Ontologías

Representación de Conocimiento

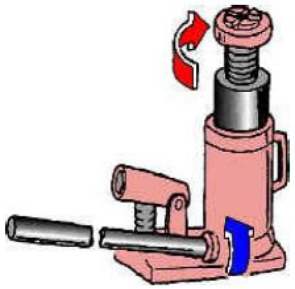
Based on: Ian Horrocks <horrocks@cs.man.ac.uk>
University of Manchester
Manchester, UK

Lenguajes de ontologías

- RDF
- DAML
- OIL
- OWL

Lenguajes de ontologías

- Con DTDs (Document Type Definition) de XML y XML Schemas se puede intercambiar información y definiciones, pero un mismo término puede significar distintas cosas dependiendo del contexto.



Lenguaje de Ontologías: RDF

RDF (Resource Description Framework) es un marco para metadatos en la WWW. Posee:

- Clases y propiedades
- Sub/super-clases (y propiedades)
- Rango y dominio (de propiedades)

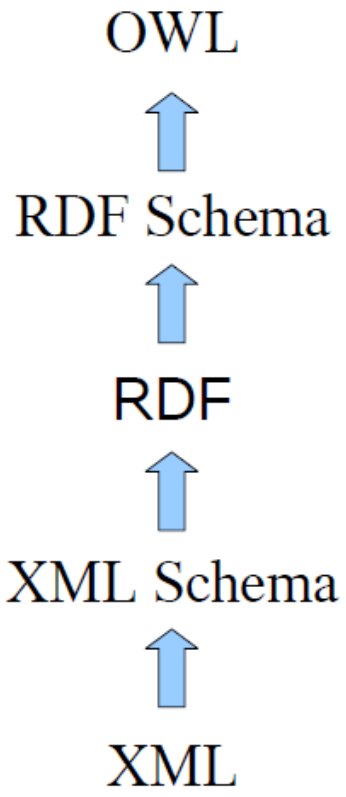
Pero RDF es demasiado débil para describir los recursos con suficientes detalles, e.g:

- No tiene restricciones de dominio y rango localizado.
- No tiene restricciones de cardinalidad/existencia.
- No tiene propiedades simétricas o inversas ni transitivas.

Y RDF tiene semánticas no estándares

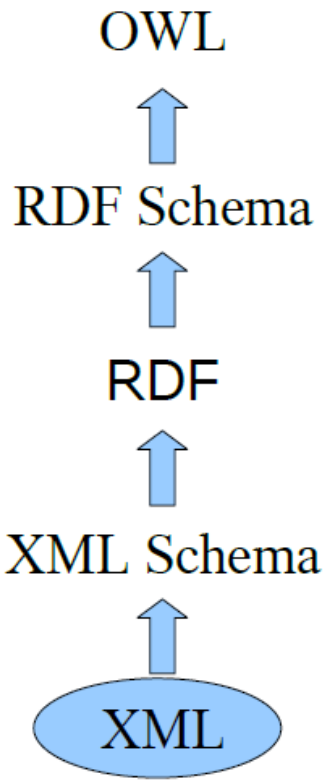
- Que dificultan soportar el razonamiento

Lenguaje de Ontologías



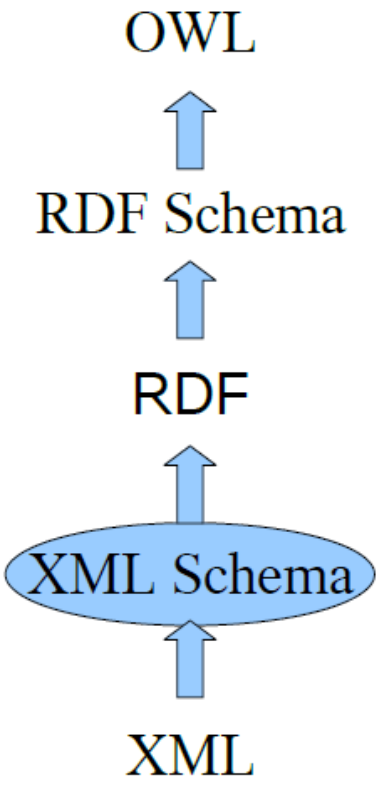
http://www.matem.unam.mx/~rajsbaum/cursos/web/ontologias2.pdf

Lenguaje de Ontologías



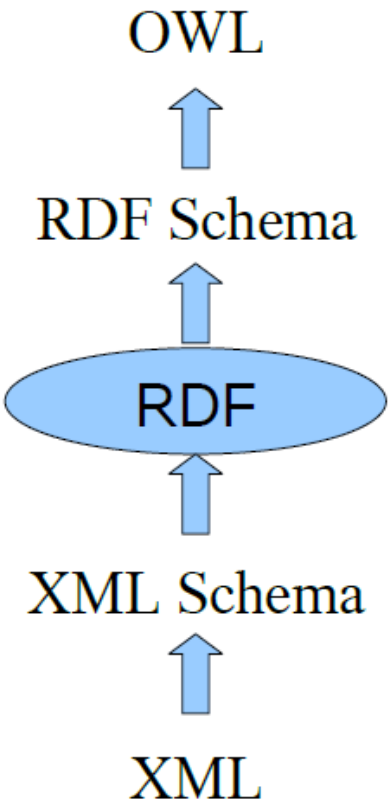
Es una sintaxis superficial para documentos semi-estructurados. Sin embargo, no proporciona información semántica

Lenguaje de Ontologías



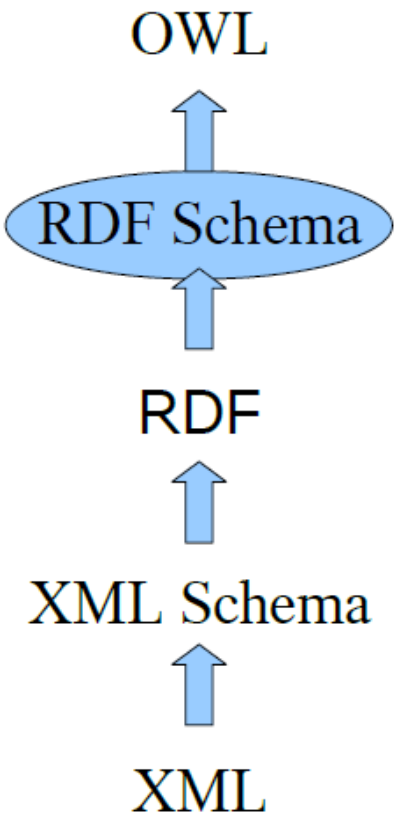
Lenguaje que restringe la estructura de XML. Además, le proporciona la capacidad de manejar tipos de datos

Lenguaje de Ontologías



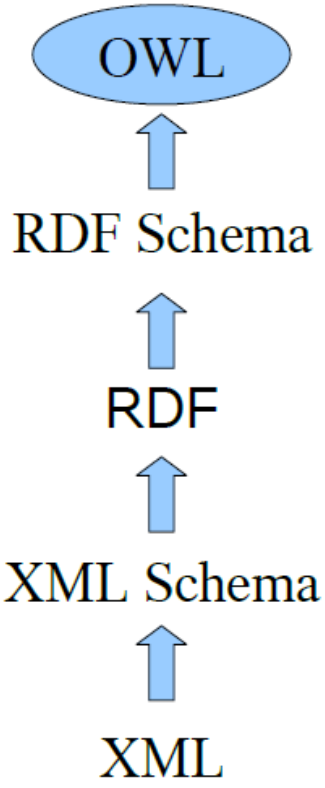
Modelo de datos para objetos (recursos) y las relaciones entre ellos. Ya tiene la capacidad de expresar cierta semántica

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

Lenguaje de Ontologías: OWL

- Puede representar el significado de los **términos** en vocabularios y las **relaciones** entre ellos (ontología)
- Construido sobre la habilidad de **XML** de definir esquemas de etiquetado y el uso de **RDF** para representar-describir datos.

Representación en owl – ver tipos de datos

```
<owl:Class rdf:ID="Mapa">
   <rdfs:subClassOf>
        <owl:Restriction>
                <owl:onProperty rdf:resource="#tieneEscalaPredeterminada"/>
                <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger"> 1
                </owl:maxCardinality>
        </owl:Restriction>
   </rdfs:subClassOf>
</owl:Class>

<owl:ObjectProperty rdf:ID="tieneEscala">
   <rdfs:domain rdf:resource="#Mapa"/>
   <rdfs:range rdf:resource="#Escala"/>

</owl:ObjectProperty> <owl:ObjectProperty rdf:ID="tieneEscalaPredeterminada">
   <rdfs:subPropertyOf rdf:resource="#tieneEscala"/>
</owl:ObjectProperty>
```

Elementos de Ontología OWL

OWL

- Individuos
- Propiedades
- Clases

Protege

- Casos (instance)
- Slots
- Classes

protégé

Lenguaje de Ontologías: OWL

Existen 3 versiones de OWL

- **OWL Lite** permite creación de jerarquías y restricciones simples.
  - Por ej. Sólo permite cardinalidad 0 y 1.
- **OWL DL** proporciona la máxima capacidad de expresión garantizando computabilidad (tiempo finito)
  - Por ej. Una clase puede ser subclase de muchas, pero no puede ser un caso de otra clase.
  - Tiene un modelo de semántico estándar (primer orden)
- **OWL full** es el máximo nivel de expresión union de la sintaxis OWL y RDF
  - La semántica RDF extendida con condiciones semánticas relevantes y axiomáticas sujeto(recurso)-predicado(propiedad)-objeto(valor).

Razonar – inferir: cómo avanzar.

- Qué es razonar?
  Para qué?
- Qué necesitamos?
- Cuáles el alcance?

Razonamiento

Make implicit information explicit

Deductive logical inference

- Start with one or more statements to reach a logically certain conclusion
- **IF** x and y are true **THEN** z is true

Use declarative definitions of the domain (Schema - Ontology)

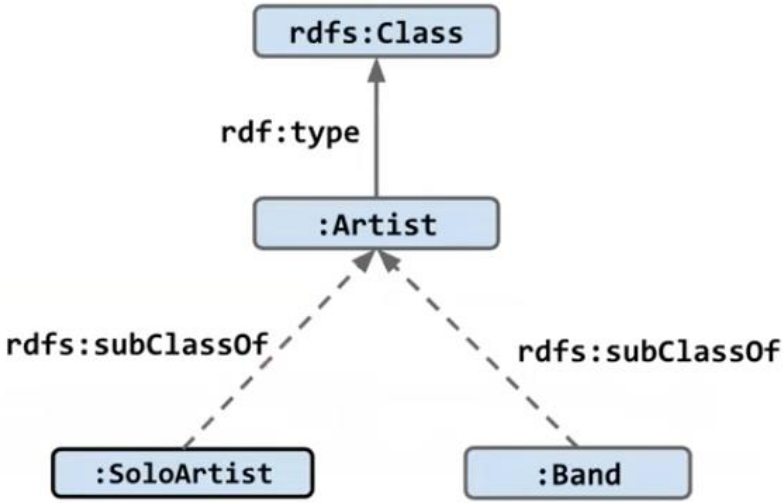
Ver: [Training] Reasoning with RDF Graphs and Ontologies
https://www.youtube.com/watch?v=p9ZE4tzQn70

Classes

Represents a set of instances with common characteristics

Organized in a hierarchy

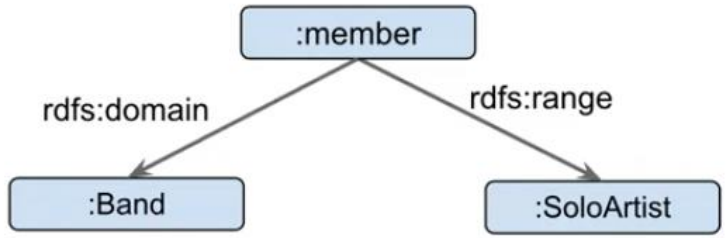
Instances can be classified automatically under the hierarchy based on their properties



Dominio y rango

For the triples where the property is used as the predicate

- Domain is the type of the subject
- Range is the type of the object

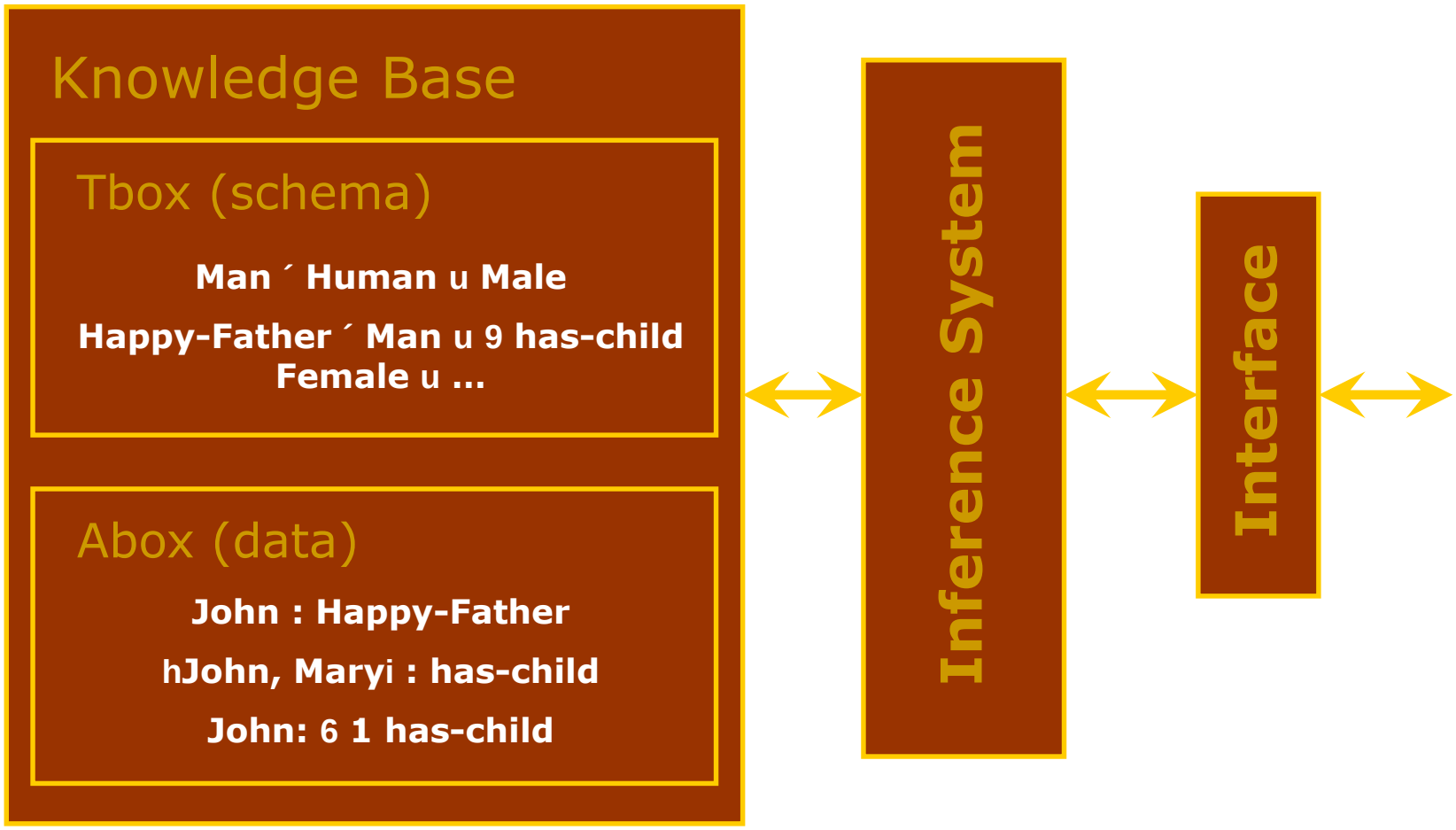


Domain and range declaration is not for validating the graph!

Problems with RDFS

- RDFS too weak to describe resources in sufficient detail
  - No localised range and domain constraints
    - Can't say that the range of hasChild is person when applied to persons and elephant when applied to elephants
  - No existence/cardinality constraints
    - Can't say that all *instances* of person have a mother that is also a person, or that persons have exactly 2 parents
  - No transitive, inverse or symmetrical properties
    - Can't say that isPartOf is a transitive property, that hasPart is the inverse of isPartOf or that touches is symmetrical
  - …
- Difficult to provide reasoning support

Volvamos a ver: DL Architecture



Class/Concept Constructors

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); x is an individual name
- XMLS datatypes as well as classes in 8P.C and 9P.C
  - Restricted form of DL concrete domains

Ontology/Tbox 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

- Obvious FO/Modal Logic equivalences
  - E.g., DL: $\mathrm{C} \sqsubseteq \mathrm{D}$ FOL: $\forall \mathrm{x}.\mathrm{C}(\mathrm{x}) \rightarrow \mathrm{D}(\mathrm{x})$ ML: $\mathrm{C} \rightarrow \mathrm{D}$
- Often distinguish two kinds of Tbox axioms
  - "Definitions" $\mathrm{C} \sqsubseteq \mathrm{D}$ or $\mathrm{C} \equiv \mathrm{D}$ where C is a concept name
  - General Concept Inclusion axioms (GCIs) where C may be complex

Web Ontology Language (OWL)

Expressive ontology language that builds on top of RDFS

Formal semantics based on Description Logics

More strict rules about definitions

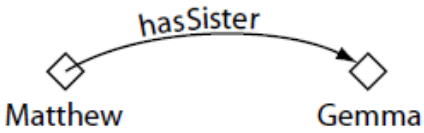
Properties

OWL Properties

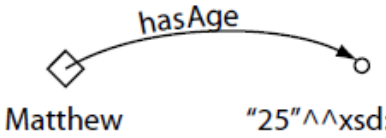
Represent relationships between two **individuals**.

Two main types:

- Object properties, link an individual to an individual;
- Datatype properties, link an individual to an XML Schema Datatype value[4]

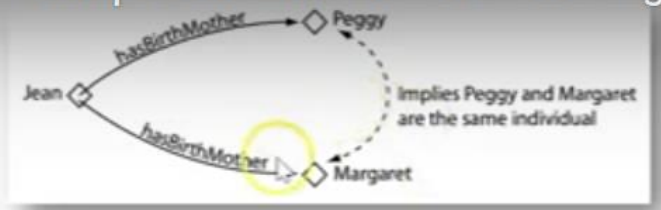


An object property linking the individual Matthew to the individual Gemma

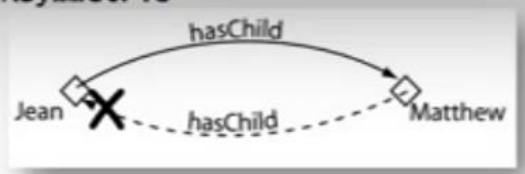


A datatype property linking the individual Matthew to the data literal '25', which has a type of an xml:integer.

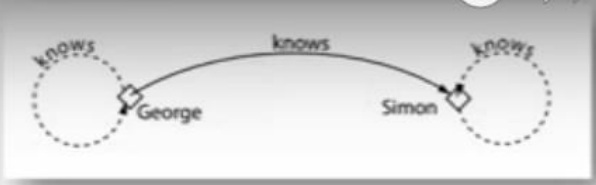
A simple tutorial on OWL Ontologies using Protege - Part 2



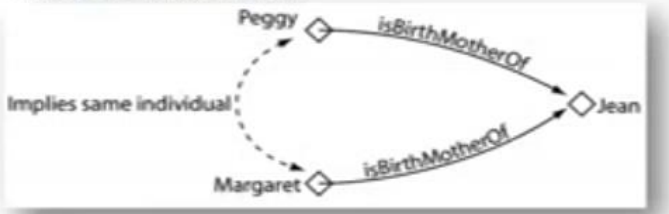
Asymmetric



Reflexive



Inverse Functional



Transitive

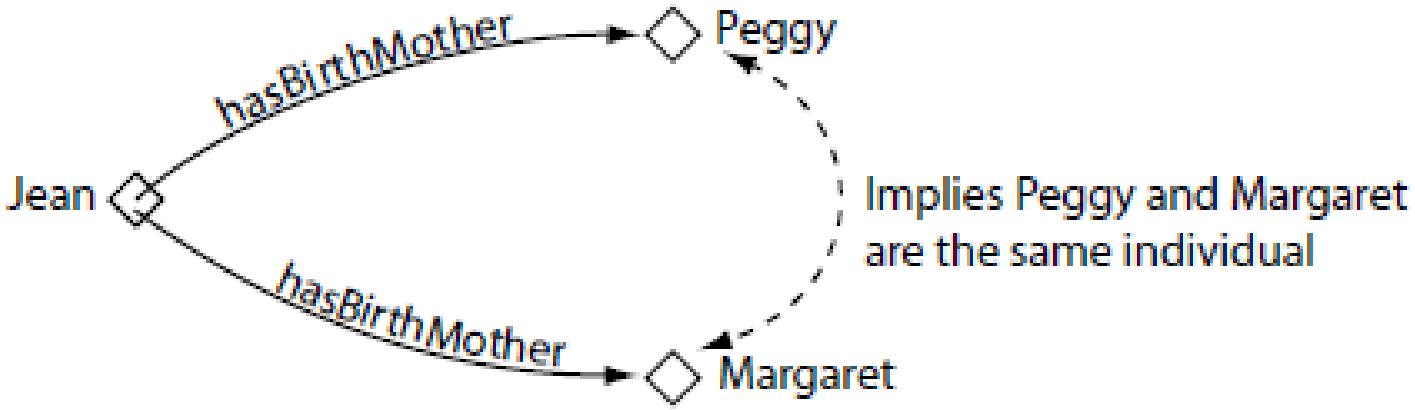
Irreflexive

Symmetric

Types of Properties

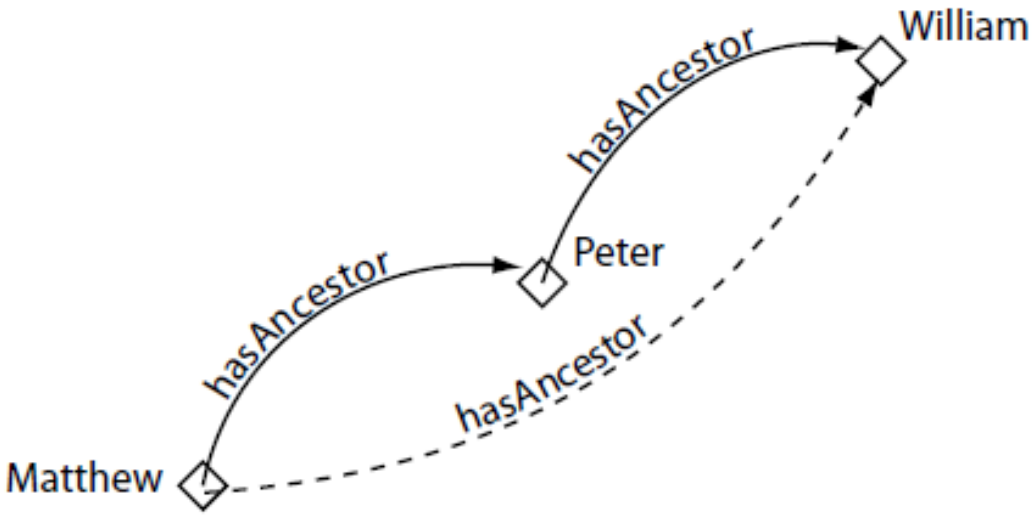
Functional Property

- If a property is functional, for a given individual, there can be at most one individual that is related to the individual via the property.

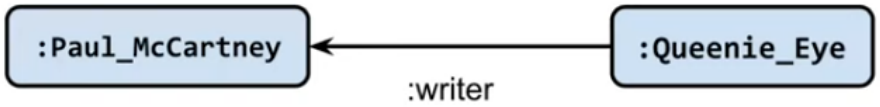


Transitive Properties

- If a property is transitive, and the property relates individual **a** to individual **b**, and also individual **b** to individual **c**, then we can infer that individual **a** is related to individual **c** via property P.



Inverse Properties



Properties are directional

`writer` direction is from Song to Songwriter

Infer `writerOf` property in reverse direction

```
:writerOf owl:inverseOf :writer
```

Inverse Properties

Properties are directional

`writer` direction is from Song to Songwriter

Infer `writerOf` property in reverse direction

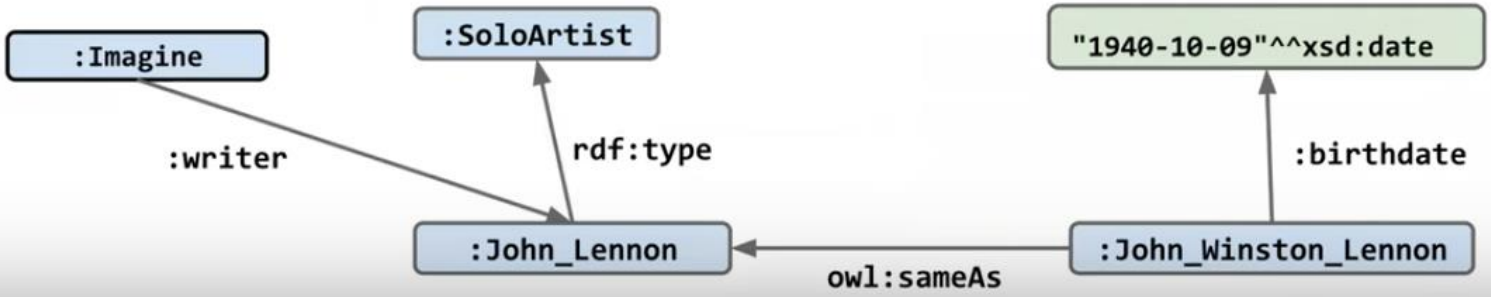
```
:writerOf owl:inverseOf :writer
```

Owl:sameAs reasoning

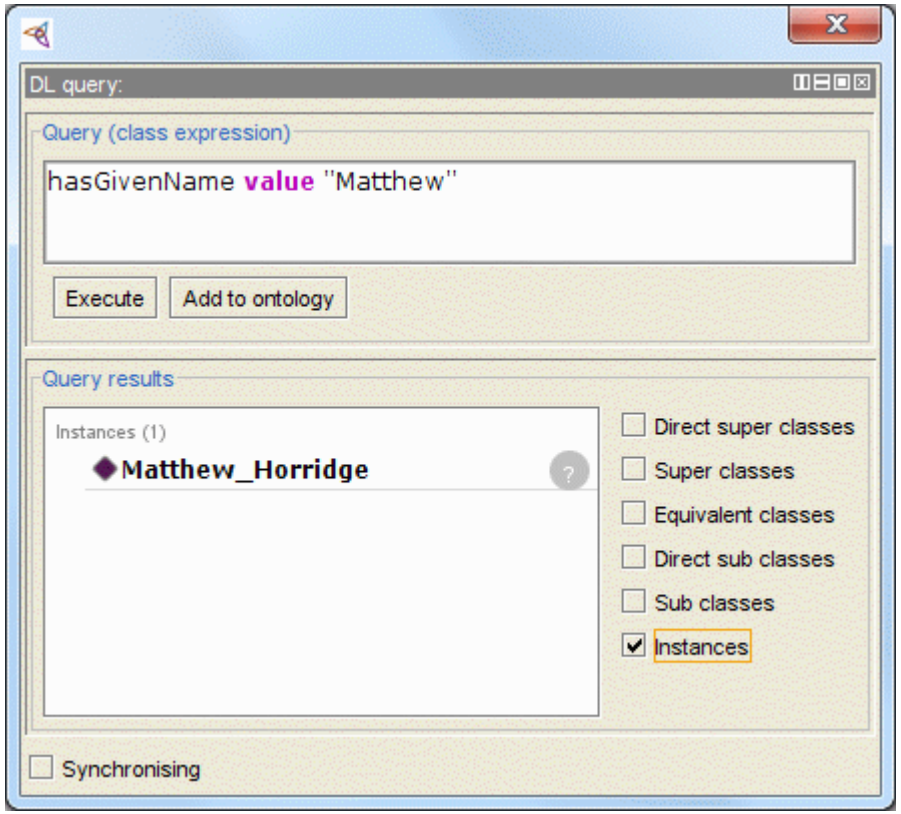
Declare two nodes are representing the same entity

Node will be merged at the logical level

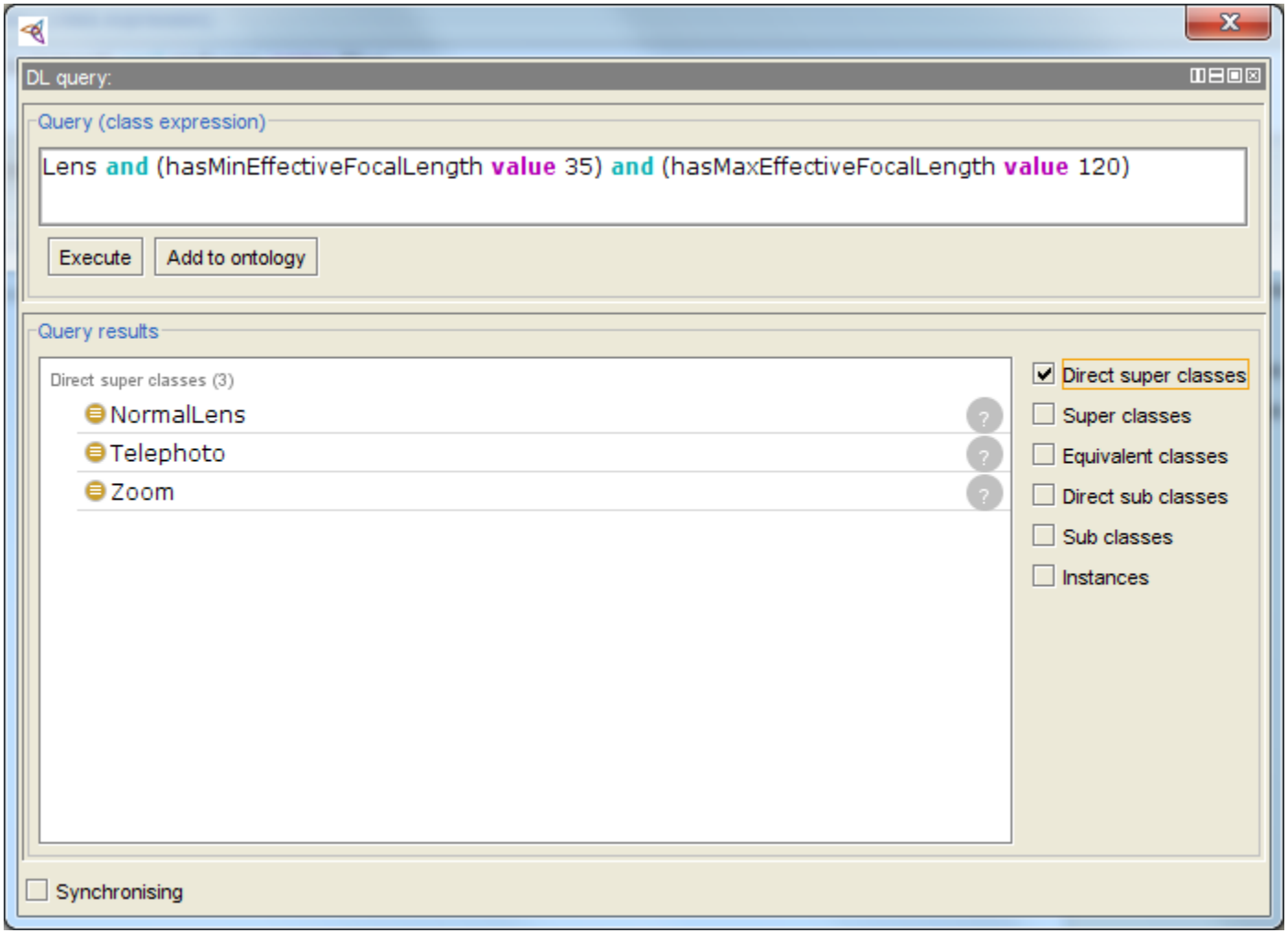
sameAs inferences are reflexive, symmetric and transitive



DL Query


DL query:
Query (class expression)
hasGivenName value "Matthew"
Execute
Add to ontology
Query results
Instances (1)
Matthew_Horridge
Direct super classes
Super classes
Equivalent classes
Direct sub classes
Sub classes
Instances
Synchronising

DL Query


DL query:
Query (class expression)
Lens and (hasMinEffectiveFocalLength value 35) and (hasMaxEffectiveFocalLength value 120)
Execute
Add to ontology
Query results
Direct super classes (3)
NormalLens
Telephoto
Zoom
Direct super classes
Super classes
Equivalent classes
Direct sub classes
Sub classes
Instances
Synchronising

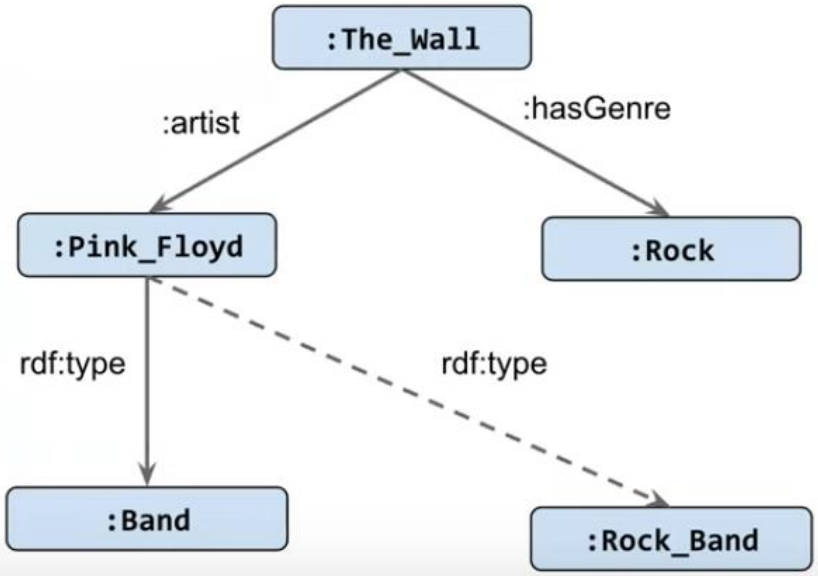
Reglas SWRL: una forma

Use graph patterns in IF/THEN rules

```
IF {
    <graph pattern>
    <filter>
    <bind>
}
THEN {
    <graph pattern>
}
```

SWRL

```
IF {
    ?album :hasGenre :Rock ;
           :artist ?band .
    ?band a :Band .
}
THEN {
    ?band a :RockBand
}
```



Lenguajes

Table 2

Reasoning Characteristics

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

Summary 1

- DLs are family of object oriented KR formalisms related to frames and Semantic networks
  - Distinguished by formal semantics and inference services
- OWL is a DL based ontology language designed for the Web
  - Exploits existing standards: XML, RDF(S)
  - Adds KR idioms from object oriented and frame systems
  - DL provides formal foundations and reasoning support

Summary 2

- Reasoning is important because
  - Understanding is closely related to reasoning
  - Essential for design, maintenance and deployment of ontologies
- Reasoning support based on DL systems
  - Sound and complete reasoning
  - Highly optimised implementations
- Challenges remain
  - Reasoning with full OWL language
  - (Convincing) demonstration(s) of scalability
  - New reasoning tasks
  - Development of (more) high quality tools and infrastructure