

DATA SCIENCE MASTER

SEMANTIC KNOWLEDGE REPRESENTATION

Ontology Formalisation and Protégé

Mounira Harzallah

[mounira.harzallah@univ-nantes.fr](mailto:mounira.harzallah@univ-nantes.fr)

*Tél: 02 28 09 21 27*

# Plan

- ❑ Formalisation of ontology with description logics (DLs)
- ❑ Reasoning with DL on ontology
- ❑ Formalisation of ontology with OWL/Protégé

# Description Logics

**Description logics** are logical formalisms for :

- Defining concepts and their relations/roles (**Terminologies**)
- Specifying properties of individuals (**Assertions**)

**Main advantages:**

- **Well-defined semantic:** *defined* by interpreting with subsets of individuals
- **Flexibility:** Formalisation of large kinds of situations

Today DLs represent one of the main language of knowledge representation, with applications in a number of areas of computer sciences

**Semantic web:** DLs are a foundation of ontology languages (OWL) which have been standardised by W3C group

**Software engineering:** one can translate UML diagrams into DL

**Database theory:** one can translate E/R-models into DL

# DL constructs for ontology formalisation

**Atomic Concepts** denote subsets of the domain that are not defined (primitive concepts). All other concepts are defined in terms of the primitive concepts

Example: Human, Animal, Plant, Female

**Roles** denote **binary relations on the domain**

Example: HasChild, HasCost, HasPart, IsAuthorOf

**Concepts constructors** denote operations for combining and building complex concepts.

Example: Conjunction ( $\sqcap$ ) , Disjunction ( $\sqcup$ )

**Role constructors** denotes operations for building complex roles

Example: transitive closure ( $*$ ) and  $R^*$  is a transitive closure on  $R$

$$R^*(x)=y \iff \text{It exists } z \text{ where } R(x)=z \text{ and } R^*(z)=y$$

# DL Constructs for ontology formalisation

## Example

- ▣ **Atomic/primitive concepts:** Human, Animal, Plant, Female
- ▣ **Role:** HasChild, Marryto
- ▣ **Complex Concepts :**
  - Animal  $\sqcap$  Plant
  - Human  $\sqcap \exists \text{HasChild}.\top$
  - Human  $\sqcap \exists \text{HasChild}.\top \sqcap \forall \text{HasChild}.\text{Female}$
  - Human  $\sqcap \neg \text{Female}$
- ▣ **Complex Role:**  $(\text{HasChild})^* = \text{HasDescendant}$   
HasDescendant (x) = HasChild\*(x)=y  $\Leftrightarrow$   
It exists z where Haschild(x)= z and HasDescendant(z)= y

# DL goals

DL is a language of description :

- How to constitute concepts and roles,
- Mechanisms to specify knowledge on **concepts and roles**, using **Terminological axioms** i.e. **TBox** : collection of definitions of concepts and their relations
- Mechanisms to specify the **properties of objects/individuals** using **Assertional axioms** i.e. **Abox**: specification of individual properties , where the properties are those defined in the TBox
- A set of mechanisms for reasoning

# DL goals

## Example

- Complex concepts:

$\text{Human} \sqcap \text{Male} \sqcap \exists \text{HasChild} \sqcap \forall \text{HasChild}.\text{Doctor}$

- Tbox

$T = \{\text{Father} \equiv \text{Human} \sqcap \text{Male} \sqcap \exists \text{hasChild},$   
 $\text{HappyFather} \subseteq \text{Father} \sqcap \forall \text{HasChild}.\text{Doctor}\}$

- Abox

$A = \{\text{HappyFather}(\text{Jean}), (\text{Jean}, \text{Marie}) \in \text{Haschild}^I\}$

- Example of reasoning

$T \models \text{HappyFather} \subseteq \exists \text{HasChild}.\text{Doctor}$   
 $T \cup A \models \text{Doctor}(\text{Marie})$

# Semantic of DL constructs

- An interpretation  $I$  is a pair  $(F, \Delta)$ , where
  - $\Delta$  is a non-empty set
  - $Fc : \text{AtomicConcepts} \rightarrow \text{Pow}(\Delta)$
  - $Fr : \text{Roles} \rightarrow \text{Pow}(\Delta \times \Delta)$

- $Fc(C) = C^I$  is the interpretation of  $C$

The set of individuals of  $C$  is one interpretation of  $C$

The set of ordered pairs of individuals of  $R$  is one interpretation of  $R$

## **Interpretation is a mean to explain**

- the semantic of DL constructs
- the reasoning made



# Interpretation and Reasoning

□ An interpretation  $I$  satisfies a Tbox, if it satisfies every axioms in the Tbox

denoted by  $I \models \text{Tbox}$

□  $\text{Tbox} \models A_i \Rightarrow$  all the interpretations of the Tbox should satisfy  $A_i$ ,  $A_i$  is an axiom

If  $\exists$  an interpretation of **Tbox** that dosent' satisfy  $A_i \Rightarrow \text{Tbox} \not\models A_i$

## Example

**Tbox** =  $\{C \sqcup D \equiv B\}$ ,  $I_1 : C^{I_1} = \{1, 2, 3\}$ ,  $D^{I_1} = \{1, 4\}$ ,  $B^{I_1} = \{1, 2, 3, 4\}$

$\{1, 2, 3\} \cup \{1, 4\} = \{1, 2, 3, 4\} \Rightarrow I_1 \models \text{Tbox}$

**A1** :  $C \sqcap D \equiv \perp$ ,  $\text{Tbox} \not\models A_1$  because  $C^{I_1} \cap D^{I_1} = \{1\}$  is not empty

# Semantic of DL constructs

Construct	Syntax	Example	Semantics
atomic concept	$A$	Doctor	$A^I \subseteq \Delta^I$
atomic role	$P$	hasChild	$P^I \subseteq \Delta^I \times \Delta^I$
atomic negation	$\neg A$	$\neg$ Doctor	$\Delta^I \setminus A^I$
conjunction	$C \sqcap D$	Hum $\sqcap$ Male	$C^I \cap D^I$
(unqual.) exist. res.	$\exists R$	$\exists$ hasChild	$\{ a \mid \exists b. (a, b) \in R^I \}$
value restriction	$\forall R.C$	$\forall$ hasChild.Male	$\{ a \mid \forall b. (a, b) \in R^I \rightarrow b \in C^I \}$
bottom	$\perp$		$\emptyset$

# Semantic of DL constructs

Construct	Syntax	Semantics
disjunction	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
top	$\top$	$\Delta^{\mathcal{I}}$
qual. exist. res.	$\exists R.C$	$\{ a \mid \exists b. (a, b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}} \}$
(full) negation	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
number	$(\geq k R)$	$\{ a \mid \#\{b \mid (a, b) \in R^{\mathcal{I}}\} \geq k \}$
restrictions	$(\leq k R)$	$\{ a \mid \#\{b \mid (a, b) \in R^{\mathcal{I}}\} \leq k \}$
qual. number	$(\geq k R.C)$	$\{ a \mid \#\{b \mid (a, b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\} \geq k \}$
restrictions	$(\leq k R.C)$	$\{ a \mid \#\{b \mid (a, b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\} \leq k \}$
inverse role	$R^{-}$	$\{ (a, b) \mid (b, a) \in R^{\mathcal{I}} \}$

# Semantic of LD constructs

$$C \sqcup D = B \Rightarrow C \sqsubseteq B$$

## Equivalences of the Negation

$$\neg(C \sqcap D) = \neg C \sqcup \neg D$$

$$\neg(C \sqcup D) = \neg C \sqcap \neg D$$

$$\neg(\exists R.C) = \forall R.\neg C$$

$$\neg(\forall R.C) = \exists R.\neg C$$

$$\neg\neg C = C$$

For reasoning, we can use interpretations and/or associate a first-order formula with free-variable  $x, y$

For an atomic concept  $A$ ,  $A(x) = A(x)$

For a role  $R$ , we use a binary relation symbol  $R(x, y)$

$$\top(x) = P(x) \vee \neg P(x) \text{ and } \perp(x) = P(x) \wedge \neg P(x)$$

$$C \sqcap D(x) = C(x) \wedge D(x)$$

$$C \sqcup D(x) = C(x) \vee D(x)$$

$$\exists R.C(x) = \exists y(R(x, y) \wedge C(y))$$

$$\forall R.C(x) = \forall y(R(x, y) \rightarrow C(y))$$

# Exercices

1. Write "Course with at most 20 participants, all of which are master or Ph.D students", (primitive concepts: Course, MasterStudent, PhDStudent, role: HasParticipant).
2.  $T = \{ \text{Father} \equiv \text{Human} \sqcap \text{Male} \sqcap \exists \text{hasChild},$   
 $\text{HappyFather} \subseteq \text{Father} \sqcap \forall \text{HasChild.Doctor} \}$   
 $A = \{ \text{HappyFather}(\text{Jean}), \text{Haschild}(\text{Jean}, \text{Marie}) \}$

To infer using interpretations :

$$\begin{aligned} T &\models \text{HappyFather} \subseteq \exists \text{HasChild.Doctor} \\ T \cup A &\not\models \text{Doctor}(\text{Marie}) \end{aligned}$$

# Exercices

$T = \{ \text{Father} \equiv \text{Human} \sqcap \text{Male} \sqcap \exists \text{hasChild},$

$\text{HappyFather} \subseteq \text{Father} \sqcap \forall \text{HasChild.Doctor} \}$

$A = \{ \text{HappyFather}(\text{Jean}), \text{Haschild}(\text{Jean}, \text{Marie}) \}$

To infer using interpretations :

$T \models \text{HappyFather} \subseteq \exists \text{HasChild.Doctor}$

$T \cup A \models \text{Doctor}(\text{Marie})$

$\text{HappyFather} \subseteq \exists \text{hasChild} \sqcap \forall \text{HasChild.Doctor}$

$\rightarrow (\forall x, \text{Happyfather}(x) \rightarrow (\exists y, \text{HasChild}(x,y) ) \wedge (\forall y \text{HasChild}(x,y) \rightarrow \text{Doctor}(y)))$

$\rightarrow (\forall x, \text{Happyfather}(x) \rightarrow \exists y, \text{HasChild}(x,y) \wedge \text{Doctor}(y))$

$\rightarrow \text{HappyFather} \subseteq \exists \text{hasChild.Doctor}$

$\text{HappyFather}(\text{Jean}) \rightarrow \forall \text{HasChild.Doctor}(\text{Jean}) \wedge$

$\text{Haschild}(\text{Jean}, \text{Marie}) \wedge \forall \text{HasChild.Doctor}(\text{Jean}) \rightarrow \text{Doctor}(\text{Marie})$

# How to formalise ontology

## Primitive Concepts

Human, Male

## Complex Concept definitions

$\text{Man} \subseteq \text{Human}$

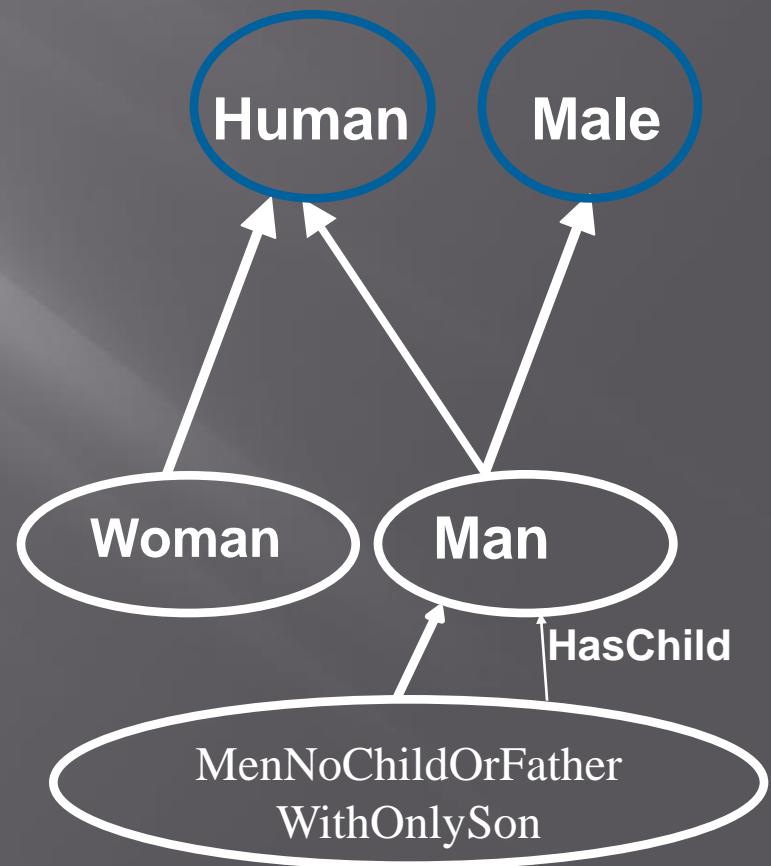
$\text{Man} \equiv (\text{Human} \sqcap \text{Male})$

$\text{Woman} \equiv \text{Human} \sqcap (\neg \text{Male})$

$\text{Father} \subseteq \exists \text{HasChild}$

$\text{Father} \equiv (\text{Man} \sqcap \exists \text{HasChild})$

$\text{MenNoChildOrFatherWithOnlySon} \equiv$   
 $(\text{Man} \sqcap \forall \text{HasChild}.\text{Man})$



# Reasoning on ontology

- ❑ Classification: classification of a concept within a taxonomy
- ❑ Improving a classification: a better classification of a concept)
- ❑ Inference of new properties of concepts, individuals or roles.
- ❑ Determination of unsatisfiability
- ❑ Identification of logical inconsistencies



# Reasoning. Classification of concepts

A concept  $C$  should be classed under the concept  $C1$  that is the most close to it.

$C1$  is the most close to  $C$ , if  $C$  is a sub class of  $C1$  where there is not another concept  $C2$  (distinct of  $C1$ ) where  $C \subseteq C2 \subseteq C1$

Person

CitizenOf(Person, County)

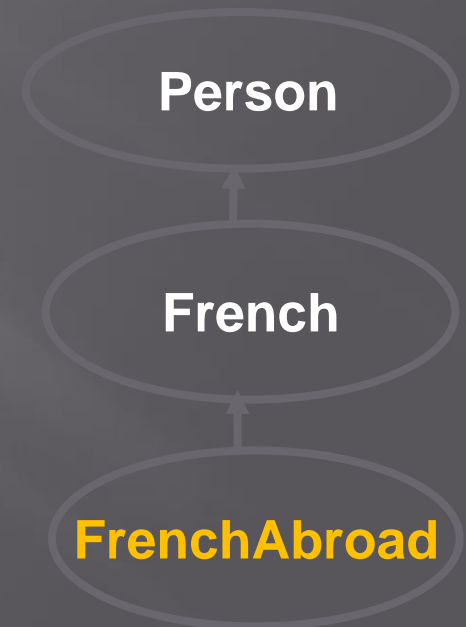
LivesIn(Person, Country)

French  $\equiv$  (Person  $\sqcap \exists$  Citizen.{France})

**FrenchAbroad** is a new concept defined as

$\equiv$  (Person  $\sqcap (\exists$  Citizen.{France}  
 $\sqcap \exists$  LivesIn. $\neg$ {France}))

Then **FrenchAbroad**  $\subseteq$  French

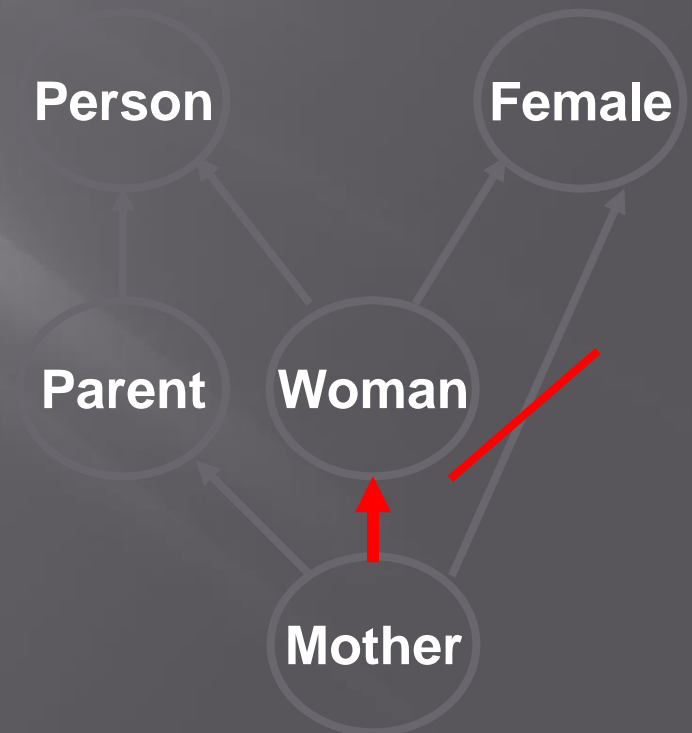


# Reasoning. Taxonomy improving

- ▣ Person
- ▣  $\text{Parent} \equiv (\text{Person} \sqcap \exists \text{HasChild}.\text{Person})$
- ▣  $\text{Woman} \equiv (\text{Person} \sqcap \text{Female})$
- ▣  $\text{Mother} \equiv (\text{Parent} \sqcap \text{Female})$

In the taxonomy Mother is-a Female

But we can deduce that  
Mother is-a Woman



# Reasoning. Logical inconsistency

Ontology inconsistency is when an ontology contains logical contradictions for which a model does not exist

Examples of ontology inconsistencies

- $O \models (C_i \sqcap C_j = \perp \text{ and } C_i^I \sqcap C_j^I = \{c\})$
- $O \models C_i = \neg C_i$

# Reasoning. Unsatisfiability

- ▣ **Satisfiability.** A concept  $C$  is satisfiable if there exists an interpretation  $I$  such that  $C^I \neq \emptyset$

**The following complex concept are satisfiable**

- $\text{Parent} \equiv (\text{Person} \sqcap \exists \text{hasChild}.\text{Person})$
- $\text{Woman} \equiv (\text{Person} \sqcap \text{Female})$
- $\text{Man} \equiv (\text{Person} \sqcap \text{Male})$
- $\text{Mother} \equiv (\text{Female} \sqcap \text{Parent})$
- $\text{NonHuman} \equiv \neg \text{Woman} \sqcap \neg \text{Man}$
- $\text{OtherMother} \equiv (\text{NonHuman} \sqcap \text{Mother})$

**We can deduce that there is not a non empty set that can interpret OtherMother**

OtherMother is Unsatisfiable

# OWL (Ontology Web Language)

OWL is developed based on DL. It defines

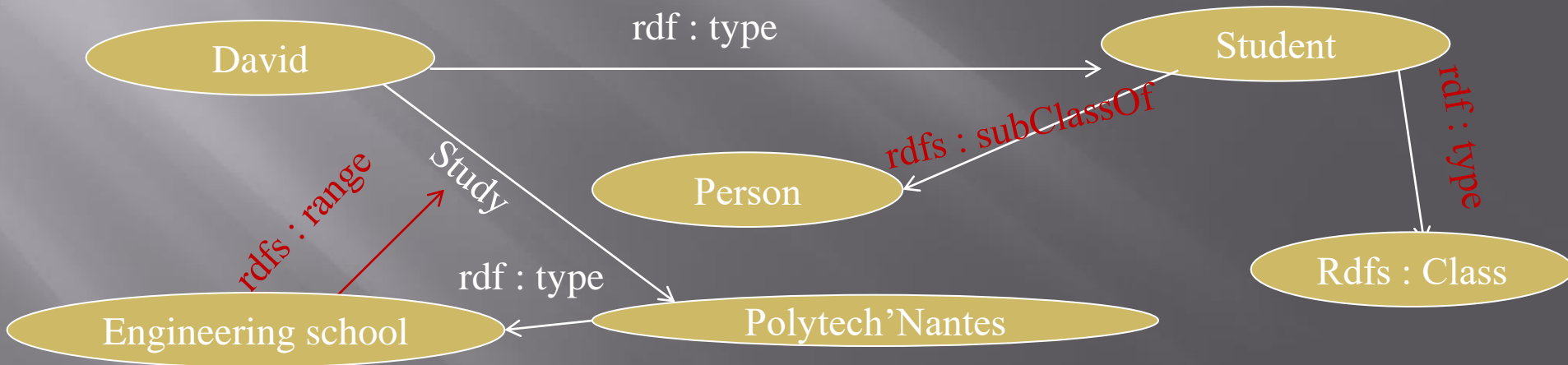
- ▣ Hierarchies of Classes
- ▣ Hierarchies of relations (or properties)
- ▣ Axioms on concepts and relations

# OWLS

RDF/RDFs are languages to describe ressources/ data on the web

RDF defines the web ressources as triplets of (objet predicat objet)

RDFs : structures the ressources of RDF



RDF/RDFs are not suffisant to specify and formalize ontologies

# OWL languages

- ▣ OWL is an extension of RDF/RDFs
- ▣ OWL languages: OWL Lite, OWL DL, OWL Full
- ▣ OWL languages support a variety of syntaxes.

# OWL vs LD

OWL constructor	DL constructor	Example
intersectionOf	$C_1 \sqcap \dots \sqcap C_n$	Human $\sqcap$ Male
unionOf	$C_1 \sqcup \dots \sqcup C_n$	Doctor $\sqcup$ Lawyer
complementOf	$\neg C$	$\neg$ Male
oneOf	$\{a_1\} \sqcup \dots \sqcup \{a_n\}$	{john} $\sqcup$ {mary}
allValuesFrom	$\forall P.C$	$\forall$ hasChild.Doctor
someValuesFrom	$\exists P.C$	$\exists$ hasChild.Lawyer
maxCardinality	$(\leq n P)$	$(\leq 1$ hasChild)
minCardinality	$(\geq n P)$	$(\geq 2$ hasChild)



# OWL vs LD

OWL 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	$C_1 \sqsubseteq \neg C_2$	Man $\sqsubseteq \neg$ Female
sameIndividualAs	$\{a_1\} \equiv \{a_2\}$	{presBush} $\equiv$ {G.W.Bush}
differentFrom	$\{a_1\} \sqsubseteq \neg\{a_2\}$	{john} $\sqsubseteq \neg$ {peter}
subPropertyOf	$P_1 \sqsubseteq P_2$	hasDaughter $\sqsubseteq$ hasChild
equivalentProperty	$P_1 \equiv P_2$	hasCost $\equiv$ hasPrice
inverseOf	$P_1 \equiv P_2^-$	hasChild $\equiv$ hasParent <sup>-</sup>

# Protégé : ontology editor

- ▣ Developed with Java
- ▣ Specific language, but based on OWL (and then on DL)
- ▣ With protégé, we can edit an ontology in OWL (in several syntaxes)
- ▣ Possibility to integrate plug-in by adding applications on the ontology
- ▣ Ontology is defined as a hierarchy of classes (is-a), slots (relations), instances and axioms

# Protege et OWL

Create a new OWL ontology → OWL/XML

Save your project

The screenshot shows the Protege OWL editor interface. The top menu bar includes 'Active Ontology', 'Entities', 'Classes', 'Object Properties', 'Data Properties', 'Individuals', 'OWLViz', 'DL Query', and 'OntoGraf'. The 'Classes' tab is active, showing a class hierarchy on the left and a usage panel on the right. The class hierarchy on the left shows a tree structure starting with 'Thing', which has subclasses 'LIVING' and 'Object'. 'LIVING' has subclasses 'Animal' and 'PLANT'. 'Animal' has subclasses 'Human' and 'hasdaughterornothing'. 'Human' has subclasses 'Men' and 'Women'. 'Men' has subclasses 'Husband' and 'Women'. 'Women' has subclasses 'Husband' and 'Women'. The usage panel on the right shows 'Usage: Human' and 'Found 12 uses of Human'. It lists 'HasChild' with domain 'Human' and range 'Human', 'Human' with superclass 'LIVING', and 'Husband' with superclass 'Human'. The 'Add classes' annotation points to the class hierarchy tree. The 'Add axioms' annotation points to the 'hasdaughterornothing' class. The 'Add properties' annotation points to the 'Object Properties' tab. The 'Add individuals' annotation points to the 'Individuals' tab.

**Add classes**

**Add axioms**

**Add properties**

**Add individuals**

# Definition of classes/subClasses and Instances with OWL

The relation Class/subClass is defined by the definition of a hierarchy

```
<owl:Class rdf:ID="Woman">  
  <rdfs:subClassOf rdf:resource="Human"/>  
</owl:Class>  
<owl:Class rdf:ID="man">  
  <rdfs:subClassOf rdf:resource="Human"/>  
</owl:Class>
```

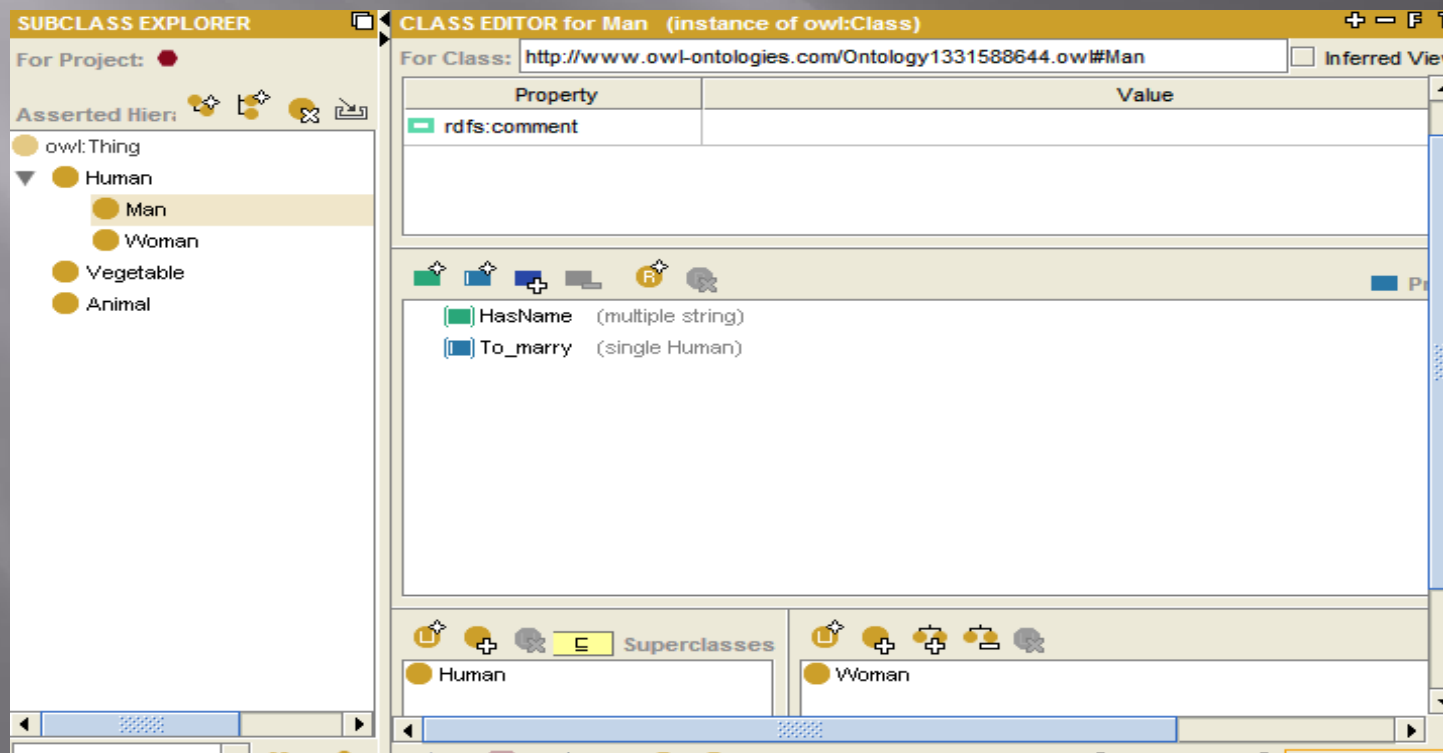
Definition of intances

```
<Woman rdf:ID="Mounira"/>
```

# Definition of axioms with OWLProtege

Example : two classes are disjoint (i.e if it doesn't exist an instance that belongs to the two classes)

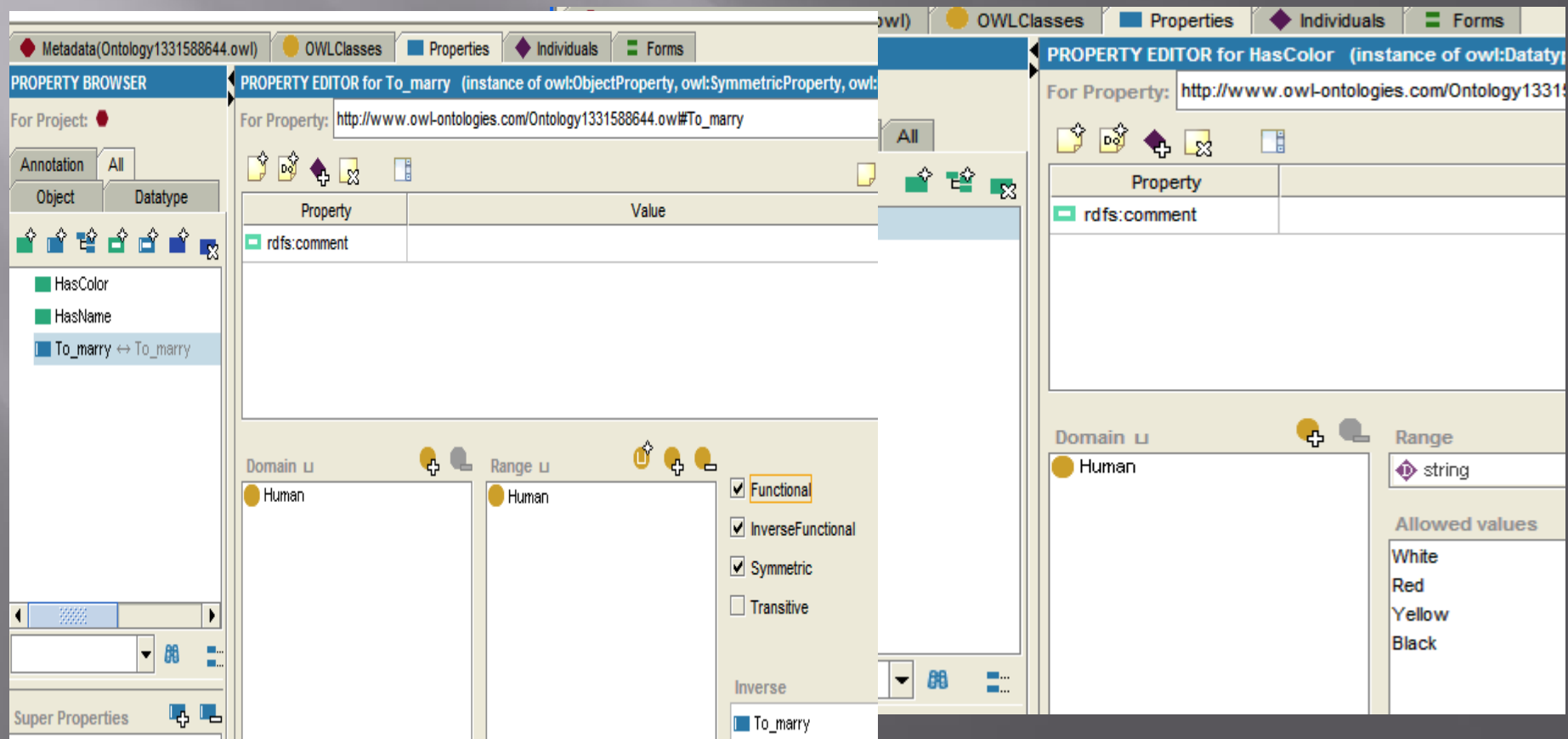
```
<owl:Class rdf:ID="Woman">  
  <rdfs:subClassOf rdf:resource="Human"/>  
  <owl:disjointWith rdf:resource="Man"/>  
</owl:Class>
```



# Definition of properties with Protege/OWL

Two kinds of properties :

- Object properties : between two individuals
- Datatype properties : between an individual and a datatype



The screenshot displays the Protege OWL editor interface with two property editors open side-by-side.

**Left Editor: PROPERTY EDITOR for To\_marry**  
(instance of owl:ObjectProperty, owl:SymmetricProperty, owl:TransitiveProperty)  
For Property: [http://www.owl-ontologies.com/Ontology1331588644.owl#To\\_marry](http://www.owl-ontologies.com/Ontology1331588644.owl#To_marry)

Property	Value
<input checked="" type="checkbox"/> rdfs:comment	

Domain: ☒ Human  
Range: ☒ Human

☒ Functional  
☒ InverseFunctional  
☒ Symmetric  
☐ Transitive

Inverse: ☒ To\_marry

**Right Editor: PROPERTY EDITOR for HasColor**  
(instance of owl:DatatypeProperty)  
For Property: <http://www.owl-ontologies.com/Ontology1331588644.owl#HasColor>

Property	Value
<input checked="" type="checkbox"/> rdfs:comment	

Domain: ☒ Human  
Range: ☒ string

Allowed values: White, Red, Yellow, Black

# Definition of axioms OWL/Protege

## Characteristics of properties :

- ▣ It can be the inverse of another property
- ▣ It can have only one value for a given instance: function
- ▣ It can be transitive
- ▣ It can be symmetric

## Restrictions on properties:

- ▣ Restrictions on the cardinalities
- ▣ Restrictions on the values (instances) (hasValue)
- ▣ Restrictions with quantifiers (existential and universal)

# Definition of axioms with OWL/Protege

## Restrictions with quantifiers

### Example

“ $\forall$ hasChild **only** Woman” : instances that have daughter or don't have child

The screenshot displays the Protege interface with three main panels:

- SUBCLASS EXPLORER:** Shows the class hierarchy for the project 'Living'. The hierarchy is: owl:Thing (parent) -> Living (child) -> Human (child) -> hasGirl\_or\_noChild (child, highlighted). Other children of Human include Class\_24, Has\_Nasser\_or\_David, haschild, HasGirl, Man, NoChild, and Woman. Vegetables are also listed as a separate class.
- CLASS EDITOR for hasGirl\_or\_noChild:** Shows the 'Property' tab with 'rdfs:comment'. Below, the 'Restrictions' list shows 'HasChild only Woman' selected.
- Restricted Property dialog:** A modal window for defining the restriction. The 'Restricted Property' list contains 'HasChild', 'hascolor', and 'To\_marry'. The 'Restriction' list contains 'allValuesFrom', 'someValuesFrom', 'hasValue', 'cardinality', 'minCardinality', and 'maxCardinality'. The 'Filler' text box contains 'Woman'. The 'Restriction' type is set to 'allValuesFrom' (indicated by a green checkmark). The 'OK' button is highlighted.



# Definition of axioms with OWL/Protege

## Restrictions with quantifiers

Examples :

$\exists$ HasChild **some** Woman : instances that has at least one daughter (someValuesFrom)

Restriction on Value (on instances)

$\exists$ HasColor **has** "White" : instances has at least a color white (here white is an instance)

The screenshot displays the Protege software interface. On the left, the 'SUBCLASS EXPLORER' shows an 'Asserted Hierarchy' with classes: owl:Thing, Human, Man, WhiteMAN, Woman, HavingDaughter\_or\_Nochild, HavingatleastDaughter, Vegetable, and Animal. The 'CLASS EDITOR for WhiteMAN' is open, showing the 'Property' tab with 'rdfs:comment'. The 'Create Restriction' dialog is active, showing the 'Restricted Property' list with 'HasChild', 'hascolor', and 'To\_marry'. The 'hascolor' property is selected. The 'Restriction' list on the right includes 'allValuesFrom', 'someValuesFrom', 'hasValue', 'cardinality', 'minCardinality', and 'maxCardinality'. The 'hasValue' restriction is selected. The 'Filler' field contains the text 'white'. The bottom of the dialog shows a toolbar with various restriction symbols and a green checkmark button.

# Definition of axioms with OWL/Protege

## Restrictions on cardinalities

The screenshot displays the Protege software interface with the 'Create Restriction' dialog box open. The dialog box is titled 'Create Restriction' and contains the following sections:

- Restricted Property:** A list of properties including 'HasChild', 'HasColor', 'HasName', and 'To\_marry'. 'HasChild' is selected.
- Restriction:** A list of restriction types including 'allValuesFrom', 'someValuesFrom', 'hasValue', 'cardinality', 'minCardinality', and 'maxCardinality'. 'minCardinality' is selected.
- Filler:** A text field containing the value '3'.
- Buttons:** A row of icons for various logical operations and a green checkmark button.

In the background, the 'CLASS EDITOR for HavingLargeFamily' is visible, showing the 'Property' tab with 'rdfs:comment' and a list of restrictions. The 'Asserted Hierarchy' pane on the left shows the ontology structure, including classes like 'Human', 'Man', 'WhiteMAN', 'Woman', 'HavingDaughter\_or\_Nochild', 'HavingatleastDaughter', 'HavingLargeFamily', 'Vegetable', and 'Animal'.