
- *Vino blanco, vino tinto (rojo), vino rosado...*
- *comida, pescado, mariscos, carne, vegetales, queso...*

4. Definición de Clases y Jerarquías



- Una clase es un **concepto** del dominio
 - clase de vinos
 - clase de bodegas
 - clase de vinos tintos
- Una clase es una **colección** de elementos con propiedades similares
- **Instancias** de clases
 - Un vaso de vino de California Ud. tendrá para la cena



4. Herencia de Clases: IS A

- Las clases constituyen una **taxonomía jerárquica** (subclass-superclass)
 - *Una instancia de una subclase es una instancia de una superclase*
- Siendo una clase un **set** de elementos, una subclase es un **subset**



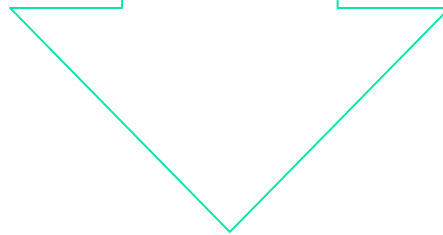
Herencia de Clase: Ej.

- Apple **es una** subclase de Fruit
 - *Cada manzana es una fruta*
- Red wine es una subclase de Wine
 - *Cada red wine es un wine*
- Chianti wine es una subclase de Red wine
 - *Cada Chianti wine es un red wine*



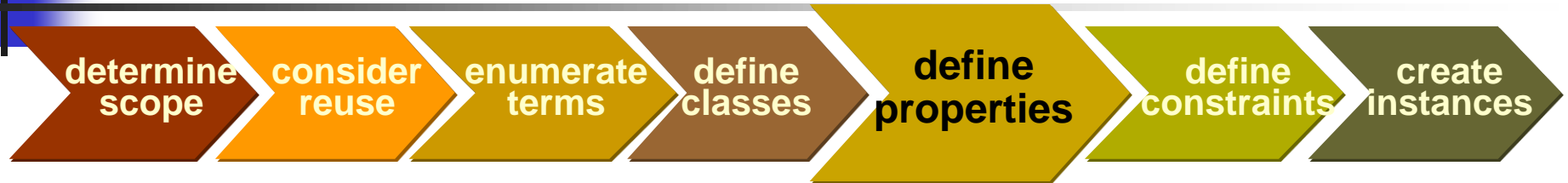
Documentación

- Clases (y slots)
 - Descripción de clases en lenguaje natural
 - Características del dominio relevantes para la def. de clases
 - Listado de sinónimos



Requisitos de Usuario

5. Definición de Propiedades de Clases: Slots



- Los **slots** describen los *atributos* de una instancia de una clase y la relación con otras instancias
 - *Por Ej. Cada wine tiene un color, contenido de azúcar...*



5. Propiedades (Slots)

■ Tipos

- intrínsecas: **flavor** y **color** de wine
- extrínsecas: **name** y **price** de wine
- Relaciones con otros objetos: **producer** de wine (bodega)

■ Simples y complejas

- simples (attributes): contienen valores (strings, numbers)
- complejas : contienen (o apuntan a) otros objetos (ej., una instancia de bodega)

5. Slots para la clase Wine

Template Slots									
Name	Type	Cardinality	Other Facets						
body	Symbol	single	allowed-values={FULL,MEDIUM,LIGHT}						
color	Symbol	single	allowed-values={RED,ROSÉ,WHITE}						
flavor	Symbol	single	allowed-values={DELICATE,MODERATE,STRONG}						
grape	Instance	multiple	classes={Wine grape}						
maker	Instance	single	classes={Winery}						
name	String	single							
sugar	Symbol	single	allowed-values={DRY,SWEET,OFF-DRY}						

Slot y Herencia




- Una subclase **hereda todos los slots** de una superclase
 - *Si un wine tiene name y flavor, un red wine también tiene name y flavor*
- Si una clase tiene **múltiples** superclases, hereda slots de todas ellas
 - *Oporto es un dessert wine y un red wine: hereda “sugar content: high” de dessert wine y “color:red” de red wine*

6. Restricciones de Propiedades



- Describen los posibles valores del slot
 - *El name de un wine es un string*
 - *El wine producer es una instancia de Winery (Bodega)*
 - *Una winery tiene una sola localización*

Facets for Slots at the Wine Class



Name	Type	Cardinality	Other Facets
S body	Symbol	single	allowed-values={FULL,MEDIUM,LIGHT}
S color	Symbol	single	allowed-values={RED,ROSÉ,WHITE}
S flavor	Symbol	single	allowed-values={DELICATE,MODERATE,STRONG}
S grape	Instance	multiple	classes={Wine grape}
S maker I	Instance	single	classes={Winery}
S name	String	single	
S sugar	Symbol	single	allowed-values={DRY,SWEET,OFF-DRY}

6. Common Facets: Restricciones



- Slot **cardinality** – the number of values a slot has
- Slot **value type** – the type of values a slot has
- **Minimum and maximum** value – a range of values for a numeric slot
- **Default** value – the value a slot has unless explicitly specified otherwise

6. Common Facets: Slot Cardinality

- **Cardinality**
 - Cardinality N means that the slot **must** have N values
- **Minimum cardinality**
 - Minimum cardinality 1 means that the slot must have a value (**required**)
 - Minimum cardinality 0 means that the slot value is **optional**
- **Maximum cardinality**
 - Maximum cardinality 1 means that the slot can have at most one value (**single-valued slot**)
 - Maximum cardinality greater than 1 means that the slot can have more than one value (**multiple-valued slot**)

6. Common Facets: Value Type

- **String**: a string of characters ("Château Lafite")
- **Number**: an integer or a float (15, 4.5)
- **Boolean**: a true/false flag
- **Enumerated type**: a list of allowed values (high, medium, low)
- **Complex type**: an instance of another class
 - Specify the class to which the instances belong
 - *The Wine class is the value type for the slot "produces" at the Winery class*

7. Create Instances



- Create an instance of a class
 - The class becomes a **direct type** of the instance
 - Any superclass of the direct type is a **type** of the instance
- Assign slot values for the instance frame
 - Slot values should conform to the facet constraints
 - Knowledge-acquisition tools often check that