



Introduction to ontologies

Amel Bouzeghoub
CSC5003



What is an ontology ?

- No exact definition
- A tool to help organize knowledge
- Or a way to convey a theory on how to represent a class of things
- The term **ontology** is originated from philosophy. In that context it is used as the name of a subfield of philosophy, namely, the study of the nature of existence.

Definition

- **Webster's Third New International Dictionary defines Ontology:**
 1. A science or study of being specifically, a branch of metaphysics relating to the nature and relations of being.
 2. A theory concerning the kinds of entities and specifically the kinds of abstract entities that are to be admitted to a language system.
- **An ontology in Computer Sciences**
 - « explicit formal specifications of the terms in the domain and relations among them » (Gruber 1993)
 - Semantic Relations
 - Composition and inheritance relationships

Definition (Hudelot 2006)

- An ontology is a **formal** and **explicit** specification of a **shared conceptualization** of a field of knowledge.
- **Conceptualization**: a certain view of the world in relation to a domain, often conceived as a set of concepts, their definition, their interrelationships.
- **Explicit**: explicit definition of the types of concepts used and constraints on their use.
- **Formal**: understandable by the machine.
- **Shared**: consensus, knowledge accepted by a group.

The Role of Ontology

■ Basic Role

- To provide a language which allows a group of people to share information reliably in a chosen area of work
- Reuse
 - Create and maintain reusable knowledge bases
 - Build knowledge bases from reusable modules
- Knowledge sharing and communication
 - Ensure interoperability between systems
 - Enable knowledge exchange between systems

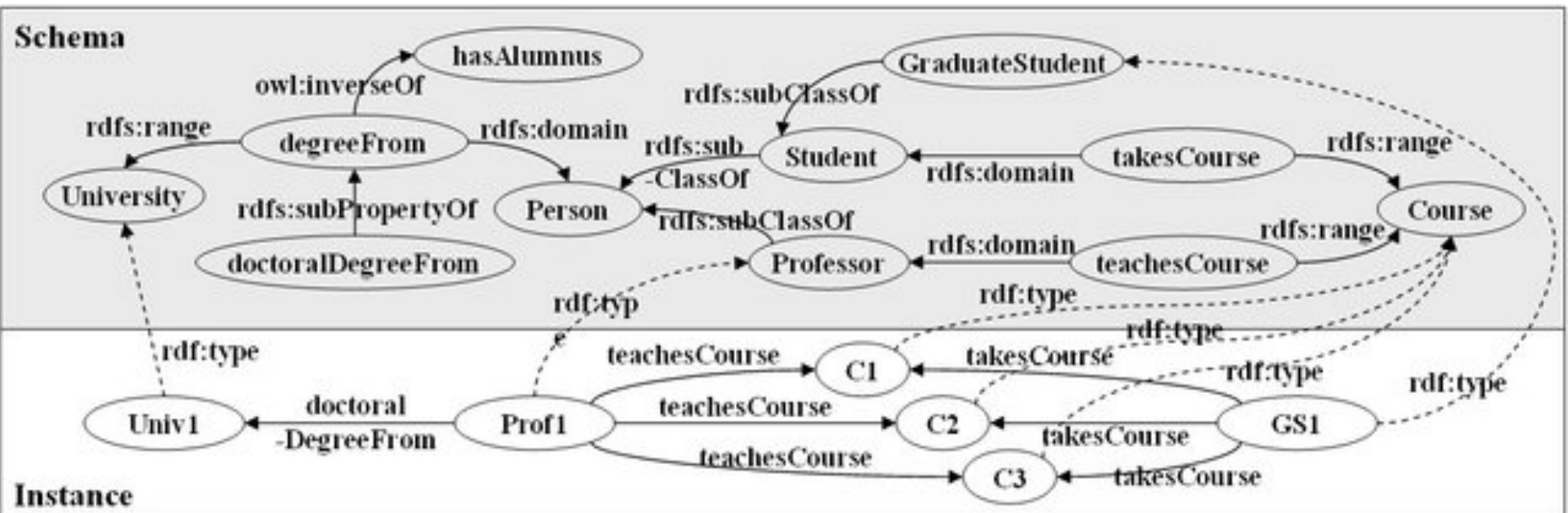
■ Some areas of application

- Indexing
- Knowledge Sharing & Reuse
- Artificial Intelligence (AI)
- Enterprise Modeling
- Software Design
- Molecular Biology
- eCommerce
- Semantic Web....

Why do I need to develop an ontology?

- **To provide more precise definition to resources and make them more suitable to machine processing**
- **To separate domain knowledge from operational knowledge**
 - Re-use domain and operational knowledge separately
- **A common reference to define concepts and relationships between concepts and other axioms in a domain**
- **To share a consistent understanding of a domain**
- **Reasoning / inference**
 - Building and discovering new information and/or knowledge from existing ontologies and resources

Example




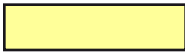
Definition: ontology structure

■ An ontology structure is a quintuplet $\mathbf{O} := \{C, \mathcal{R}, \mathcal{H}^C, \mathcal{R}el, \mathcal{A}^O\}$

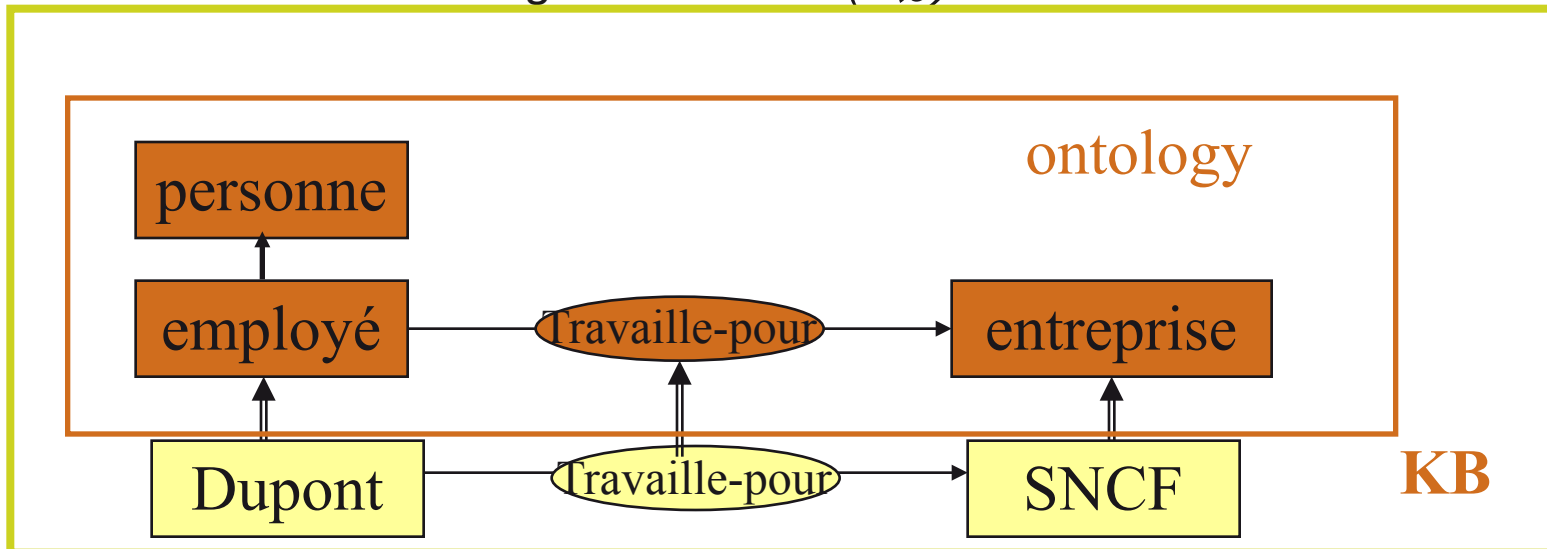
- C and \mathcal{R} : disjointed sets of concepts and relations
- \mathcal{H}^C **hierarchy (taxonomy) of concepts** : $\mathcal{H}^C \subseteq C \times C$, $\mathcal{H}^C(C_1, C_2)$ means that C_1 is a sub-concept of C_2 (oriented relation)
- $\mathcal{R}el$: **relation** $rel: \mathcal{R} \rightarrow C \times C$ (defines non-taxonomic semantic relations) with 2 associated functions
 - $dom: \mathcal{R} \rightarrow C$ with $dom(\mathcal{R}) := \Pi_1(rel(\mathcal{R}))$
 - $range: \mathcal{R} \rightarrow C$ with $range(\mathcal{R}) := \Pi_2(rel(\mathcal{R}))$ co-domaine
 - $rel(\mathcal{R}) = (C_1, C_2)$ or $\mathcal{R}(C_1, C_2)$
- \mathcal{A}^O : set of axioms, expressed in an adapted logical language (description logic, 1st order logic)

Definition: structure of a knowledge base

■ Structure of a knowledge base : quadruplet $\mathcal{KB} := \{O, I, inst, instr\}$

- $O := \{C, \mathcal{R}, \mathcal{H}^C, rel, \mathcal{A}^O\}$ is an ontology 
- I is a set of individual 
- $inst : C \rightarrow 2^I$ Concept instantiation
- $instr : \mathcal{R} \rightarrow 2^{I \times I}$ Relationship instantiation

■ Lexicon of a knowledge base $\mathcal{L}^{\mathcal{KB}} := (\mathcal{L}^I, \mathcal{J})$

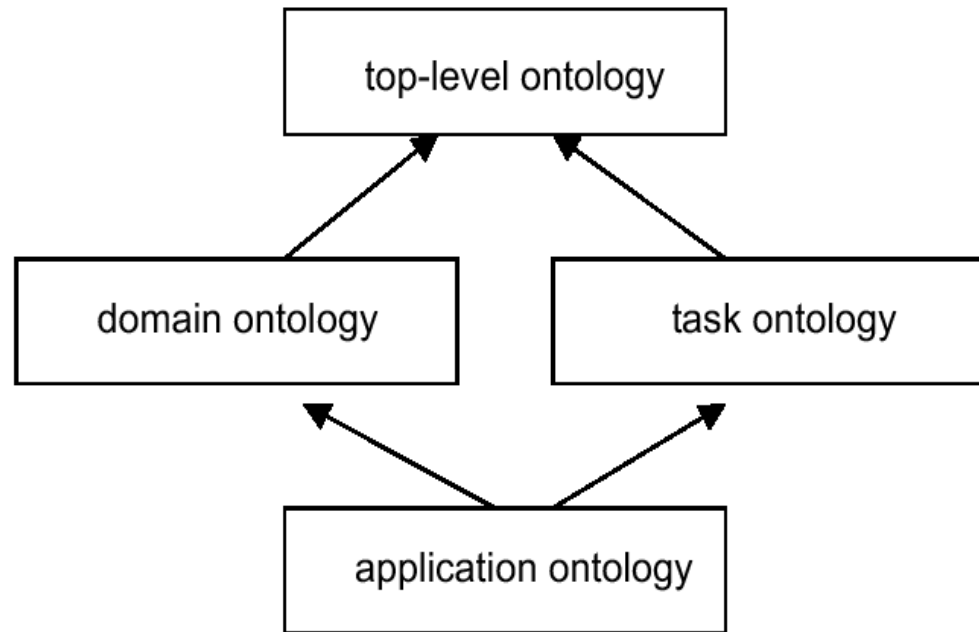


Different types of ontologies

[Guarino, 98]

Describe **very general concepts** like space, time, event, which are independent of a particular problem or domain. It seems reasonable to have unified top-level ontologies for large communities of users.

Describe the vocabulary related to a **generic domain** by specializing the concepts introduced in the top-level ontology.



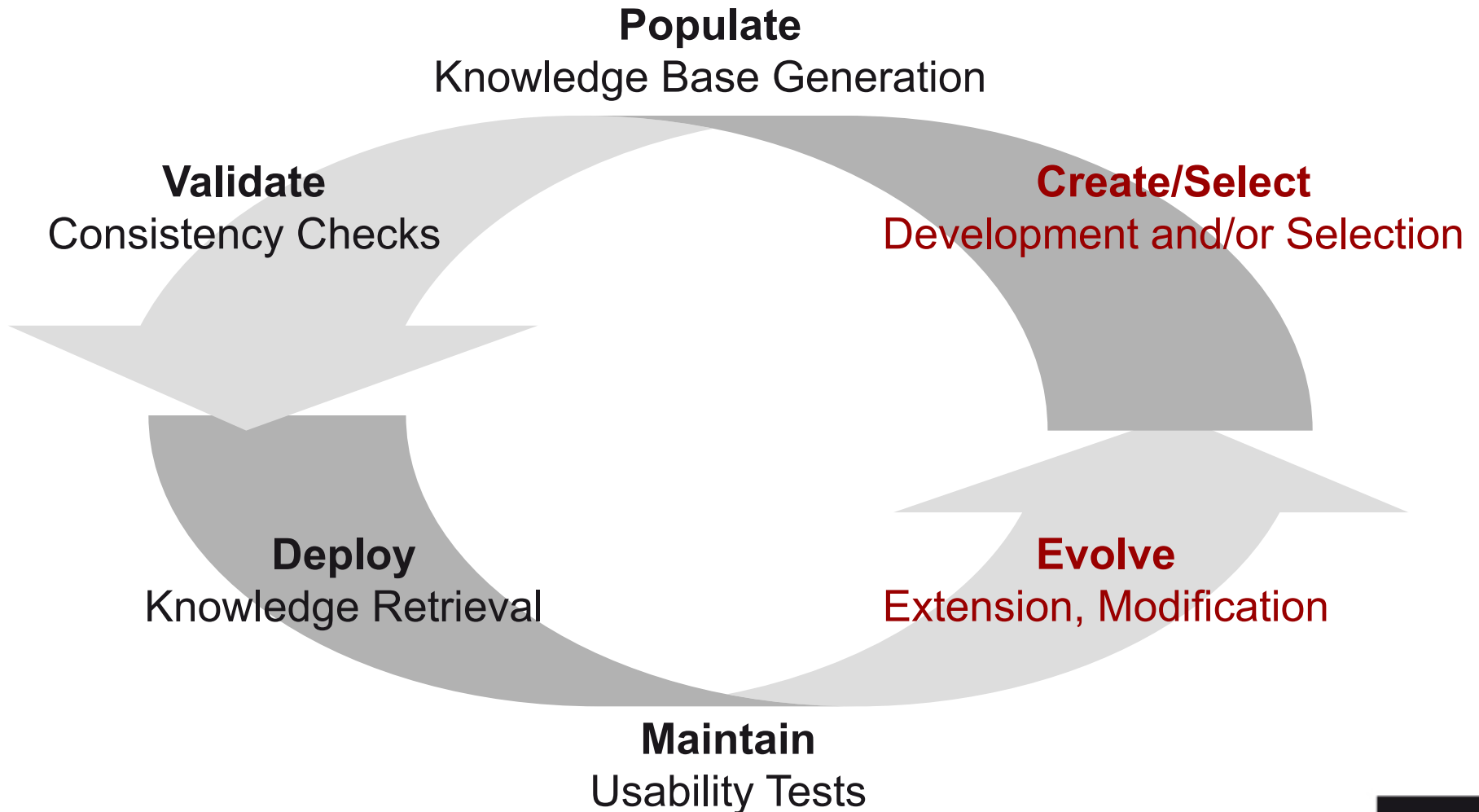
Describe the vocabulary related to a **generic task or activity** by specializing the top-level ontologies.

These are the most specific ontologies. Concepts in application ontologies often correspond to **roles played by domain entities while performing a certain activity**.

Heavyweight vs. Lightweight

- **They differ in their expressiveness, reasoning capacity, complexity, decidability.**
 - Lightweight
 - E-R diagrams, UML
 - Heavyweight
 - Description logic (DL), 1st order logic
- **There are standards for each case (RDF, RDF Schema, OWL)**

Life cycle of an ontology





Knowledge representation



Knowledge representation

■ Definition :

Knowledge representation refers to a set of tools and technologies designed to organize human knowledge for use and sharing.

■ Some solutions :

- Ontologies
- Logical Representations
- Production Rules
- Semantic Networks
- Conceptual graphs, frames
- Description Logics

From data to knowledge (1/4)

■ Data:

- Non-interpreted signals
- **Example** : « It is 2:00 pm »
→ data= 2:00 (**discrete**)
- **Example** : « The temperature in the room B313 »
→ data= [20° , 25°] (**continue**)

■ Information:

- An interpretation of the data
- **Example** : « today it is grey, the temperature is below 10° C and we are in room B313 »
→ climate = grey
→ Temperature = [0 ° C, 10° C]
→ room= B313
- **Representation** :
→ Couple (attribute, value)

From data to knowledge (2/4)

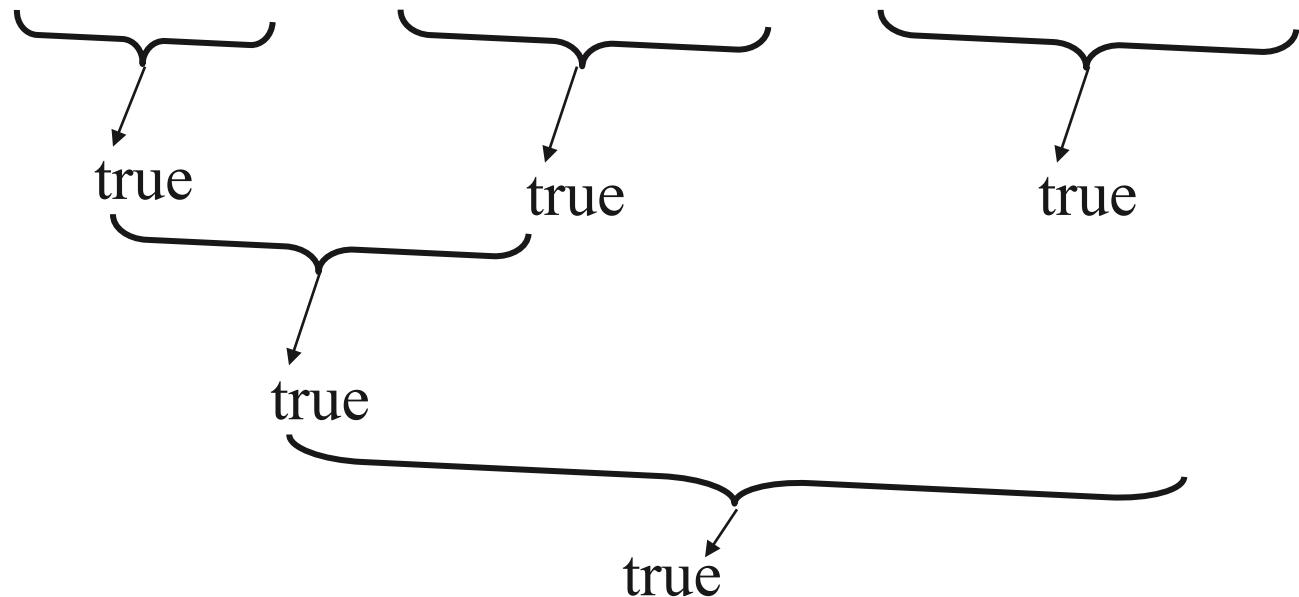
■ Knowledge:

- **Links between data and information**
- uses information in the context of actions, with a specific goal:
 - Logic operators: AND, OR, EXCEPT
 - Set operators: $\in, \notin, \cup, \cap, \supset, \supseteq$

From data to knowledge (3/4)

■ Expression of knowledge :

- (T° , 10) AND (room, B313) AND (climate, grey)



From data to knowledge (4/4)

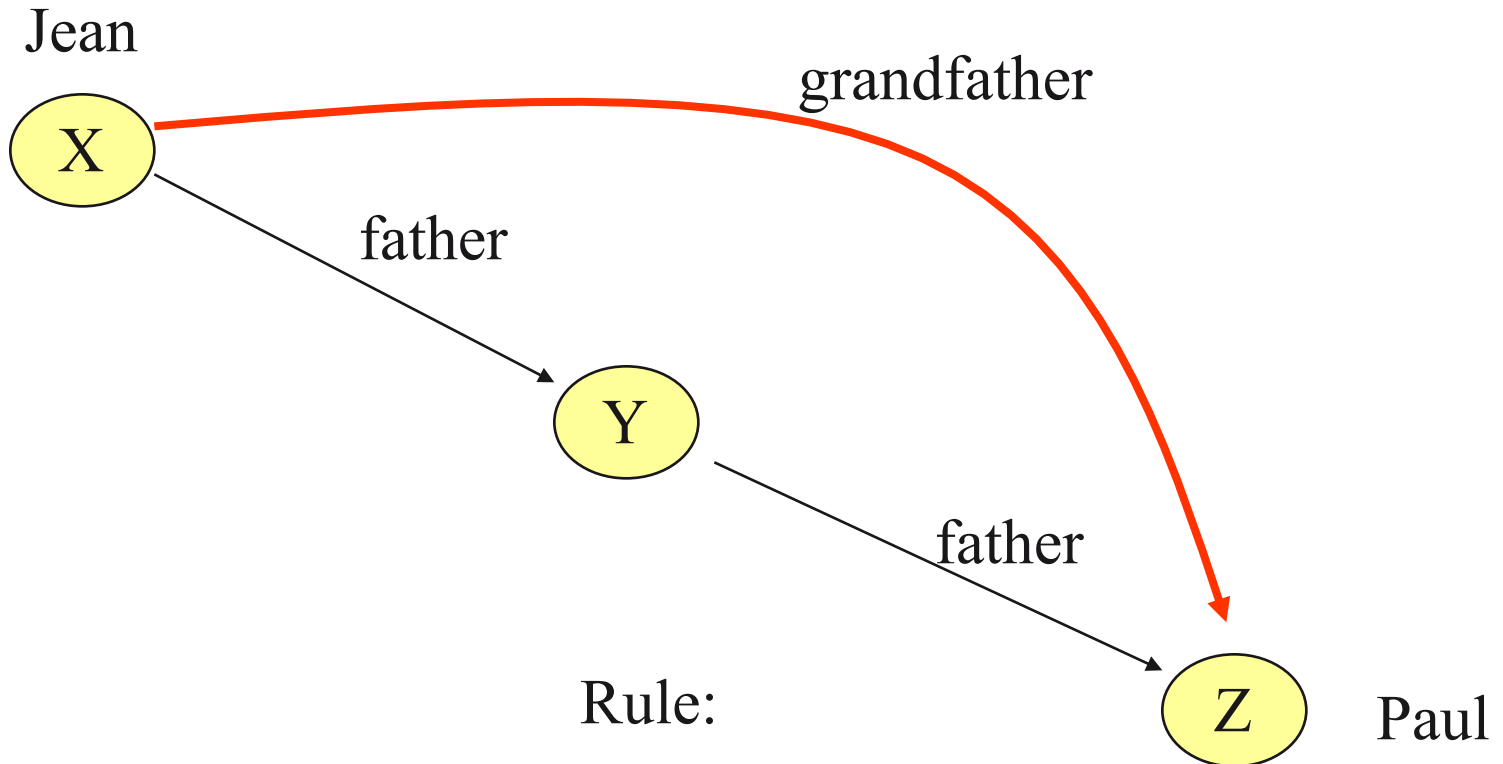
- **Knowledge = logical object(s)**
- **Logical object = properties, axioms and rules.**
 - **Property** = data / information
 - **Axiom** = data/information that admits a truth value (true, false).
 - **Rule** = way of inferring new true information (inferring new knowledge)

If ($T^{\circ} > 20^{\circ} \text{ C}$) and (sky is blue) then "it is sunny".

Example of rule

father(X,Y): "X is the father of Y".

grandfather(X,Y): "X is the grandfather of Y".



If **father(X,Y)** and **father(Y,Z)** Then **grandfather(X,Z)**.

Exercise

■ Axioms : (true facts)

- father(jean, paul)
- father(max, nadine)
- father(paul, max)

■ Question : Grandfather(X,Y) ?

■ Answer: 2 solutions (new facts inferred)

- grandfather(jean,max)
- grandfather(paul,nadine)

■ Question : grandfather(Pierre,Y) ?

- No answer (not applicable facts)

Several ways to represent knowledge

■ Logics

- Propositional Logic
- Predicate Logic
- Temporal Logic
- ...

■ Structured

- Ontologies
- Semantic networks

■ Degrees of belief

- Bayesian networks
- Fuzzy logic

Different Logics

■ Order 0 : propositional logic

- If Ferrari and Michael then fast

■ Order 0+ : typed propositional logic (attribute-value)

- Si car= Ferrari and pilot=Michael then speed=fast

■ Order 1 : predicate logic (first-order logic)

- X, Y : IF car(X) and X =Ferrari and drives(Y, X) and Y =Michael then fast(X)

■ Order 2 : second-order logic

- R, X, Y : IF type(R)=symmetric and $R(X, Y)$ then $R(Y, X)$

Reasoning: rules of inference

- Modus ponens

$$\frac{A, A \rightarrow B}{B}$$

- Modus tollens

$$\frac{\neg B, A \rightarrow B}{\neg A}$$

- ...

Reasoning: the basic cycle

■ Detection

- Determines the rules and relevant facts by means of "pattern matching" unifications

■ Choice

- Decides which of the applicable rules should actually be triggered

■ Execution

- Executes the action part of the rule.
- Updates the Knowledge base.

The selection phase

- **Selects a rule from the set of applicable rules**
- **Methods:**
 - The first applicable rule (e.g. Prolog)
 - The most specific rule
 - The most useful rule (according to a utility value to be calculated)
- **The metarules**
 - Rules controlling the selection of rules to be applied

The triggering phase

■ Three inference algorithms :

- Forward chaining (data driven)

$$\frac{A, A \rightarrow B}{B}$$

- to deduce a particular fact: the rules whose premises are known are triggered - up to - the fact to be deduced is also known - or that no rule can be triggered any more

- Backward chaining (goal oriented)

$$\frac{B, A \rightarrow B}{A}$$

- start from the fact to prove,
- search for all the rules that conclude on that fact,
- draw up a list of facts that need only be proved in order for them to be triggered and then
- apply recursively the same mechanism to the facts contained in those lists
- These facts in turn become facts to be proven.
- Recursion stops when the goal to prove belongs to the fact base
- Mixed chaining

Forward Chaining example

- **R1: B et D et E → F**
- **R2: D et G → A**
- **R3: C et F → A**
- **R4: C → D**
- **R5: D → E**
- **R6: A → H**
- **R7: B → X**
- **R8: X et C → A**

B Vrai
C Vrai
H ?

Strategy 1: the first rule

B, C _____ B, C, D _____ B, C, D, E _____ B, C, D, E, F _____
B, C, D, E, F, A _____ B, C, D, E, F, A, **H**

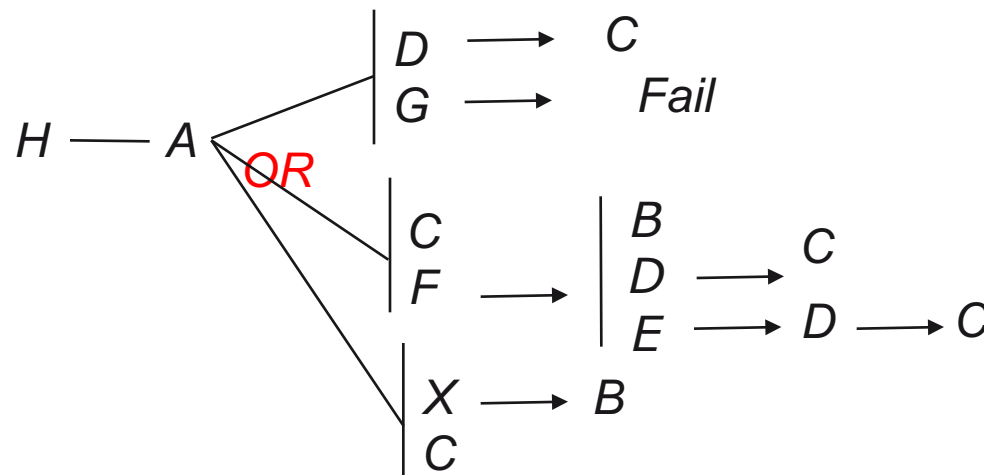
Strategy 2: the last rule

B, C _____ B, C, X _____ B, C, X, A _____ B, C, X, A, **H**

Backward Chaining Example

- **R1: B et D et E → F**
- **R2: D et G → A**
- **R3: C et F → A**
- **R4: C → D**
- **R5: D → E**
- **R6: A → H**
- **R7: B → X**
- **R8: X et C → A**

B Vrai
C Vrai
H ?



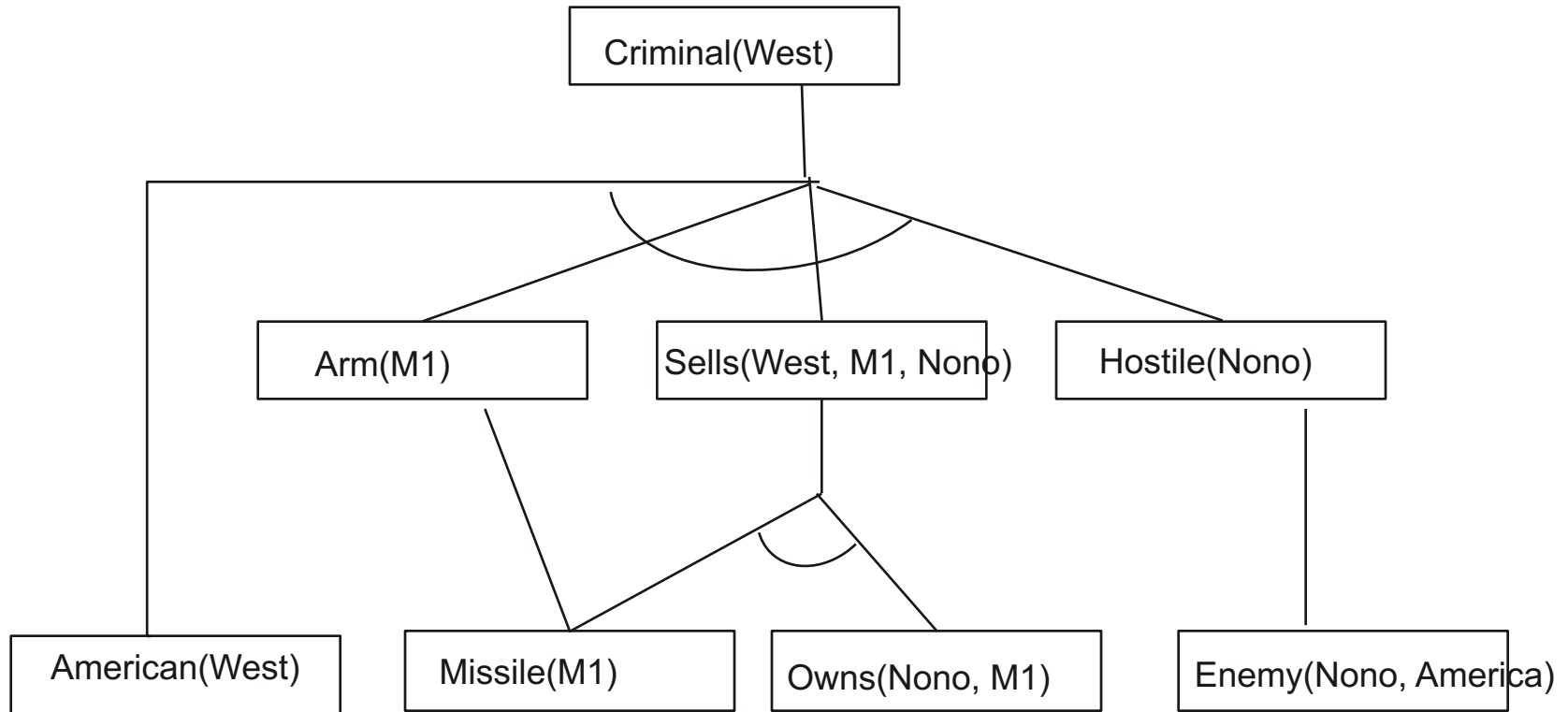
Example 2 : Knowledge base

- **The law states that it is a crime for an American to sell arms to hostile nations. The country Nono, an enemy of America, has some missiles, and all of its missiles were sold to it by Colonel West, who is American.**
- **⇒ Let's prove that West is a criminal**

Example 2 : Knowledge base

- **“... it is a crime for an American to sell arms to hostile nations.”:**
 - $American(x) \wedge Arm(y) \wedge Sells(x, y, z) \wedge Hostile(z) \Rightarrow Criminal(x)$
- **“Nono ... has some missiles”: i.e $\exists x Missile(x) \wedge Owns(Nono, x)$**
 - $Own(Nono, M1) \wedge Missile(M1)$
- **“.. all of its missiles were sold to it by Colonel West” :**
 - $\forall x Missile(x) \wedge Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$
- **Missiles are armes :**
 - $\forall x Missile(x) \Rightarrow Arme(x)$
- **An enemy of America count as « hostile » :**
 - $\forall x Enemy(x, America) \Rightarrow Hostile(x)$
- **“...West, who is American”:**
 - $American(West)$
- **“The country Nono, an enemy of America” :**
 - $Enemy(Nono, America)$

Forward Chaining Proof



$American(x) \wedge Arm(y) \wedge Sells(x,y,z) \wedge Hostile(z) \Rightarrow Criminal(x)$

$Owns(Nono, M_1) \wedge Missile(M_1)$

$Missile(x) \wedge Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$

$Missile(x) \Rightarrow Arme$

$Enemy(x, America) \Rightarrow Hostile(x)$

$American(West)$

$Enemy(Nono, America)$



OWL



Introduction

- is based on research carried out in the field of description logic
- used to describe ontologies, i.e. it enables terminologies to be defined to describe concrete domains
- is an important step forward in the representation and organisation of knowledge available on the Web
- is designed as an extension of the Resource Description Framework (RDF) and RDF Schema (RDFS).

Introduction

■ RDF and RDFS alone are too limited :

- Cannot specify the nature of the relationships between resources (reflexivity, etc.)
- No capacity for reasoning
- Very limited logic

■ The need for OWL :

- Derives from RDF + RDFS
- Logical connectors between classes (union, intersection, etc.)
- Cardinality on properties
- Characterization of properties (transitivity, inverse, etc.)

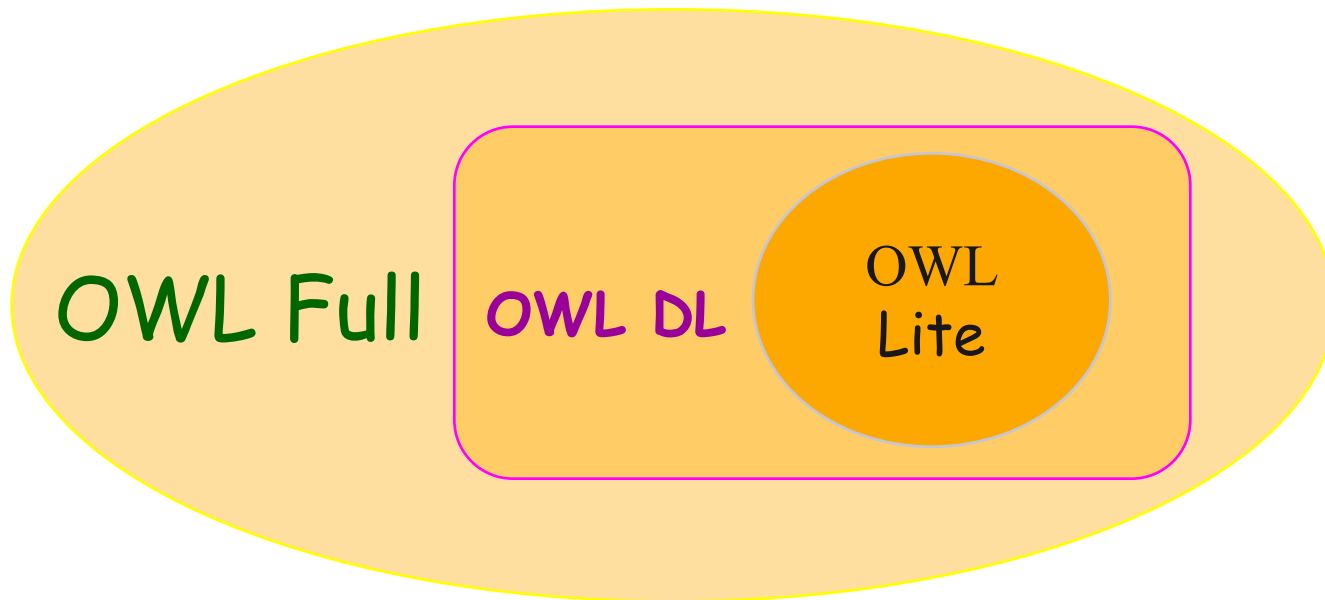
<i>Logic</i>	OWL
<i>Metadata</i>	RDF + RDF Schema
<i>Syntax</i>	XML + Schema XML

Main advantages

- Brings better **integration, evolution, sharing** and easier **inference** of ontologies
- Adds the concepts of **equivalent classes, equivalent properties, equality** of two resources, their **differences**, the **opposite, symmetry** and **cardinality**
- Thanks to its formal semantics based on a widely studied logical foundation, allows to define more **complex associations of resources** as well as the **properties** of their respective classes
- Is suitable for the Semantic Web, as it offers a strictly defined syntax, and depending on the level can allow automated reasoning on knowledge inferences and conclusions

OWL sub-languages

- OWL has three expressive languages for use by different communities of developers and users.



Structure of an owl ontology

■ Based on RDFS

■ An OWL ontology is an *OWL document* (file extension **.rdf** or **.owl**) with:

- Namespace declarations (owl, rdf, and others)
- The header (<owl:Ontology>) to describe the content of the ontology
- The definition of classes
- The definition of properties
- Assertion of facts

■ **Extensibility of existing ontologies :**

- <owl:import> to use other OWL ontologies and extend them :

Definition of OWL classes

■ A class can be declared in several ways:

- By naming the class or,
- By enumeration of its individuals
- By restricting the properties of its individuals
- By intersection (AND), union (OR) or complement (NOT) of another class

==> Anonymous Classes: The members of an anonymous class are the set of Individuals that satisfy its logical definition

- **There is an inheritance mechanism (<owl:subClassOf>)**
- **The superclass owl:Thing is the mother of all the other classes**
- **owl:Nothing is subclass of all classes**
- **In OWL Full, a class can be an instance of another class (a "metaclass").**
- **The set of instances of a class is called "the extension".**

Restriction Types

\exists	Existential, someValuesFrom	“Some”, “At least one”
\forall	Universal, allValuesFrom	“Only”
\in	hasValue	“equals x”
$=$	Cardinality	“Exactly n”
\leq	Max Cardinality	“At most n”
\geq	Min Cardinality	“At least n”

Property Characteristics

- Domain and range can be set
- OWL offers a mechanism for property inheritance:

```
owl:ObjectProperty rdf:ID="aPourFrere">  
<rdfs:subPropertyOf rdf:resource="#estDeLaFamilleDe" />  
<rdfs:range rdf:resource="#Humain" />  
<rdfs:domain rdf:resource="#Humain" />  
</owl:ObjectProperty>
```

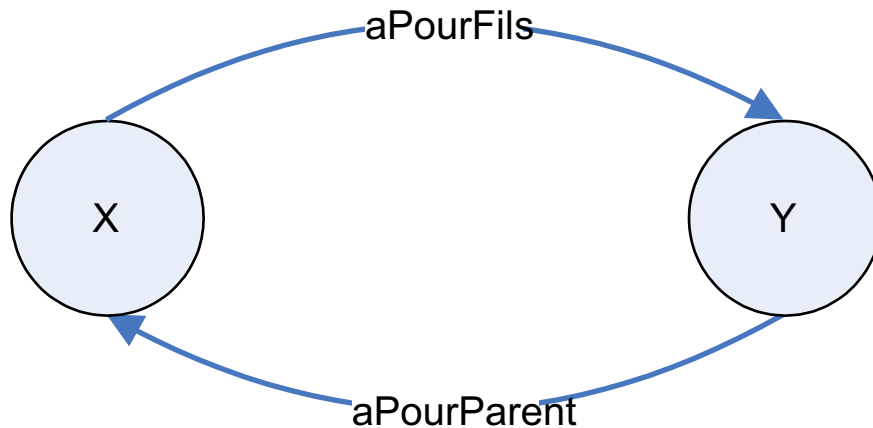
- Properties can be characterized:

- Inverse
- Transitivité
- Symétrie
- Fonctionnelle
- fonctionnelle Inverse

```
<<owl:ObjectProperty rdf:ID="aPourFrere">  
<rdf:type rdf:resource="#owl:SymmetricProperty" />  
<rdfs:range rdf:resource="#Humain" />  
<rdfs:domain rdf:resource="#Humain" />  
</owl:ObjectProperty>
```

Inverse Property

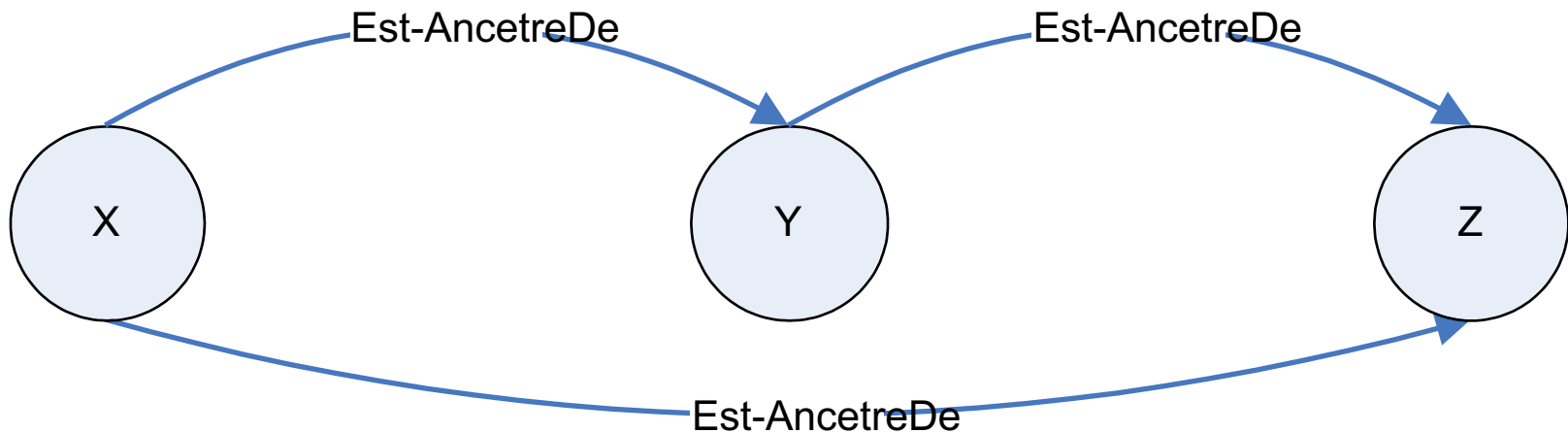
- $P1(X, Y) \text{ iff } P2(Y, X)$



- X mange Y iff Y estMangéPar X
- X aPourParent Y iff Y aPourFils X

Transitive Property

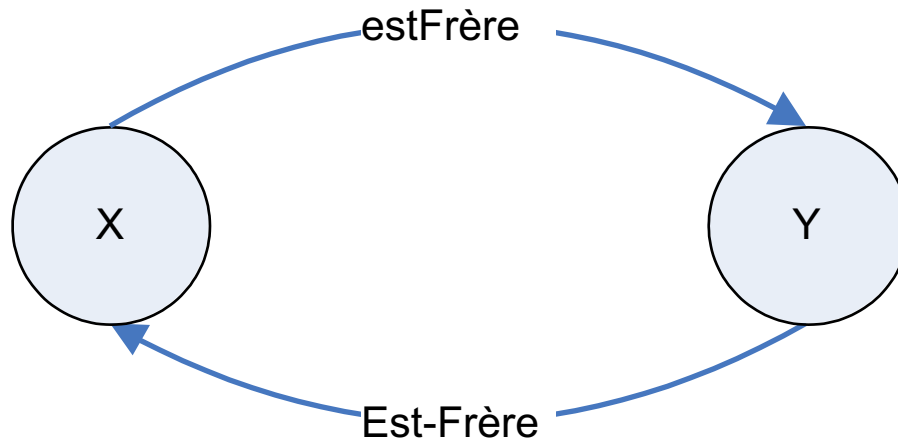
- If $P(X,Y)$ and $P(Y,Z)$ then $P(X,Z)$



- X ancetreDe Y, Y ancetreDe Z, then X ancetreDe Z

Symmetric Property

- $P(X,Y) \text{ iff } P(Y,X)$

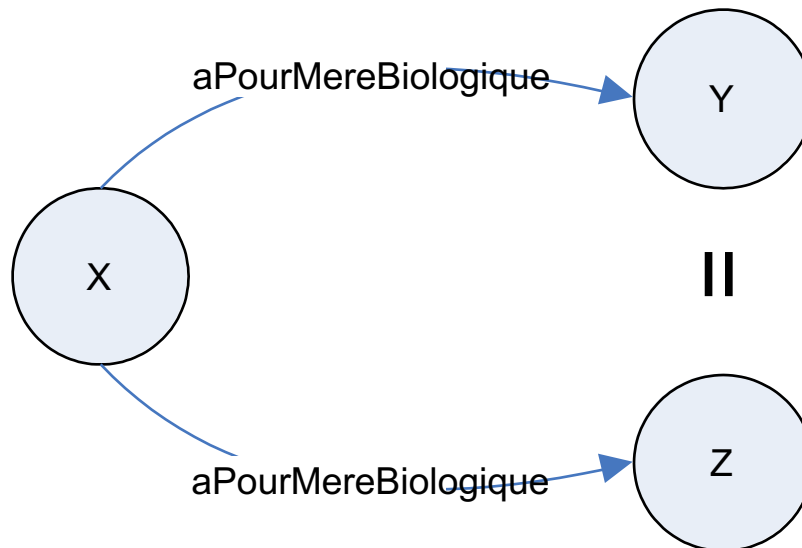


- $X \text{ estFrère } Y \text{ iff } Y \text{ estFrère } X$

Functional Property

■ Unicity

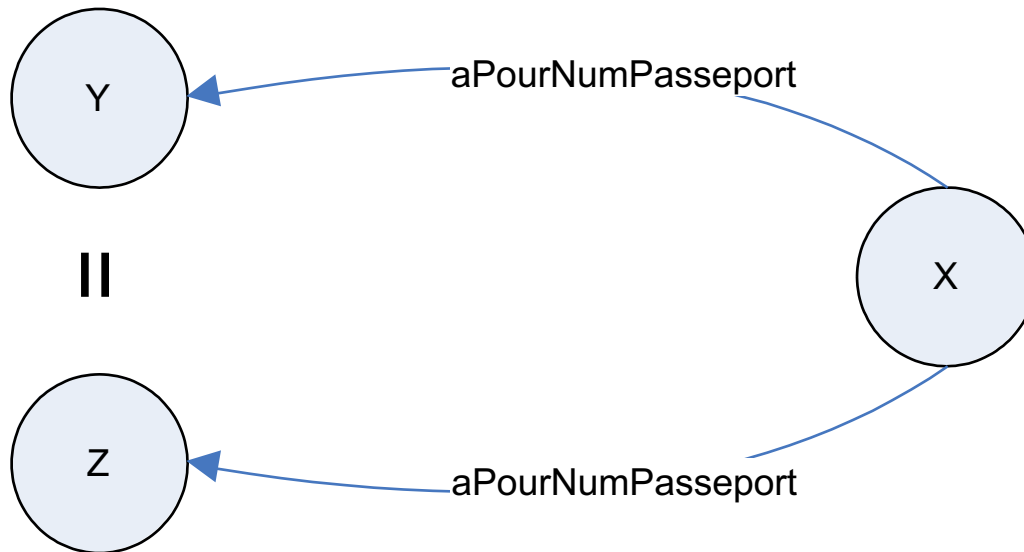
- Only one instance can be linked
If $P(X,Y)$ and $P(X,Z)$ then $Y=Z$



- If $X \text{ aPourMereBiologique } Y$ and $X \text{ aPourMereBiologique } Z$
then $Y=Z$

Inverse Functional Property

- $P(Y,X)$ and $P(Z,X)$ then $Y=Z$



- $Y aPourNumPasseport X$ and $Z aPourNumPasseport X$
then $Y=Z$

OWL Class Constructors

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	$\exists^{\geq n} y.P(x, y)$