



Ministry of Higher Education and Scientific Research  
Dr. Tahar Moulay University of Saida  
School of Science

Computer Science Department

**Textual knowledge engineering.**

Author :

**Dr. Mebarka YAHLALI**

**Associate Professor in Computer Science**

**October 2024**

---

## Preface

*This course is intended to second year Master MICR<sup>1</sup> students of University of Saida "Dr Tahar Moulay". Textual knowledge engineering aims to transform unstructured textual information into structured, actionable knowledge that is easily accessed, utilized, and maintained.*

*This handout is organized into four chapters: Basic Concepts, Descriptive Logic, OWL and Mapping and alignment of ontologies. We have started the course by presenting the main basic concepts in the context of the subject (data, Information, Knowledge, Knowledge graphs and ontologies....). The second part is dedicated to the Descriptive Logic (DL). This method is a formalism used in knowledge representation and reasoning. Its primary goal is to provide a structured way to represent knowledge about the world in a manner that enables efficient reasoning and inference. Chapter three presents the essentials of Web Ontology Language (OWL). In the last chapter, we study Mapping and alignment of ontologies.*

---

<sup>1</sup> MICR: *Modélisation Informatique des Connaissances et du Raisonnement*

---

# Table of Contents

**Preface**

**Table of Contents**

**Figure List**

**General Introduction .....6**

## **Chapter I : Basic Concepts**

### **Introduction**

I. Data, information and knowledge .....8

II. Knowledge Engineering .....9

III. Model .....9

### **Knowledge graphs and ontologies**

I. Definitions .....11

II. Terminology vs. Ontology .....11

III. Ontology Components .....12

IV. Examples .....15

V. Ontology Lifecycle .....15

VI. Formal representation of ontologies (languages) .....18

VI.1. DefOnto (operational language) : .....18

VI.2. Ontology exchange languages on the Web .....20

## **Chapter II: Descriptive logic**

I. Introduction (Knowledge representation) .....26

II. Semantic networks.....26

III. Description logics (DL) .....27

III.1. DL components : .....27

III.2. Description Level .....27

III.3. Description languages .....29

III.4. Semantics of the AL language .....33

III.5. Description logic properties.....35

## **Chapter III : Web Ontology Language**

I. Introduction .....38

II. Structure of an OWL document (Syntax) .....	35
II.1. Namespace Declaration (header) .....	39
II.2. Header of the Ontology .....	40
II.3. Elements of language.....	40
III. Exercises .....	47
<b>Chapter IV: Mapping and alignment of ontologies.</b>	
I. Introduction .....	49
II. Mapping .....	49
III. Alignment .....	50
IV. Steps in a mapping process .....	51
V. Alignment Approaches .....	52
<b>General Conclusion.....</b>	<b>54</b>
<b>Annex .....</b>	<b>55</b>
<b>Bibliography.....</b>	<b>57</b>

## Figure List

Fig1. Data,Information and Knowledge .....	8
Fig2. Modeling Process. ....	10
Fig 3. Ontology Lifecycle.....	16
Fig4. Design and evolution stage.....	17
Fig5. DefOnto example stage.....	19
Fig 6: HTML vs XML.....	21
Fig7: RdF Statement .....	21
Fig8: RdF Statement -Example-.....	22
Fig9: Structured Value -Example-.....	23
Fig10: RDF Model : Example of Abstract Syntax .....	24
Fig11: Semantic Networks: Example .....	26
Fig 12: Namespace Declaration (header) .....	38
Fig 13: OWL: Elements of language .....	39
Fig 14: Example of alignment of two ontologies.....	50
Fig 15: Mapping Process .....	51

## GENERAL INTRODUCTION

In today's data-driven world, it is increasingly important to be able to harness and utilize information. With the exponential growth of unstructured textual data, such as articles and reports, and social media posts, there is a pressing need for effective techniques to extract, manage, and represent knowledge. This module on textual knowledge engineering aims to provide students with the foundational concepts and practical skills necessary to navigate this complex landscape.

Textual Knowledge Engineering encompasses a diverse range of interdisciplinary methodologies, incorporating disciplines such as natural language processing (NLP), machine learning, information retrieval, and knowledge representation. The objective is to convert raw textual data into structured knowledge that can be easily analyzed and utilized for various applications.

Throughout this module, we will explore key topics including:

**1. Knowledge Representation:**

- ✓ Ontologies and knowledge graphs
- ✓ *Descriptive Logic*
- ✓ *OWL*

**2. Mapping and alignment of ontologies.**

By the end of this module, students will not only have a solid theoretical understanding of Textual Knowledge Engineering, but also have the hands-on experience necessary to apply these concepts to real-world problems.

# **Chapter I : Basic Concepts**

Textual Knowledge Engineering is the field of Knowledge Engineering which is interested in texts, Its objective is to acquire knowledge from texts or to exploit the products and methods of knowledge engineering – and more broadly AI – to interpret texts and access their content.

## I. Data, information and knowledge



**Fig 1: Data, information and knowledge**

### I.1. Data

The concept of data or ‘raw’ data is a collection of text, numbers and symbols with no meaning. [1]

#### Example

- cat, dog, rabbit,...
- 161.2, 175.3, 166.4, 164.7, 169.3

### I.2. Information

Information is data that has meaning.

⇒ **Data + Meaning = Information.** [1]

#### Example

- cat, dog, rabbit is a list of household pets
- 161.2, 175.3, 166.4, 164.7, 169.3 are the heights of 15-year-old students.

### I.3. knowledge

According to experts in the field, knowledge is:

- the result of the understanding of information (Hayes, 1992)



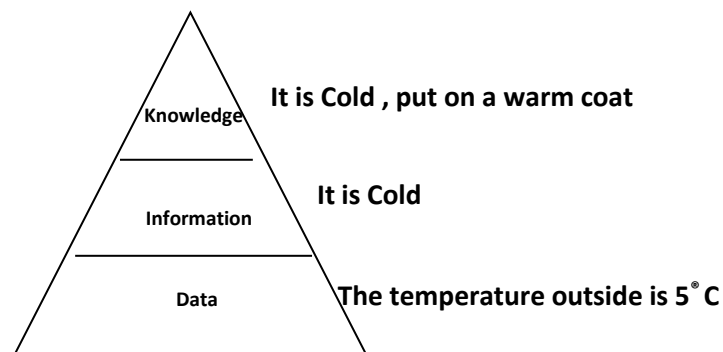
- the result of internalising information (Hayes, 1992)
  - collected information about an area of concern (Senn, 1990)
- information with direction or intent—it facilitates a decision or an action

**Information + application or use = Knowledge**

**Example1 :**

- ⇒ The tallest student is 175.3cm.
- ⇒ A lion is not a household pet as it is not in the list and it lives in the wild.

**Example 2:**



## II. Knowledge Engineering

- Knowledge engineering is a field of artificial intelligence (AI).
- In computer science, knowledge engineering refers to techniques for manipulating knowledge on a computer.
- Integration of artificial intelligence techniques and software engineering to design and build expert systems. [2]

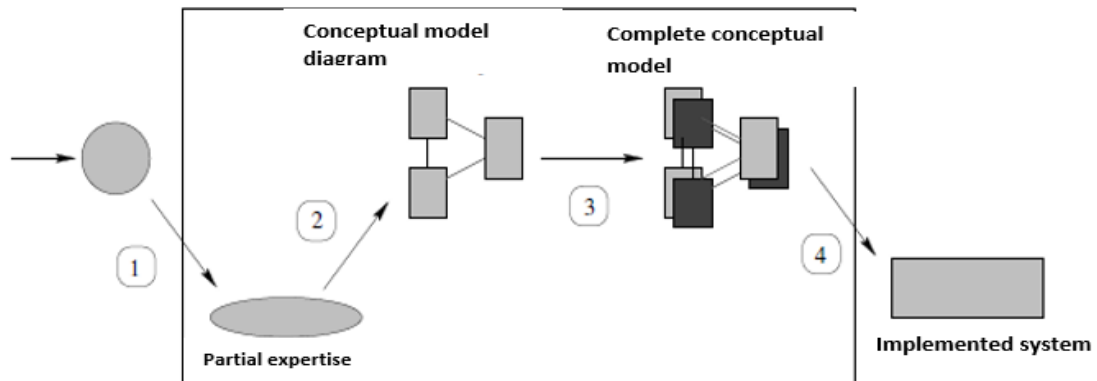
## III. Model

### III.1. What is a model?

A model is an abstraction that reduces complexity by focusing on certain aspects, according to certain goals. [4]

- ⇒ **BUT a model should manipulate objects and interpret the results.**

### III.2. The modeling process



**Fig 2: Modeling Process[4]**

- **Phase 1: Data-Driven Acquisition** : Corpus analysis tools are used to collect raw data (primary data)
- **Phases 2 and 3 : (2)Construction of the conceptual model diagram and (3) Instantiation of the conceptual model diagram** : Phases 2 and 3 contribute to the development of the conceptual model. Phase 2 develops the diagrams of this model phase 3 extends these diagrams with data from the application domain (refine) .
- **Phase 4 : Operationalization of the model ( implementation )** : corresponds to the choice of a language for the implementation of the final system

## I. Definitions

- The word ontology comes from the Greek :
  - a. **ontos** : means being, what exists, the existing
  - b. **logos**: means speech
- The word “ontology” is used with different senses in different communities. Ontology, as a branch of philosophy, is the nature of being and existence . In Computer Science, we refer to an ontology as a special kind of information, object or computational artifact.  
⇒ In simple terms, ontology seeks the classification and explanation of entities.  
It is a discourse on what is, what exists. According to Willard Van Orman Quine, ontology answers the question: What exists? how is the World constituted? [2]
- Gruber originally defined the notion of an ontology as an “*explicit specification of a conceptualization*” [4] . Borst in [5] defined an ontology as a “*formal specification of a shared conceptualization*”
  - a. The expression “**explicit specification**” means: representation in a language (natural language or formal language).
  - b. The term “**conceptualization**” refers to a system of concepts (entities, attributes, processes, their definitions and their interrelations) [3]
- An ontology defines the basic terms and relationships of a domain's vocabulary as well as the rules that indicate how to combine terms and relationships in order to be able to expand the vocabulary [1].

## II. Terminology vs. Ontology

### Terminology:

- Set of terms representing a system of concepts for a particular domain.
- Association of linguistic information to terms (linguistic entities).
- No formal organization of terms (redundancy problems)

### Ontology:

- Formal description of concepts and relationships for a particular domain.
- Association of properties to concepts
- Intended for automatic processing =>the ontology must be readable by a machine, which excludes natural language.

## **Two complementary resources => Terminology can be used to populate an ontology**

### **III. Ontology Components**

The formalization of Knowledge in ontologies is based on five components: classes, instances, relations, functions, axioms , instances and Attributes

**III.1. Concepts (classes):** are relevant abstractions of a segment of reality [1]. A Class is a set, entity, collection, or type of objects sharing common characteristics.

The concept can be defined as an entity composed of three distinct elements [5]:

- 1. The term(s):** is a symbolic representation, often linguistic, expressing the concept (label).  
**e.g:** Person, Human,..... Car, Auto....
- 2. The meaning of the concept:** also called “notion” or “intension” of the concept. → Attributes  
**e.g:** Person (Name, Age, gender ...)
- 3. The object(s) denoted by the concept,** also called “realization” or “extension” of the concept. They are the real and concrete basic objects. These are the leaves in the ontology diagram.  
**e.g (Mohamed , 25, male )**

Two classes of concepts can be distinguished

- a. Primitive concept:** Human, Male, Female
- b. Defined concept :**
  - Man: Human and Male
  - Woman: Human and Female

**III.2. Relations (object property):** the relations are the links or interactions that a class or a class instance can have with other classes or class instances in the targeted domain [6].

Formally a relation R is defined as a subset of a Cartesian product of n sets:

$$\mathbf{R: C_1 \times C_2 \times \dots \times C_n}$$

### III.2.1. Relationship types

a. **The specialization relationship** : (subsumption, generalization, subclass), \*means inheritance (is-a, is-a); (called **Hyponymy** (upper element, ) /**Hyponymy** (lower element)).

This relationship allows the inheritance of properties (hierarchical links).

e.g:

- “father” is a “person”  
“person” is the hyperonym “father”

b. **The relation of composition** (meronymy), (aggregation or composition) (is-a-part-of); are semantic links.

eg:

- A Branch is a part of a Tree.
- A Leaf is a part of a Branch

c. **Association** ; instance-of....

### III.2.1. Characteristics of relationships:

#### 1. Algebraic Characteristics:

- **Symmetric:** a relation R is symmetric if for all  $x, y \in E / x R y$  if  $y R x$ .

E.g: Brother\_of

- **Transitive:** a relation R is transitive if for all  $x, y, z \in E /$  if  $x R y \wedge y R z \Rightarrow x R z$ .

e.g : Parent\_of

If Mohamed **parent\_of** Karim  $\wedge$  Karim **parent\_of** Ali  
 $\Rightarrow$  Mohamed **parent\_of** Ali

- **Inverse:**

if  $R :: x R y / x \in E1 \wedge y \in E2$  the inverse relation  
 $R^{-1} :: y R x / y \in E2 \wedge x \in E1$

e.g: mother\_of and son\_of (son\_of = mother\_of<sup>-1</sup>)

#### 2. Cardinality:

- **minCardinality**: any instance of the class will be connected by the property :**has at least x individuals**  
**e.g:** A teacher has at least one diploma: **minCardinality=1**
- **maxCardinality**: any instance of the class will be connected by the property :**has at most x individuals**  
**e.g:**
  - A mother has at most 10 children: **maxCardinality=10**
  - By Muslim law, men can have, at the most, 4 wives in total: **maxCardinality=10**
  - **cardinality**: any instance of the class will be linked by the property : **has exactly x individuals**  
**e.g:** A child with exactly one and only one mother: **cardinality=1**

**III.3. Functions** :Are special cases of relations in which the nth element of the relation is uniquely defined from the previous n-1 elements (The nth element of the relation is unique for the preceding n – 1 elements.)

$$F: C_1 \times C_2 \dots \times C_{n-1} \rightarrow C_n$$

**e.g:** Ternary function:

Price of a used car: **Model x Years x Kilometers --> Price**

**III.4.Axiom: [rules]:** Set of axioms or inference rules allowing the definition of properties of relationships. They are assertions accepted as true for:

- Define the components signification.
- Set restrictions on attribute values.
- Define the arguments of a relationship.
- Check the validity of specified information or derive new information.

**III.5.Instances:** (individuals or extension of the concept) : are the real and concrete basic objects. The instances are the leaves in the ontology diagram.

**III.6.Attribute: (intension/notion or meaning):** or Data properties. The attributes are the set of characteristics or parameters that describe the concepts (instances).

**e.g:** Age, last name, first name for the concept person

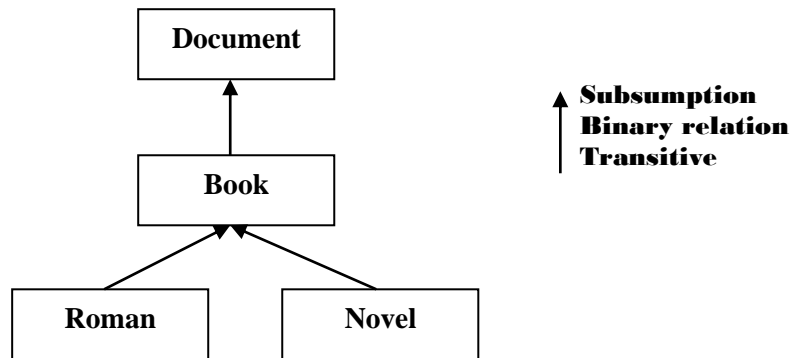
#### IV. Examples :

##### Informal :

“A novel and a Roman are books.”

“A book is a document.”

##### Formal



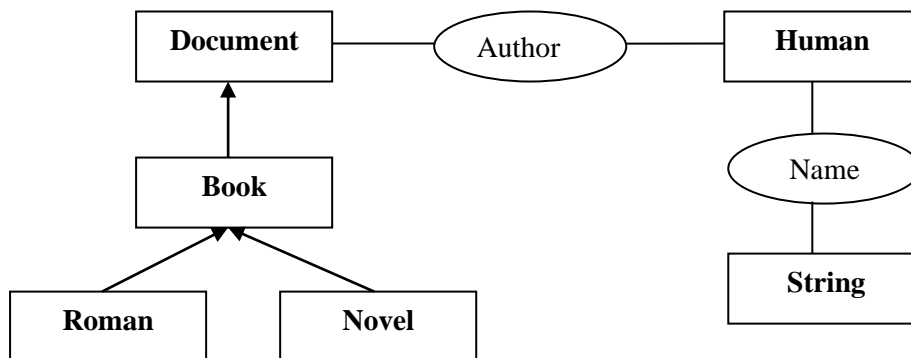
##### Informal :

“A document has an author.”

“An author is human”

A human has name ... name is String

##### Formal



#### V. Ontology Lifecycle

The ontology lifecycle identifies the different stage to develop an ontology. Several ontology construction methods have been proposed. These methods share the following main steps [34]:

- Identification of the ontology domain and its scope of application;
- Definition of the expected objectives of the ontology;
- Informal specification of concepts;
- Coding the ontology by formally representing its components (concepts, relation , axioms....)
- Evaluation of the ontology....

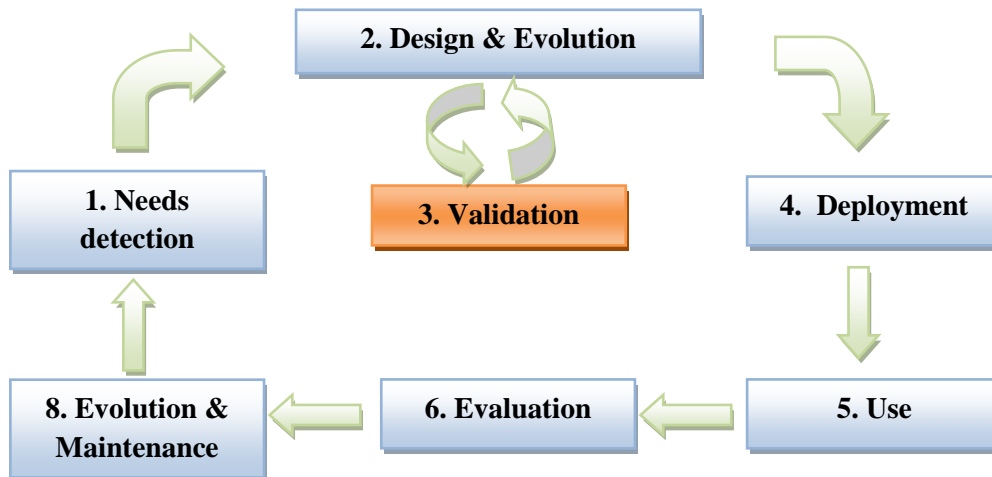
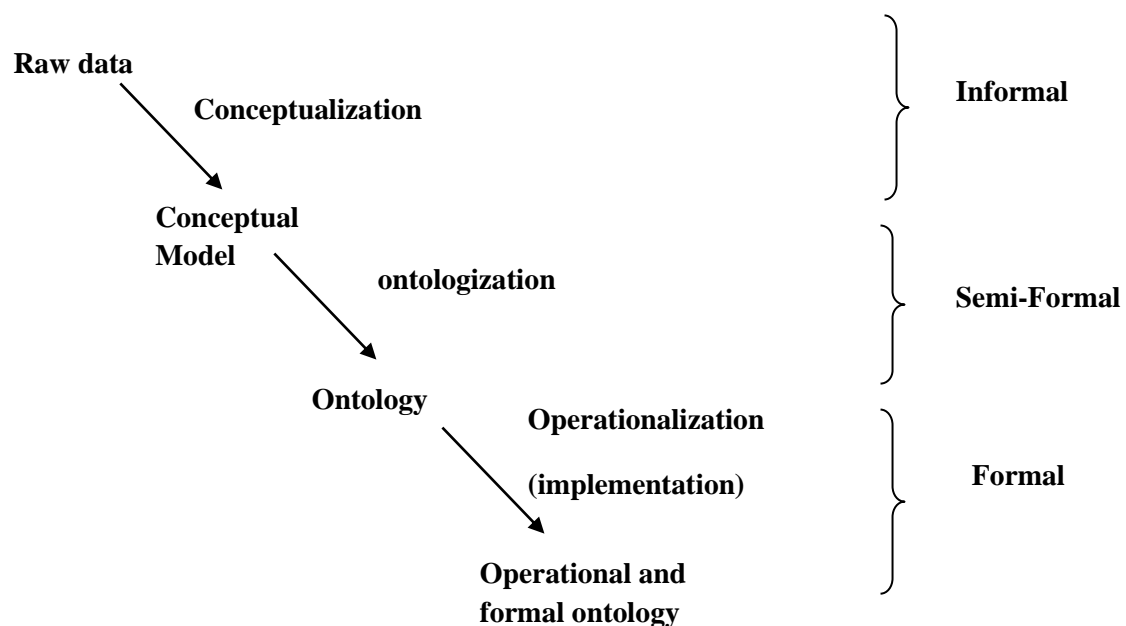


Fig 3. Ontology Lifecycle [6]

1. **The needs detection (specification)** : needs detection phase starts with a detailed inventory of the domain and the various purposes:
  - ✓ **The operational objective:** it is important to determine the operational use of the ontology through use case diagrams; (scenario)
  - ✓ **Domain of knowledge**
  - ✓ **Users**
2. **Design and evolution:** Consists of identifying from raw corpus data all concepts and the relationships between them.





**Fig4. Design and evolution stage [6]**

The design and development phases share a certain number of points:

- ✓ **Conceptualization and modeling:** sort the knowledge found in the specific corpus (Select specific terms of the domain). Then it is necessary to specify the concepts, the relationships, the properties of the concepts and relationships, the rules and constraints, ... → conceptual aspect.
- ✓ **Formalization (Ontologization and Operationalization):** Sometimes called representation, is a transcription of the ontology into a formal and operational knowledge representation language (description logics, conceptual graphs, semantic web formalisms RDF, RDF(S) and OWL) .

### 3. Validation (Documentation and evaluation)

- **Documentation:** Produce formal, informal and complete definitions to clarify the meaning of ontology terms.
- **Evaluation:** performed by tests corresponding to the operational objective of the ontology.

**4. The deployment, diffusion and use phase:** the ontology is distributed and it is put at the service of users.

**5. Evolution and Maintenance:**

The ontology must evolve if:

- ✓ It does not exactly meet the needs expressed at the start of the project (The evaluation of the ontology failed:) → It will be necessary to modify certain parts of the ontology.
  
- ✓ Over time the context of use is modified, or the domain of knowledge is expanded => needs change automatically. It is important to :
  - a) Build a new ontology with the knowledge to add and integrate it into the ontology already established
  - b) Or directly aggregate new knowledge into the existing ontology

**VI. Formal representation of ontologies (languages)**

To construct an ontology, several languages can be used. They are adapted to the different stages:

1. Natural language or informal modeling language to acquire ontological knowledge,
2. Or formal and executable representation languages. Formal languages offer data structures adapted to the representation of concepts. Among these languages, we distinguish:
  - Operational languages which implement ontologies for inference purposes, to constitute a component of an information system.
  - Languages for exchanging ontologies on the Web, whose syntax is based on XML.

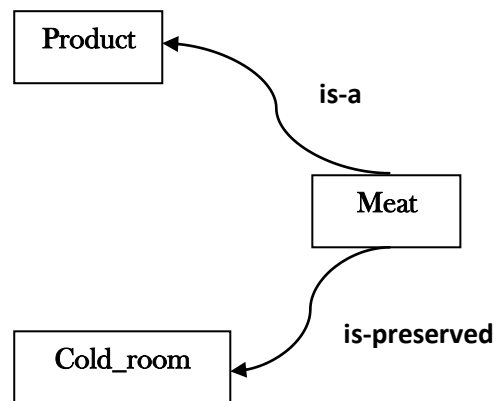
**VI.1. DefOnto (operational language) :**

DefOnto allows the representation of meta-knowledge (ex: properties of properties), an objective of DefOnto is to provide the language with modular compilation mechanisms to facilitate the development and maintenance of formal ontologies. Each type of conceptual entity corresponds to a particular language construction, defined using one of the following "constructors" [7] :

- **DefGenConcept** : to define generic concepts. The generic concept represents an undefined object.
- **DefRelation** to define relationships;
- **DefIndConcept** : to define individual concepts. The individual concept represents a specific object. In natural language, it corresponds to a proper noun or a common noun preceded by a definite article. For example, Algeria is represented by the concept [COUNTRY: Algeria].

```

DefGenConcept Product
DefGenConcept Meat
is-a Product
DefRelation is_preserved
RelationProperties
domain Meat
range Cold_room
    
```



**Example :**

« *Electronic document: An ELECTRONIC DOCUMENT is a DOCUMENT which HAS A SUPPORT electronic. Every ELECTRONIC DOCUMENT HAS FOR FORMAT a FORMAT. The ELECTRONIC DOCUMENTS are opposed to PAPER DOCUMENTS* »

```

DefGenConcept #document
    =[#object]
DefGenConcept #paper_document
    = [#document]
DefGenConcept #electronic_document
    = [#document] -> (MI#has_a_support) ->“électronic”
ObjectProperties
    ->(AE#has_for_format)->[#format]
SetProperties
    ->(#is_disjoint_with)->[#paper_document]
    
```

**Fig5. DefOnto example [7]**

**Remarks :**

- The letters 'A' (=ALL) and 'E' (Exist), preceding the name of the relationship possesses,
- The letter 'I' (I = Individual) indicates the presence of a constant.
- The letter 'M' represents a pseudo-quantifier. The M introduces a relative and reads "who..." or "whose..." or even "having...".


**VI.2. Ontology exchange languages on the Web**

**VI.2.1. XML :**


**What is XML?**

- XML stands for eXtensible Markup Language
- XML is a markup language much like HTML
- XML was designed to store and transport data
- XML is a W3C Recommendation
- All major browsers have a built-in XML parser to access and manipulate XML.[8]

*The Adventures of Tom Sawyer*



Front piece of *The Adventures of Tom Sawyer*

<b>Author</b>	Mark Twain
<b>Cover artist</b>	created by <a href="#">Mark Twain</a>
<b>Country</b>	United States
<b>Language</b>	English, Limited Edition(Spanish)
<b>Genre</b>	Bildungsroman, picaresque, satire, folk, children's novel
<b>Publisher</b>	American Publishing Company
<b>Publication date</b>	1876 <sup>[1]</sup>
<b>OCLC</b>	47052486 
<b>Dewey Decimal</b>	Fic. 22
<b>LC Class</b>	PZ7.T88 Ad 2001
<b>Followed by</b>	<i>Adventures of Huckleberry Finn</i>

**HTML: focus on presentation**

```
<h2>The adventures of Tom Sawyer</h2>
...
<b>Author: </b> Mark Twain <br>
<b>Cover artist: <b> created by <a href="http://...">Mark Twain </a>
...
```

**XML: focus on metadata**

```
<book>
  <title> The adventures of Tom Sawyer </title>
  <author> Mark Twain </author>
  <genre> Bildungsroman </genre>
  <genre> picaresque </genre>
  ...
  <publisher> American Publishing Company </publisher>
  <year>1876</year>
</book>
```

Fig 6: HTML vs XML

## VI.2.2. Resource Description Framework (RDF)

- A language for representing Web resources and information about them in the form of metadata<sup>2</sup> [9]

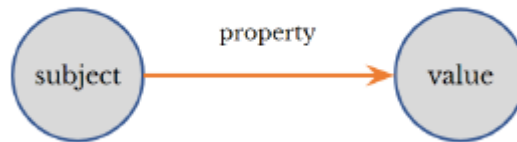
### What is a resource?

- A resource is an object or anything that has identity
- For example, : electronic document, Authors, books, publishers, an image, a service (e.g., "today's weather report for Algiers "), and a collection of other resources.
- All resources are identified by a URI (Uniform Resource Identifier).
- Resources are described in terms of simple statements specifying properties and property values.

<sup>2</sup> *Metadata* is defined as the information that describes and explains data

## RDF Data Model (Abstract Syntax)

A statement is a triplet of : subject- predicate –value



**Fig7: RdF Statement**

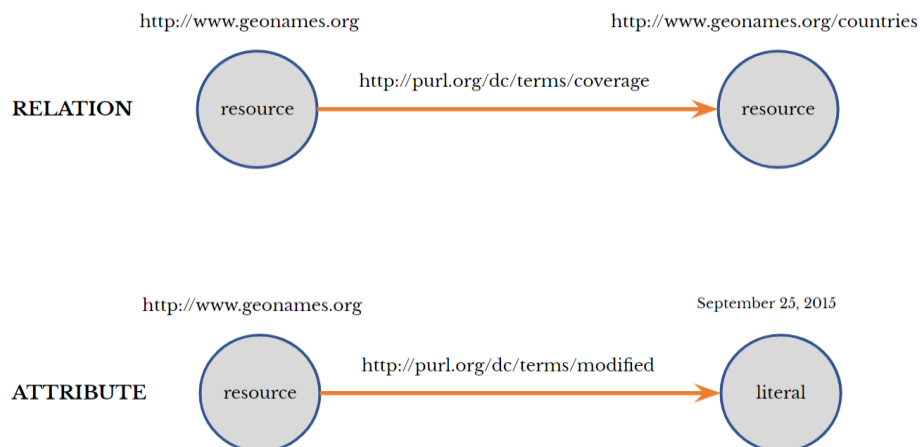
1. **A subject** : a “thing” we want to talk about.
2. **A predicate ( or property about the subject)**

Special kind of resources:

- Describe relations between resources, for example “written by”, “age”, “title”, and so on.
- Are also identified by URIs (and in practice by URLs).
- 

3. **An object (the value of predicate)**

- Values can be resources (for relations) or literals (for attributes)



**Fig8: RdF Statement -Example-**

### Example :

- Statement: "The author of <http://www.w3schools.com/rdf> is Refsnes ".  
The **subject** of the statement is: **<http://www.w3schools.com/rdf>**

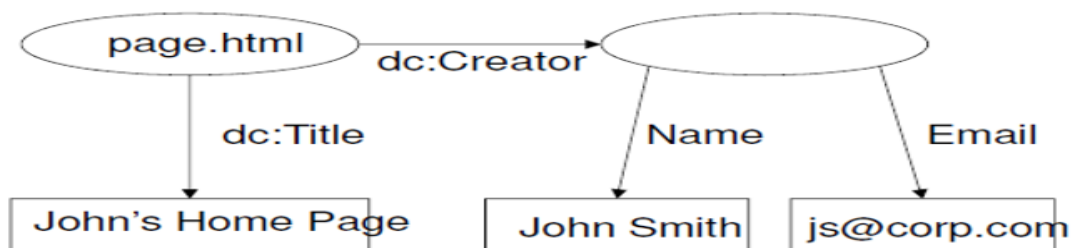
The **predicate** (or property) is: **author**  
The **object** is: **Refsnes**.

- Statement: "The homepage of <http://www.w3schools.com/rdf> is <http://www.w3schools.com>".

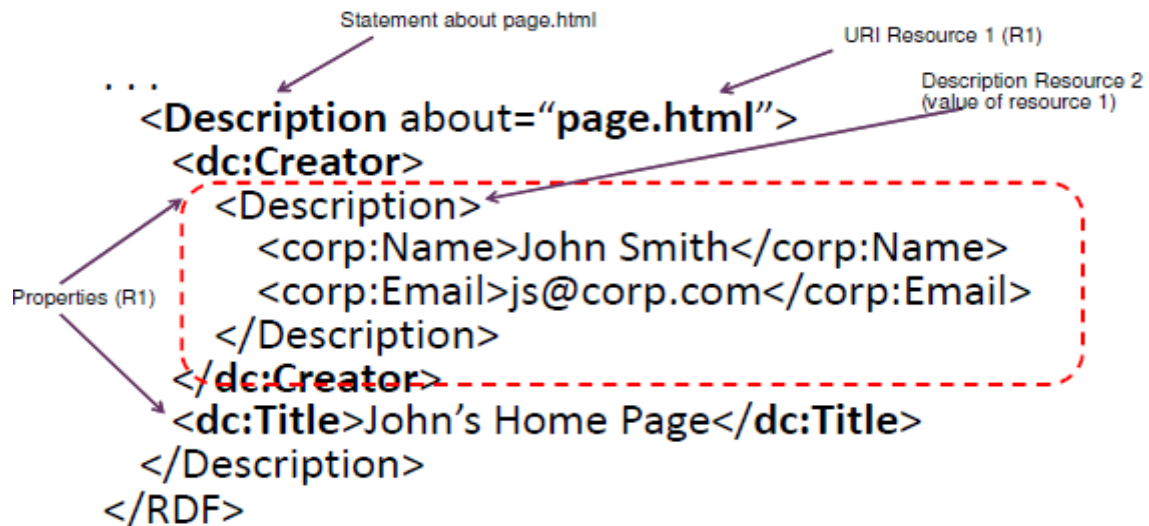
The **subject** of the statement above is: **<http://www.w3schools.com/rdf>**  
The **predicate** is: **homepage**  
The **object** is: **<http://www.w3schools.com>**

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf=" http://www.w3.org/1999/02/22-rdf-syntax-ns"
xmlns:dc="http://purl.org/dc/terms#">

<rdf:Description about=" http://www.w3schools.com/rdf">
<dc:Creator> Refsnes </dc:Creator>
<dc: homepage rdf:resource = "http://www.w3schools.com" />
</rdf:Description>
</rdf:RDF>
```



**Fig9: Structured Value -Example**



**Fig10: RDF Model : Example of Abstract Syntax**

### Exercise

Write an RDF model representing the following statements:

- Document 1 has been created by Mohamed
- Document 2 and document 3 have been created by the same author (who is unknown)
- Document 3 says that document 1 has been published by W3C

Use predicates `dc:Creator` and `dc:Publisher`, and assume that the three documents are represented by URIs `doc1`, `doc2`, `doc3`, respectively.



# Chapter II :

# Descriptive Logic

## I. Introduction (Knowledge representation)

Knowledge is considered as a collection of information appropriate and relating to a particular subject. In the field of AI the main objective of knowledge representation is to store knowledge so that programs can process and carry out calculations. i.e the knowledge representation model should facilitate the reasoning process.

Knowledge representation approaches can be classified into two categories:

1. **Non-logical approach:** The meaning is decided by users' and programmers' intuition. Therefore, different knowledge representation systems based on the same model could have different interpretations. Example: semantic networks.
2. **Logical approach:** They have surpassed intuition in non-logical representations. They clearly use predicate calculations to capture facts about the world. [10]

**II. Semantic networks:** Knowledge is represented in the form of a labeled directed graph. Vertices denote concepts and objects, and arcs represent the various relationships between concepts.

There are two types of arcs : [10]

1. is-a arcs which introduce hierarchical relationships between concepts and instance relationships between objects and concepts.  
e.g :
  - Teacher is-a Person (hierarchical relationships)
  - Amina is-a Teacher (instance relationships)
2. Property arcs that assign properties to concepts and objects.

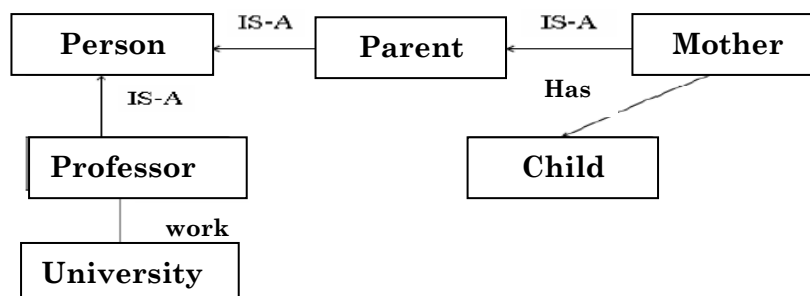


Fig 11 . Semantic networks : Example

Semantic networks do not have formal semantics. For example: the property arc "work" from "PROFESSOR" to "UNIVERSITY" may indicate that the university is the only establishment for professors, furthermore, it may mean that a professor works in at least one university and also in other establishments.

→ **Evolution: logical definition**

### III. Description logics (DL) :

Description logics are a family of knowledge representation languages. DLs are used to represent knowledge of an application domain in a formal and unambiguous way. [11]

#### III.1. DL components [11]:

- **Atomic concepts (unary predicates):** classes of objects or set of properties **Examples:** Woman, Human, Mother!
- **individuals or objects (constants):** identifiable by a name **Examples:** Amina, Mohamed, Ali!
- **Atomic roles (binary predicates) :** relationships between objects examples: hasChild, hasHusband!
- **Constructors:**

$\neg, \sqcap, \sqcup, \sqsubseteq, \exists, \forall$

#### III.2. Description Level

In a description logic knowledge base, there are two components:

- **TBox (Terminological Box):** contains all the axioms defining the concepts of the domain.
- **ABox (Assertional Box):** contains assertions about individuals: specification of their class and their attributes. [12]

**III.2.1.TBox:** Defines concepts (classes) and roles (relationships). The TBox contains: [11][12]

1. **Atomic Entities:** atomic concepts and atomic roles constituting the elementary entities of the LD.
2. **Minimal predefined atomic concepts and roles:**
  - The concept  $\top$  and the role  $\top R$ : ( $\top$ : universal TOP): root of all concepts.  
By definition, for any concept C, we have the following axiom:

$$C \sqsubseteq \top$$

- The concept  $\perp$  and the role  $\perp R$ : The impossible concept: the most specific (empty set).

**3. Composite Entities:** atomic concepts and roles can be combined using constructors to form composite entities

**Note:** Some conventions

- A and B denote atomic concepts
- C and D denote composite or atomic concepts
- R denotes a role 4.
- The names of concepts begin with a capital letter: **Ex: Man, Woman,**
- Role names with a lowercase letter: **Ex: parent, child, etc**

### III.2.2.ABox:

- Describes individuals in terms of concepts and relationships.
- Provides a description of the world: The ABox introduces individuals by specifying their names, the concepts they belong to, and their relations with other individuals. i.e. Describe a concrete situation by stating properties of individuals.
- Corresponds to ***Data*** (ground facts) in database settings

An ABox contains 2 types of assertions about individuals:

**1. Element-of (belongs-to) assertions:** specify the class and attributes of individuals:

ex: Meriem is a woman,

Meriem is a Mother ((individual/instance of the mother class))

**2. role assertions:** specify the existing relationships between individuals (real relationships)

**Example:** a mother must have at least one child: the ABox must contain at least one other individual, and a relationship between this one and Meriem indicating that he is one of her children.

**III.3. Description languages:** There are several concepts and roles description languages. The minimal language called AL (Attributive Language), which is gradually enriched with new constructors. [13]

A	Atomic Concept
$\top$	Universal Concept
$\perp$	Impossible Concept
$\neg A$	Atomic Negation
$C \sqcap D$	Concept Intersection
$\forall R.C$	Value Restriction
$\exists R. \top$	Limited existential qualification

**Table 1: AL Syntax[12]**

- $\exists R.C$ : individuals who have at least one type R relationship with a type C individual.
- $\forall R.C$ : individuals whose all type R relationships are with type C individuals.

**Example :**

**Assuming**

**Set C = {a,b,c}**

**Relationship R {(a,b) (a,d) ,(b,c) }**

$\exists R.C = \{a,b\}$
$\forall R.C \{b\}$

**III.3.1 Description logic families (some Extensions)**

The AL language can be enriched with constructors. Among these constructors we can note:

- The negation of primitive or defined concepts (complement), which is noted (not C) or  $\neg C$ . The corresponding extension of AL is  $ALC^3 = AL \sqcup \{\neg C\}$ .
- The disjunction of concepts, which is noted (C or D) or  $(C \sqcup D)$ . The corresponding extension of AL is  $ALU = AL \sqcup \{C \sqcup D\}$ .
- The cardinality on the roles, which is noted (atleast n r) or  $\geq n r$ , and (atmost n r) or  $\leq n r$ . The corresponding extension of AL is  $ALN = AL \sqcup \{\geq n r, \leq n r\}$ .

The constructors  $\leq n r$  and  $\geq n r$  set the cardinality => minimum and maximum number of elementary values of the role with which they are associated. In particular, construction  $(\exists r)$  is equivalent to construction  $(\geq 1 r)$ .

---

<sup>3</sup> Attributive concept Language with Complements

### III.3.2. Description of concepts and roles (TBOX)

Axioms are constructed from a set of concepts. Two possible formulas for the axioms:  $C \sqsubseteq D$  or  $C \equiv D$ .

1.  $C \sqsubseteq D$ : stating (express) inclusion relations (C is included in D): declares that any entity of class C also belongs to class D,
2.  $C \equiv D$ : stating equivalence (definition) relations between concepts: C equivalent by definition to D. A definition is an axiom in the form of  $C \equiv D$  where C is an atomic concept. It actually serves to associate a name with a complex concept.

**Ex: some statement (declaration)**

- Humain is an animal,

$$\mathbf{Humain \sqsubseteq Animal}$$

- A human is a reasoning animal

$$\mathbf{Humain \equiv Animal \sqcap Reasonable}$$

3. On DL we can define a concept by restrictions on roles (relationships)

**Ex :**

- Person with at least one child:  $\exists a \text{Child.Human}$
- The class of individuals whose children are all women:  $\forall a \text{Child.Woman}$

4. Number restrictions are used to restrict the number of successors in the given role for the given concept.

$$\mathcal{I}(\geq n R) = \{a \in \Delta \mid |\{b \mid (a, b) \in \mathcal{I}(R)\}| \geq n\}$$

$$\mathcal{I}(\leq n R) = \{a \in \Delta \mid |\{b \mid (a, b) \in \mathcal{I}(R)\}| \leq n\}$$

Woman who has at least 3 sons :  $\mathbf{Womthreesun \equiv Woman \sqcap (\geq 3 \text{hasChildMan})}$

Concept of father who has exactly 2 children:  $\mathbf{Homme \sqcap \geq 2 aEnfant \sqcap \leq 2 aEnfant}$

### III.3.3. Declaration of individuals in a terminology

- In certain situations, we would like to be able to designate individuals in the terminology (enumeration).

**Setname**  $\equiv \{a_1, a_2, a_3, \dots, a_n\}$

**Example :** Traffic Lights Colors

**TrafficLC**  $\equiv \{\text{Red, Green, Yellow}\}$

- To describe the set of individuals who are connected to a specific individual (a) by a relationship R

**R : a**

**Example :** to represent the concept of Algerian Citizen, which is defined as follows: **person born in Algeria**

**AlgerianCitizen**  $\equiv \text{born: Algeria}$

### III.3.4. Description of the world (ABOX)

1. **C(a)** to indicate that a is an individual belonging to the set denoted by the concept C.

**Example:** Mohammed is a Student: Student (Mohammed)

2. **R(b; c)** to indicate that b and c are linked by the relation R or, in other words, that individual c fulfills the role R for individual b.

**Example:** aChild(Amina, Ali) to indicate that Ali is Amina's son

**Example**

**TBOX**

Woman  $\sqsubseteq$  Person

Mother  $\sqsubseteq$  Woman  $\sqcap \exists \text{hasChild. Person}$

**ABOX**

hasChild(Mohamed, Amina)

Woman(Amina)

Person(Mohamed)

**Exercise .**

1. **Men and women are human**

Men  $\sqsubseteq$  Humain

Women  $\sqsubseteq$  Humain

2. **Amina is a Woman**

Woman (Amina)

3. **Person who does not have children: we restrict the value of the relationship hasChild to the impossible concept:**

$$\forall \text{hasChild}. \perp$$

4. **The mother is a woman who has children**

$$\text{Mother} \equiv \text{Woman} \sqcap \exists \text{hasChild}. \text{Person}$$

5. **The father is a man who has children**

$$\text{Father} \equiv \text{Man} \sqcap \exists \text{hasChild}. \text{Person}$$

6. **Parent?**

$$\text{Parent} \equiv \text{Father} \sqcup \text{Mother}$$

$$\text{Grandmère} \equiv \text{Mère} \sqcap \exists a \text{Enfant}. \text{Parent}$$

$$\text{MèreDeFamilleNombreuse} \equiv \text{Mère} \sqcap \geq 3 a \text{Enfant}$$

$$\text{MèreSansFille} \equiv \text{Mère} \sqcap \forall a \text{Enfant}. \neg \text{Femme}$$

$$\text{Épouse} \equiv \text{Femme} \sqcap \exists \text{mariéAvec}. \text{Homme}$$

$$\text{Célibataire} \equiv \text{Personne} \sqcap \forall \text{mariéAvec}. \perp$$

7. **The class of people who do not have male children**

$$\neg \exists \text{hasChild}. \text{Men}$$

8. **Grand Parent ??**

$$\text{GrandParent} \equiv \text{Person} \sqcap \exists \text{hasChild} . \exists \text{hasChild} . \top$$

9. **Amina has at most two children**

$$(\leq 2 \text{ hasChild})(\text{Amina})$$

10. **Mohamed has only one son Karim**

$$\text{hasChild}(\text{Mohamed}, \text{Karim})$$

$$(\leq 1 \text{ hasChild})(\text{Mohamed})$$

### III.4. Semantics of the AL language

The semantics of the AL language (and DLs in general) is expressed in terms of interpretation. It uses set theory: each concept is associated with a set of individuals denoted by this concept. [14]

In order to define the semantics of concept terms, we consider interpretations  $I = (\Delta^I, I)$ , which consist of:



1. A non-empty set  $\Delta^I$  (the interpretation domain).
2. An interpretation function  $I$  assigning:
  - for each atomic concept  $A$ , a set  $A^I$ , such that  $A^I \subseteq \Delta^I$
  - for each atomic role  $R$ , a binary relation  $R^I$ , such that  $R^I \subseteq \Delta^I \times \Delta^I$

Here is how this interpretation function is defined for the other possible descriptions:

$$\begin{aligned}
 \mathcal{I}(\top) &= \Delta \\
 \mathcal{I}(\perp) &= \emptyset \\
 \mathcal{I}(\neg A) &= \Delta \setminus A^I \\
 \mathcal{I}(C \sqcap D) &= \mathcal{I}(C) \cap \mathcal{I}(D) \\
 \mathcal{I}(\forall R.C) &= \{a \in \Delta \mid \forall b. (a, b) \in \mathcal{I}(R) \rightarrow b \in \mathcal{I}(C)\} \\
 \mathcal{I}(\exists R.\top) &= \{a \in \Delta \mid \exists b. (a, b) \in \mathcal{I}(R)\}
 \end{aligned}$$

- **Ensemble**  $\equiv \{a_1, a_2, a_3, \dots, a_n\}$   
 $\mathcal{I}(\{a_1, a_2, a_3, \dots, a_n\}) = \{I(a_1), I(a_2), I(a_3), \dots, I(a_n)\}$

### Example :

Let the domain interpretation  $\Delta$

$$\Delta = \{\text{Mohamed, Amine; Ibrahim; Fatima; Sara; hp, Dell}\}$$

### Example of interpretations:

- $I(\text{Person}) = \{\text{mohamed; amine; ibrahim; fatima; sara}\}$
- $I(\text{Man}) = \{\text{mohamed; amine; Ibrahim}\}$
- $I(\text{hasChild}) = \{(\text{mohamed; amine}); (\text{mohamed; sara}), (\text{fatima; sara}); (\text{amine; ibrahim})\}$
- $I(\exists \text{hasChild. Person}) = \{\text{mohammed; amine; Fatima}\}$
- $I(\forall \text{hasChild. Man}) = \{\text{amine}\}$

**Equivalence of concepts:** Two concepts  $C$  and  $D$  are equivalent, denoted  $C \equiv D$ , if we have  $I(C) = I(D)$ , By definition we have the following equivalences:

- $\top \equiv \neg \perp$
- $\neg \top \equiv \perp$
- $C \sqcap \neg C \equiv \perp$
- $C \sqcup \neg C \equiv \top$

**Inclusion of concepts:** The concept C includes the concept D, noted  $C \sqsubseteq D$ , iff we have  $I(C) \sqsubseteq I(D)$ , By definition, for any concept C we have:  $C \sqsubseteq T$

**Exercise :**

Let the following definitions:

$\Delta = \{a, b, c, d, e, f, g\}$

$I(\text{man}) = \{a, b, c, g\}$

$I(\text{hasChild}) = \{(a, c), (b, d), (b, e), (c, g)\}$

$I(\text{marryWith}) = \{(b, f), (f, b)\}$

$\text{Parent} \equiv \exists \text{hasChild}. \top$

$\text{ParentOfWoman} \equiv \exists \text{hasChild}. T \sqcap \forall \text{hasChild}. \neg \text{Man}$

$\text{Single} \equiv \forall \text{marrywith}. \perp$

$\text{Marriedman} \equiv \text{Man} \sqcap \exists \text{marrywith}. T$

Specify an interpretation for each of the classes defined above?

$\text{Parent} : \{a, b, c\}$   
 $\text{ParentDeFemme} : \{b\}$   
 $\text{OWLCélibataire} : \{a, c, d, e, g\}$   
 $\text{HommeMarié} : \{b\}$   
 $\text{GrandParentChoyé} : \{a, d, e, f, g\}$

### III.5. Description logic properties

In description logic, there are four properties to prove for a Tbox T:

1. **Satisfaction:** a concept is satisfiable if there exists at least one entity of the described world which can belong to the set described by this concept

• **A concept C is satisfiable if there exists an interpretation I such that:  $CI \neq \emptyset$**

**Example:**  $I(\text{Man} \sqcap \neg \text{Man}) = \emptyset \rightarrow$  It is unsatisfiable

2. **Subsumption:** A concept C1 is subsumed by a concept C2 if for any interpretation I:  $I(C1) \sqsubseteq I(C2)$

**Example :**  $I(\text{Mother}) \subseteq I(\text{Woman.})$  Mother is subsumed by Woman.

**3. Equivalence:** Two concepts C and D are if for any interpretation I if  
 $I(C) = I(D)$

**4. Disjunction:** Two concepts C and D are disjointed if for any interpretation I:

$$I(C) \cap I(D) = \emptyset.$$

**Example:**  $I(C) \cap I(D) = \emptyset$  Father and Mother are disjointed

**Exercise:** Consider the following description in descriptive logic:

$$A \equiv \forall R.A \sqcap \exists R.T \sqcap B$$

Say, for each of the following interpretations, whether it satisfies this description (justify!!):

**Interpretation 1:**

- $I(A) = \{a; b; c; d\}$
- $I(B) = \{e; f\}$
- $I(R) = \{(a; b); (b; c); (c; d); (d; a); (e; a); (f; a)\}$

**Interpretation 2 :**

- $I(A) = \{a; b; c\}$
- $I(B) = \{e\}$
- $I(R) = \{(a; a); (a; b); (a; c); (b; a); (b; c); (c; a); (e; e)\}$

# **Chapter III : Web Ontology Language -OWL-**

## I. Introduction

OWL is not a real acronym. "Web Ontology Language" should have used "WOL", but the Working Group did not wish to keep this abbreviation and therefore decided to use OWL. The design of OWL took into account the distributed nature of the Semantic Web and, as such, incorporated the possibility of extending existing ontologies, or of employing various existing ontologies to complete the definition of a new ontology. The more complete a tool is, the more complex it is. OWL has three sub-languages offering increasing expression capabilities and, naturally, intended for different communities of users:

- OWL Lite: is the simplest OWL sublanguage. It is intended for users who need a simple hierarchy of concepts.
- OWL DL: is more complex than OWL Lite. It allows for much greater expressiveness. OWL DL is based on description logic (hence its name, OWL Description Logics). Despite its relative complexity compared to OWL Lite, OWL-DL guarantees the completeness of the reasoning and their decidability.
- OWL Full: is the most complex version of OWL, but also the one that allows the highest level of expressiveness. OWL Full is intended for situations where it is more important to have a high level of description ability.

There is a dependency of a hierarchical nature between these three sublanguages: any valid OWL Lite ontology is also a valid OWL DL ontology, and any valid OWL DL ontology is also a valid OWL Full ontology.

## II. Structure of an OWL document (Syntax)

OWL is based on RDF and uses RDF's XML syntax. The general structure of an owl document is presented as follows [15]:

```
<rdf:RDF
[namespace declaration]>
  <owl:Ontologie rdf:about= ' '> [Header of the Ontology]
  [class declaration]
  [declaration of properties and relationships]
  [instance declaration]
  </owl:ontologie>
/rdf:RDF>
```

## II.1. Namespace Declaration (header):

Namespaces were introduced in XML in order to be able to mix several vocabularies within the same document. The initial component of an ontology includes a set of XML namespace declarations, contained within an **rdf:RDF** opening tag.

```
<rdf:RDF
xmlns = "URI#"
xmlns:prefixname = "URI#"
xmlns:base = "URI#"
xmlns:prefixname2 = "URI2"
xmlns:owl = "http://www.w3.org/2002/07/owl#"
xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs = "http://www.w3.org/2000/01/rdf-schema#"
xmlns:xsd = http://www.w3.org/2001/XMLSchema#
>
```

**Namespace of the current Ontology**

**Namespaces of other Ontologies: 1 line /Ontology**

**Namespaces of other languages**

Fig 12. Namespace Declaration (header) [15]

Namespace of the current Ontology:

1. **xmlns = "URI#"**: the default namespace. Used to qualify elements that are not qualified (are not prefixed). for example <name> □ <Person: name>
2. **xmlns:prefixname = "URI#"**: identify the namespace of the current ontology with the prefix prefixname
3. **xmlns:base = "URI#"**: identify the base URI address of this document

**Example:** Header of the Humanite Ontology in which we will integrate the Alive Ontology.

```
<rdf:RDF
xmlns = "http://domain.tld/path/humanite#"
xmlns:humanite = "http://domain.tld/path/humanite#"
xmlns:base = "http://domain.tld/path/humanite#"
xmlns:alive = "http://otherdomain.tld/otherpath/alive#"
xmlns:owl = "http://www.w3.org/2002/07/owl#"
xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs = "http://www.w3.org/2000/01/rdf-schema#"
xmlns:xsd = "http://www.w3.org/2001/XMLSchema#"
>
```

**Namespace of the current Ontology**

**Namespaces of other Ontologies:**

**Namespaces of other languages**

## II.2. Header of the Ontology

The owl:Ontology tag groups together a set of assertions about ontology.

```
<owl:Ontology rdf:about="">
```

```
<rdfs:comment> "Comments describing the Ontology" </rdfs:comment>
```

```
<owl:imports> the inclusion of other ontologies (1 line/ ontology) </owl:imports>
```

```
<rdfs:label>Ontology Label</rdfs:label>
```

**Note:** If the URI is not specified in the base attribute, it must be mentioned in the about attribute.

### Example :

```
<owl:Ontology rdf:about="">
```

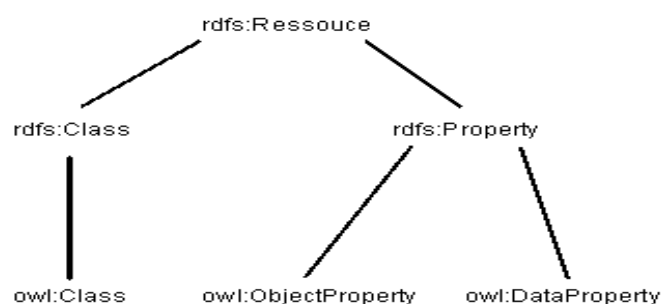
```
<rdfs:comment>Ontology describing humanity</rdfs:comment>
```

```
<owl:imports
```

```
  rdf:resource="http://otherdomain.tld/otherpath/vivant"/>
```

```
<rdfs:label>Ontology on humanity</rdfs:label>
```

## II.3. Elements of language



**Fig 13: OWL: Elements of language [16]**

### II.3. 1. The Class:

There are two predefined classes which are:

- owl:Thing: (TOP or universal) Each individual in the OWL world is a member of the owl:Thing class. Each user-defined class is therefore implicitly a subclass of owl:Thing
- owl: noThing (impossible or empty): subclass of all classes owl (class without instances)

A class can be declared in six different ways: Class identification, Instance Enumeration, Property Restriction, Union, Intersection, Complement

1. **Class identification:** Classes are defined with owl:Class (Subclass of rdfs:Class).

**Syntax:** <owl:Class rdf:ID="Term" />

The class can be called within the document by #Term

**Example:** <owl:Class rdf:ID="Human" />

The Human class can be called by #Human

The subClassOf property is used to express inheritance. For example to express the inheritance between the document and mail classes:

```
<owl:Class rdf:ID="Document" />
```

```
<owl:Class rdf:ID="Mail">
```

```
<rdfs:subClassOf rdf:resource="#Document" />
```

```
</owl:Class>
```

#### 2. Instance Enumeration (OWL DL and OWL FULL)

This type of description is done by listing the instances of the class, using the owl:oneOf property:

**Syntax:**

```
<owl:Class>
```

```
<owl:oneOf rdf:parseType="Collection">
```

```
<owl:Thing rdf:about="#A"/>
```

```
<owl:Thing rdf:about="#C"/>
```

```
...
```

```
</owl:oneOf>
```

```
</owl:Class>
```

**Example :**



```
<owl:Class>  
<owl:oneOf rdf:parseType="Collection">  
<owl:Thing rdf:about="#Mohamed" />  
<owl:Thing rdf:about="#Amine" />  
<owl:Thing rdf:about="#Alia" />  
</owl:oneOf>  
</owl:Class>
```

### 3. Property Restriction

A property restriction is a kind of class description. It defines a class composed of all instances of owl:Thing that satisfy one or more properties (restriction). OWL distinguishes two kinds of restrictions: value constraints and cardinality constraints. Property restrictions have the following general form:

```
<owl:Restriction>  
<owl:onProperty rdf:resource="(some property) " />  
(specifying a value or cardinality constraint)  
</owl:Restriction>
```

#### 3.1. Value constraints (hasValue, allValuesFrom, someValuesFrom)

A value constraint is exerted on the value of a certain property of the individual

- allValuesFrom: class for which all the values of a property come from another class

**Example:** Individuals of an anonymous class whose parents are People.

```
<owl:Restriction>  
<owl:onProperty rdf:resource="#hasParent" />  
<owl:allValuesFrom rdf:resource="#Person" />  
</owl:Restriction>
```

- someValuesFrom: at least some property values come from a specific class

**Example:** Individuals from an anonymous class of which some parents are Women.

```
<owl:Restriction>  
<owl:onProperty rdf:resource="#haveParent" />  
<owl:someValuesFrom rdf:resource="#Female" />  
</owl:Restriction>
```

- hasValue: the defined class can only have one value for the targeted property

**Example:** The class of individuals whose parent is the individual Amel

```
<owl:Restriction>
<owl:onProperty rdf:resource="#hasParent" />
<owl:hasValue rdf:resource="#Amel" />
</owl:Restriction>
```

### 3.2. Cardinality constraints: (minCardinality, Cardinality, maxCardinality)

relates to the number of values that a property can take. The following example describes a class of individuals having at least one parent.

```
<owl:Restriction>
<owl:onProperty rdf:resource="#hasParent" />
<owl:minCardinality rdf:datatype="&xsd; nonNegativeInteger">1
</owl:minCardinality>
</owl:Restriction>
```

#### Example :

Write in OWL the class of people with two brothers.

```
<owl:Restriction>
<owl:onProperty rdf:resource="#aForBrother" />
<owl:cardinality rdf:datatype="&xsd;nonNegativeInteger"> 2 </owl:cardinality>
</owl:Restriction>
```

#### 4. Union: Declare a class by union of two or more classes:

```
<owl:Class>
<owl:unionOf rdf:parseType="Collection">
<owl:Class rdf:about="#class1" />
...
<owl:Class rdf:about="#classen" />
</owl:unionOf>
</owl:Class>
```

#### 5. Intersection: Declare a new class by intersection of two or more classes:

```
<owl:Class>
<owl:intersectionOf rdf:parseType="Collection">
<owl:Class rdf:about="#class1"/>
```

```
.....  
<owl:Class rdf:about="#classen"/>  
</owl:intersectionOf>  
</owl:Class>
```

**6. Complement: Declare a class that contains all individuals in the speech domain not belonging to another class.**

```
<owl:Class rdf:ID="Nonclass">  
<owl:complementOf rdf:resource="#class"/>  
</owl:Class>
```

**Example:** the expression “not student” can be written:

```
<owl:Class>  
<owl:complementOf rdf:about="#Student"/>  
</owl:Class>
```

**Exercise:** write in OWL the class of students with two brothers

```
<owl:Class>  
  
<owl:intersectionOf rdf:parseType="Collection">  
  
<owl:Class rdf:about="#students" />  
  
<owl:Restriction>  
  
<owl:onProperty rdf:resource="#aForBrother" />  
  
<owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">  
2  
</owl:cardinality>  
</owl:Restriction>  
</owl:intersectionOf>  
</owl:Class>
```

### II.3.2. Properties

OWL distinguishes between two types of properties:

**1. Object properties (owl:ObjectProperty):** object properties allow you to link instances to other instances: it is necessary to specify the domain and the image of the property.

**Example:** the property lives has the Human class as its domain and the Country class as its image: it connects instances of the Human class to instances of the Country class.

```
<owl:ObjectProperty rdf:ID="SubmittedBy">
<rdfs:domain rdf:resource="#mail"/>
<rdfs:range rdf:resource="#issuer"/>
</owl:ObjectProperty>
```

Data type properties (owl:DatatypeProperty): in the case of a data type property, the image of the property can be a data type

**Example:** declaration of the object property of the Mail class

```
<owl:DatatypeProperty rdf:id="object">
<rdfs:domain rdf:resource="#mail" />
<rdfs:range rdf:resource="&xsd:string"/>
</owl:DatatypeProperty>
```

**Data Types:**

<b>String &amp; its derivatives</b>	xsd: string xsd:token xsd:language xsd:Name xsd: NCName	xsd:normalizedString xsd:NMTOKEN
<b>Boolean</b>	xsd: boolean	
<b>Digital data</b>	xsd: decirnal double xsd:integer xsd:nonPositiveInteger xsd:negativeInteger xsd:nonNegativeInteger xsd: long xsd:unsignedLong xsd:unsignedShort	xsd: float xsd: positiveInteger xsd: int xsd: short xsd: byte xsd:unsignedInt xsd:unsignedByte
<b>Time-related types</b>	xsd:dateTirne, xsd:gYearMonth xsd:gDay	xsd:tirne xsd:gYear xsd:gMonthDay xsd : gMonth

**Table 2: OWL data type**

### II.3. 3. The Instances

Simply declare that an instance belongs to a class:

**Syntax:**

```
<Class_ID rdf: ID= "Individual_ID" />
```

**Example :**

```
<Mail rdf:ID="Mail14582"/>
```

Properties of individuals

**Syntax:**

```
<Class_ID rdf: ID= "Individual_ID" >  
< property1> property-value < /property1>  
</ Class_ID >
```

**Example :**

```
<Mail rdf:ID="Mail14582">  
<object>presidentialvisit</object>  
<mail_num>14582</mail_num>  
</Mail>
```

### II.3.6. Relationships between properties:

- owl:equivalentProperty: the two properties have the same extension, but are not identical.
- owl:inverseOf: one property is the inverse of the other.

```
<owl:ObjectProperty rdf:ID="child">  
  <owl:inverseOf rdf:resource="#parent"/>  
</owl:ObjectProperty>
```

**Logical constraints:**

- owl:SymmetricProperty (husband)
- owl:TransitiveProperty (ancestor)

**Exercise:** Express the following expressions in Owl:

1. Classes: Person, Man, Woman: (Man and Woman are Persons and Man and Woman are disjointed)
2. A person has a name, age and nationality

3. A person is the parent of another person
4. A woman is a person's mother
5. The parent class: instances of this class are parents of at least one person
6. Class Father
7. ClassMother
8. Farid, 40, son of Ahmed, of Algerian nationality
9. A herbivore is an animal that feeds mainly on plants,

**Exercises:** Express the following expressions in Owl:

1. **Classes: Person, Man, Woman:** (Man and Woman are Persons and Man and Woman are disjointed)

```
<owl:Class rdf:ID=" Person"/>
```

```
<owl:Class rdf:ID="Man ">
```

```
<rdfs:subClassOf rdf:resource="# Person " />
```

```
< owl:disjointWith rdf: resource = " # Woman " />
```

```
</owl:Class>
```

```
<owl:Class rdf:ID="Woman ">
```

```
<rdfs:subClassOf rdf:resource="# Person " />
```

```
</owl:Class>
```

2. **A person has a name, age and nationality**

```
<owl: DatatypeProperty rdf:ID=" nom ">  
  <rdfs:domain rdf:resource="# Personne" />  
  <rdfs:range rdf:resource="&xsd;String" />  
</owl:DatatypeProperty>
```

```
<owl: DatatypeProperty rdf:ID=" age ">  
  <rdfs:domain rdf:resource="# Personne" />  
  <rdfs:range rdf:resource="&xsd;Integer" />  
</owl:DatatypeProperty>
```

```
<owl: DatatypeProperty rdf:ID=" nationalité ">  
  <rdfs:domain rdf:resource="# Personne" />  
  <rdfs:range rdf:resource="&xsd;String" />  
</owl:DatatypeProperty>
```

**3. A man is a person's father**

```
<owl:ObjectProperty rdf:ID="fatherOf">  
<rdfs:domain rdf:resource="# man " />  
<rdfs:range rdf:resource="# Person " />  
</owl:ObjectProperty>
```

# Chapter IV: Mapping and alignment of ontologies



## I. Introduction

Semantic interoperability between information sources is an important issue due to the increasing number of information sources available on the web. The user often faces several problems when trying to use independently developed ontologies, or when existing ontologies are adapted for new purposes.

Incompatibilities between concepts in different ontologies can exist on two levels [17]:

**Level 1: Language incompatibility:** Language-level incompatibilities occur when ontologies written in different ontology languages are combined.

**Level 2: Incompatibility at the ontology level:** For example :

- **Concept structuring:** Concerns the difference in the design of ontologies, for example one person can use the concept “address” as a single entity while others can separate it over several entities: City , Street name (number) , House Number ...
- **Synonymous terms:** Multilanguage incompatibilities occur when using different synonyms for the same concept, Example : “house” and “home .”; Human and Person , ...
- **Homonymous Terms:** This is called overlapping terminology. For example, “table” can be a piece of furniture or a list of information arranged in columns and rows.
- **Coding:** incompatibilities occur when the Ontologies are built in different formats such as a date represented by “dd/mm/yyyy” or “dd-mm-yyyy”.

Ontology mapping is seen as a promising solution for making heterogeneous systems and semantic web applications (where information is presented by ontologies) interoperable.

**II. Mapping:** is the formal expression of a semantic relationship between two entities belonging to two different ontologies. Ontology mapping can be seen as a process that semantically links two different vocabularies.

The result of the mapping process is called an alignment which represents the set of functions that connect the concepts of different ontologies. There are six types of mapping cardinalities (1:1), (1:n), (n:1), (1:null), (null:1), and (n:m). For example: the entity "Name" corresponds to two entities "First name" and "Last name":

Type de mapping	O1	O2	Expressions de mapping
1 : 1	Faculty	Academic staff	$O1.Faculty = O2.Academicstaff$
1 : n	Name	First name, Last name	$O1.Name = O2.Firstname + O2.Lastname$
n : 1	Cost, Tax ratio	Price	$O1.Cost * (1 + O1.Taxratio) = O2.Price$
1 : null	AI		

### III. Alignment:

An ontology alignment is a set of correspondences between the entities (classes, properties, predicates, etc.) forming ontologies. The alignment process called Matching is the action which allows to find these correspondences which are relation such as equivalence ( $\equiv$ ), subsumption (more general ( $\supseteq$ ), more specific ( $\subseteq$ )) and disjunction ( $\vee$ ).

An alignment is described by a quintuple A:  $\langle id, e_1, e_2, r, n \rangle$  such that:

1. Id: unique identifier of an alignment,
2. e: an entity to align belonging to O (ontology 1): class, property, constraint, instance
3.  $e'$ : an entity to align belonging to O' (ontology 2)
4. r a relation linking e and  $e'$  ( $=, \subseteq, \supseteq, \dots$ )
5. n: the confidence measure of the relationship r.

The following figure shows an example of alignment of two ontologies:

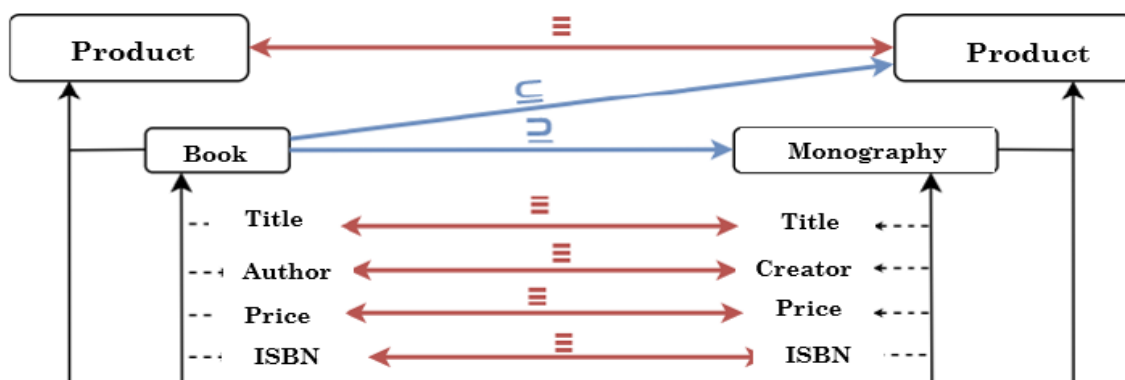
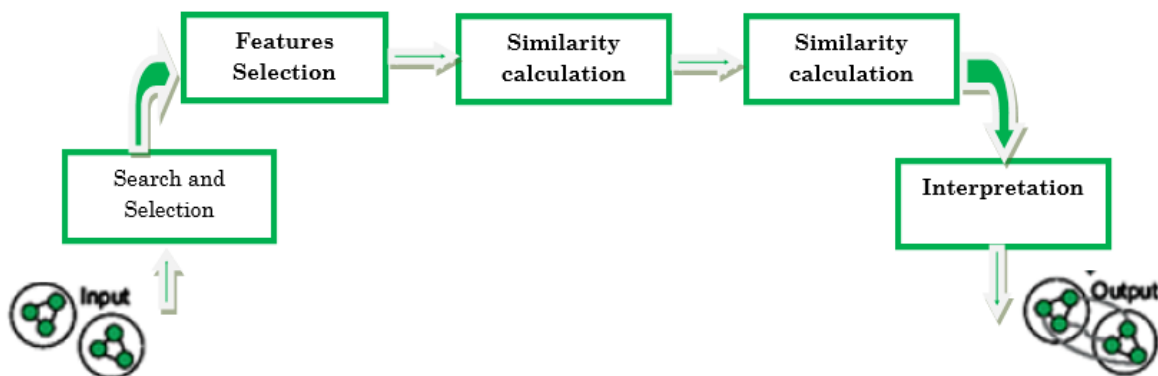


Fig 14: Example of alignment of two ontologies

#### IV. Steps in a mapping process

The mapping process can be defined in a generalized way as being a series of steps illustrated in the following figure:



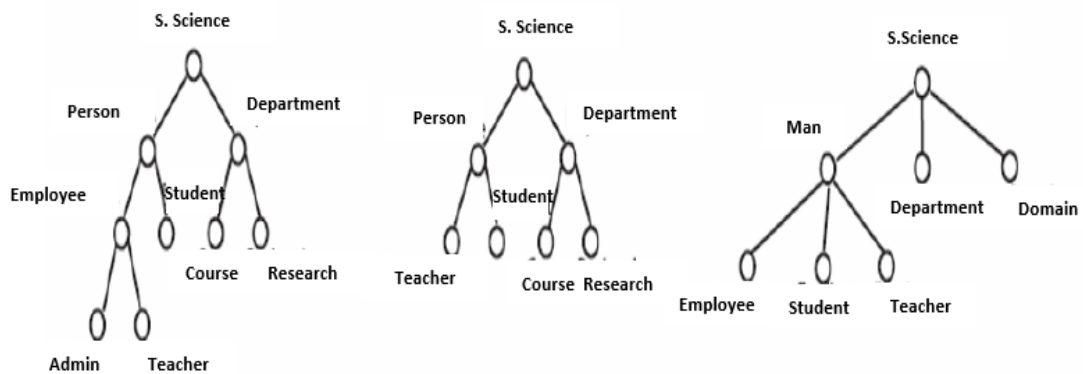
**Fig 15: Mapping Process [18]**

- **Input:** A pair of ontologies for which we want to establish alignment. It is possible to have an alignment at the start that helps find other alignments.
- **Features Selection :** the set of characteristics to be compared (for example, we can consider the labels of the concepts, the instances, etc.); The result of this step is the set of characteristics to be compared in the search and selection phases and calculation of similarity.
- **Search and selection:** In this step, the choice of entities that will be taken into account. For example, we can compare all the entities of the source ontology with those of the target ontology or compare only similar entities (concepts with concepts, properties with properties, relations with relations or instances with instances ). The result of this step are the candidate entities for the similarity calculation.
- **Similarity calculation:** this is a semantic similarity. In general, these measurements return values that belong to the interval  $[0,1]$ . The application of a single measurement produces an individual matching.
- **Similarity aggregation:** The results produced in the previous phase (in the case of multiple matching) are aggregated using aggregation functions such as Max, Average, weighted sum, weighted product etc.
- **Interpretation:** Interpretation uses individual or aggregated matching results to derive relevant alignments between entities.
- **Iteration:** depending on the results obtained, we can decide to do another iteration in order to refine the results obtained in the previous iteration

## V. Alignment Approaches:

There are several alignment approaches. The following example shows an approach to merging ontologies using the hierarchical concept classification technique

Example :



Hierarchical classification of concepts and construction of the SYN set: Use of a similarity measure to calculate the similarity between two concepts. Each class contains the synonymous concepts of the different ontologies

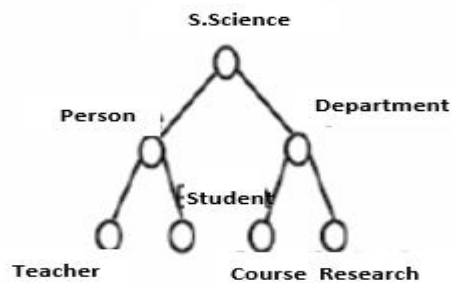
$SYN = \{$   
 $\{F.Sciences, Department, F.science\}, \{Person, Person, MAN\},$   
 $\{Department, Department, Department\}, \{Student, Student, Student\},$   
 $\{Teacher, Teacher, Teacher\}, \{Employee, Employee\}, \{Course,$   
 $Course\}, \{Research, Research\}, \{Admin\}, \{Domain\}$

Teacher Course Research

- Construction of the SYN set which contains the “Global” concepts (Maintain the link between new concepts and those of local ontologies.)

$SYNGlobal = \{F.Sciences, Person, Department, Student, Teacher, Employee,$   
 $Course, Admin, Research, Domain\}$

- For each ontology, we deduce the set SUB<sub>i</sub> of pairs (father, son)



SUB2={ (F.sciences, Person), (F.sciences, Department), (Person, Teacher), (Person, Student), (Department, Course), (Department, Research) }

- SUBG: We merge the SUB<sub>i</sub> sets to have the total SUB set which contains all pairs of all ontologies.

SUBG = { (F.sciences, Person), (F.sciences, Department), (Person, Employee), (Person, Student), (Department, Course), (Department, Research), (Employee, Admin.), (Employee, Teacher), (F.sciences, Person), (F.sciences, Department), (Person, Teacher), (Person, Student), (Department, Course), (Department, Research), (F.sciences, MAN), (F.sciences, Department), (F.sciences, Field), (Man, Employee), (MAN, Student), (MAN, Teacher) }

- In the SUBG set, we replace each concept with the equivalent concept plus Global from the SYN set
- Removal of redundant pairs in the SUB set

SUBG = { (F.sciences, Person), (F.sciences, Department), (Person, Employee), (Person, Student), (Department, Course), (Department, Research), (Employee, Admin.), (Employee, Teacher), (Person, Teacher), (F.sciences, Field) }

## GENERAL CONCLUSION

This handout highlights the crucial significance of knowledge representation. By examining ontologies, knowledge graphs, descriptive logic, and OWL, we gain valuable insights into structured methods for arranging and disseminating information. The mapping and alignment of ontologies is crucial for the integration of disparate data sources, thereby facilitating seamless communication between systems.

In general, mastering these concepts equips us with the capabilities to construct more intelligent systems capable of comprehending and utilizing intricate knowledge, ultimately propelling advancements in fields such as artificial intelligence, data science, and semantic web technologies. This fundamental knowledge is imperative for effectively addressing real-world challenges that necessitate efficient information sharing and collaboration.

**TP 1 :**

1. Downloaded the Pizza ontology: (<http://protege.stanford.edu/ontologies/pizza/pizza.owl>),
2. Load it into Protégé
3. Identify : the main classes, the properties and the complex classes
4. Check the consistency of the ontology and propose solutions to remove inconsistencies if there are any.

**TP2:**

The goal of this work is to design an ontology for e-Learning.

This ontology allows:

- to provide a conceptual vocabulary that can be shared between the community of teachers and students
- to annotate and search for educational documents on a distance learning platform.

**Required work :**

- A report explaining the components of the proposed ontology (Concepts, intensions, relationships, etc.)
- editing the ontology with “Protégé” or another Ontology editor

### **TP3 :**

The objective is to build an that aims to model **art works**.

#### **Specifications :**

- **Classes :**
  1. It contains three first-level classes: Artist, Collection and Work, disjoint from each other.
  2. Collection contains two disjoint subclasses Private and Museum. Any instance of Collection is necessarily an instance of one of these two subclasses.
  3. Work contains two subclasses Painting and Sculpture.
- **Properties (object properties) :**
  1. The author property links a Work to one or more Artist(s) who created it.
  2. The creation property links an Artist to each of the Works he/she has created.
  3. The contains property links a Collection to each of the Works it contains.
  4. The collaborator property links two Artists each time they have created a Work together. In other words, an Artist has as collaborators the authors of his creations.
  5. Every Work has at least one author.
  6. Every Artist has at least one creation.
  7. Every Collection contains at least one Work.
- **Instances**
  1. Louvre is an instance of Museum.
  2. Monalisa and Baptism are instances of Painting.
  3. grancavallo is an instance of Sculpture.
  4. Rachida-Haddad, Leonardo and Yerrocchio are instances of Artist.
  5. louvre contains monalisa; monalisa, grancavallo and baptism all have leonardo as author. baptism has verrocchio as author.
- **Defined Classes**
  1. Anything that has created a Painting is a Painter.
  2. Anything that has created a Sculpture is a Sculptor.
  3. Anything that has created a Painting and something that is not a Painting is a PainterVersatile.
  4. Any Collection that contains only Paintings is a Gallery

#### **Questions :**

Provide the DL query that answers to.

- Q1: Is it possible to be both a Painter and a Sculptor?
- Q2: Is it possible to be an Artist without having created any Painting or Sculpture?
- Q3: Is it possible for a Gallery to contain a Work whose author is a Sculptor?
- Q4: Is it possible for an Artist to have no collaborators?



## Bibliography

- [1] Hans Muller & Christiaan Maasdorp (2011). The data, information, and knowledge hierarchy and its ability to convince. Research Challenges in Information Science (RCIS).
- [2] Dr. Roman & V Belavkin (2012). Lecture 1: Data, Information and Knowledge. Middlesex University, UK .
- [3] J. M., & McElroy, M. W. (2003). Key issues in the new knowledge management. KMCI Press.
- [4] Jean Charlet, (2003). 'knowledge engineering Developments, results and perspectives For medical knowledge management' . Dissertation qualification for supervising research.
- [5] Dr. Roman & V Belavkin (2012). Lecture 8: Ontologies. Middlesex University, UK.
- [6] Razika Driouche (2017), Towards Ontology Lifecycle: Building, Matching and Evolution to Semantically Integrate Application Ontologies, International Journal of Computer Applications Technology and Research ,Volume 6–Issue 2, 109-116, 2017, ISSN:-2319–8656
- [7] [https://perso.liris.cnrs.fr/amille/enseignements/DEA-ECD/ontologies/construction\\_ontologie.htm](https://perso.liris.cnrs.fr/amille/enseignements/DEA-ECD/ontologies/construction_ontologie.htm) visited 17.11.2023
- [8] Extensible Markup Language (XML) 1.0 , W3C Recommendation 10-Feb-98
- [9] <https://www.w3.org/TR/rdf-schema/> visited 07.09.2023
- [10] P.A. Bonatti, S. Decker, A. Polleres, V. Presutti, Knowledge graphs: new directions for knowledge representation on the Semantic Web (dagstuhl seminar 18371). Dagstuhl Rep. 8(9), 29–111 (2019)
- [11] Hollunder, B., & B. R. (1999). *Description Logics: An Overview*. In *Handbook of Logic in Artificial Intelligence and Logic Programming* (Vol. 4).
- [12] Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., & Patel-Schneider, P. F. (2003). *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press.
- [13] Markus Krötzsch and all ,(2013), Description Logics, University of Oxford, UK.
- [14] Leif Harald Karlsen,(2015) Description Logic 1: Syntax and Semantics.
- [15] <https://www.w3.org/TR/owl-ref/> visited 2.3. 2023
- [16] [https://iaoa.org/isc2012/docs/Guarino2009\\_What\\_is\\_an\\_Ontology.pdf](https://iaoa.org/isc2012/docs/Guarino2009_What_is_an_Ontology.pdf) visited 2.3. 2023
- [17] Jérôme Euzenat, (2007) , Introduction to ontology matching and alignment, Montbonnot, France.
- [18] Helen L And All ,(2007), A Fault Model For Ontology Mapping, Alignment, And Linking Systems, Pacific Symposium On Biocomputing 12:233-268