

# Semantic Web

Yue Ma

Laboratoire de Recherche en Informatique (LRI)  
Université Paris Sud  
[ma@lri.fr](mailto:ma@lri.fr)

# Outline

## A basic Description Logic

# Description Logics

A subfield of knowledge representation  
which is a subfield of artificial intelligence

## Description

comes from concept description :  
a formal expression that determines a set of individuals

## Logics

comes from the semantics being defined using logic :  
most Description Logics are fragments of First Order Logic

# Knowledge Representation

## Goal [Brachman & Nardi, 2003]

develop formalisms for providing high-level description of the world  
that can be effectively used to build intelligent applications

- ▶ **formalism** : well-defined syntax ; formal, unambiguous semantics
- ▶ **high-level description** : only relevant aspects represented
- ▶ **intelligent applications** : reason about the knowledge ; infer implicit knowledge
- ▶ **effectively used** : practical reasoning tools and efficient implementations

# Syntax

- ▶ explicit symbolic representation of the knowledge
- ▶ not implicit (as e.g. in neural networks)

*Woman*  $\equiv$  Person  $\sqcap$  Female  
*Man*  $\equiv$  Person  $\sqcap$   $\neg$ Female  
*Mother*  $\equiv$  Woman  $\sqcap$   $\exists$ hasChild.T

hasChild(stephen, marc)  
hasChild(marc, anna)  
hasChild(john, maria)  
hasChild(anna, jason)

Male(john)  
Male(marc)  
Male(stephen)  
Male(jason)  
Female(jill)  
Female(anna)  
Female(maria)

# Semantics

connection between the symbolic representation and the “real world” entities it represents

## Declarative semantics

- ▶ map symbols to an abstraction of the “world” (interpretation)
- ▶ notion of when a symbolic expression is true in the world (model)

## NO procedural semantics :

should not just express how specific programs should behave

# Expressive Power

(what can be expressed) depends on syntax and semantics

## Equilibrium

**not too low** : can all the relevant knowledge be represented ?

**not too high** : are all the elements really necessary for the application ?

# Reasoning

deduce implicit knowledge from the explicit representation

$$\forall x. \forall y. (male(y) \wedge \exists z. (child(x, z) \wedge child(z, y))) \rightarrow grandson(x, y)$$

*child(john, mary)*

*child(mary, paul)*

*male(paul)*

*grandson(john, paul)*

## Knowledge Representation Systems

should provide inference tools to deduce implicit consequences

answers should depend on **semantics**; not on the **syntactic**

**representation** (same semantics must yield the same answer)

# Reasoning Procedures

## Decision Procedure

- ▶ **sound** : all positive answers are correct
- ▶ **complete** : all negative answers are correct
- ▶ **terminating** : always provides an answer in finite time

## Efficient

- ▶ ideally, **optimal** w.r.t. worst-case complexity of the problem

## Practical

- ▶ easy to implement and optimize
- ▶ behave well in applications

# Examples of Formalisms

## First-order logic

satisfiability in FOL **does not have** a decision procedure

is thus not an appropriate knowledge representation formalism

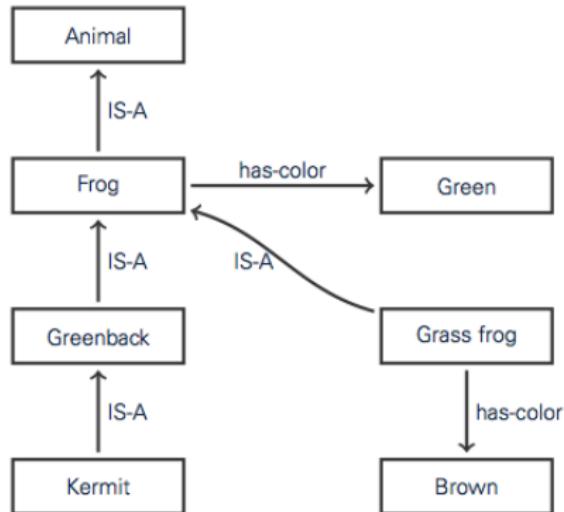
## Propositional logic

satisfiability in propositional logic has a decision procedure the problem is **NP-complete**

highly optimized SAT solver behave well in practice

however, expressive power is often insufficient

# Terminological Knowledge



# Terminological Knowledge

formalize the terminology (names) of the application domain :

- ▶ define important notions of the domain (classes, relations, objects)
- ▶ constrain the interpretations of these notions
- ▶ deduce consequences : subclass, instance relationships

Example (university domain)

- **classes (concepts)** : Person, Teacher, Course, Student, . . .
- **relations (roles)** : gives, attends, likes, . . .
- **objects (individuals)** : DL WS13, Marcel, Daniel, . . .
- **constraints** :
  - every course is given by a teacher,
  - every student must attend at least one course

# Ontologies

the modern name for knowledge bases

## Applications

- ▶ **semantic web** enable a common understanding of notions for semantic labeling of Web content
- ▶ **information retrieval** support automatic extraction of information from text
- ▶ **medicine** formal definitions that can be used by doctors, patients, insurance companies, etc, to communicate with each other

# Description Logics

a class of logic-based knowledge representation formalisms for representing terminological knowledge

## Prehistory

early approaches for representing terminological knowledge

- ▶ semantic networks (Quillian, 1968)
- ▶ frames (Minsky, 1975)

problems with missing semantics led to

- ▶ structured inheritance networks
- ▶ the first DL system KL-ONE

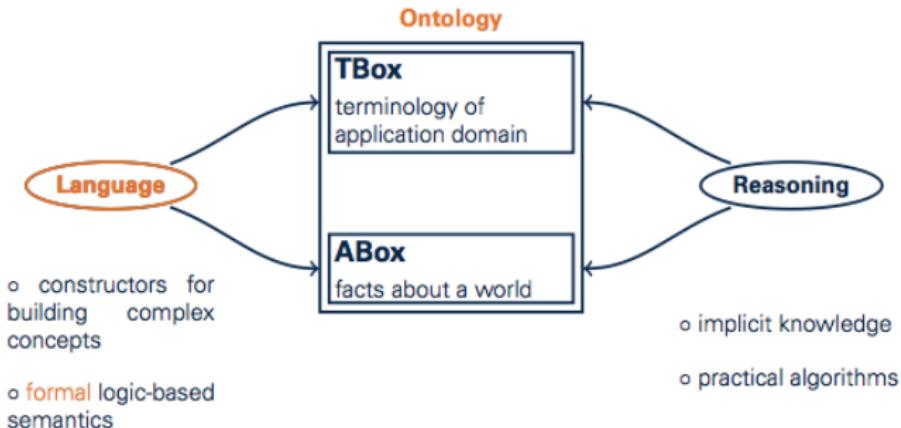
# History of Description Logics

- ▶ Phase 1
  - implementation of systems based on incomplete structural subsumption algorithms
- ▶ Phase 2
  - tableau-based algorithms and complexity results
  - first tableau-based systems (Kris, Crack)
  - first formal study of optimization methods
- ▶ Phase 3
  - tableau-based algorithms for very expressive DLs
  - highly-optimized tableau-based systems (FaCT, Racer)
  - relationship to modal logic and FOL

# History of Description Logics

- ▶ Phase 4
  - Web Ontology Language (OWL) based on DL
  - industrial-strength reasoners and ontology editors
  - light-weight (tractable) DLs
- ▶ Phase 5
  - non-standard reasoning
  - ontology management
  - semantic extensions

# Structure of Description Logic Systems



# Outline

A basic Description Logic

## Syntax of $\mathcal{ALC}$ : Notations

- ▶ concept names are also called **atomic concepts**
  - ▶ all other concepts are called **complex**
  - ▶ instead of  $\mathcal{ALC}$  concept, we often say **concept**
- 
- ▶  $A, B$  stand for concept names
  - ▶  $C, D$  for (complex) concepts
  - ▶  $r, s$  for role names

Example.

We can have *Female*, *Student*, *Pizza*, *Win*, *Country* as atomic concepts, *hasChild*, *hasTeacher*, *hasTopping*, *locatedIn* as role names, and  $\text{Female} \sqcap \exists \text{hasChild}.\text{Student}$  a complex concept.

How is the complex concept constructed ?

And what does it mean (intuitively) ?

## Syntax of $\mathcal{ALC}$ : constructing complex concepts

Let  $N_C$  and  $N_R$  two disjoint sets of concept names and role names, respectively.  $\mathcal{ALC}$  concepts are defined by induction :

- ▶ if  $A \in N_C$ , then  $A$  is an  $\mathcal{ALC}$  concept
- ▶ if  $C, D$  are  $\mathcal{ALC}$  concepts and  $r \in N_R$ , then the following are  $\mathcal{ALC}$  concepts :
  - $C \sqcap D$  (conjunction)
  - $C \sqcup D$  (disjunction)
  - $\neg C$  (negation)
  - $\exists r.C$  (existential restriction)
  - $\forall r.C$  (value restriction)
- ▶ Abbreviations :
  - $\top = A \sqcup \neg A$  (top)
  - $\perp = A \sqcap \neg A$  (bottom)

Example.  $\text{Female} \sqcap \exists \text{hasChild}.\text{Student}$

For you : try to write down your (atom, complex)  $\mathcal{ALC}$  concepts.

## Syntax of $\mathcal{ALC}$ : constructing complex concepts

Let  $N_C$  and  $N_R$  two disjoint sets of concept names and role names, respectively.  $\mathcal{ALC}$  concepts are defined by induction :

- ▶ if  $A \in N_C$ , then  $A$  is an  $\mathcal{ALC}$  concept
- ▶ if  $C, D$  are  $\mathcal{ALC}$  concepts and  $r \in N_R$ , then the following are  $\mathcal{ALC}$  concepts :
  - $C \sqcap D$  (conjunction)
  - $C \sqcup D$  (disjunction)
  - $\neg C$  (negation)
  - $\exists r.C$  (existential restriction)
  - $\forall r.C$  (value restriction)
- ▶ Abbreviations :
  - $\top = A \sqcup \neg A$  (top)
  - $\perp = A \sqcap \neg A$  (bottom)

Example.  $\text{Female} \sqcap \exists \text{hasChild}.\text{Student}$

For you : try to write down your (atom, complex)  $\mathcal{ALC}$  concepts.

## Syntax of $\mathcal{ALC}$ : constructing complex concepts

Let  $N_C$  and  $N_R$  two disjoint sets of concept names and role names, respectively.  $\mathcal{ALC}$  concepts are defined by induction :

- ▶ if  $A \in N_C$ , then  $A$  is an  $\mathcal{ALC}$  concept
- ▶ if  $C, D$  are  $\mathcal{ALC}$  concepts and  $r \in N_R$ , then the following are  $\mathcal{ALC}$  concepts :
  - $C \sqcap D$  (conjunction)
  - $C \sqcup D$  (disjunction)
  - $\neg C$  (negation)
  - $\exists r.C$  (existential restriction)
  - $\forall r.C$  (value restriction)
- ▶ Abbreviations :
  - $\top = A \sqcup \neg A$  (top)
  - $\perp = A \sqcap \neg A$  (bottom)

Example.  $\text{Female} \sqcap \exists \text{hasChild}.\text{Student}$

For you : try to write down your (atom, complex)  $\mathcal{ALC}$  concepts.

## Syntax of $\mathcal{ALC}$ : constructing complex concepts

Let  $N_C$  and  $N_R$  two disjoint sets of concept names and role names, respectively.  $\mathcal{ALC}$  concepts are defined by induction :

- ▶ if  $A \in N_C$ , then  $A$  is an  $\mathcal{ALC}$  concept
- ▶ if  $C, D$  are  $\mathcal{ALC}$  concepts and  $r \in N_R$ , then the following are  $\mathcal{ALC}$  concepts :
  - $C \sqcap D$  (conjunction)
  - $C \sqcup D$  (disjunction)
  - $\neg C$  (negation)
  - $\exists r.C$  (existential restriction)
  - $\forall r.C$  (value restriction)
- ▶ Abbreviations :
  - $\top = A \sqcup \neg A$  (top)
  - $\perp = A \sqcap \neg A$  (bottom)

Example.  $\text{Female} \sqcap \exists \text{hasChild}.\text{Student}$

For you : try to write down your (atom, complex)  $\mathcal{ALC}$  concepts.

## Syntax of $\mathcal{ALC}$ : constructing complex concepts

Let  $N_C$  and  $N_R$  two disjoint sets of concept names and role names, respectively.  $\mathcal{ALC}$  concepts are defined by induction :

- ▶ if  $A \in N_C$ , then  $A$  is an  $\mathcal{ALC}$  concept
- ▶ if  $C, D$  are  $\mathcal{ALC}$  concepts and  $r \in N_R$ , then the following are  $\mathcal{ALC}$  concepts :
  - $C \sqcap D$  (conjunction)
  - $C \sqcup D$  (disjunction)
  - $\neg C$  (negation)
  - $\exists r.C$  (existential restriction)
  - $\forall r.C$  (value restriction)
- ▶ Abbreviations :
  - $\top = A \sqcup \neg A$  (top)
  - $\perp = A \sqcap \neg A$  (bottom)

Example.  $\text{Female} \sqcap \exists \text{hasChild}.\text{Student}$

For you : try to write down your (atom, complex)  $\mathcal{ALC}$  concepts.

## Syntax of $\mathcal{ALC}$ : constructing complex concepts

Let  $N_C$  and  $N_R$  two disjoint sets of concept names and role names, respectively.  $\mathcal{ALC}$  concepts are defined by induction :

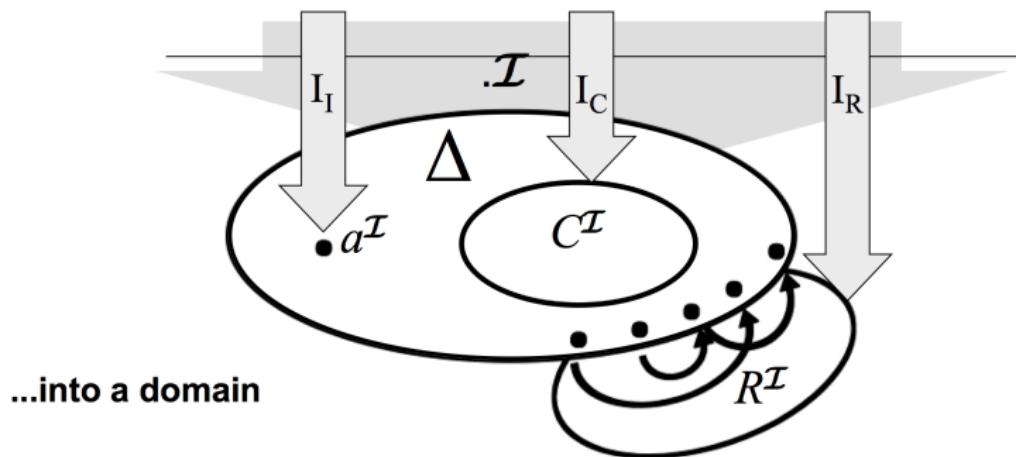
- ▶ if  $A \in N_C$ , then  $A$  is an  $\mathcal{ALC}$  concept
- ▶ if  $C, D$  are  $\mathcal{ALC}$  concepts and  $r \in N_R$ , then the following are  $\mathcal{ALC}$  concepts :
  - $C \sqcap D$  (conjunction)
  - $C \sqcup D$  (disjunction)
  - $\neg C$  (negation)
  - $\exists r.C$  (existential restriction)
  - $\forall r.C$  (value restriction)
- ▶ Abbreviations :
  - $\top = A \sqcup \neg A$  (top)
  - $\perp = A \sqcap \neg A$  (bottom)

Example.  $\text{Female} \sqcap \exists \text{hasChild}.\text{Student}$

For you : try to write down your (atom, complex)  $\mathcal{ALC}$  concepts.

# Semantics of Description Logics

- model-theoretic semantics
- starts with interpretations
- an interpretation  $\mathcal{I}$  maps  
individual names, class names and property names...



## Semantics of $\mathcal{ALC}$

An interpretation  $I = (\Delta^I, \cdot^I)$  consists of :

- ▶ a non-empty domain  $\Delta^I$ , and
- ▶ an extension mapping  $\cdot^I$  (also called interpretation function) :
  - $A^I \subseteq \Delta^I$  for all  $A \in N_C$  (concepts interpreted as sets)
  - $r^I \subseteq \Delta^I \times \Delta^I$  for all  $r \in N_R$  (roles interpreted as binary relations)

The extension mapping is extended to complex concepts :

$$(C \sqcap D)^I := C^I \cap D^I$$

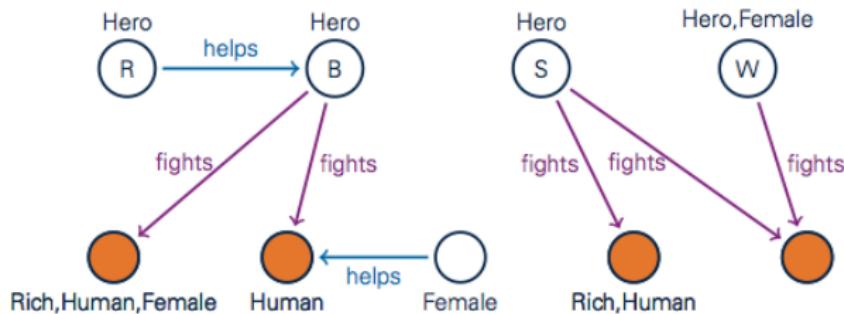
$$(C \sqcup D)^I := C^I \cup D^I$$

$$(\neg C)^I := \Delta^I \setminus C^I$$

$$(\exists r.C)^I := \{d \in \Delta^I \mid \text{there is } e \in \Delta^I \text{ with } (d, e) \in r^I \text{ and } e \in C^I\}$$

$$(\forall r.C)^I := \{d \in \Delta^I \mid \text{for all } e \in \Delta^I \text{ with } (d, e) \in r^I, \text{ it holds } e \in C^I\}$$

# Interpretation Example



$$(Hero \sqcap \exists \text{fights}.\text{Human})^I = \{B, S\}$$

$$(Hero \sqcap \forall \text{fights}.(\text{Rich} \sqcup \neg \text{Human}))^I = \{R, S, W\}$$

$$(\forall \text{helps}.\text{Human})^I = \Delta^I \setminus \{R\}$$

## Interpretation Example

Can you give an interpretation for the following concept ?

*Female*  $\sqcap \exists hasChild. Student$

## A step further towards ontologies

How we can express "There is a wash basin in either a kitchen or a bathroom" ?

To define what is an ontology, we need the following definitions :

- ▶ TBox and ABox
  - ▶ TBox : terminology box containing concept definitions and/or GCIs
  - ▶ ABox : assertion box, containing individual information.

# Defining a Concept (aka. Concept Definitions)

Definitions.

- ▶ A concept definition is of the form  $A = C$  where
  - ▶  $A$  is a concept name.
  - ▶  $C$  is a concept description.
- ▶ An interpretation  $I$  satisfies, also called a model of, the concept definition  $A = C$  if  $A^I = C^I$

Examples.

$\text{Heroine} = \text{Hero} \sqcap \text{Female}$

$\text{Student} = \text{People} \sqcap \exists \text{registers}.(\text{School} \sqcup \text{University})$

# Defining a Concept (aka. Concept Definitions)

Definitions.

- ▶ A concept definition is of the form  $A = C$  where
  - ▶  $A$  is a concept name.
  - ▶  $C$  is a concept description.
- ▶ An interpretation  $I$  satisfies, also called a model of, the concept definition  $A = C$  if  $A^I = C^I$

Examples.

$\text{Heroine} = \text{Hero} \sqcap \text{Female}$

$\text{MutantCriminal} = \text{Criminal} \sqcap \exists \text{fights}.\text{Mutant}$

## Defining a Concept (aka. Concept Definitions)

Definitions.

- ▶ A concept definition is of the form  $A = C$  where
  - ▶  $A$  is a concept name.
  - ▶  $C$  is a concept description.
- ▶ An interpretation  $I$  satisfies, also called a model of, the concept definition  $A = C$  if  $A^I = C^I$

Examples.

$\text{Heroine} = \text{Hero} \sqcap \text{Female}$

So, is the following interpretation  $I$  satisfying this concept definition ?

$$\Delta^I = \{a, b\}$$

$$\text{Heroine}^I = \{a\}$$

$$\text{Hero}^I = \{a, b\}$$

$$\text{Female}^I = \{b\}$$

If yes, why ?

If no, modify  $I$  to make it satisfy the concept definition.

# Defining a Concept (aka. Concept Definitions)

Definitions.

- ▶ A concept definition is of the form  $A = C$  where
  - ▶  $A$  is a concept name.
  - ▶  $C$  is a concept description.
- ▶ An interpretation  $I$  satisfies, also called a model of, the concept definition  $A = C$  if  $A^I = C^I$

Examples.

$Student = People \sqcap \exists registers.(School \sqcup University)$

For you : give a model of this concept definition

# GCI s (General Concept Inclusions) and TBoxes

Definitions.

- ▶ A general concept inclusion (GCI) is of the form  $C \sqsubseteq D$ , where  $C, D$  are concepts.
- ▶ A TBox  $\mathcal{T}$  is a finite set of GCIs and/or concept definitions.
- ▶ An interpretation  $I$  satisfies the GCI  $C \sqsubseteq D$  iff  $C^I \subseteq D^I$ .  $I$  is a model of a TBox  $\mathcal{T}$  iff it satisfies all GCIs in  $\mathcal{T}$ .
- ▶ Two TBoxes are equivalent if they have the same models.

Examples.

$\text{Hero} \sqcap \text{Villain} \sqsubseteq \perp$  : two concepts are disjoint

$\text{Heroine} = \text{Hero} \sqcap \text{Female}$

$\text{Kitchen} \sqcup \text{Bathroom} \sqsubseteq \exists \text{hasWashbasin}.\top$

$\text{Cold} \sqcap \exists \text{causedBy}.\text{Virus} \sqsubseteq \text{Disease}$

# GCI s (General Concept Inclusions) and TBoxes

Definitions.

- ▶ A general concept inclusion (GCI) is of the form  $C \sqsubseteq D$ , where  $C, D$  are concepts.
- ▶ A TBox  $\mathcal{T}$  is a finite set of GCIs and/or concept definitions.
- ▶ An interpretation  $I$  satisfies the GCI  $C \sqsubseteq D$  iff  $C^I \subseteq D^I$ .  $I$  is a model of a TBox  $\mathcal{T}$  iff it satisfies all GCIs in  $\mathcal{T}$ .
- ▶ Two TBoxes are equivalent if they have the same models.

Examples.

$\text{Hero} \sqcap \text{Villain} \sqsubseteq \perp$  : two concepts are disjoint

$\text{Heroine} = \text{Hero} \sqcap \text{Female}$

$\text{Kitchen} \sqcup \text{Bathroom} \sqsubseteq \exists \text{hasWashbasin}.\top$

$\text{Cold} \sqcap \exists \text{causedBy}.\text{Virus} \sqsubseteq \text{Disease}$

# GCI s (General Concept Inclusions) and TBoxes

Definitions.

- ▶ A general concept inclusion (GCI) is of the form  $C \sqsubseteq D$ , where  $C, D$  are concepts.
- ▶ A TBox  $T$  is a finite set of GCIs and/or concept definitions.
- ▶ An interpretation  $I$  satisfies the GCI  $C \sqsubseteq D$  iff  $C^I \subseteq D^I$ .  $I$  is a model of a TBox  $T$  iff it satisfies all GCIs in  $T$ .
- ▶ Two TBoxes are equivalent if they have the same models.

Examples.

$\text{Hero} \sqcap \text{Villain} \sqsubseteq \perp$  : two concepts are disjoint

$\text{Heroine} = \text{Hero} \sqcap \text{Female}$

$\text{Kitchen} \sqcup \text{Bathroom} \sqsubseteq \exists \text{hasWashbasin}.\top$

$\text{Cold} \sqcap \exists \text{causedBy}.\text{Virus} \sqsubseteq \text{Disease}$

# GCI s (General Concept Inclusions) and TBoxes

Definitions.

- ▶ A general concept inclusion (GCI) is of the form  $C \sqsubseteq D$ , where  $C, D$  are concepts.
- ▶ A TBox  $T$  is a finite set of GCIs and/or concept definitions.
- ▶ An interpretation  $I$  satisfies the GCI  $C \sqsubseteq D$  iff  $C^I \subseteq D^I$ .  $I$  is a model of a TBox  $T$  iff it satisfies all GCIs in  $T$ .
- ▶ Two TBoxes are equivalent if they have the same models.

Examples.

$\text{Hero} \sqcap \text{Villain} \sqsubseteq \perp$  : two concepts are disjoint

$\text{Heroine} = \text{Hero} \sqcap \text{Female}$

$\text{Kitchen} \sqcup \text{Bathroom} \sqsubseteq \exists \text{hasWashbasin.} \top$

$\text{Cold} \sqcap \exists \text{causedBy.} \text{Virus} \sqsubseteq \text{Disease}$

For you : give a model for these GCIs and definitions

# GCI s (General Concept Inclusions) and TBoxes

Definitions.

- ▶ A general concept inclusion (GCI) is of the form  $C \sqsubseteq D$ , where  $C, D$  are concepts.
- ▶ A TBox  $T$  is a finite set of GCIs and/or concept definitions.
- ▶ An interpretation  $I$  satisfies the GCI  $C \sqsubseteq D$  iff  $C^I \subseteq D^I$ .  $I$  is a model of a TBox  $T$  iff it satisfies all GCIs in  $T$ .
- ▶ Two TBoxes are equivalent if they have the same models.

Examples.

$\text{Hero} \sqcap \text{Villain} \sqsubseteq \perp$  : two concepts are disjoint

$\text{Heroine} = \text{Hero} \sqcap \text{Female}$

$\text{Kitchen} \sqcup \text{Bathroom} \sqsubseteq \exists \text{hasWashbasin}.\top$

$\text{Cold} \sqcap \exists \text{causedBy}.\text{Virus} \sqsubseteq \text{Disease}$

For you : give a model for these GCIs and definitions

# Assertions and ABoxes

Definitions.

- ▶ An assertion is of the form  $C(a)$  (concept assertion) or  $r(a, b)$  (role assertion) where  $C$  is a concept,  $r$  a role, and  $a, b$  are individual names from a set  $N_I$  (disjoint with  $N_C, N_R$ )
- ▶ An ABox  $\mathcal{A}$  is a finite set of assertions
- ▶ An interpretation  $I$  is a model of the ABox  $\mathcal{A}$  if it satisfies all its assertions :
  - $a' \in C^I$  for all  $C(a) \in \mathcal{A}$ .
  - $(a', b') \in R^I$  for all  $R(a, b) \in \mathcal{A}$ .

Examples.

$Kitchen(room1)$  : the room1 is a kitchen

$locatedIn(whitechair1, room1)$  : the whitechair1 is in the room1

$\neg Student(Jean)$  : Jean is not a student

# Assertions and ABoxes

Definitions.

- ▶ An assertion is of the form  $C(a)$  (concept assertion) or  $r(a, b)$  (role assertion) where  $C$  is a concept,  $r$  a role, and  $a, b$  are individual names from a set  $N_I$  (disjoint with  $N_C, N_R$ )
- ▶ An ABox  $\mathcal{A}$  is a finite set of assertions
- ▶ An interpretation  $I$  is a model of the ABox  $\mathcal{A}$  if it satisfies all its assertions :
  - $a' \in C^I$  for all  $C(a) \in \mathcal{A}$ .
  - $(a', b') \in R^I$  for all  $R(a, b) \in \mathcal{A}$ .

Examples.

$Kitchen(room1)$  : the room1 is a kitchen

$locatedIn(whitechair1, room1)$  : the whitechair1 is in the room1

$\neg Student(Jean)$  : Jean is not a student

For you : name some models for these ABox assertions

# Ontology

Definitions.

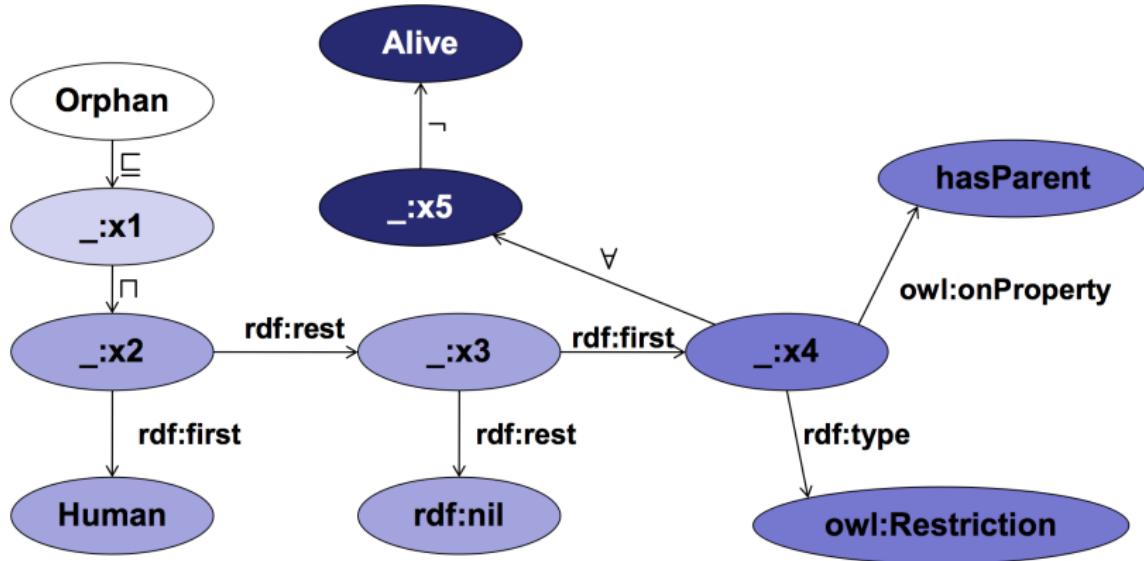
- ▶ An ontology  $O = (\mathcal{T}, \mathcal{A})$  consists of a TBox  $\mathcal{T}$  and an ABox  $\mathcal{A}$ .
- ▶ The interpretation  $I$  is a model of the ontology  $O = (\mathcal{T}, \mathcal{A})$  iff it is a model of  $\mathcal{T}$  and a model of  $\mathcal{A}$ .

Examples.

Many ontologies from <http://swoogle.umbc.edu> and  
<http://bioportal.bioontology.org>

# Description Logics and Graphs

Orphan  $\sqsubseteq$  Human  $\sqcap \forall \text{hasParent}. \neg \text{Alive}$



# OWL and Description Logics

```
:Mary rdf:type :Woman .
```

```
:John :hasWife :Mary .
```

```
:John owl:differentFrom :Bill .
```

**{John} ⊓ {Bill} ⊑ ⊥**

```
:James owl:sameAs :Jim.
```

**{John} ≡ {Jim}**

```
:John :hasAge "51"^^xsd:nonNegativeInteger .
```

```
[] rdf:type owl:NegativePropertyAssertion ;  
owl:sourceIndividual :Bill ;  
owl:assertionProperty :hasWife ;  
owl:targetIndividual :Mary .
```

**¬hasWife(Bill,Mary)**

```
[] rdf:type owl:NegativePropertyAssertion ;  
owl:sourceIndividual :Jack ;  
owl:assertionProperty :hasAge ;  
owl:targetValue 53 .
```

# OWL and Description Logics

```
:Woman rdfs:subClassOf :Person .  
:Person owl:equivalentClass :Human .  
[] rdf:type owl:AllDisjointClasses ;  
    owl:members ( :Woman :Man ) .
```

**Woman ⊓ Man ⊑ ⊥**

```
:hasWife rdfs:subPropertyOf :hasSpouse .  
:hasWife rdfs:domain :Man ;  
         rdfs:range :Woman .
```

# OWL and Description Logics

```
:Mother owl:equivalentClass [  
    rdf:type owl:Class ;  
    owl:intersectionOf ( :Woman :Parent )  
] .
```

**Mother ≡ Woman  $\sqcap$  Parent**

```
:Parent owl:equivalentClass [  
    rdf:type owl:Class ;  
    owl:unionOf ( :Mother :Father )  
] .
```

**Parent ≡ Mother  $\sqcup$  Father**

```
:ChildlessPerson owl:equivalentClass [  
    rdf:type owl:Class ;  
    owl:intersectionOf ( :Person [ owl:complementOf :Parent ] )  
] .
```

**ChildlessPerson ≡ Person  $\sqcap$   $\neg$ Parent**

```
:Grandfather rdfs:subClassOf [  
    rdf:type owl:Class ;  
    owl:intersectionOf ( :Man :Parent )  
] .
```

# OWL and Description Logics

```
:Jack rdf:type [  
    rdf:type owl:Class ;  
    owl:intersectionOf ( :Person  
        [ rdf:type owl:Class ;  
          owl:complementOf :Parent ]  
    )  
] .
```

**Person ⊓ ¬Parent (Jack)**

# OWL and Description Logics

```
:Parent owl:equivalentClass [  
    rdf:type          owl:Restriction ;  
    owl:onProperty   :hasChild ;  
    owl:someValuesFrom :Person  
] .
```

**Parent  $\equiv \exists \text{hasChild}.\text{Person}$**

```
:Orphan owl:equivalentClass [  
    rdf:type          owl:Restriction ;  
    owl:onProperty   [ owl:inverseOf :hasChild ] ;  
    owl:allValuesFrom :Dead  
] .
```

**Orphan  $\equiv \forall \text{hasChild}^{-}.\text{Dead}$**

# OWL and Description Logics

```
:JohnsChildren owl:equivalentClass [  
    rdf:type          owl:Restriction ;  
    owl:onProperty   :hasParent ;  
    owl:hasValue     :John  
] .
```

**JohnsChildren  $\equiv \exists \text{hasParent}.\{\text{John}\}$**

```
:NarcisticPerson owl:equivalentClass [  
    rdf:type          owl:Restriction ;  
    owl:onProperty   :loves ;  
    owl:hasSelf      "true"^^xsd:boolean .  
] .
```

**NarcisticPerson  $\equiv \exists \text{loves}.\text{Self}$**

# OWL and Description Logics

**$\leq 4 \text{ hasChild.} \text{Parent (John)}$**

```
:John rdf:type [  
    rdf:type owl:Restriction ;  
    owl:maxQualifiedCardinality "4"^^xsd:nonNegativeInteger ;  
    owl:onProperty :hasChild ;  
    owl:onClass :Parent  
] .
```

**$\geq 2 \text{ hasChild.} \text{Parent (John)}$**

```
:John rdf:type [  
    rdf:type owl:Restriction ;  
    owl:minQualifiedCardinality "2"^^xsd:nonNegativeInteger ;  
    owl:onProperty :hasChild ;  
    owl:onClass :Parent  
] .
```

**$=3 \text{ hasChild.} \text{Parent (John)}$**

```
:John rdf:type [  
    rdf:type owl:Restriction ;  
    owl:qualifiedCardinality "3"^^xsd:nonNegativeInteger ;  
    owl:onProperty :hasChild ;  
    owl:onClass :Parent  
] .
```

# OWL and Description Logics

```
:John rdf:type [  
    rdf:type owl:Restriction ;  
    owl:cardinality "5^^xsd:nonNegativeInteger ;  
    owl:onProperty :hasChild  
] .
```

**=5 hasChild.T (John)**

```
:MyBirthdayGuests owl:equivalentClass [  
    rdf:type owl:Class ;  
    owl:oneOf ( :Bill :John :Mary )  
] .
```

**MyBirthdayGuests ≡ {Bill, John, Mary}**

# OWL and Description Logics

```
:hasParent owl:inverseOf :hasChild .  
  
:Orphan owl:equivalentClass [  
    rdf:type owl:Restriction ;  
    owl:onProperty [ owl:complementOf :hasChild ] ;  
    owl:allValuesFrom :Dead  
] .  
  
:hasSpouse rdf:type owl:SymmetricProperty .  
:hasChild rdf:type owl:AsymmetricProperty .  
:hasParent owl:propertyDisjointWith :hasSpouse .  
:hasRelative rdf:type owl:ReflexiveProperty .  
:parentOf rdf:type owl:IrreflexiveProperty .  
:hasHusband rdf:type owl:FunctionalProperty .  
:hasHusband rdf:type owl:InverseFunctionalProperty .  
:hasAncestor rdf:type owl:TransitiveProperty .
```

**Orphan  $\equiv \forall \text{hasChild} . \neg \text{Dead}$**

# OWL and Description Logics

```
:hasGrandparent owl:propertyChainAxiom ( :hasParent :hasParent ).
```

**hasParent  $\circ$  hasParent  $\sqsubseteq$  hasGrandParent**

```
:Person owl:hasKey ( :hasSSN ) .
```

In OWL 2 a collection of (data or object) properties can be assigned as a key to a class expression. This means that each named instance of the class expression is uniquely identified by the set of values which these properties attain in relation to the instance.

# OWL and Description Logics

```
:personAge owl:equivalentClass
[ rdf:type rdfs:Datatype;
  owl:onDatatype xsd:integer;                               Datatype facets
  owl:withRestrictions (
    [ xsd:minInclusive "0"^^xsd:integer ]
    [ xsd:maxInclusive "150"^^xsd:integer ]
  )
]
.

:majorAge owl:equivalentClass
[ rdf:type rdfs:Datatype;
  owl:intersectionOf (
    :personAge
    [ rdf:type rdfs:Datatype;
      owl:datatypeComplementOf :minorAge ]
  )
]
```

# OWL and Description Logics

Feature	Related OWL vocabulary	FOL	DL
top/bottom class	<code>owl:Thing/owl:Nothing</code>	(axiomatise)	$\top/\perp$
Class intersection	<code>owl:intersectionOf</code>	$\wedge$	$\sqcap$
Class union	<code>owl:unionOf</code>	$\vee$	$\sqcup$
Class complement	<code>owl:complementOf</code>	$\neg$	$\neg$
Enumerated class	<code>owl:oneOf</code>	(ax. with $\approx$ )	{a}
<b>Property restrictions</b>	<code>owl:onProperty</code>		
Existential	<code>owl:someValueFrom</code>	$\exists y \dots$	$\exists R.C$
Universal	<code>owl:allValuesFrom</code>	$\forall y \dots$	$\forall R.C$
Min. cardinality	<code>owl:minQualifiedCardinality</code> <code>owl:onClass</code>	$\exists y_1 \dots y_n \dots$	$\geq n S.C$
Max. cardinality	<code>owl:maxQualifiedCardinality</code> <code>owl:onClass</code>	$\forall y_1 \dots y_{n+1} \dots \rightarrow \dots$	$\leq n S.C$
Local reflexivity	<code>owl:hasSelf</code>	$R(x,x)$	$\exists R.Self$

Let us now define the interesting reasoning services that we can benefit from ontologies

# Terminological Reasoning of an Ontology

Let  $\mathcal{T}$  be a TBox. Terminological reasoning refers to deciding the following problems :

- ▶ **Satisfiability** : a concept  $C$  is satisfiable w.r.t.  $\mathcal{T}$  if and only if there is a model  $I$  of  $\mathcal{T}$  such that  $C^I \neq \emptyset$ .
- ▶ **Subsumption** :  $C$  is subsumed by  $D$  w.r.t.  $\mathcal{T}$  if and only if  $C^I \subseteq D^I$  for all models  $I$  of  $\mathcal{T}$ .
- ▶ **Equivalence** :  $C$  is equivalent to  $D$  w.r.t.  $\mathcal{T}$  if and only if  $C^I = D^I$  for all models  $I$  of  $\mathcal{T}$ .
- ▶ **Entailment** : An ontology  $O$  entails  $C \sqsubseteq D$ , written  $O \models C \sqsubseteq D$  if and only if  $C^I \subseteq D^I$  (resp.  $a^I \in C^I$ ,  $(a^I, b^I) \in R^I$ ) for all models  $I$  of  $O$ .

Examples (next page)

# Ontology : Reasoning Problems and Services

Examples.

- ▶  $A \sqcap \neg A$  is unsatisfiable
- ▶  $\forall r.A \sqcap \forall r.\neg A$  is unsatisfiable
- ▶  $\forall r.A \sqcap \exists r.\neg A$  is satisfiable
- ▶  $\exists r.(A \sqcap B)$  is subsumed by  $\exists r.A$
- ▶  $\{A \sqsubseteq B, B \sqsubseteq C\} \models A \sqsubseteq C$  (entailment)
- ▶  $\{A \sqsubseteq \exists r.B, B \sqsubseteq C\} \models A \sqsubseteq \exists r.C$  (entailment)

## Assertional Reasoning of an Ontology

Let  $O = (\mathcal{T}, \mathcal{A})$  be an ontology with TBox  $\mathcal{T}$  and ABox  $\mathcal{A}$ .

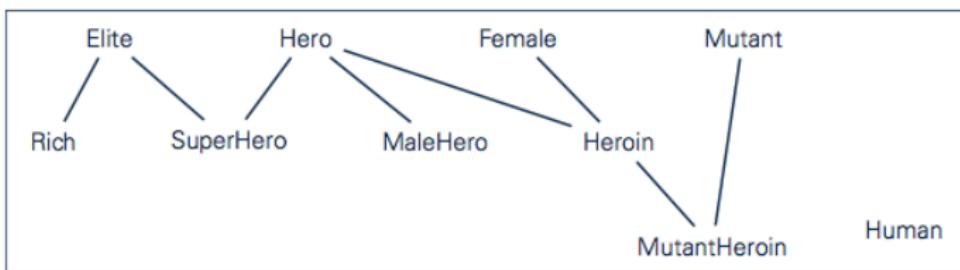
Assertional reasoning refers to deciding the following problems :

- ▶ **Consistency** :  $O$  is consistent iff  $O$  has a model.
- ▶ **Instance** :  $a$  is an instance of a concept  $C$  w.r.t  $O$  iff  $a^I \in C^I$  for all models  $I$  of  $O$

# TBox Classification

Computing the **subsumption relations** between all **concept names** in  $\mathcal{T}$ :

$$\begin{aligned} \text{Heroine} &\equiv \text{Hero} \sqcap \text{Female} \\ \text{MaleHero} &\equiv \text{Hero} \sqcap \neg\text{Female} \\ \text{MutantHeroine} &\equiv \text{Heroine} \sqcap \text{Mutant} \\ \text{Elite} &\equiv \text{Rich} \sqcup \neg\text{Human} \\ \text{Superhero} &\equiv \text{Hero} \sqcap \text{Elite} \end{aligned}$$



# Realization

Computing **the most specific** concept names to which **an individual** belongs

$$\begin{aligned}\text{Heroine} &\equiv \text{Hero} \sqcap \text{Female} \\ \text{MaleHero} &\equiv \text{Hero} \sqcap \neg\text{Female} \\ \text{MutantHeroine} &\equiv \text{Heroine} \sqcap \text{Mutant} \\ \text{Elite} &\equiv \text{Rich} \sqcap \neg\text{Human} \\ \text{Superhero} &\equiv \text{Hero} \sqcap \text{Elite}\end{aligned}$$

Hero(Superman)

Superman is an instance of

Hero, MaleHero, Elite, Superhero

# Deduction systems for $\mathcal{ALC}$

Reduction to (un)satisfiability.

Idea :

- ▶ Given ontology  $K$
- ▶ Attempt construction of a tree (called Tableau), which represents a model of  $K$ . (It's actually rather a Forest.)
- ▶ If attempt fails,  $K$  is unsatisfiable.

# The Tableau

- ▶ Nodes represent elements of the domain of the model
  - ▶ Every node  $x$  is labeled with a set  $L(x)$  of class expressions.  
 $C \in L(x)$  means : “ $x$  is in the extension of  $C$ ”
- ▶ Edges stand for role relationships :
  - ▶ Every edge  $\langle x, y \rangle$  is labeled with a set  $L(\langle x, y \rangle)$  of role names.  $R \in L(\langle x, y \rangle)$  means : “ $(x,y)$  is in the extension of  $R$ ”

# Simple Example

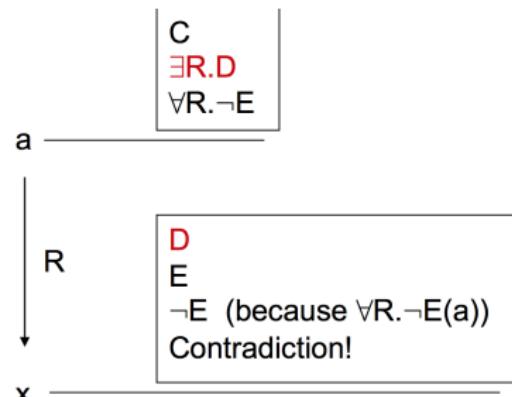
$C(a)$

$C \sqsubseteq \exists R.D$

$D \sqsubseteq E$

Does this entail  
 $(\exists R.E)(a)$ ?

(add  $\forall R.\neg E(a)$   
and show  
unsatisfiability)



## Another Example

$C(a)$

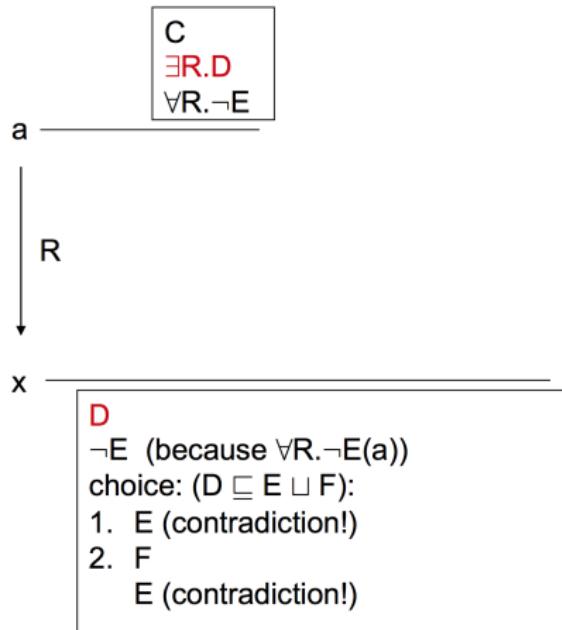
$C \sqsubseteq \exists R.D$

$D \sqsubseteq E \sqcup F$

$F \sqsubseteq E$

Does this entail  
 $(\exists R.E)(a)$ ?

(add  $\forall R.\neg E(a)$   
and show  
unsatisfiability)



# The Tableau

Tableaux algorithm with blocking for termination  
(see next lecture)

## DL vs FOL

- ▶ ALC can be seen as a fragment of First-order Logic
  - concept names are **unary predicates**
  - role names are **binary predicates**
- ▶ **Interpretations** can obviously be seen as first-order interpretations for this signature
- ▶ Concepts correspond to FOL formulae with one free variable

## Translation to FOL

Syntactic translation  $C \mapsto \pi_x(C)$

$$\pi_x(A) := A(x) \text{ for } A \in N_C$$

$$\pi_x(C \sqcap D) := \pi_x(C) \wedge \pi_x(D)$$

$$\pi_x(C \sqcup D) := \pi_x(C) \vee \pi_x(D)$$

$$\pi_x(\neg C) := \neg \pi_x(C)$$

$$\pi_x(\exists r.C) := \exists y.(r(x,y) \wedge \pi_y C), y \text{ new variable different from } x$$

$$\pi_x(\forall r.C) := \forall y.(r(x,y) \rightarrow \pi_y C)$$

Example.

$$\pi_x(\forall r.(A \sqcap \exists s.B)) = ?$$

## Translation to FOL

Syntactic translation  $C \mapsto \pi_x(C)$

$$\pi_x(A) := A(x) \text{ for } A \in N_C$$

$$\pi_x(C \sqcap D) := \pi_x(C) \wedge \pi_x(D)$$

$$\pi_x(C \sqcup D) := \pi_x(C) \vee \pi_x(D)$$

$$\pi_x(\neg C) := \neg \pi_x(C)$$

$$\pi_x(\exists r.C) := \exists y.(r(x,y) \wedge \pi_y C), y \text{ new variable different from } x$$

$$\pi_x(\forall r.C) := \forall y.(r(x,y) \rightarrow \pi_y C)$$

Example.

$$\pi_x(\forall r.(A \sqcap \exists s.B)) = ?$$

Lemma (optional).

$C$  and  $\pi_x(C)$  have the same extension ; that is,

$$C^I = \{d \in \Delta^I \mid I \models \pi_x(C)(d)\}.$$