

## Artificial Intelligence

# Description Logic (and business rules)

Dr. Mustafa Jarrar

[Sina Institute, University of Birzeit](#)

[mjarrar@birzeit.edu](mailto:mjarrar@birzeit.edu)

[www.jarrar.info](http://www.jarrar.info)





**Watch this lecture and download the slides from**  
<http://jarrar-courses.blogspot.com/2011/11/artificial-intelligence-fall-2011.html>

# This lecture

- What and Why Description Logic
- *ALC* Description Logic
- Reasoning services in Description Logic

## Lecture Keywords:

Logic, Description Logic, DL, ALC Description Logic, SHOIN, AL, DLR, Tbox, Abox, Reasoning, Reasoning services, Reasoners, Racer, HermiT, Business Rules, Conceptual Modeling, satisfiability, Unsatisfiability,

المنطق، المنطق الوصفي، الاستنباط، الاستنتاج المنطقي، مهام الاستنتاج، قواعد  
الاستنتاج، قواعد العمل، النمذجة المفاهيمية، طرق الاستنتاج، صحة الجمل  
المنطقية، الحدود، التناقض

# Reading Material

1. **All slides + everything I say**
2. Prof. Enrico Franconi: Lecture notes on Description Logic  
<http://www.inf.unibz.it/~franconi/dl/course/>
3. D. Nardi, R. J. Brachman. **An Introduction to Description Logics**. In the Description Logic Handbook, edited by F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, P.F. Patel-Schneider, Cambridge University Press, 2002, pages 5-44.  
<http://www.inf.unibz.it/~franconi/dl/course/dlhb/dlhb-01.pdf>
4. Sean Bechhofer, “The DIG Description Logic Interface: DIG/1.1 ”  
<http://racer-systems.com/dl.php?file=NativeLibraries%252FDIGinterface11.pdf&typ=file&name=DIGinterface11.pdf>

Only Sections 2.1 and 2.2 are required (= the first 32 pages)

\* The slides in this lecture are based on and modify material largely from [2]

# Why Description Logics?

Based on [2]

If FOL is directly used without some kind of restriction, then

- The structure of the knowledge/information is lost (no variables, concepts as classes, and roles as properties),
- The expressive power of FOL is too high for having good (computational properties and efficient procedures).

# Description Logics

Description logics are a family of logics concerned with knowledge representation.

A description logic is a **decidable** fragment of first-order logic, associated with a set of automatic **reasoning procedures**.

The basic constructs for a description logic are the notion of a **concept** and the notion of a **relationship**.

**Complex concept** and relationship expressions can be constructed from atomic concepts and relationships with suitable constructs between them.

Example:

$$HumanMother \sqsubseteq Female \sqcap \exists HasChild.Person$$

# Axioms, Disjunctions and Negations

Based on [2]

$$\forall x. \text{Teaching-Assistant}(x) \rightarrow \neg \text{Undergrad}(x) \vee \text{Professor}(x)$$

$$\text{Teaching-Assistant} \sqsubseteq \neg \text{Undergrad} \sqcup \text{Professor}$$

A necessary condition in order to be a teaching assistant is to be either not undergraduated or a professor. Clearly, a graduated student being a teaching assistant is not necessarily a professor; moreover, it may be the case that some professor is not graduated.

$$\forall x. \text{Teaching-Assistant}(x) \leftrightarrow \neg \text{Undergrad}(x) \vee \text{Professor}(x)$$

$$\text{Teaching-Assistant} \doteq \neg \text{Undergrad} \sqcup \text{Professor}$$

When the left-hand side is an atomic concept, the  $\sqsubseteq$  symbol introduces a primitive definition (giving only necessary conditions) while the  $\doteq$  symbol introduces a real definition, with necessary and sufficient conditions.

In general, it is possible to have complex concept expressions at the left-hand side as well.



# Description Logics

Most known description logics are :

*FL*<sup>-</sup> The simplest and less expressive description logic.  
 $C, D \rightarrow A \mid C \sqcap D \mid \forall R.C \mid \exists R$

*ALC* A more practical and expressive description logic.  
 $C, D \rightarrow A \mid \top \mid \perp \mid \neg A \mid C \sqcap D \mid \forall R.C \mid \exists R.\top$

*SHOIN* Very popular description logic.  
The logic underlying OWL.

*DLR*<sub>idf</sub> Very expressive description logic,  
Capable of representing most database constructs.



# *ALC* Description logic (Syntax and Semantic)

Constructor	Syntax	Semantics
Primitive concept	$A$	$A^I \subseteq \Delta^I$
Primitive role	$R$	$R^I \subseteq \Delta^I \times \Delta^I$
Top	$\top$	$\Delta^I$
Bottom	$\perp$	$\emptyset$
Complement	$\neg C$	$\Delta^I \setminus C^I$
Conjunction	$C \sqcap D$	$C^I \cap D^I$
Disjunction	$C \sqcup D$	$C^I \cup D^I$
Universal quantifier	$\forall R.C$	$\{x \mid \forall y. R^I(x,y) \rightarrow C^I(y)\}$
Extensional quantifier	$\exists R.C$	$\{x \mid \exists y. R^I(x,y) \wedge C^I(y)\}$

## Examples:

Woman  $\sqsubseteq$  Person  $\sqcap$  Female

Parent  $\sqsubseteq$  Person  $\sqcap \exists \text{hasChild}.\top$

Man  $\sqsubseteq$  Person  $\sqcap \neg \text{Female}$

NotParent  $\sqsubseteq$  Person  $\sqcap \exists \text{hasChild}.\perp$

# Closed Propositional Language

Based on [2]

**Conjunction** ( $\sqcap$ ) is interpreted as *intersection of sets of individuals*.

**Disjunction** ( $\sqcup$ ) is interpreted as *union of sets of individuals*.

**Negation** ( $\neg$ ) is interpreted as *complement of sets of individuals*.

$$\exists R. \top \Leftrightarrow \exists R.$$

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

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

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

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

# Formal Semantics

Based on [2]

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

a nonempty set  $\Delta^I$  (the *domain*)

a function  $.^I$  (the *interpretation function*)

that maps

- every *individual* to an element of  $\Delta^I$
- every *concept* to a subset of  $\Delta^I$
- every *role* to a subset of  $\Delta^I \times \Delta^I$

An interpretation function  $.^I$  is an **extension function** if and **only if it satisfies the** semantic definitions of the language.

# DL Knowledge Base

DL Knowledge Base ( $\Sigma$ ) normally separated into two parts:

$$\Sigma = \langle \mathbf{Tbox}, \mathbf{Abox} \rangle$$

- TBox (**Terminological Box**) is a set of axioms in the form of  $(\mathbf{C} \sqsubseteq \mathbf{D}, \mathbf{C} \doteq \mathbf{D})$  describing structure of domain (i.e., schema),

Example:

HappyFather  $\doteq$  Man  $\sqcap$   $\exists$ hasChild.Female

Elephant  $\sqsubseteq$  Animal  $\sqcap$  Large  $\sqcap$  Grey

- ABox (Assertion Box) is a set of axioms in the form of  $(\mathbf{C(a)}, \mathbf{R(a, b)})$  describing a concrete situation (data),

Example:

HappyFather (John)

hasChild(John, Mary)

# Knowledge Bases (Example)

Based on [2]

## Tbox:

$\text{Student} \doteq \text{Person} \sqcap \exists \text{NAME.String} \sqcap$   
 $\exists \text{ADDRESS.String} \sqcap$   
 $\exists \text{ENROLLED.Course}$   
 $\exists \text{TEACHES.Course} \sqsubseteq \neg \text{Undergrad} \sqcap \text{Professor}$

## Abox:

$\text{Student}(\text{Ali})$   
 $\text{ENROLLED}(\text{Ali}; \text{Comp338})$   
 $(\text{Student} \sqcup \text{Professor})(\text{Dima})$

# TBox: Descriptive Semantics

Based on [2]

An interpretation  $I$  satisfies the statement  $C \sqsubseteq D$  if  $C^I \subseteq D^I$ .

An interpretation  $I$  satisfies the statement  $C \doteq D$  if  $C^I = D^I$ .

An interpretation  $I$  is a model for a *TBox*  $T$  if  $I$  satisfies all statements in  $T$ .

# Abox Interpretation

Based on [2]

If  $I = (\Delta^I, .^I)$  is an interpretation,

$C(a)$  is satisfied by  $I$  if  $a^I \in C^I$ .

$R(a, b)$  is satisfied by  $I$  if  $(a^I, b^I) \in R^I$ .

A set  $A$  of assertions is called an ABox.

An interpretation  $I$  is said to be a *model* of the ABox  $A$  if every assertion of  $A$  is satisfied by  $I$ . The ABox  $A$  is said to be *satisfiable* if it admits a model.

An interpretation  $I = (\Delta^I, .^I)$  is said to be a *model* of a knowledge base  $\Sigma$  if every axiom of  $\Sigma$  is satisfied by  $I$ .

A knowledge base  $\Sigma$  is said to be *satisfiable* if it admits a model.



# Logical Implication

Based on [2]

$\Sigma \models \alpha$  if every model of  $\Sigma$  is a model of  $\alpha$

*Example:*

TBox:

$\exists \text{TEACHES.Course} \sqsubseteq \neg \text{Undergrad} \sqcup \text{Professor}$

ABox:

$\text{TEACHES}(\text{Rami}, \text{Comp338}), \text{Course}(\text{comp388}),$   
 $\text{Undergrad}(\text{Rami})$

$\Sigma \models \text{Professor}(\text{Rami})$  ?

# Logical Implication

Based on [2]

*What if:*

TBox:

$\exists \text{TEACHES.Course} \sqsubseteq \text{Undergrad} \sqcup \text{Professor}$

ABox:

$\text{TEACHES}(\text{Rami}, \text{Comp388}), \text{Course}(\text{Comp388}),$   
 $\text{Undergrad}(\text{Rami})$

$\Sigma \models \text{Professor}(\text{Rami}) \text{ ?}$

$\Sigma \models \neg \text{Professor}(\text{Rami}) \text{ ?}$

# Reasoning Services

Based on [2]

- Remember that a DL is typically associated with reasoning procedures.
- There are several primitive/common reasoning services that most DL reasoners support:

## Concept Satisfiability

$$\Sigma \models C \equiv \perp \quad \text{Student} \sqcap \neg \text{Person}$$

the problem of checking whether  $C$  is satisfiable w.r.t.  $\Sigma$ , i.e. whether there exists a model  $I$  of  $\Sigma$  such that  $C^I \neq \emptyset$

## Subsumption

$$\Sigma \models C \sqsubseteq D \quad \text{Student} \sqsubseteq \text{Person}$$

the problem of checking whether  $C$  is subsumed by  $D$  w.r.t.  $\Sigma$ , i.e. whether  $C^I \subseteq D^I$  in every model  $I$  of  $\Sigma$

## Satisfiability

$$\Sigma \models \quad \text{Student} \doteq \neg \text{Person}$$

the problem of checking whether  $\Sigma$  is satisfiable, i.e. whether it has a model.

# Reasoning Services (cont.)

Based on [2]

## Instance Checking

$\Sigma \models C(a)$       Professor(john)

the problem of checking whether the assertion  $C(a)$  is satisfied in every model of  $\Sigma$

## Retrieval

$\{a \mid \Sigma \models C(a)\} \text{ Professor} \Rightarrow \text{Dima}$

## Realization

$\{C \mid \Sigma \models C(a)\} \text{ Dima} \Rightarrow \text{Professor}$

# Reduction to Satisfiability

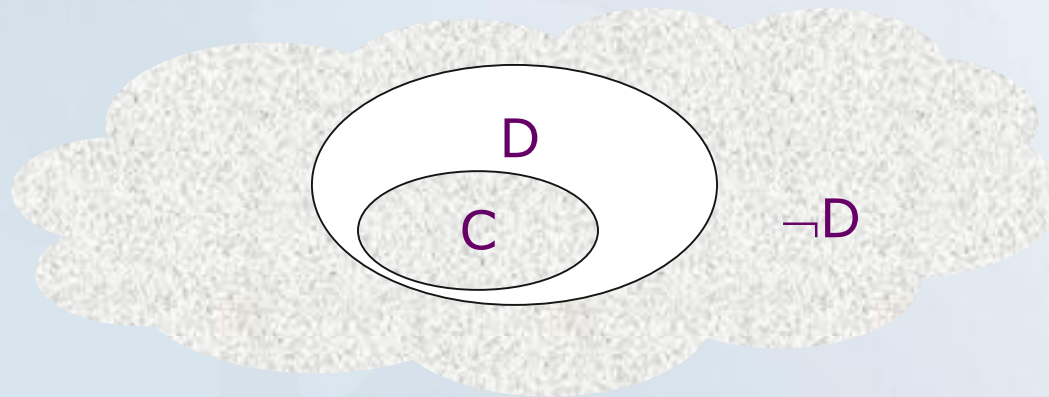
Based on [2]

## Concept Satisfiability

$\Sigma \models C \equiv \perp \leftrightarrow$  exists  $x$  s.t.  $\Sigma \cup \{C(x)\}$  has a model.

## Subsumption

$\Sigma \models C \sqsubseteq D \leftrightarrow \Sigma \cup \{C \sqcap \neg D(x)\}$  has no models.



## Instance Checking

$\Sigma \models C(a) \leftrightarrow \Sigma \cup \{\neg C(x)\}$  has no models.

# Some extensions of *ALC*

Constructor	Syntax	Semantics
Primitive concept	$A$	$A^I \subseteq \Delta^I$
Primitive role	$R$	$R^I \subseteq \Delta^I \times \Delta^I$
Top	$\top$	$\Delta^I$
Bottom	$\perp$	$\emptyset$
Complement	$\neg C$	$\Delta^I \setminus C^I$
Conjunction	$C \sqcap D$	$C^I \cap D^I$
Disjunction	$C \sqcup D$	$C^I \cup D^I$
Universal quantifier	$\forall R.C$	$\{x \mid \forall y. R^I(x,y) \rightarrow C^I(y)\}$
Extensional quantifier	$\exists R.C$	$\{x \mid \exists y. R^I(x,y) \wedge C^I(y)\}$

# Some extensions of *ALC*

Constructor	Syntax	Semantics
Primitive concept	$A$	$A^I \subseteq \Delta^I$
Primitive role	$R$	$R^I \subseteq \Delta^I \times \Delta^I$
Top	$\top$	$\Delta^I$
Bottom	$\perp$	$\phi$
Complement	$\neg C$	$\Delta^I \setminus C^I$
Conjunction	$C \sqcap D$	$C^I \cap D^I$
Disjunction	$C \sqcup D$	$C^I \cup D^I$
Universal quantifier	$\forall R.C$	$\{x \mid \forall y. R^I(x,y) \rightarrow C^I(y)\}$
Extensional quantifier	$\exists R.C$	$\{x \mid \exists y. R^I(x,y) \wedge C^I(y)\}$
Cardinality ( $N$ )	$\geq n R$	$\{x \mid \#\{y \mid R^I(x,y)\} \geq n\}$
	$\leq n R$	$\{x \mid \#\{y \mid R^I(x,y)\} \leq n\}$
Qual. cardinality ( $Q$ )	$\geq n R.C$	$\{x \mid \#\{y \mid R^I(x,y) \wedge C^I(y)\} \geq n\}$
	$\leq n R.C$	$\{x \mid \#\{y \mid R^I(x,y) \wedge C^I(y)\} \leq n\}$
Enumeration ( $O$ )	$\{a_1 \dots a_n\}$	$\{a_1^I \dots a_n^I\}$
Selection ( $F$ )	$f : C$	$\{x \in \text{Dom}(f^I) \mid C^I(f^I(x))\}$



# Cardinality Restriction

Based on [2]

Role quantification *cannot express that a woman has at least 3 (or at most 5) children.*

Cardinality restrictions can express conditions on the number of fillers:

$$\text{BusyWoman} \doteq \text{Woman} \sqcap (\exists^{\geq 3} \text{CHILD})$$

$$\text{ConsciousWoman} \doteq \text{Woman} \sqcap (\exists^{\leq 5} \text{CHILD})$$

Notice:

$$(\exists^{\geq 1} R) \Leftrightarrow (\exists R.)$$

# Cardinality Restriction

Based on [2]

$\text{BusyWoman} \doteq \text{Woman} \sqcap (\exists^{\geq 3} \text{CHILD})$

$\text{ConsciousWoman} \doteq \text{Woman} \sqcap (\exists^{\leq 5} \text{CHILD})$

Mary: Woman,  
CHILD:John,  
CHILD:Sui,  
CHILD:Karl

$\models \text{ConsciousWoman}(\text{Mary})$  ?

# Roles as Functions

Based on [2]

A role is *functional*, if the filler functionally depends on the individual, i.e., the role can be considered as a function:

$$R(x, y) \Leftrightarrow f(x) = y.$$

For example, the roles CHILD and PARENT are not functional, while the roles MOTHER and AGE are functional.

If a role is functional, we write:

$$\exists f.C \equiv f.c \quad (\textit{selection operator})$$

# Individuals

Based on [2]

In every interpretation different individuals are assumed to denote different elements, i.e. for every pair of individuals  $a, b$ , and for every interpretation  $I$ , if  $a \neq b$  then  $a^I \neq b^I$ .

This is called the *Unique Name Assumption* and is usually assumed in database applications.

**Example:** How many children does this family have?

`Family(f), Father(f, john), Mother(f, sue),  
Son(f, paul), Son(f, george), Son(f, alex)`

`| = (≥3 Son) (f)`

# Enumeration Type (one-of)

$\text{Weekday} \doteq \{\text{mon}, \text{tue}, \text{wed}, \text{thu}, \text{fri}, \text{sat}, \text{sun}\}$

$\text{Weekday}^I \doteq \{\text{mon}^I, \text{tue}^I, \text{wed}^I, \text{thu}^I, \text{fri}^I, \text{sat}^I, \text{sun}^I\}$

$\text{Citizen} \doteq (\text{Person} \sqcap \forall \text{LIVES}.\text{Country})$

$\text{Palestinian} \doteq (\text{Citizen} \sqcap \forall \text{LIVES}.\{\text{Palestine}\})$

# Racer ( <http://www.racer-systems.com/products/racerpro/index.phtm> )

Racer Systems GmbH & Co. KG : Products - Mozilla Firefox

http://www.racer-systems.com/products/racer

Racer logic

## Racer

Products Downloads About us News Contact Impressum

- Home
- Products
  - RacerPro**
    - Version 2.0 preview
    - Features
    - Release notes
    - Manual
  - RacerPorter
  - RacerPlus
  - License options
  - Update guidelines
  - Feedback / Support
  - How to order
  - Tools
  - Downloads
  - Allegro CL and Franz Inc. products
- Services
- Technology
- Company
- Website

Expand menu

[ English · German · Italian ]

Last modified: 10/31/2009 8:39 PM

[ Feedback for this page ]

### What is RacerPro? An overview:

RACER stands for **R**enamed **A**Box and **C**oncept **E**xpression **R**easoner. RacerPro is the commercial name of the software.

The origins of RacerPro are within the area of description logics. Since description logics provide the foundation of international approaches to standardize ontology languages in the context of the so-called semantic web, RacerPro can also be used as a system for managing semantic web ontologies based on OWL (e.g., it can be used as a reasoning engine for ontology editors such as Protégé). However, RacerPro can also be seen as a semantic web information repository with optimized retrieval engine because it can handle large sets of data descriptions (e.g., defined using RDF). Last but not least, the system can also be used for modal logics such as Km.

### RacerPro as a Semantic Web Reasoning System and Information Repository

The semantic web is aimed at providing "machine-understandable" web resources or by augmenting existing resources with "machine-understandable" meta data. An important aspect of future systems exploiting these resources is the ability to process OWL (Web Ontology Language) documents (OWL KBs), which is the official semantic web ontology language. Ontologies may be taken off the shelf or may be extended for domain-specific purposes (domain-specific ontologies extend core ontologies). For doing this, a reasoning system is required as part of the ontology editing system. RacerPro can process OWL Lite as well as OWL DL documents (knowledge bases). Some restrictions apply, however. OWL DL documents are processed with approximations for nominals in class expressions and user-defined XML datatypes are not yet supported.

A first implementation of the semantic web rule language (SWRL) is provided with RacerPro 1.9 (see the [Release Notes](#) and the [User Guide](#) for more information about a description of the semantics of rules in this initial version).

# Description Logic Reasoners

- For example:

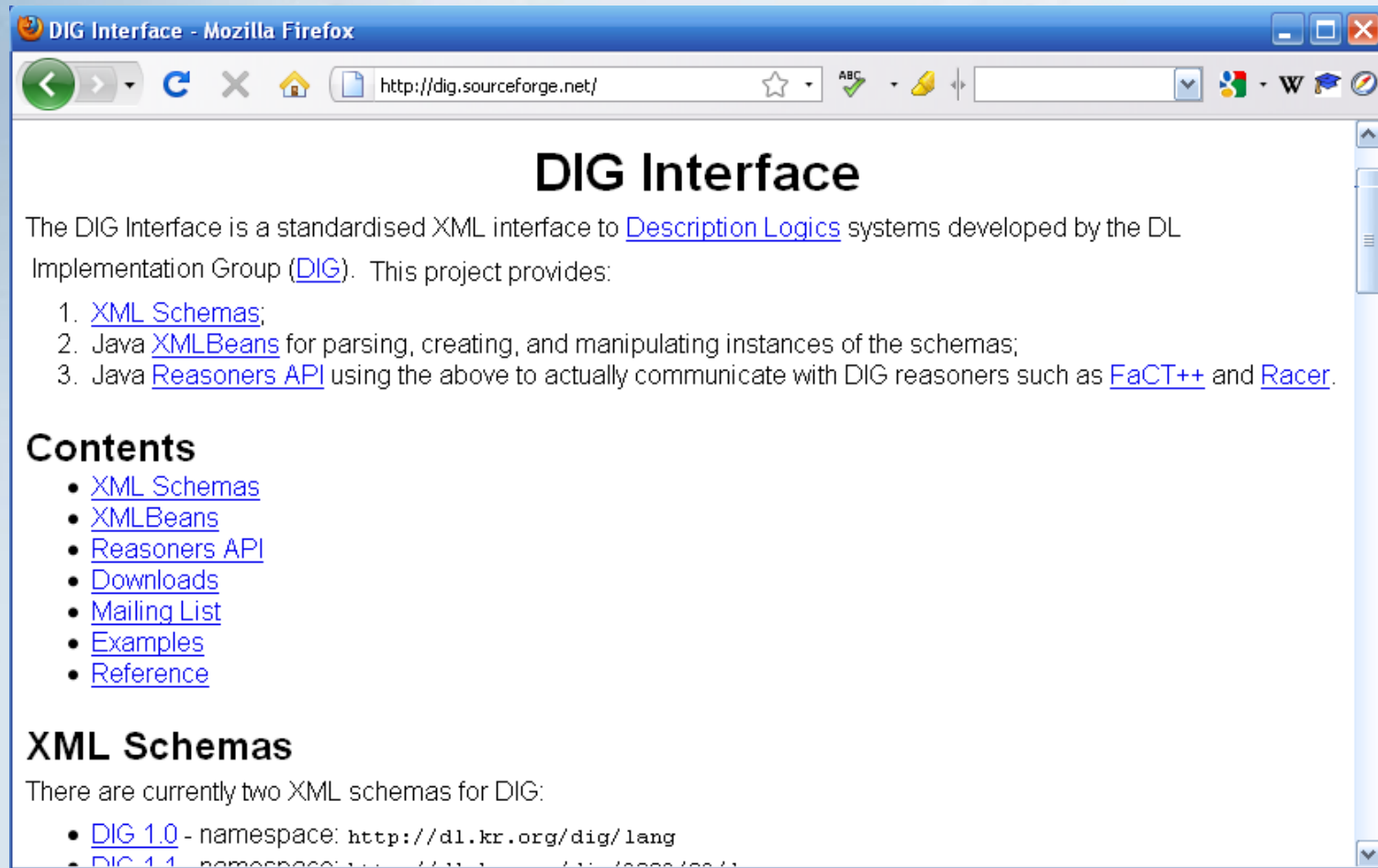


- They offer reasoning services for multiple TBoxes and ABoxes.
- They run as background reasoning engines.
- They understand DIG, which is a simple protocol (based on HTTP) along with an XML Schema.
- Example:  $Student \sqsubseteq Person$

```
<impliesc>
  <catom name="Student"/>
  <catom name="Person"/>
</impliesc>
```



# DIG Interface (<http://dig.sourceforge.net/>)



# DIG Protocol

DIG is only an XML schema for a description logic along with **ask/tell** functionality

You write a new Knowledge base (using the DIG XML syntax), and send it to Racer using the TELL functionality.

You can then write you Query/Question (using the DIG, XML syntax), and send it to Racer using the ASK functionality.

You may communicate with the Racer through HTTP or SOAP.

# Create e a new Knowledge Base

## The newKB Message

```
<?xml version="1.0" encoding="UTF-8"?>
<newKB
xmlns="http://dl.kr.org/dig/2003/02/lang"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://dl.kr.org/dig/2003/02/lang
http://dl-web.man.ac.uk/dig/2003/02/dig.xsd"/>
```

## The Response Message

```
<?xml version="1.0" encoding="UTF-8"?>
<response
xmlns="http://dl.kr.org/dig/2003/02/lang"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://dl.kr.org/dig/2003/02/lang
http://dl-web.man.ac.uk/dig/2003/02/dig.xsd">
<kb uri="urn:uuid:abcdefgh-1234-1234-12345689ab"/>
```

This URI should then be used during TELL and ASK requests made against the knowledge base

# Tell Syntax

Based on [4]

A TELL request must contain in its body a tells element, which itself consists of a number of tell statements.

Example:      $\text{Driver} \sqsubseteq \text{Person} \sqcap \exists \text{Drives.Vehicle}$

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<tells
  xmlns="http://dl.kr.org/dig/2003/02/lang"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://dl.kr.org/dig/2003/02/lang
http://dl-web.man.ac.uk/dig/2003/02/dig.xsd"
  uri="urn:uuid:abcdefgh-1234-1234-12345689ab">
  <defconcept name="driver"/>
  <equalc>
    <catom name="driver"/>
    <and>
      <catom name="person"/>
      <some>
        <ratom name="drives"/>
        <catom name="vehicle"/>
      </some>
    </and>
  </equalc>
  <defconcept name="person"/>
  <defconcept name="vehicle"/>
  <defrole name="drives"/>
</tells>
```

# Tell Syntax

Tell Language	
Primitive Concept Introduction	<code>&lt;defconcept name="CN"/&gt;</code> <code>&lt;defrole name="CN"/&gt;</code> <code>&lt;deffeature name="CN"/&gt;</code> <code>&lt;defattribute name="CN"/&gt;</code> <code>&lt;defindividual name="CN"/&gt;</code>
Concept Axioms	<code>&lt;implies&gt;C1 C2&lt;/implies&gt;</code> <code>&lt;equalc&gt;C1 C2&lt;/equalc&gt;</code> <code>&lt;disjoint&gt;C1... Cn&lt;/disjoint&gt;</code>
Role Axioms	<code>&lt;impliesr&gt;R1 R2&lt;/impliesr&gt;</code> <code>&lt;equalr&gt;R1 R2&lt;/equalr&gt;</code> <code>&lt;domain&gt;R E&lt;/domain&gt;</code> <code>&lt;range&gt;R E&lt;/range&gt;</code> <code>&lt;rangeint&gt;R&lt;/rangeint&gt;</code> <code>&lt;rangestring&gt;R&lt;/rangestring&gt;</code> <code>&lt;transitive&gt;R&lt;/transitive&gt;</code> <code>&lt;functional&gt;R&lt;/functional&gt;</code>
Individual Axioms	<code>&lt;instanceof&gt;I C&lt;/instanceof&gt;</code> <code>&lt;related&gt;I1 R I2&lt;/related&gt;</code> <code>&lt;value&gt;I A V&lt;/value&gt;</code>

Concept Language	
Primitive Concepts	<code>&lt;top/&gt;</code> <code>&lt;bottom/&gt;</code> <code>&lt;catom name="CN"/&gt;</code>
Boolean Operators	<code>&lt;and&gt;E1... En&lt;/and&gt;</code> <code>&lt;or&gt;E1... En&lt;/or&gt;</code> <code>&lt;not&gt;E&lt;/not&gt;</code>
Property Restrictions	<code>&lt;some&gt;R E&lt;/some&gt;</code> <code>&lt;all&gt;R E&lt;/all&gt;</code> <code>&lt;atmost num="n"&gt;R E&lt;/atmost&gt;</code> <code>&lt;atleast num="n"&gt;R E&lt;/atleast&gt;</code> <code>&lt;iset&gt;I1... In&lt;/iset&gt;</code>
Concrete Domain Expressions	<code>&lt;defined&gt;A&lt;/defined&gt;</code> <code>&lt;stringmin val="s"&gt;A&lt;/stringmin&gt;</code> <code>&lt;stringmax val="s"&gt;A&lt;/stringmax&gt;</code> <code>&lt;stringequals val="s"&gt;A&lt;/stringequals&gt;</code> <code>&lt;stringrange min="s" max="t"&gt;A&lt;/stringrange&gt;</code> <code>&lt;intmin val="i"&gt;A&lt;/intmin&gt;</code> <code>&lt;intmax val="i"&gt;A&lt;/intmax&gt;</code> <code>&lt;intequals val="i"&gt;A&lt;/intequals&gt;</code> <code>&lt;intrange min="i" max="j"&gt;A&lt;/intrange&gt;</code>
Role Expressions	<code>&lt;ratom name="CN"/&gt;</code> <code>&lt;feature name="CN"/&gt;</code> <code>&lt;inverse&gt;R&lt;/inverse&gt;</code> <code>&lt;attribute name="CN"/&gt;</code> <code>&lt;chain&gt;F1... FN A&lt;/chain&gt;</code> Individuals <code>&lt;individual name="CN"/&gt;</code>

# Ask Syntax

An ASK request must contain in its body an asks element.  
Multiple queries in one request is possible.

```
<?xml version="1.0"?>
<asks
  xmlns="http://dl.kr.org/dig/2003/02/lang">
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://dl.kr.org/dig/2003/02/lang"
  http://dl-web.man.ac.uk/dig/2003/02/dig.xsd"
  uri="urn:uuid:abcdefgh-1234-1234-12345689ab">
```

**KB  $\models$  Vehicle**  
asks about satisfiability of  
the Vehicle concept

```
<satisfiable id="q1">
  <catom name="Vehicle"/>
</satisfiable>
```

**$a \mid \Sigma \models \text{Peron}(a) \sqcap \exists \text{Drives.Vehicle}$**   
asks for all those concepts subsumed by  
the description given, i.e. all the drivers

```
<descendants id="q2">
  <and>
    <catom name="person"/>
    <some>
      <ratom name="drives"/>
      <catom name="vehicle"/>
    </some>
  </and>
</descendants>
```

**$C \mid \Sigma \models C(\text{JohnSmith})$**

asks for the known types of the  
given individual

```
<types id="q3">
  <individual name="JohnSmith"></individual>
</types>
```

```
</asks>
```

# Ask Syntax

Ask Language	
Primitive Concept Retrieval	<code>&lt;allConceptNames/&gt;</code> <code>&lt;allRoleNames/&gt;</code> <code>&lt;allIndividuals/&gt;</code>
Satisfiability	<code>&lt;satisfiable&gt;C&lt;/satisfiable&gt;</code> <code>&lt;subsumes&gt;C1 C2&lt;/subsumes&gt;</code> <code>&lt;disjoint&gt;C1 C2&lt;/disjoint&gt;</code>
Concept Hierarchy	<code>&lt;parents&gt;C&lt;/parents&gt;</code> <code>&lt;children&gt;C&lt;/children&gt;</code> <code>&lt;ancestors&gt;C&lt;/ancestors&gt;</code> <code>&lt;descendants&gt;C&lt;/descendants&gt;</code> <code>&lt;equivalents&gt;C&lt;/equivalents&gt;</code>
Role Hierarchy	<code>&lt;rparents&gt;R&lt;/rparents&gt;</code> <code>&lt;rchildren&gt;R&lt;/rchildren&gt;</code> <code>&lt;rancestors&gt;R&lt;/rancestors&gt;</code> <code>&lt;rdescendants&gt;R&lt;/rdescendants&gt;</code>
Individual Queries	<code>&lt;instances&gt;C&lt;/instances&gt;</code> <code>&lt;types&gt;I&lt;/types&gt;</code> <code>&lt;instance&gt;I C&lt;/instance&gt;</code> <code>&lt;roleFillers&gt;I R&lt;/roleFillers&gt;</code> <code>&lt;relatedIndividuals&gt;R&lt;/relatedIndividuals&gt;</code>

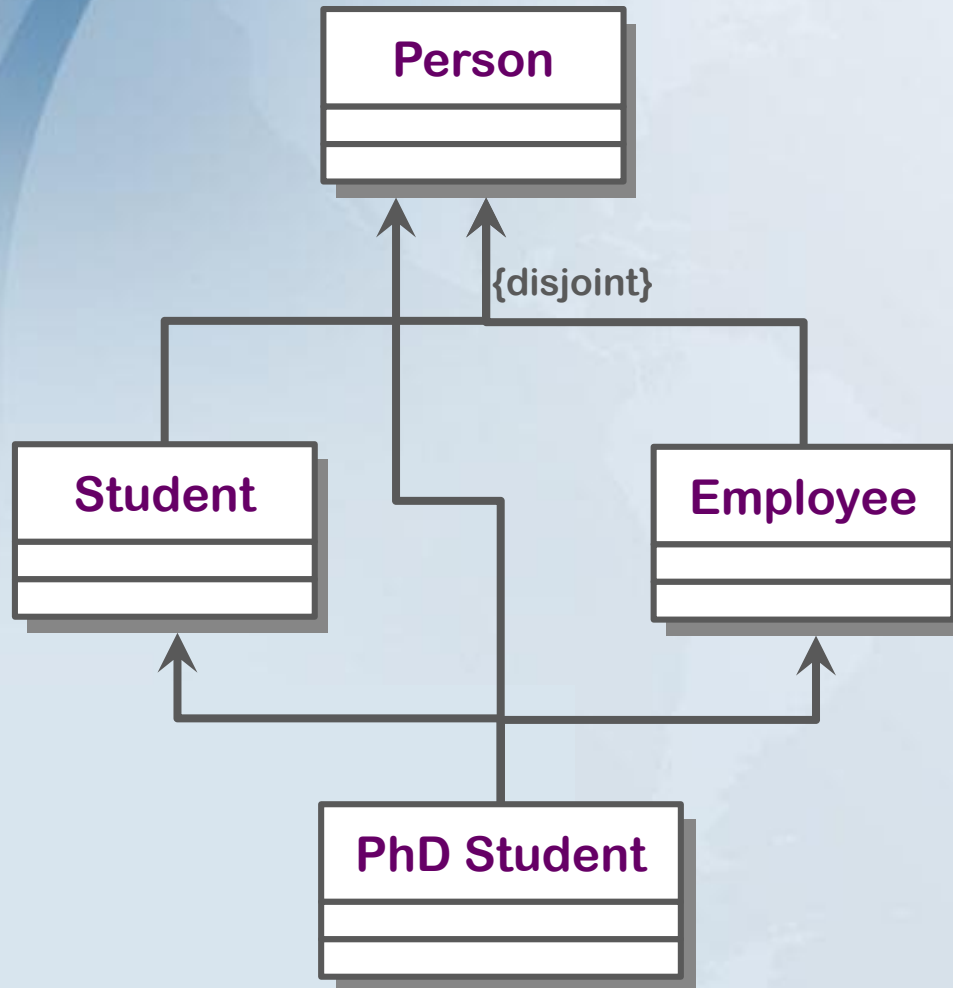




# ◦ Examples of Using Description Logic in Conceptual Modeling and Business rules

# UML Class diagram

(with a contradiction and an implication)



$\text{Student} \sqsubseteq \text{Person}$

$\text{PhDStudent} \sqsubseteq \text{Person}$

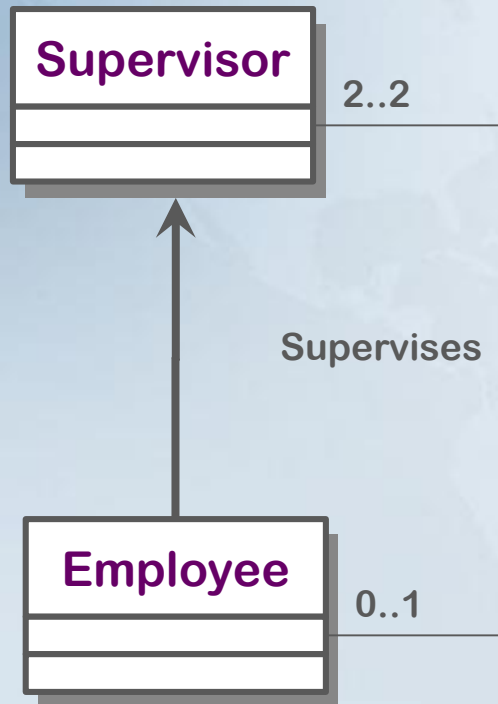
$\text{Employee} \sqsubseteq \text{Person}$

$\text{PhD Student} \sqsubseteq \text{Student} \sqcap \text{Employee}$

$\text{Student} \sqcap \text{Employee} \doteq \perp$

# Infinite Domain: the democratic company

Based on [3]



$\text{Supervisor} \sqsubseteq \exists^{=2} \text{Supervises.Employee}$

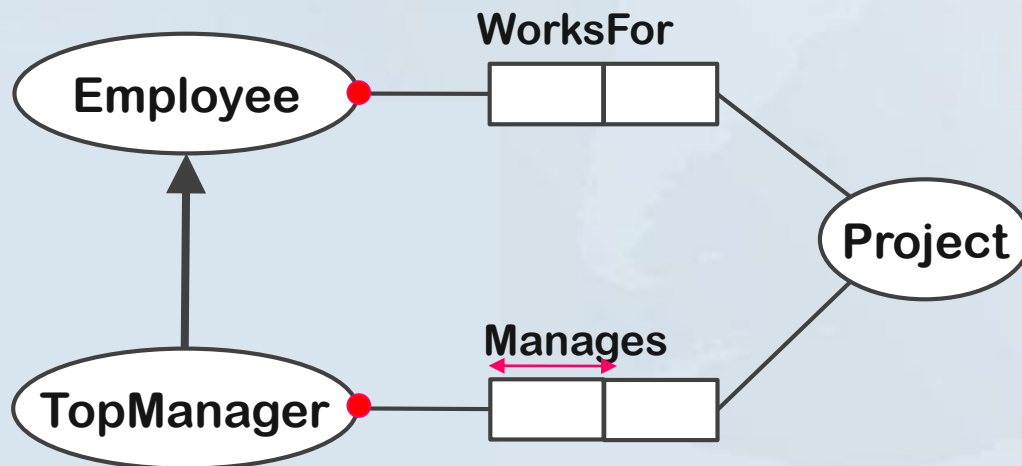
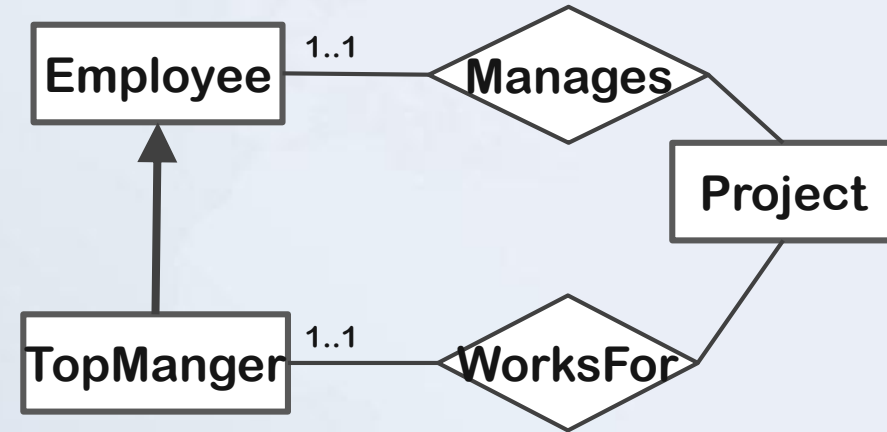
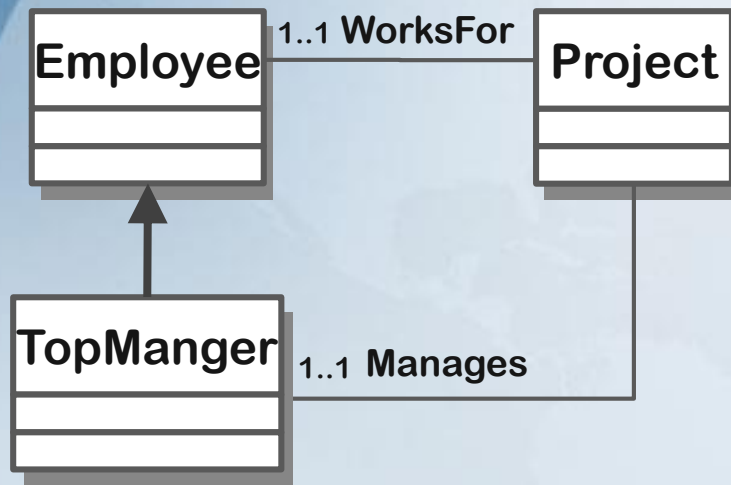
$\text{Employee} \sqsubseteq \text{Supervisor}$

*implies*

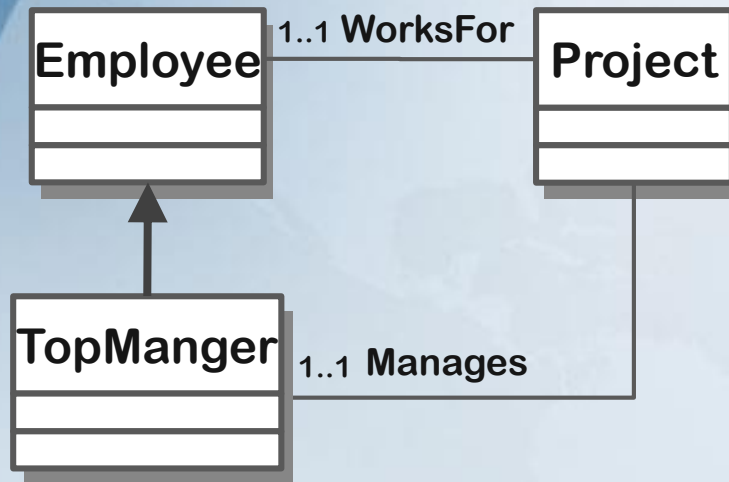
“the classes **Employee** and **Supervisor** necessarily contain an infinite number of instances”.

Since legal world descriptions are **finite** possible worlds satisfying the constraints imposed by the conceptual schema, **the schema is inconsistent**.

# Example (in UML, EER and ORM)



# Example (in UML, EER and ORM)



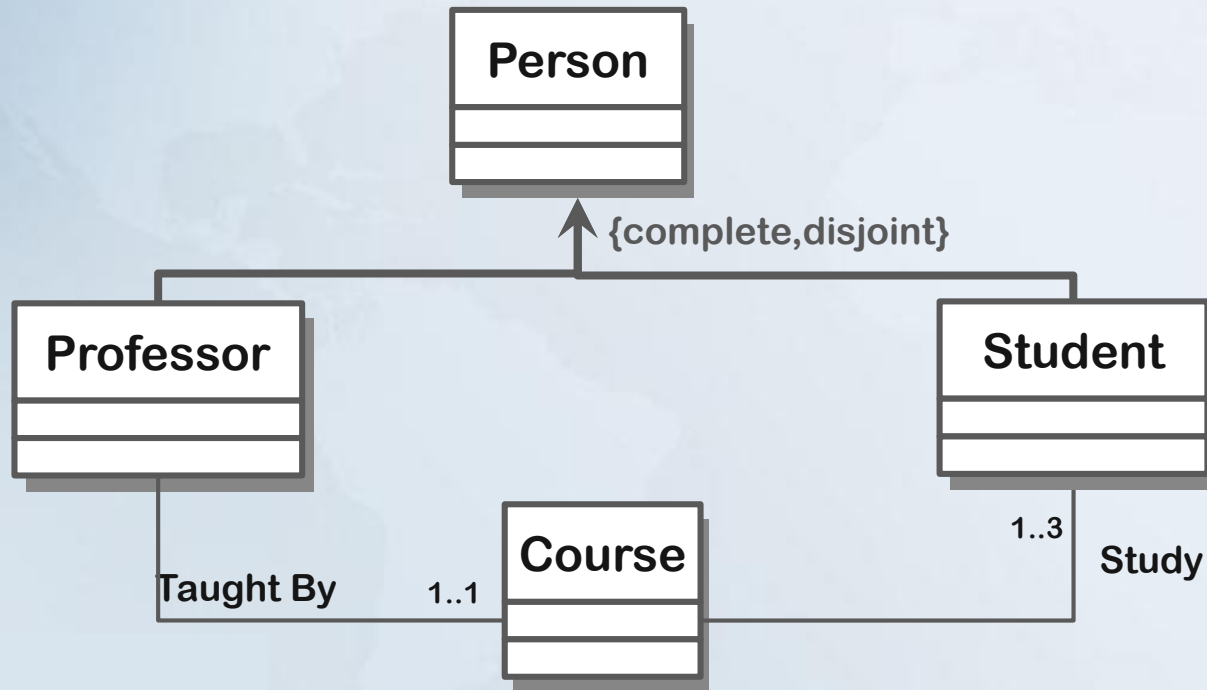
$\text{Employee} \sqsubseteq \exists^1 \text{WorksFor.Project}$

$\text{TopManager} \sqsubseteq \text{Employee} \sqcap \exists^1 \text{Manages.Project}$

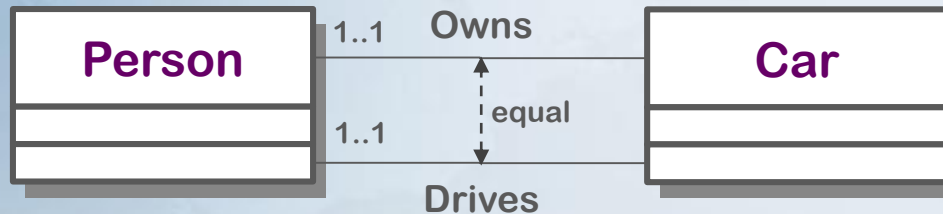


$\text{WorksFor.Project} \sqsubseteq \text{Manages.Project}$

# Another Example



# Another Example



$\text{Person} \sqsubseteq \exists^{=1} \text{Owns.Car} \sqcap \exists^{=1} \text{Drives.Car}$

$\text{Drive} \doteq \text{Owns}$

- The first 1..1 cardinality constraint means that every person must own one car.
- The second 1..1 cardinality constraint means that every person must drive one car.
- The equal constraint means that every person who owns a car is allowed to only drive that car, and vice versa.

➔ The equal constraint is implied by both cardinality constraints.

# Homework (Reason about UML/EER Diagrams)

- 1- Create a UML/EER diagram that contains some contradictions and implications.
- 2- Formulate this diagram in description logic,
- 3- Write at least 5 questions (reasoning services) to know whether the schema/concept/rule is satisfiable, and 3 questions whether something in the schema is implied?

Hint: contradictions\implication can be achieved throw the wrong use of disjointness and cardinality constraints (see examples in the next slide).

➔ Please remark that this project is not only to help you practice Description Logics, but also: 1) build correct UML/EER models and find problems automatically, 2) Reason about rules and business rules, and 3) you think of another usage (open your mind)!

Each student should deliver one pdf file, contains: 1) the diagram 2) its formalization in DL, 3) the reasoning questions.



# Ontology

Recall that a TBox can be used to specify the meaning of a terminology.  
That is, specify meaning in logic.

Recall that a TBox can be depicted in EER/UML

➔ You may build your TBox in OWL (the Ontology Web Language), and share it on the web, so that others can use it as a reference to meaning of a terminology (ontology).

➔ This will be the topic of the coming lectures.