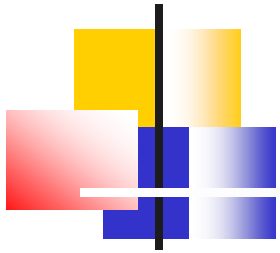


Representación del Conocimiento y Razonamiento con Ontologías

IIA-LCC

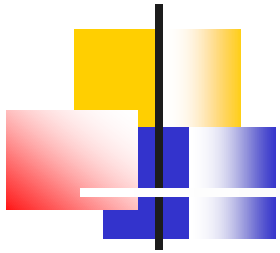
Ana Casali

Transparencias base de Pilar Bulacio (IIA-LCC) y Ian Horrocks http://www.cs.man.ac.uk/~horrocks/Slides/

Herramienta semántica: Ontologías

Las Ontologías sirven para:

- Explicitar y representar el conocimiento de un dominio
  - Vocabulario común
  - Restricciones
- Comunicación:
  - Entre personas/aplicaciones (protocolos)
- Reusar el KW del dominio
  - Facilita la modificación/actualización



Qué necesito

- Representar el conocimiento
  - Formalismo de representación (DL: Description Logic)
  - Lenguaje (OWL: Ontology Web Language)
- Framework
  - Protégé
  - OBO Edit
  - …
- Razonador

Qué es una Ontología

Una definición:

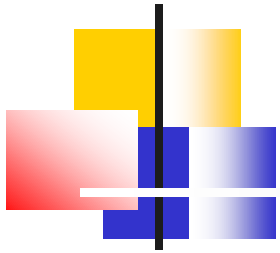
«Una ontología es una especificación explícita de una conceptualización». Una conceptualización es una abstracción, una vista simplificada del mundo que queremos representar.

Gruber (2003)

Qué es una Ontología

En IA ontología significa 2 cosas relacionadas

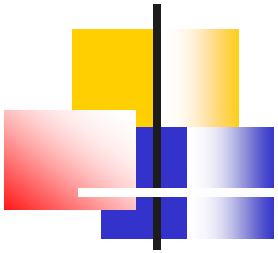
- Un vocabulario de representación especializado para algún dominio.
- Un cuerpo de conocimiento que describe algún dominio, por lo general un dominio de conocimiento de sentido común.



Qué es una Ontología

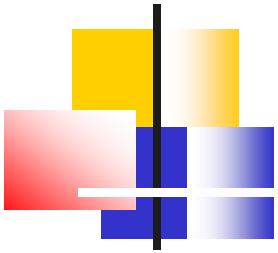
Una ontología modela un dominio de conocimiento usando primitivas representativas que son típicamente:

- ✓ clases (*sets*), personas, padres, madres
- ✓ atributos (*properties*) edad, sexo, dni
- ✓ relaciones (*class members relationships*). es-hijo-de, es-madre-de, es-hermano



Ontologías computacionales

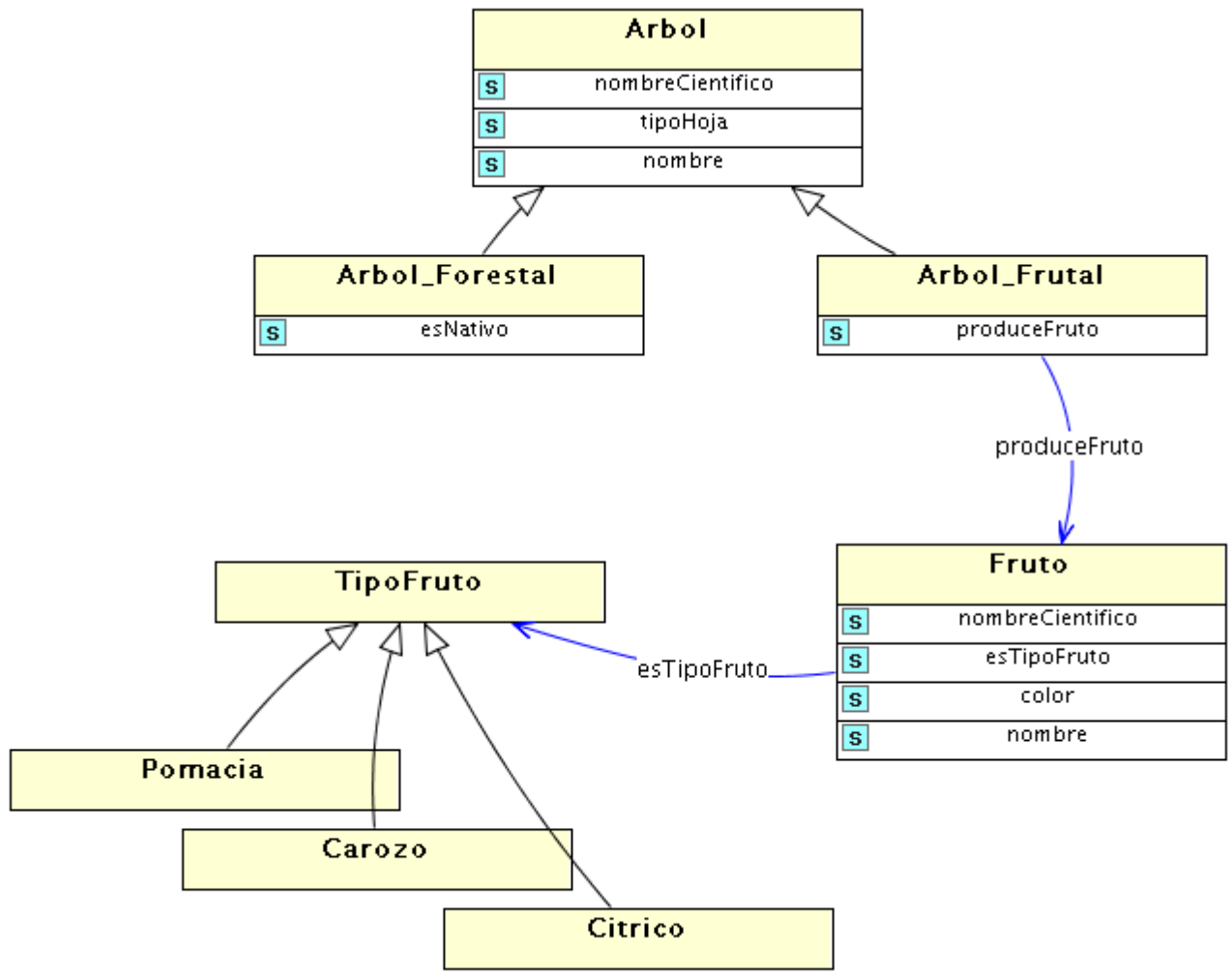
- Describe formalmente un sistema conceptual
- Estructura: grafo
  - Cada *nodo* es un concepto
  - Los nodos se *unen* por conectores tipados
  - El conector “is-a” genera un grafo acíclico dirigido (DAG):
    - Rooted: tiene una raíz
    - Directed: conectores con un sentido
    - Acyclic: no hay referencias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



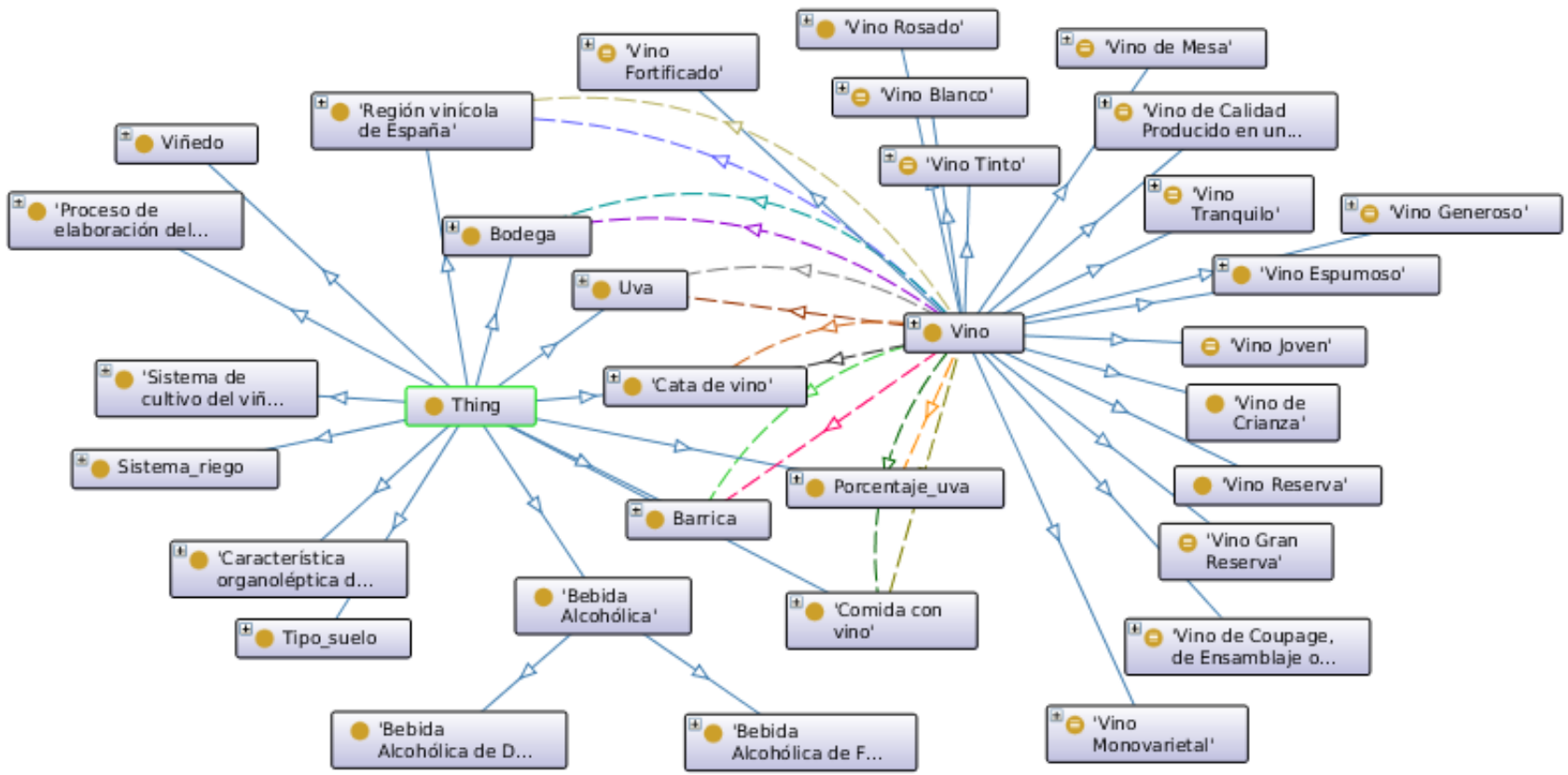
De ontología a KB…

- 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 la industria; puede contener afirmaciones sobre cierto tipo de fallas ...

Ejemplo

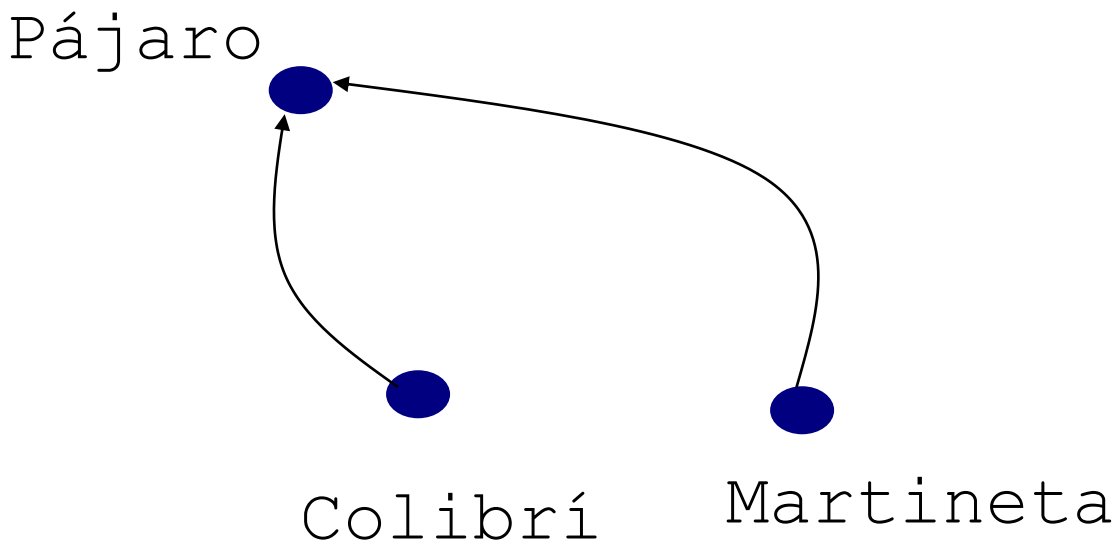

Arbol
nombreCientifico
tipoHoja
nombre
Arbol_Forestal
esNativo
Arbol_Frutal
produceFruto
produceFruto
Fruto
nombreCientifico
esTipoFruto
color
nombre
esTipoFruto
TipoFruto
Pomacia
Carozo
Citrico

Ejemplo: Ontología E-Baco



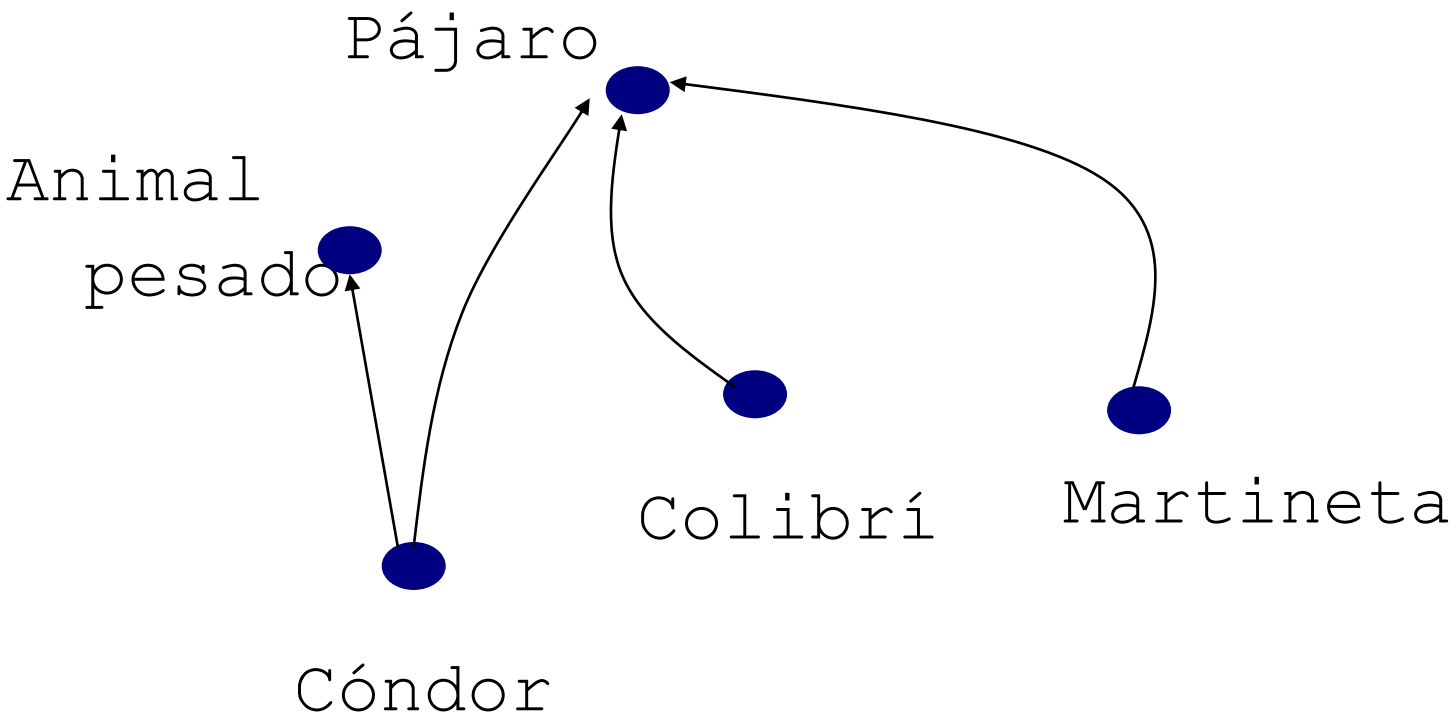
http://ebaco.github.io/es/introduction/

Ej. 1: cómo empezar?

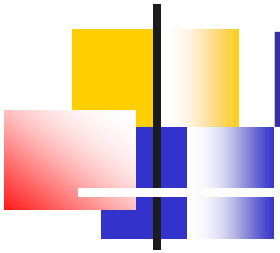


Ej. 1

Apunto a conceptos más generales

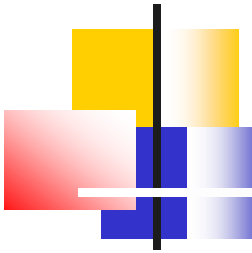


Herencia múltiple…



Ej. 2 cómo representar?

Luis tiene un perro
llamado Fido

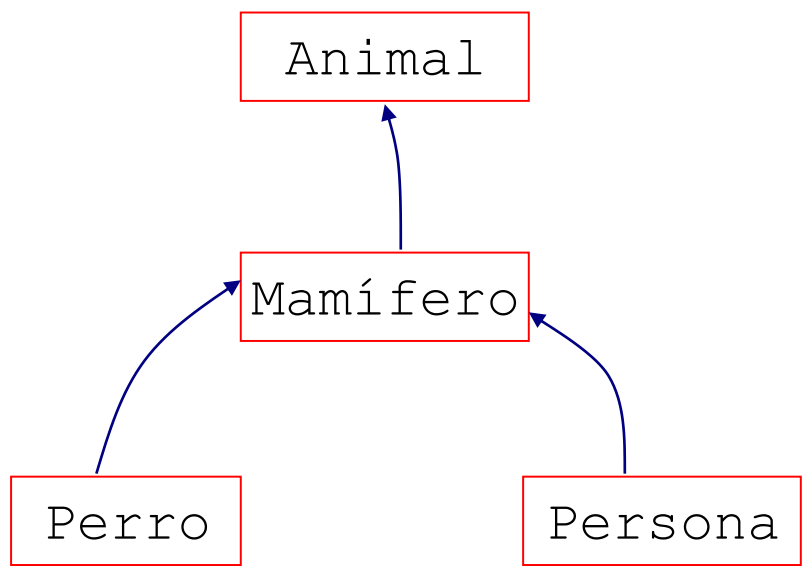


Ej. 2

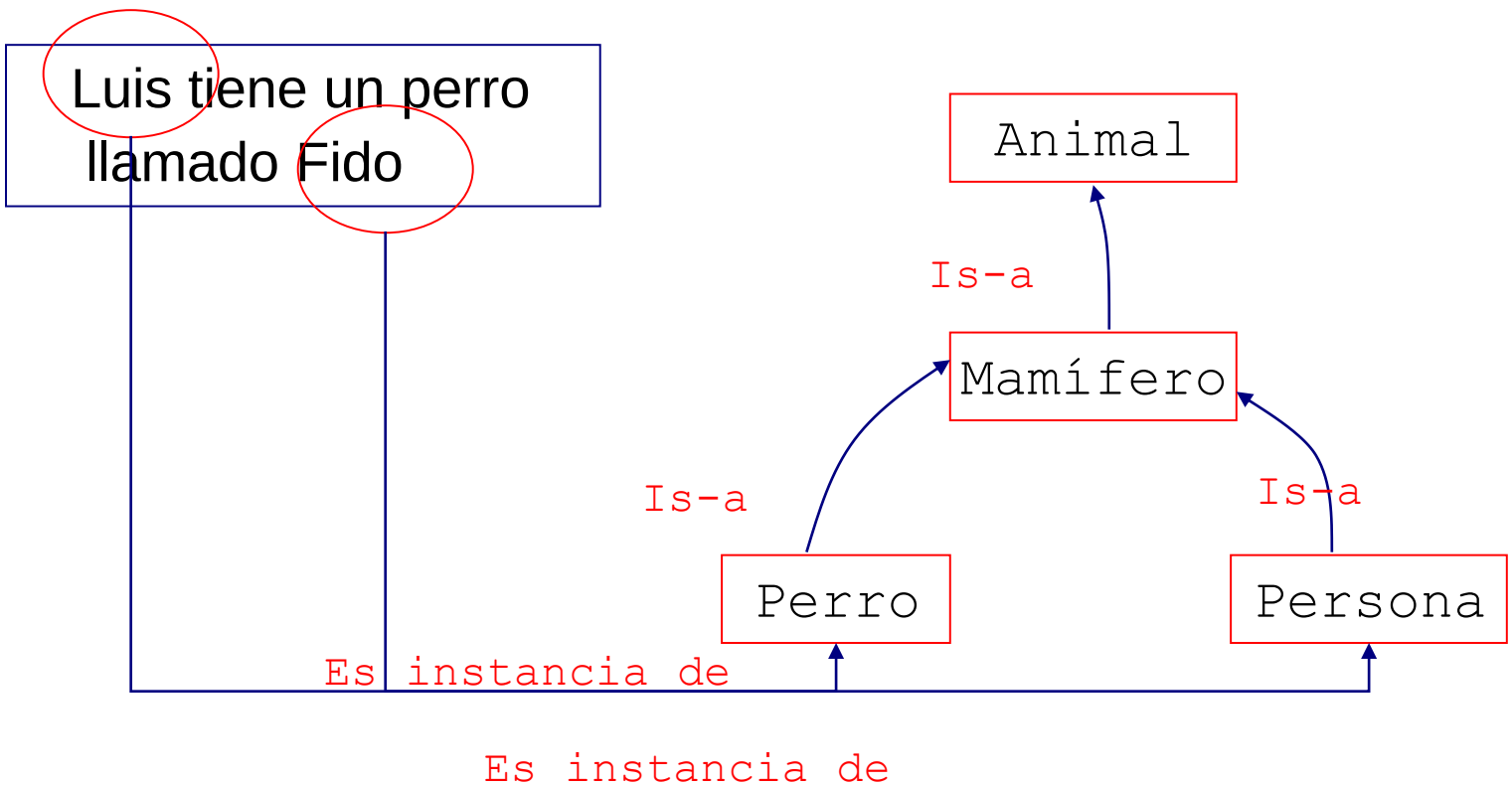
Luis tiene un perro llamado Fido

Ej. 2

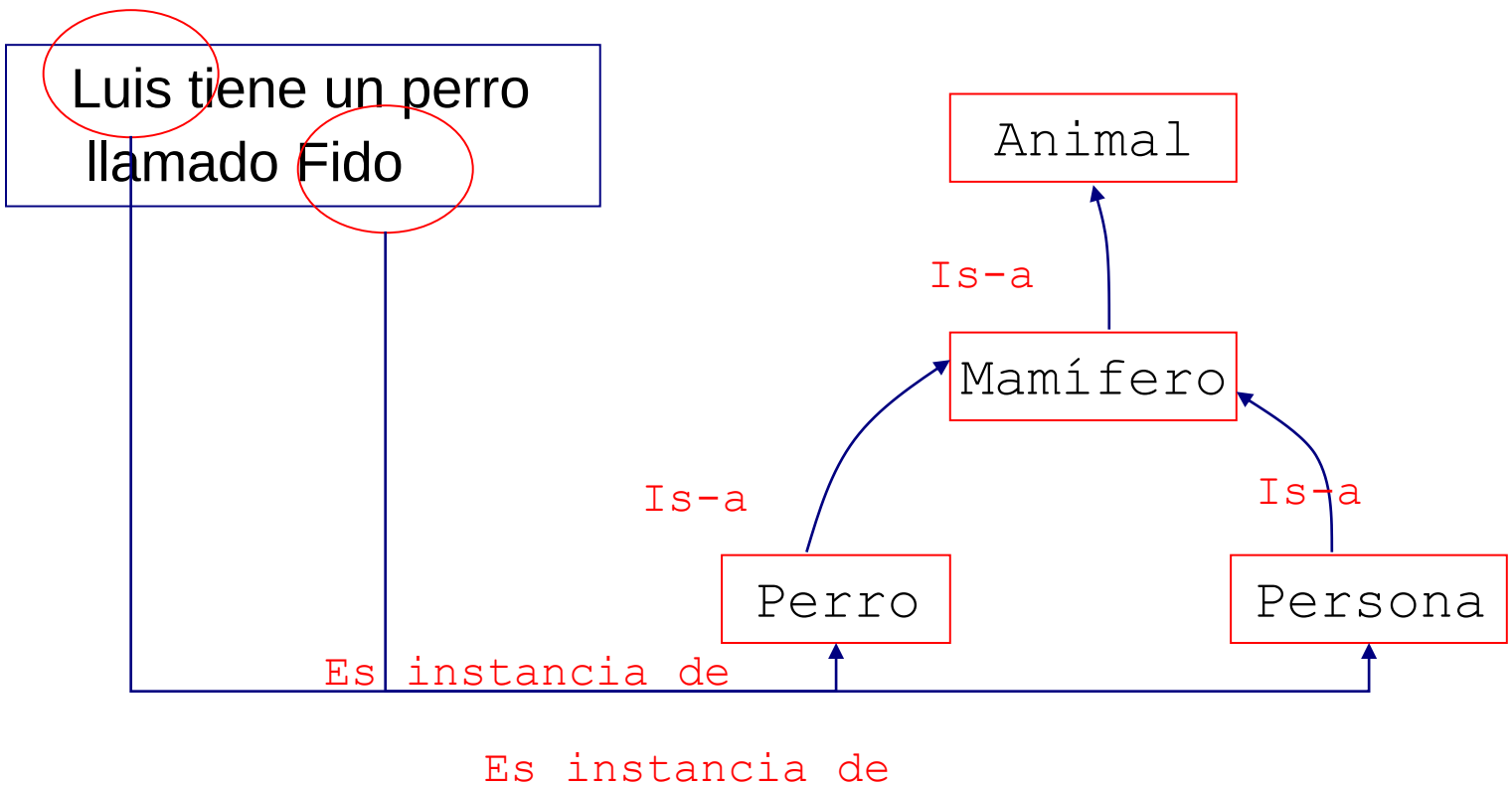
Luis tiene un perro llamado Fido



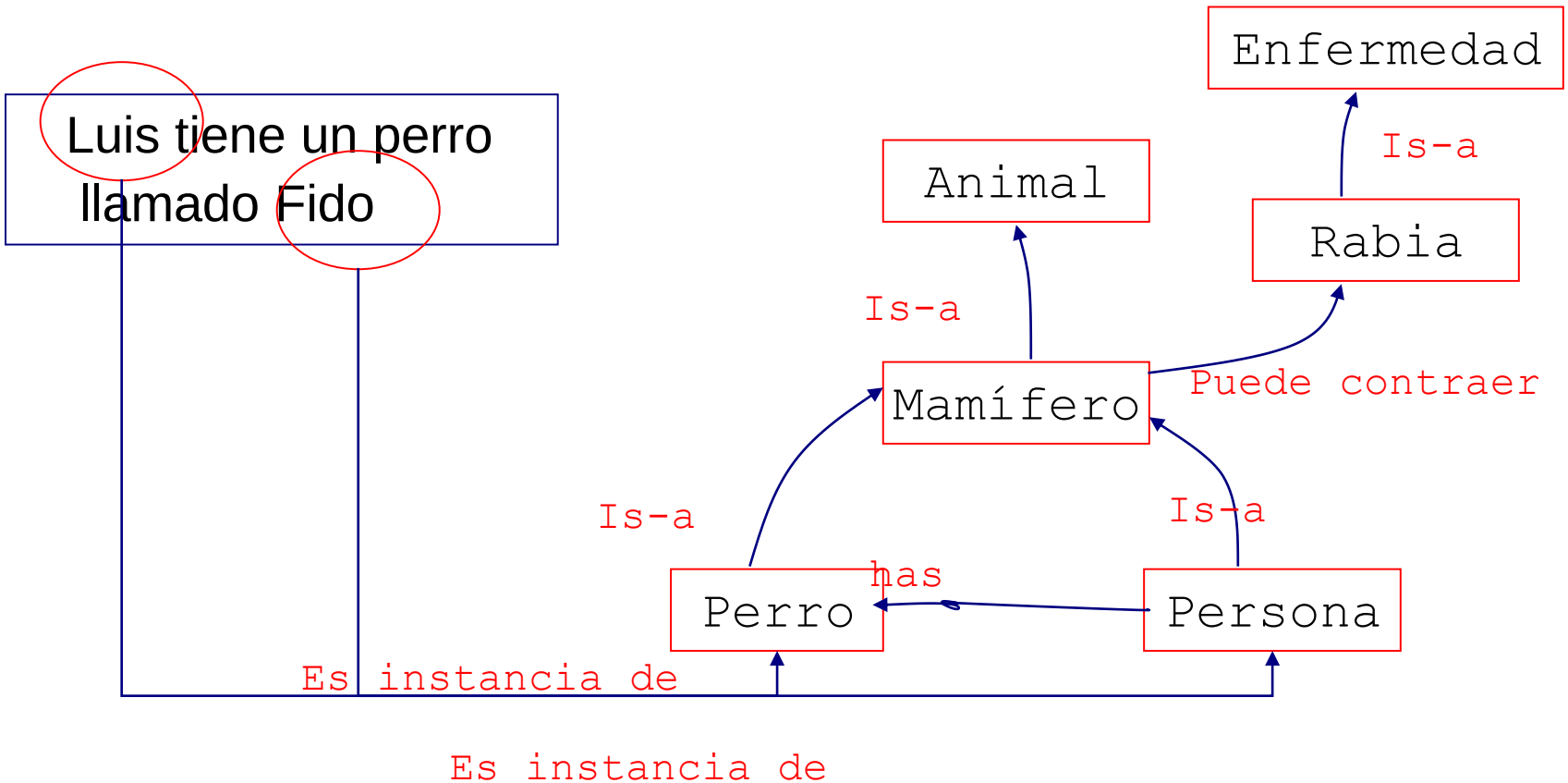
Ej. 2

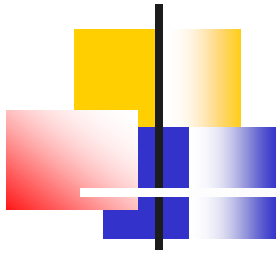

Luis tiene un perro llamado Fido
Animal
Is-a
Mamífero
Is-a
Is-a
Perro
Persona
Es instancia de
Es instancia de

Ej. 2


Luis tiene un perro llamado Fido
Animal
Is-a
Mamífero
Is-a
Is-a
Perro
Persona
Es instancia de
Es instancia de

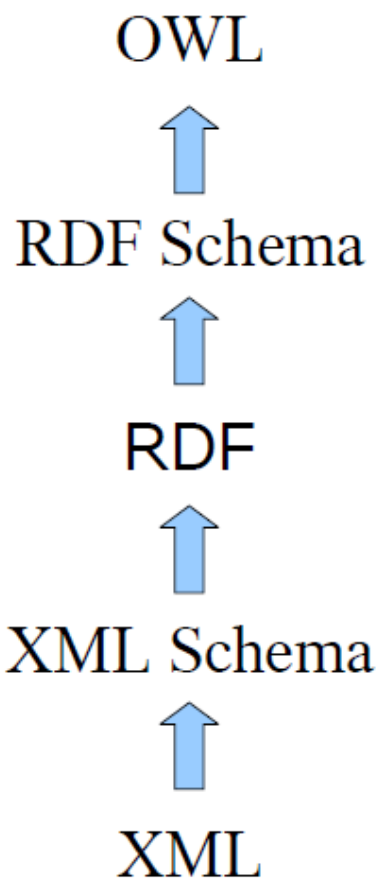
Ej. 2


Luis tiene un perro llamado Fido
Enfermedad
Is-a
Animal
Rabia
Is-a
Mamífero
Puede contraer
Is-a
Is-a
has
Perro
Persona
Es instancia de
Es instancia de



LENGUAJES DE ONTOLOGÍAS

Lenguaje de Ontologías



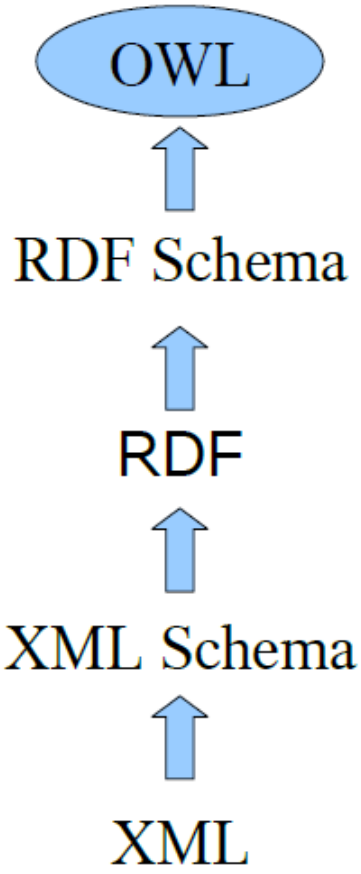
Ontology Web Language

Resource Description Framework

Extensible Markup Language

http://www.matem.unam.mx/~rajsbaum/cursos/web/ontologias2.pdf

Lenguaje de Ontologías

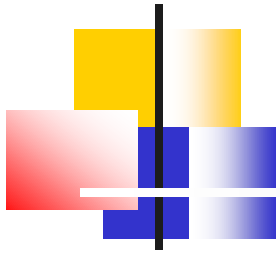


Ontology Web Language

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

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

Base Lógica de Ontologías: Lógicas descriptivas (DL)



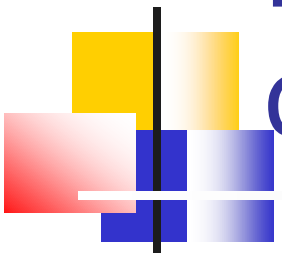
- Descripciones de conceptos usadas para describir un dominio;

  +

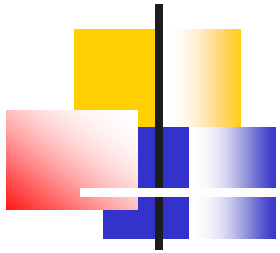
- La semántica que establece una equivalencia entre las fórmulas de DL y expresiones en lógica de predicados de primer orden.

http://es.wikipedia.org/wiki/Lógica_de_descripción

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: axiomas terminológicos que introducen descripciones complejas y propiedades de la *terminología descriptiva*.
- Un formalismo asertivo: que introduce propiedades de individuos.
- Son capaces de inferir nuevo conocimiento a partir del conocimiento dado. Tienen algoritmos de razonamiento que son *decidibles*.

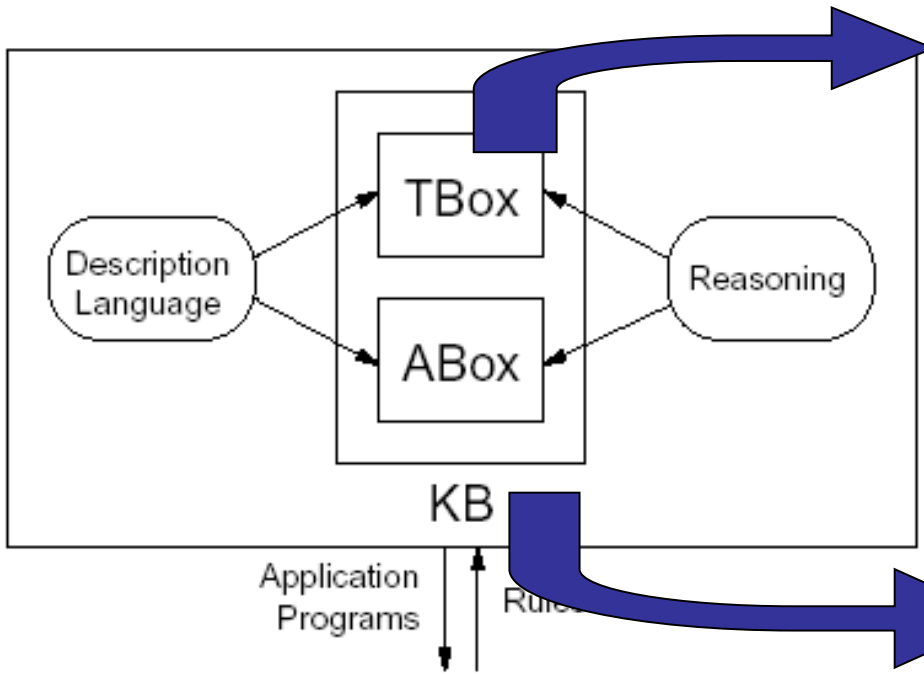


DL: Elementos Básicos

- Conceptos (Clases)
  - E.g., Person, Doctor, HappyParent, (Doctor ⊔ Lawyer)
- Roles (Propiedades – Slots)
  - E.g., hasChild, loves
- Individuos (Instancias - Instances)
  - E.g., John, Mary, Italy
- Operators (para formar conceptos y roles):
  - Computables (decidable) y si es posible, de baja complejidad

Lógica Descriptiva

Familia



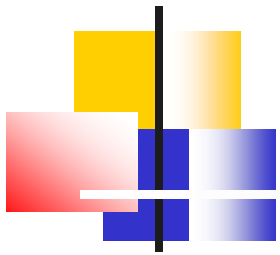
$$
\begin{aligned}
\text{Woman} &\equiv \text{Person} \sqcap \text{Female} \\
\text{Man} &\equiv \text{Person} \sqcap \neg\text{Woman} \\
\text{Mother} &\equiv \text{Woman} \sqcap \exists \text{hasChild}.\text{Person} \\
\text{Father} &\equiv \text{Man} \sqcap \exists \text{hasChild}.\text{Person} \\
\text{Parent} &\equiv \text{Father} \sqcup \text{Mother} \\
\text{Grandmother} &\equiv \text{Mother} \sqcap \exists \text{hasChild}.\text{Parent} \\
\text{MotherWithManyChildren} &\equiv \text{Mother} \sqcap \geqslant 3\,\text{hasChild} \\
\text{MotherWithoutDaughter} &\equiv \text{Mother} \sqcap \forall \text{hasChild}.\neg\text{Woman} \\
\text{Wife} &\equiv \text{Woman} \sqcap \exists \text{hasHusband}.\text{Man}
\end{aligned}
$$

Male

$$
\begin{aligned}
&\text{MotherWithoutDaughter}(\text{MARY}) \qquad \text{Father}(\text{PETER}) \\
&\text{hasChild}(\text{MARY}, \text{PETER}) \qquad \text{hasChild}(\text{PETER}, \text{HARRY}) \\
&\text{hasChild}(\text{MARY}, \text{PAUL})
\end{aligned}
$$

TBox (caja terminológica) contiene sentencias describiendo conceptos generales

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



DL KB

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

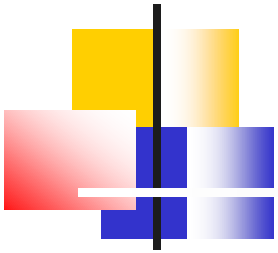
$$\{\text{Doctor} \sqsubseteq \text{Person},\ \text{HappyParent} \equiv \text{Person} \sqcap \forall \text{hasChild}.(\text{Doctor} \sqcup \exists \text{hasChild}.\text{Doctor})\}$$

  - 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: 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

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

DL: Lenguaje básico AL

- Sintaxis:

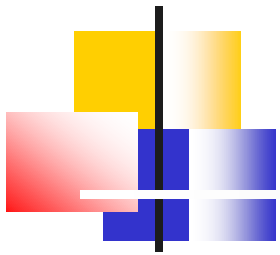
$$
\begin{array}{lll}
C, D & \longrightarrow & A \mid \\
 & & \top \mid \\
 & & \bot \mid \\
 & & \neg A \mid \\
 & & C \sqcap D \mid \\
 & & \forall R.C \mid \\
 & & \exists R.\top
\end{array}
$$

-concepto atómico
-concepto universal
-concepto bottom
-negación atómica
-intersección
-restricción de valor
-cuantificación existencial limitada

-Ejemplos:

$\text{Person} \sqcap \neg \text{Female}$	Personas no femeninas
$\text{Person} \sqcap \exists \text{hasChild}.\top$	Personas que tienen hijos
$\text{Person} \sqcap \forall \text{hasChild}.\bot$	Personas que no tienen hijos
$\text{Person} \sqcap \forall \text{hasChild}.\text{Female}$	Personas que tienen solo hijas

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., Parent (Father ⊔ Mother)

DL: Ejemplo

Ejemplo: Padre Feliz

$$\begin{aligned} &\texttt{Man} \sqcap (\exists \texttt{has-child.Blue}) \sqcap \\ &\qquad (\exists \texttt{has-child.Green}) \sqcap \\ &\qquad (\forall \texttt{has-child.Happy} \sqcup \texttt{Rich}) \end{aligned}$$

Class/Concept Constructors OWL

Constructor	DL Syntax	Example	FOL Syntax
intersectionOf	$C_1 \sqcap \ldots \sqcap C_n$	Human $\sqcap$ Male	$C_1(x) \wedge \ldots \wedge C_n(x)$
unionOf	$C_1 \sqcup \ldots \sqcup C_n$	Doctor $\sqcup$ Lawyer	$C_1(x) \vee \ldots \vee C_n(x)$
complementOf	$\neg C$	$\neg$Male	$\neg C(x)$
oneOf	$\{x_1\} \sqcup \ldots \sqcup \{x_n\}$	{john} $\sqcup$ {mary}	$x = x_1 \vee \ldots \vee x = x_n$
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	$\leqslant nP$	$\leqslant$1hasChild	$\exists^{\leqslant n} y.P(x,y)$
minCardinality	$\geqslant nP$	$\geqslant$2hasChild	$\exists^{\geqslant n} y.P(x,y)$

- C is a concept (class); P is a role (property); $\mathrm{x_i}$ 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

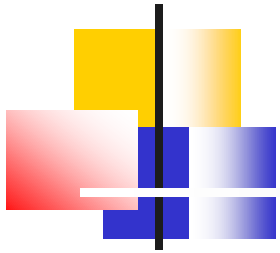
Person ⊓ ∀hasChild.(Doctor ⊔ ∃hasChild.Doctor)

```
<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>
```

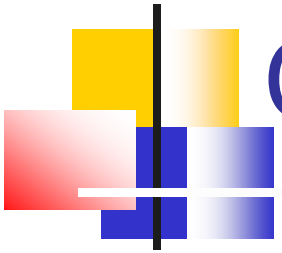
Dónde se usan las Ontologías?

- Semantic Web
- e-Science
  - Bioinformatics: Protein Ontology (MGED)data
- Medicine
  - Terminologies
- Databases
  - Integration
  - Query answering

- User interfaces
- Linguistics
- IoT



INGENIERÍA DE ONTOLOGÍAS

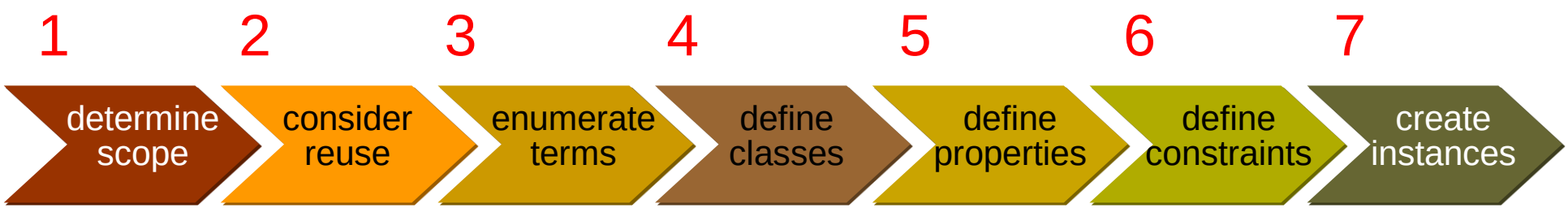


Qué es la "Ingeniería de Ontología"?

- Ingeniería de Ontología: Representa el conocimiento de un dominio utilizando ontologías
  - 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



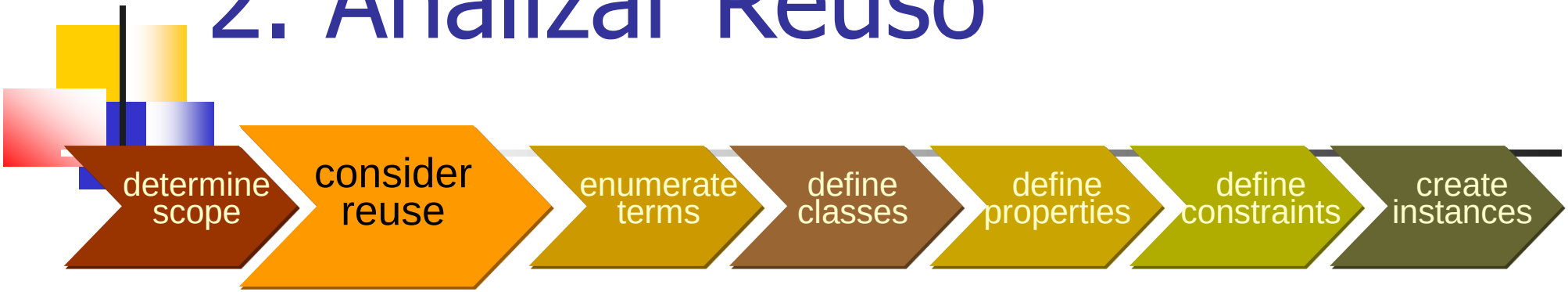
1. Determinar Dominio y Alcance

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

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?**

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?

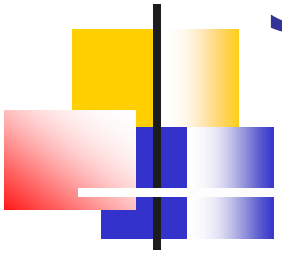
- Ontology libraries
  - DAML ontology library (www.daml.org/ontologies)
  - Ontolingua ontology library
  (www.ksl.stanford.edu/software/ontolingua/)

- Recursos semánticos - Tesauros

3. Enumerar Términos Importantes

- **De qué *términos* necesitamos hablar?**
- **Cuáles 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 Malbec que 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
  - Ej. *Alma Mora blanco* es un vino blanco que es un vino
  - Rex es un perro que es un vertebrado

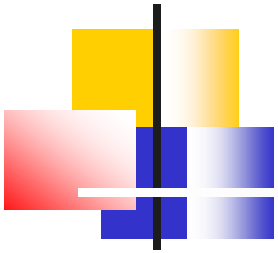
5. Definición de Propiedades de Clases: Slots

- Los slots describen los *atributos* de una instancia de una clase y la relación con otras instancias
  - *Por Ej. Cada vino tiene un color, contenido de azúcar…*
  - *Cada Animal tiene una forma de reproducción, un hábitad, pelaje o piel, etc…*

5. Propiedades (Slots)

- Tipos
  - intrínsicas: flavor y color de wine
  - extrínsicas: name y price de wine
  - Relaciones con otros objetos: producer de wine (bodega)
- Simples y complejas
  - simples (attributes): contienen valores (strings, numbers) → Data properties
  - complejas: apuntan a otros objetos (ej., una instancia de bodega) → Object properties

Slot y Herencia

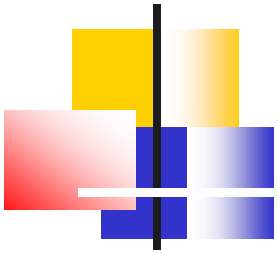


- Una subclase hereda todos los slots de una superclase
  - *Si un wine tiene name y flavor, un red wine también tiene name y flavor*
- Si una clase tiene múltiples superclases, hereda slots de todas ellas
  - *Oporto es un dessert wine y un red wine: hereda "sugar content: high" de dessert wine y "color:red"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

6. Restricciones de Propiedades

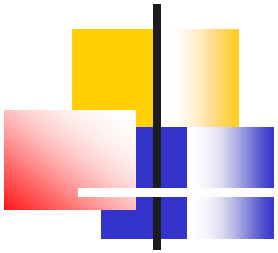
- Describen los posibles valores del slot
  - *El name de un wine es un string*
  - *El wine producer es una instancia de Winery (Bodega)*
  - *Una winery tiene una sola localización*

6. Common Facets: Restricciones



- Slot cardinality – el número de valores que tiene el slot
- Slot value type – el tipo de valores
- Minimum and maximum value – el rango de valores para un slot numérico
- Default value – el valor que tiene un slot a menos que se defina explícitamente otro

6. Common Facets: Value Type

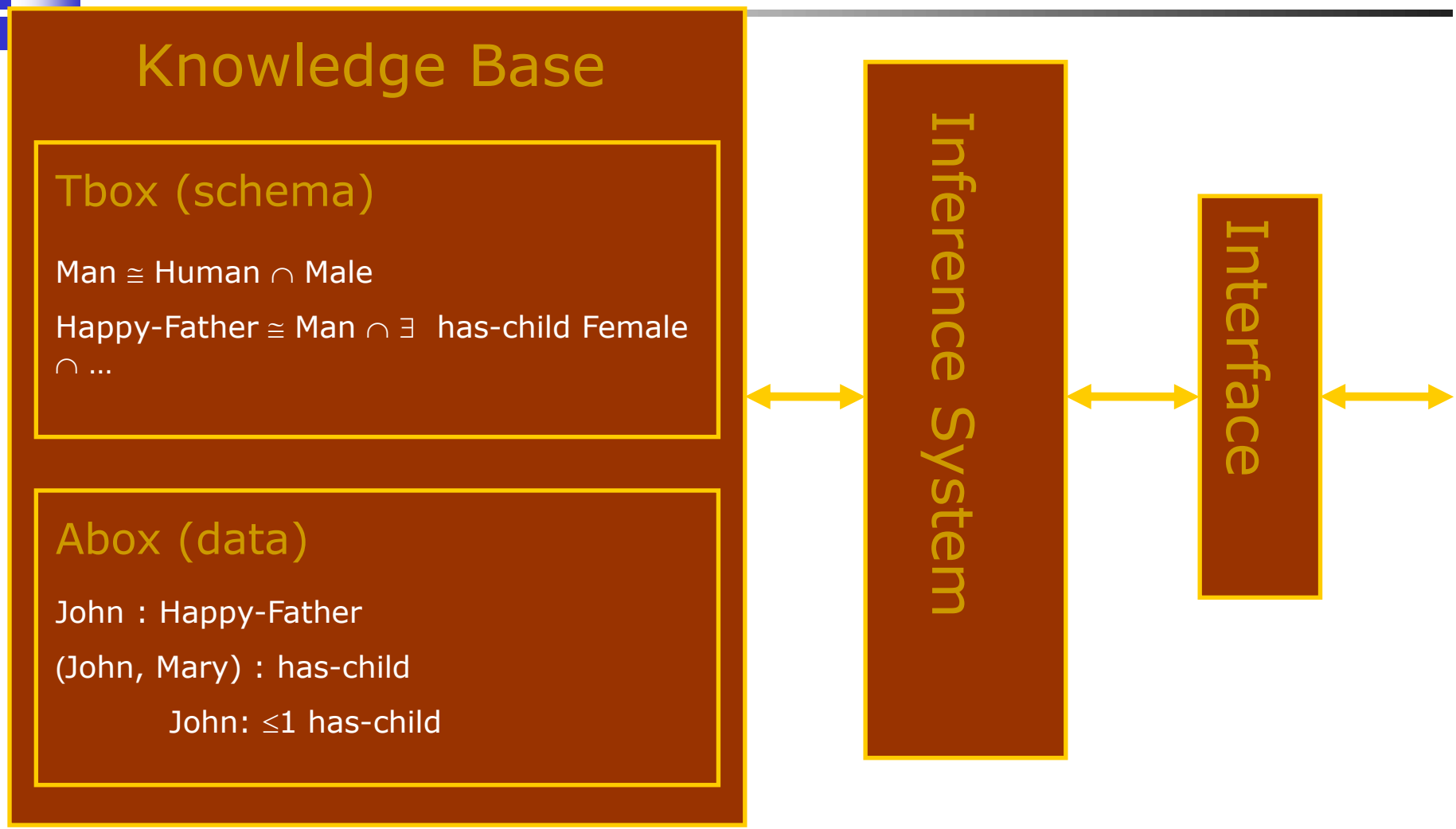


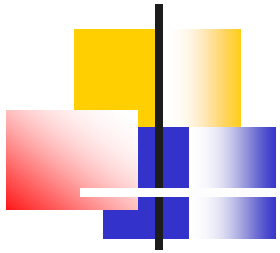
- String: string de caracteres (“Alma Mora Malbec”)
- Number: integer o float
- Boolean: true/false
- Enumerated type: una lista de valores permitidos (high, medium, low)
- Complex type: una instancia de otra clase
  - *In the Wine class the value type for the slot “maker” is at the Winery class*

7. Create Instances

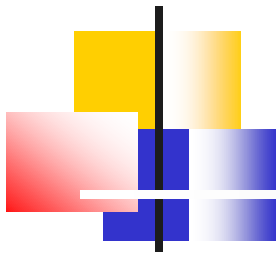
- Crear instancias de las distintas clases
- Asignar valores a los slots de cada instancia
  - Slot values deben respetar las restricciones

KB System Architecture





RAZONAR CON ONTOLOGÍAS



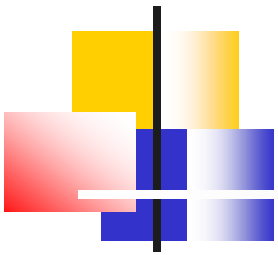
Razonar con Ontologías

Es necesario contar con herramientas que ayuden:

- Diseñar y mantener ontologías de alta calidad, e.g.:
  - **Significativas** — todas las clases tienen instancias
  - **Correctas**— sin inconsistencias y que capturan intuiciones del dominio
  - **Mínimamente redundantes** — no hay sinónimos no intencionales
  - **Axiomatizada ricamente** — con suficientes descripciones detalladas
- Answer queries sobre clases e instancias de ontology e.g.:
  - Encontrar clases más generales/específicas
  - Recuperar individuos/tuples matching una query
- Integrar y alinear múltiples ontologías

Necesitamos razonamientos correctos

- Las inferencias más interesantes/útiles son aquellas que fueron inesperadas
  - Es probable que se ignore/descarte si se sabe que el razonador no es confiable
- Muchas aplicaciones web serán agente ↔ agente sin intervención humana para detectar fallas en el razonamiento
  - Se necesita tener un alto nivel de confianza en el razonador

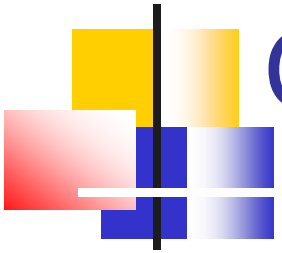


Necesitamos razonamientos decidibles

- OWL constructores/axiomas han sido restringidos de modo que el razonamiento sea decidible
- Consistente con la arquitectura en capas de la Semantic Web:
  - XML: una capa sintáctica de intercambio
  - RDF(S): un lenguaje relacional básico y primitivas ontológicas simples
  - OWL: un lenguaje ontológico más poderoso pero aún decidible
- W3C requiere para las implementaciones
  - “Algoritmos Prácticos” para razonamientos correctos y completos.

Cómo podemos razonar? Inferencias de DL

- **Validación de la consistencia de una ontología:** el razonador puede comprobar si hay 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.
- **Resolución de consultas:** 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.



Cómo razonamos?

Querying KB

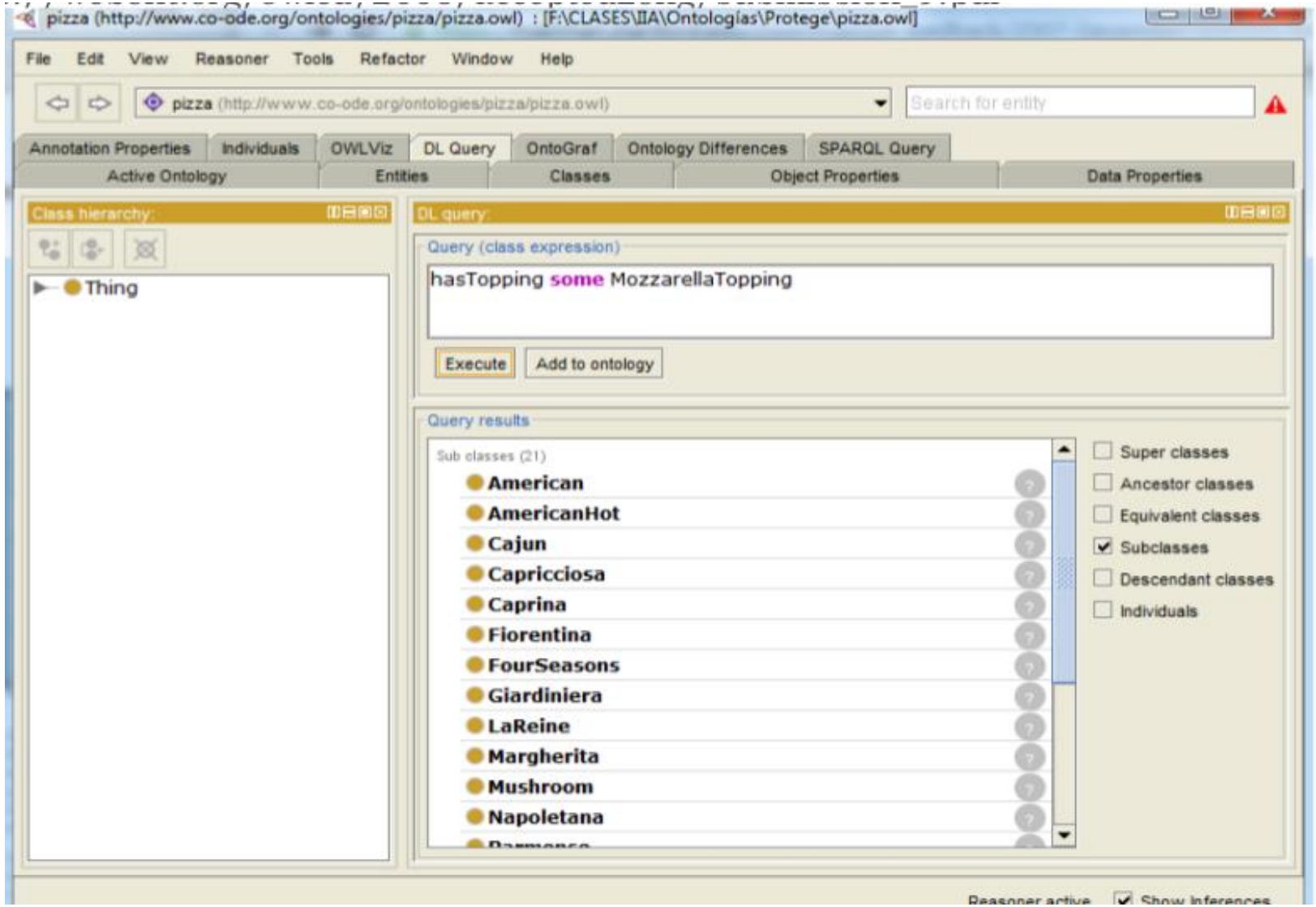
- Es x una instancia de C w.r.t. O?
  - Es Rex un perro? Es Rex un mamífero?

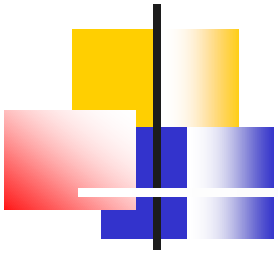
- Es [x,y] una instancia of R w.r.t. O?
  - Es Rex la mascota de Pedro?

- Estas consultas y las validaciones pueden resolverse usando DL reasoners

DL Query:Pizza Ontology

http://webont.org/owled/2006/acceptedLong/submission_9.pdf

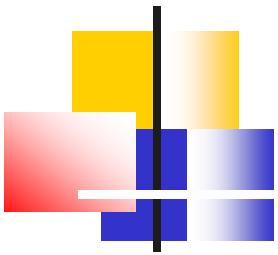




DL Queries: ejemplos

- hasTopping some MozzarellaTopping
- hasTopping only (MozzarellaTopping or PeperoniSausageTopping or TomatoTopping)
- hasTopping some PeperoniSausageTopping
- hasTopping some TomatoTopping
- not (hasTopping some FishTopping)

Ontologías: resumen



✓ Una Ontología es un artefacto de modelado conceptual e ingenieril que consiste en:

- Un vocabulario de términos
- Una especificación explícita de su significado

✓ Las Ontologías están jugando un rol fundamental en muchas aplicaciones de e-Science, IoT, Databases, Semantic Web, etc.

✓ El razonamiento es importante porque:

- permite realizar chequeos de consistencia
- detectar relaciones que estaban implícitas
- realizar queries

…. La investigación continúa!!! (aumentar el poder expresivo, tratar con ontologías muy grandes, otros razonamientos, etc…)