Ontologías

Representación de Conocimiento

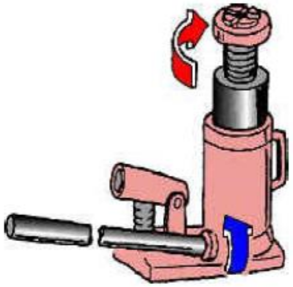
Based on: Ian Horrocks <horrocks@cs.man.ac.uk>
University of Manchester
Manchester, UK

Lenguajes de ontologías

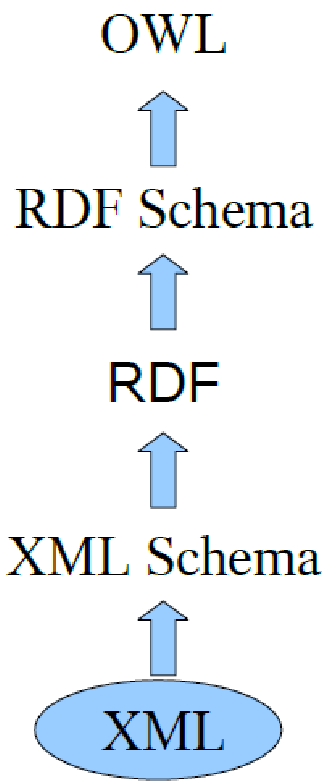
- RDF
- DAML
- OIL
- OWL

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.

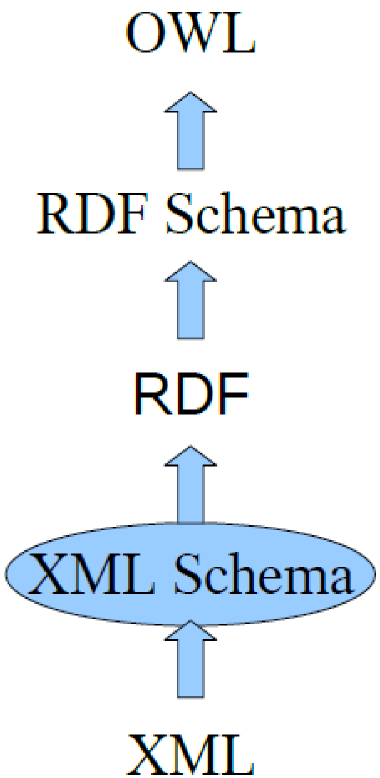


Lenguaje de Ontologías



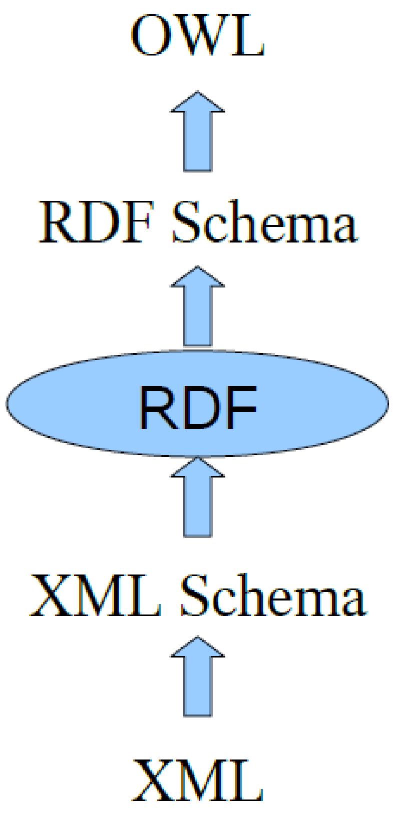
Es una sintaxis superficial
para documentos semi-estructurados.
Sin embargo, no
proporciona información
semántica

Lenguaje de Ontologías



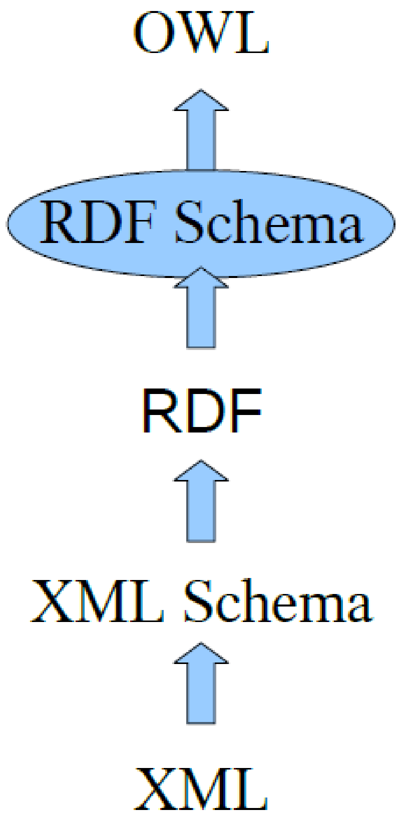
Lenguaje que restringe la estructura de XML. Además, le proporciona la capacidad de manejar tipos de datos

Lenguaje de Ontologías



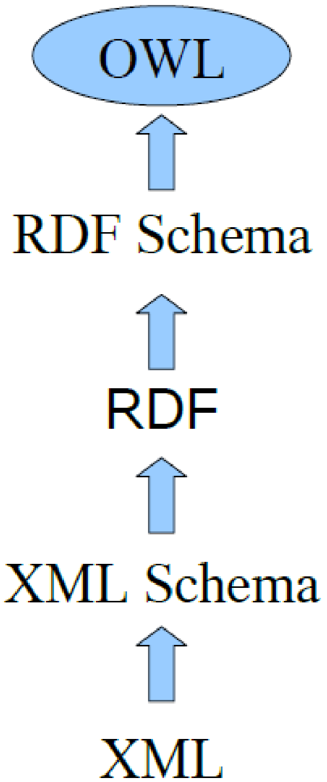
Modelo de datos para objetos (recursos) y las relaciones entre ellos. Ya tiene la capacidad de expresar cierta semántica

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

Elementos de Ontología OWL

OWL

- Individuos
- Propiedades
- Clases

Protege

- Casos (instance)
- Slots
- Classes

Lógicas descriptivas (DL) – OWL-DL

- Formalismos basados en lógica de primer órden para **Representación de Conocimiento**.
  - Describen al **dominio** en función de **conceptos** (classes), **roles** (relationships) e **individuos**.
  - Describen a la **semántica** que establece equivalencias entre *fórmulas lógicas de descripción* (*lógicas de predicados*).
    - Concepts (**formulae**)
      - E.g., Person, Doctor, HappyParent, (Doctor **t** Lawyer)
    - Roles (**modalities**)
      - E.g., hasChild, loves
    - Individuals (**nominals**)
      - E.g., John, Mary, Italy
    - Operators (para formar conceptos y roles):
      - Computables y si es posible, de baja complejidad

Representación en OWL

```
<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>
```

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

OWL Axioms

Axiom	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
disjointWith	$C_1 \sqsubseteq \neg C_2$	Male $\sqsubseteq \neg$Female
sameIndividualAs	$\{x_1\} \equiv \{x_2\}$	{President_Bush} $\equiv$ {G_W_Bush}
differentFrom	$\{x_1\} \sqsubseteq \neg\{x_2\}$	{john} $\sqsubseteq \neg${peter}
subPropertyOf	$P_1 \sqsubseteq P_2$	hasDaughter $\sqsubseteq$ hasChild
equivalentProperty	$P_1 \equiv P_2$	cost $\equiv$ price
inverseOf	$P_1 \equiv P_2^-$	hasChild $\equiv$ hasParent$^-$
transitiveProperty	$P^+ \sqsubseteq P$	ancestor$^+$ $\sqsubseteq$ ancestor
functionalProperty	$\top \sqsubseteq \leqslant 1P$	$\top \sqsubseteq \leqslant$1hasMother
inverseFunctionalProperty	$\top \sqsubseteq \leqslant 1P^-$	$\top \sqsubseteq \leqslant$1hasSSN$^-$

OWL Class 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	$\exists^{\geqslant n} y.P(x,y)$

Razonar – inferir: cómo avanzar.

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

Porque…

- Applications such as the Semantic Web aim at "machine understanding"
- Understanding is closely related to reasoning
  - Recognising semantic similarity in spite of syntactic differences
  - Recognising implicit consequences given explicitly stated facts

Razones Prácticas

- Las Ontologías son útiles como tools y services si brindan a los usuarios:
  - Design and maintain high quality ontologies, e.g.:
    - Meaningful — all named classes can have instances
    - Correct — captured intuitions of domain experts
    - Minimally redundant — no unintended synonyms
    - Richly axiomatised — (sufficiently) detailed descriptions
  - Answer queries over ontology classes and instances, e.g.:
    - Find more general/specific classes
    - Retrieve individuals/tuples matching a given query
  - Integrate and align multiple ontologies

Razonamiento computable

- OWL constructors/axioms have been restricted so reasoning is decidable
- Consistent with Semantic Web's layered architecture
  - XML provides syntax transport layer
  - RDF(S) provides basic relational language and simple ontological primitives
  - OWL provides powerful but still decidable ontology language
  - Further layers (e.g. SWRL) will extend OWL
    - Will almost certainly be undecidable
- W3C requirement for “implementation experience”
  - “Practical” algorithms for sound and complete reasoning
  - Several implemented systems
  - Evidence of empirical tractability

Los razonadores DL: servicios de inferencia

- **Validación de la consistencia** de una ontología: comprobar si hay hechos contradictorios
- **Validación del cumplimiento de los conceptos**: determinar 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**: computar 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**: inferir cuáles son las clases a las que directamente pertenece (generalizadoras). Pertenencia de clases o individuos dentro de la ontología.

Cómo?

DL Reasoning: Basics

- Tableau algorithms used to test satisfiability (consistency)
- Try to build a tree-like model of the input concept C
- Decompose C syntactically
  - Apply tableau expansion rules
  - Infer constraints on elements of model
- Tableau rules correspond to constructors in logic ($\sqcap$, $\sqcup$ etc)
  - Some rules are nondeterministic (e.g., $\sqcup$, $\leqslant$)
  - In practice, this means search
- Stop when no more rules applicable or clash occurs
  - Clash is an obvious contradiction, e.g., $\mathrm{A}(\mathrm{x})$, $\neg\mathrm{A}(\mathrm{x})$
- Cycle check (blocking) may be needed for termination
- C satisfiable iff rules can be applied such that a fully expanded clash free tree is constructed

- Un razonador *tableaux* posee sólo la funcionalidad de verificar consistencias de un ABox respecto a un TBox. .

DL Reasoning…

- Satisfiability w.r.t. an Ontology $\mathcal{O}$
  - For each axiom $C \sqsubseteq D \in \mathcal{O}$, add $\neg C \sqcup D$ to every node label
- More expressive DLs
  - Basic technique can be extended to deal with
    - Role inclusion axioms (role hierarchy)
    - Number restrictions
    - Inverse roles
    - Concrete domains/datatypes
    - Aboxes
    - etc.
  - Extend expansion rules and use more sophisticated blocking strategy
  - Forest instead of Tree (for Aboxes)
    - Root nodes correspond to individuals in Abox

Web Ontology Language (OWL)

Expressive ontology language that builds on top of RDFS

Formal semantics based on Description Logics

More strict rules about definitions

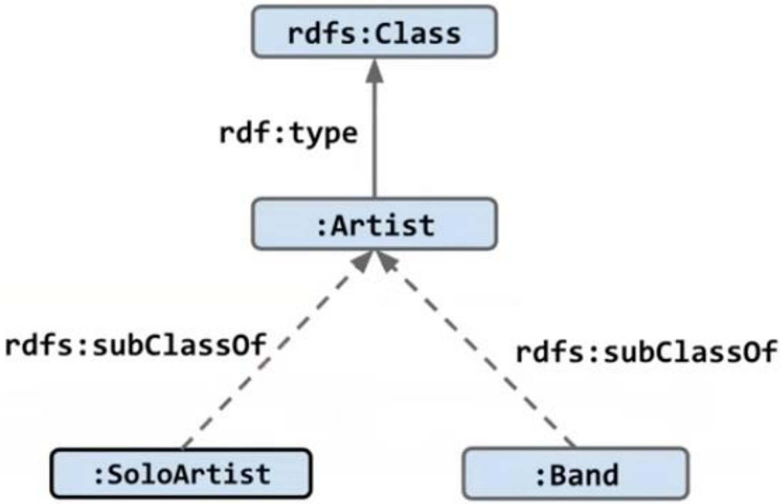
Properties/ROLES

Classes

Represents a set of instances with common characteristics

Organized in a hierarchy

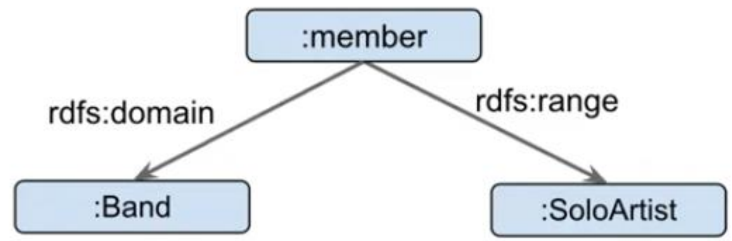
Instances can be classified automatically under the hierarchy based on their properties



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!

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)

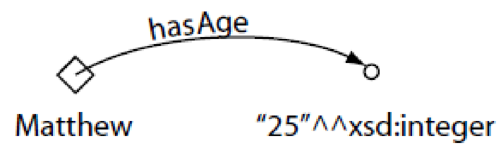
OWL Properties

Represent relationships between two **individuals**.

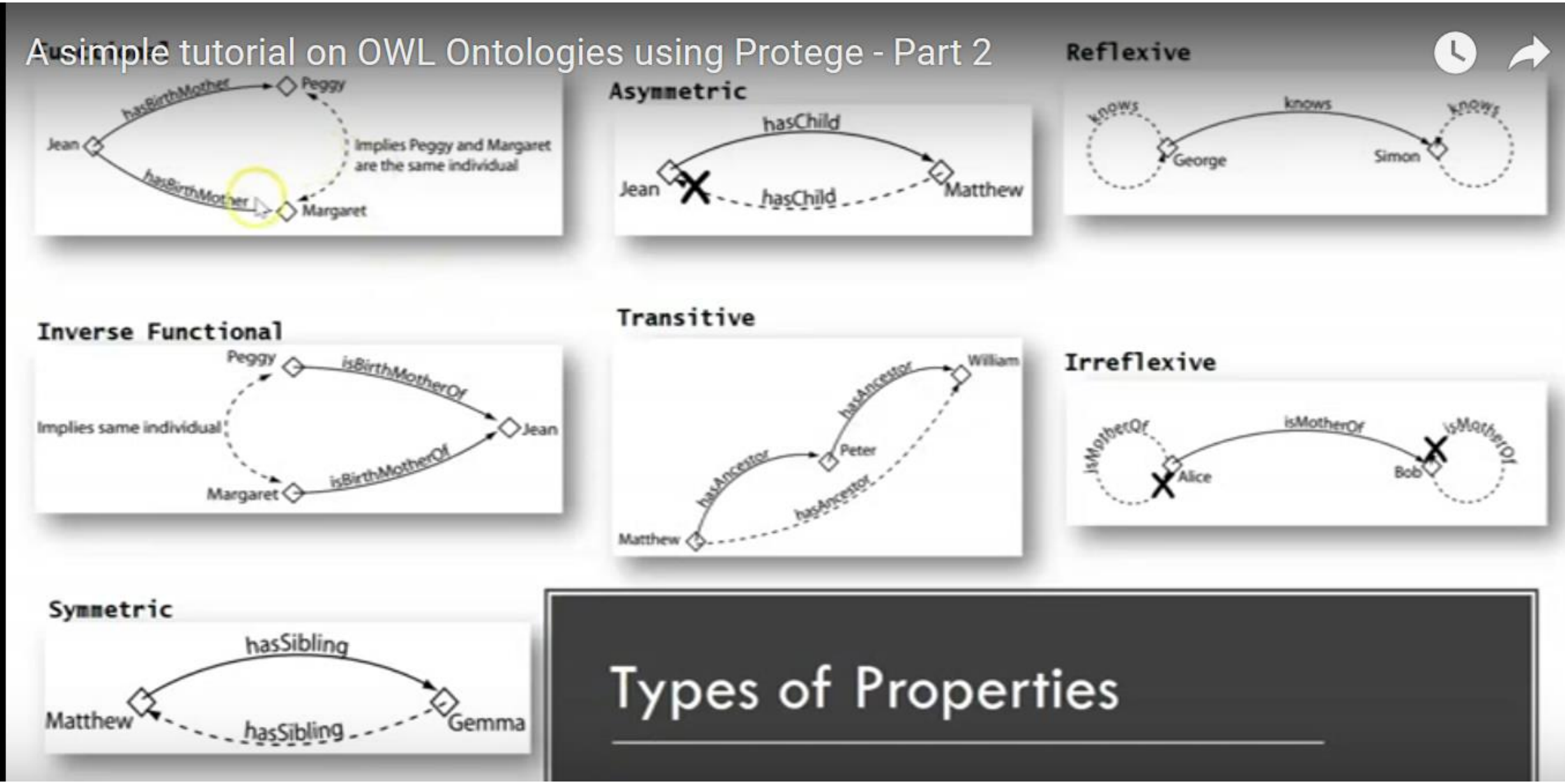
Two main types:

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

An object property linking the individual Matthew to the individual Gemma

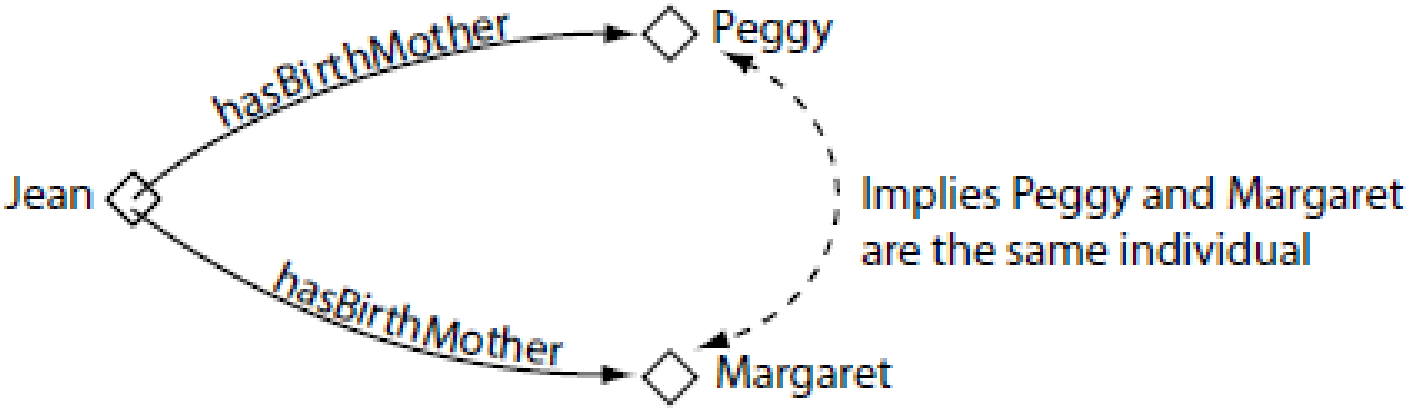


A datatype property linking the individual Matthew to the data literal '25', which has a type of an xml:integer.


A simple tutorial on OWL Ontologies using Protege - Part 2
hasBirthMother
Peggy
Jean
Implies Peggy and Margaret are the same individual
hasBirthMother
Margaret
Asymmetric
hasChild
Jean
hasChild
Matthew
Reflexive
knows
knows
knows
George
Simon
Inverse Functional
Peggy
isBirthMotherOf
Implies same individual
Jean
isBirthMotherOf
Margaret
Transitive
William
hasAncestor
Peter
hasAncestor
hasAncestor
Matthew
Irreflexive
isMotherOf
isMotherOf
isMotherOf
Alice
Bob
Symmetric
hasSibling
Matthew
hasSibling
Gemma
Types of Properties

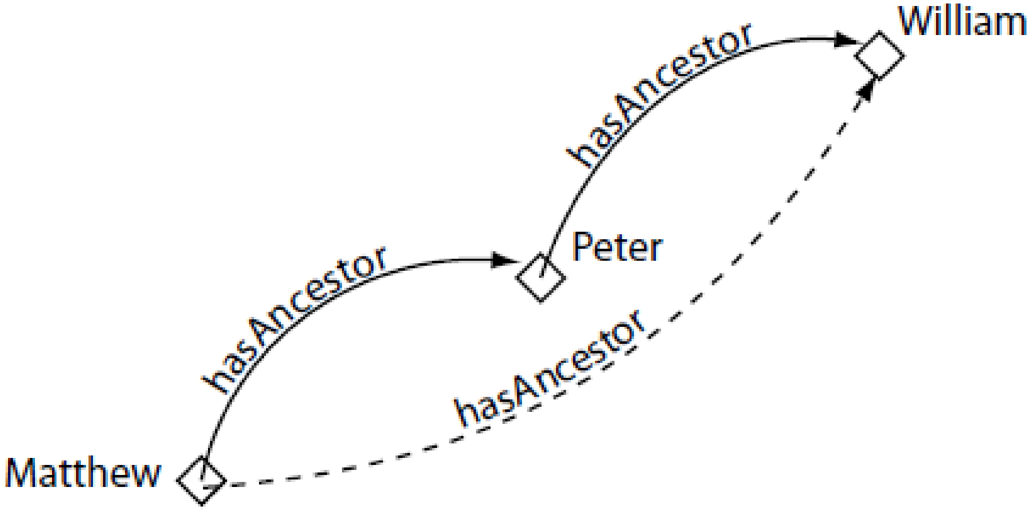
Functional Property

- 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.

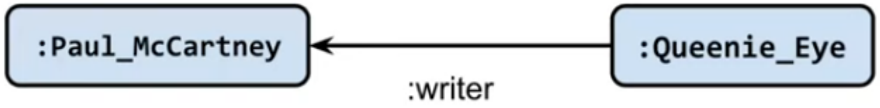


Transitive Properties

- 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



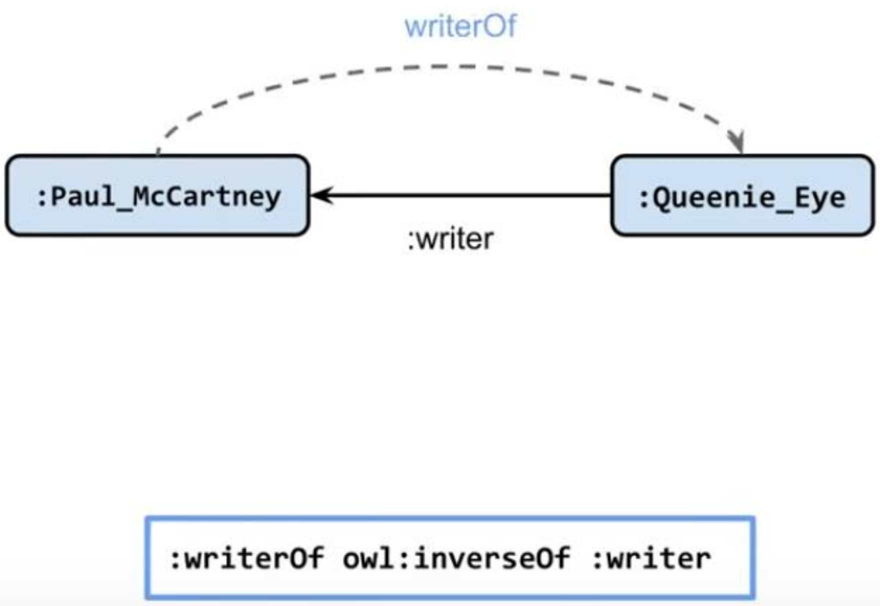

```
:writerOf owl:inverseOf :writer
```

Properties are directional

`writer` direction is from Song to Songwriter

Infer `writerOf` property in reverse direction

Inverse Properties



Properties are directional

`writer` direction is from Song to Songwriter

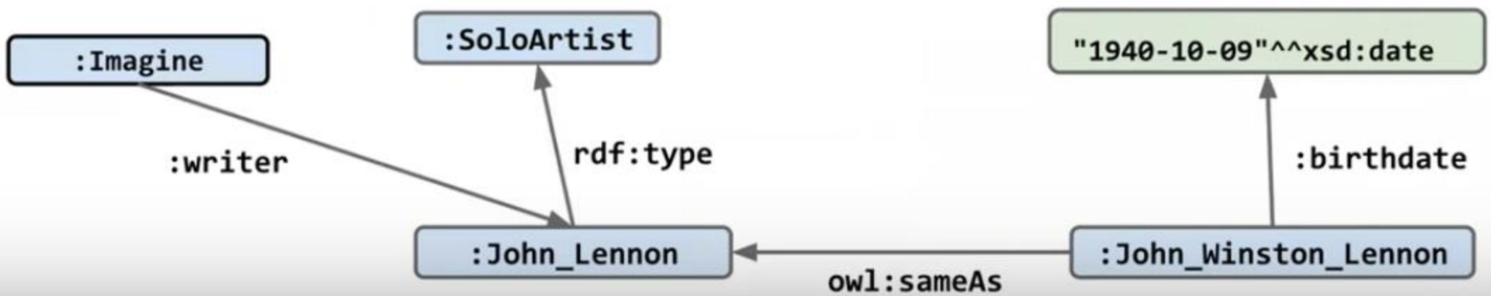
Infer `writerOf` property in reverse direction

Owl:sameAs reasoning

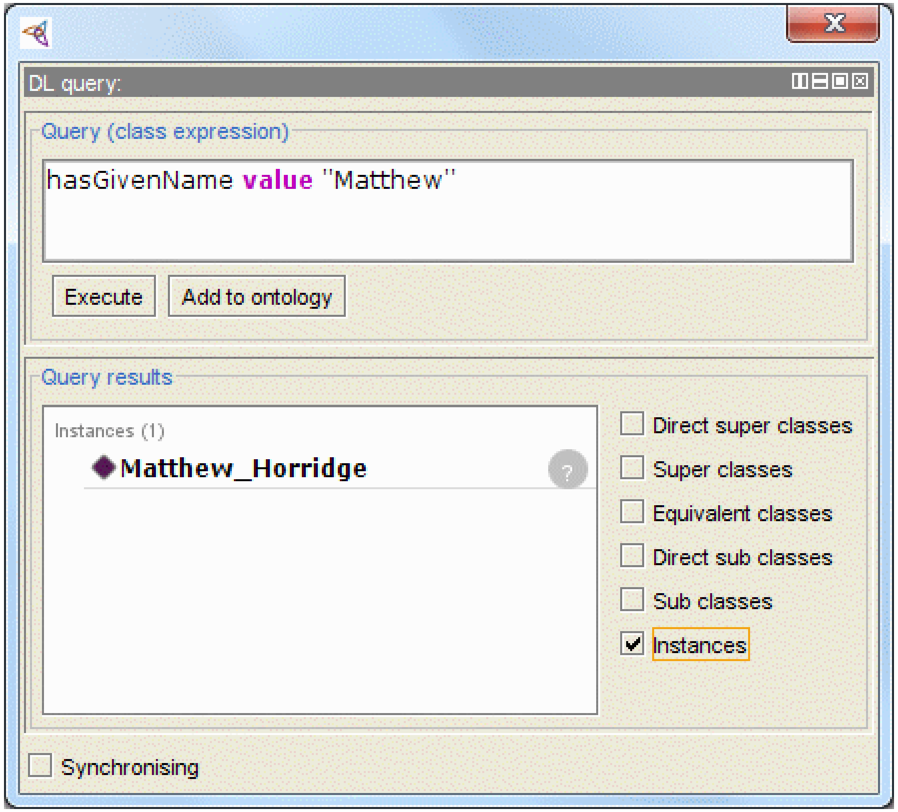
Declare two nodes are representing the same entity

Node will be merged at the logical level

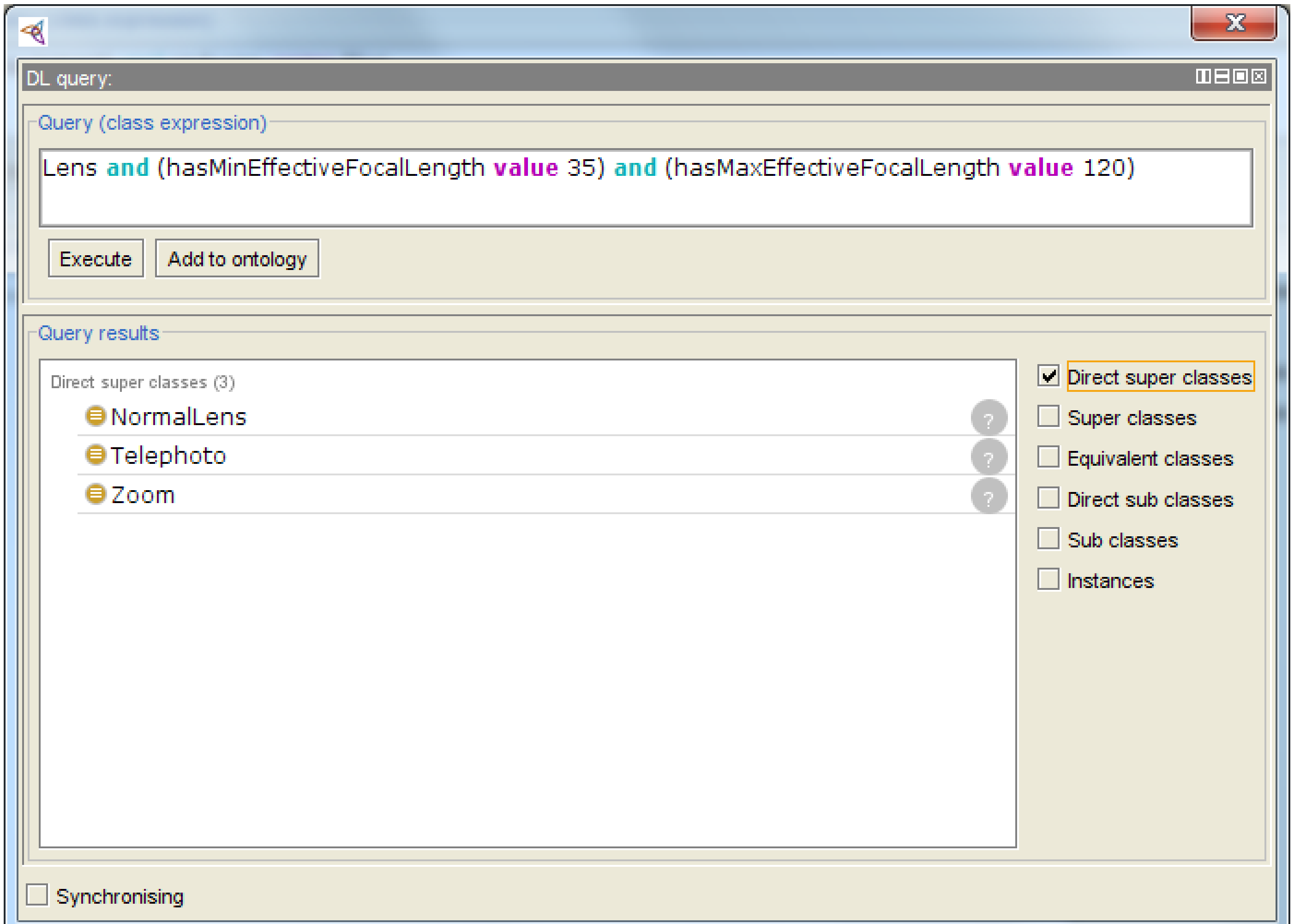
sameAs inferences are reflexive, symmetric and transitive



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

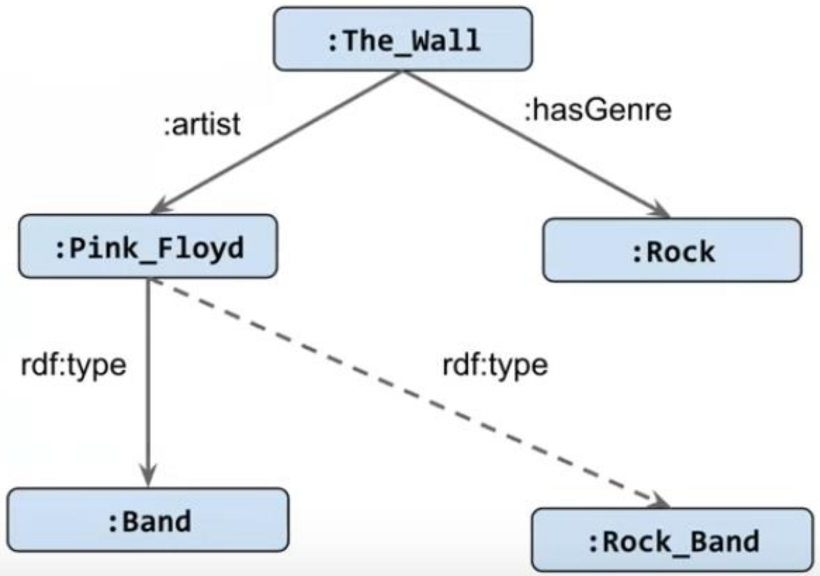
Reglas SWRL: una forma

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 $\mathcal{SHIF}$	$\mathcal{EL}^+$	$\mathcal{SROIQ}(\mathcal{D})$	$\mathcal{SROIQ}(\mathcal{D})$	$\mathcal{SROIQ}(\mathcal{D})$	$\mathcal{SHIQ}(\mathcal{D}-)$	$\mathcal{EL}^+$	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