Artificial Intelligence

Description Logic
(and business rules)

Dr. Mustafa Jarrar

Sina Institute, University of Birzeit

mjarrar@birzeit.edu

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
- $\mathcal{ALC}$ 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]

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

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

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.

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

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

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 :

$\mathcal{FL}^-$ The simplest and less expressive description logic.

$C, D \rightarrow A \mid C \sqcap D \mid \forall R.C \mid \exists R$

$\mathcal{ALC}$ A more practical and expressive description logic.

$C, D \rightarrow A \mid \top \mid \bot \mid \neg A \mid C \sqcap D \mid \forall R.C \mid \exists R.\top$

$\mathcal{SHOIN}$ Very popular description logic.
The logic underlying OWL.

$\mathcal{DLR}_{idf}$ Very expressive description logic,
Capable of representing most database constructs.

$\mathcal{ALC}$ 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	$\top$	Δ^I
Bottom	$\bot$	ϕ
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:

$\text{Woman} \sqsubseteq \text{Person} \sqcap \text{Female}$

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

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

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

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.*

$$\exists R.\top \Leftrightarrow \exists \mathrm{R}.$$
$$\neg(C \sqcap D) \Leftrightarrow \neg C \sqcup \neg D$$
$$\neg(C \sqcup D) \Leftrightarrow \neg C \sqcap \neg D$$
$$\neg(\forall \mathrm{R}.\mathrm{C}) \Leftrightarrow \exists R.\neg C$$
$$\neg(\exists R.C) \Leftrightarrow \forall R.\neg C$$

Formal Semantics

Based on [2]

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

a nonempty set Δ^I (the *domain)*

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

that maps

– every *individual to an element of* Δ^I

– every *concept to a subset of* Δ^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 (Σ) normally separated into two parts:

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

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

  Example:

  $\text{HappyFather} \doteq \text{Man} \sqcap \exists \text{hasChild.Female}$

  $\text{Elephant} \sqsubseteq \text{Animal} \sqcap \text{Large} \sqcap \text{Grey}$

- ABox (Assertion Box) is a set of axioms in the form of (C(a), 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:

Student(Ali)

ENROLLED(Ali; Comp338)

(Student ⊔ Professor)(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* Σ if every axiom of Σ is satisfied by I.

A knowledge base Σ is said to be *satisfiable if it admits a model.*

Logical Implication

Based on [2]

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

Example:

TBox:

$\exists$TEACHES.Course $\sqsubseteq$ $\neg$Undergrad $\sqcup$ Professor

ABox:

TEACHES(Rami,Comp338), Course(comp388),
Undergrad(Rami)

$\Sigma \models$ Professor(Rami) ?

Logical Implication

Based on [2]

What if:

TBox:

$\exists$TEACHES.Course $\sqsubseteq$ Undergrad $\sqcup$ Professor

ABox:

TEACHES(Rami,Comp388), Course(Comp388), Undergrad(Rami)

$\Sigma \models$ Professor(Rami) ?

$\Sigma \models \neg$Professor(Rami) ?

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

$\Sigma \not\models C \equiv \bot$ Student $\sqcap \neg$Person

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

Subsumption

$\Sigma \models C \sqsubseteq D$ Student $\sqsubseteq$ Person

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

Satisfiability

$\Sigma \models$ Student $\doteq \neg$Person

the problem of checking whether Σ 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 Σ

Retrieval

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

Realization

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

Reduction to Satisfiability

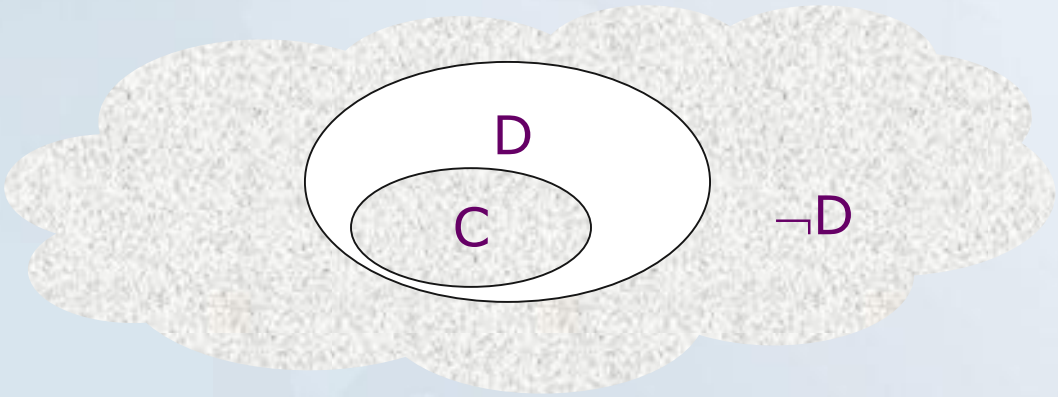
Based on [2]

Concept Satisfiability

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

Subsumption

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



Instance Checking

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

Some extensions of $\mathcal{ALC}$

Constructor	Syntax	Semantics
Primitive concept	A	$A^I \subseteq \Delta^I$
Primitive role	R	$R^I \subseteq \Delta^I \times \Delta^I$
Top	$\top$	Δ^I
Bottom	$\bot$	ϕ
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 $\mathcal{ALC}$

Constructor	Syntax	Semantics
Primitive concept	A	$A^I \subseteq \Delta^I$
Primitive role	R	$R^I \subseteq \Delta^I \times \Delta^I$
Top	$\top$	Δ^I
Bottom	$\bot$	ϕ
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)	$\geq n\ R$	$\{x \mid \#\{y \mid R^I(x,y)\} \geq n\}$
	$\leq n\ R$	$\{x \mid \#\{y \mid R^I(x,y)\} \geq n\}$
Qual. cardinality (Q)	$\geq n\ R.C$	$\{x \mid \#\{y \mid R^I(x,y) \wedge C^I(y)\} \geq n\}$
	$\leq n\ R.C$	$\{x \mid \#\{y \mid R^I(x,y) \wedge C^I(y)\} \geq n\}$
Enumeration (O)	$\{a_1 \ \ldots\ a_n\}$	$\{a^I_1 \ \ldots\ a^I_n\}$
Selection (F)	$f : C$	$\{x \in \mathrm{Dom}(f^I) \mid C^I(f^I(x))\}$

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

$\models$ ConsciousWoman(Mary) ?

Roles as Functions

Based on [2]

A role is *functional, is 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:

$\exists f.C \equiv f.c$ *(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)

Description Logic Reasoners

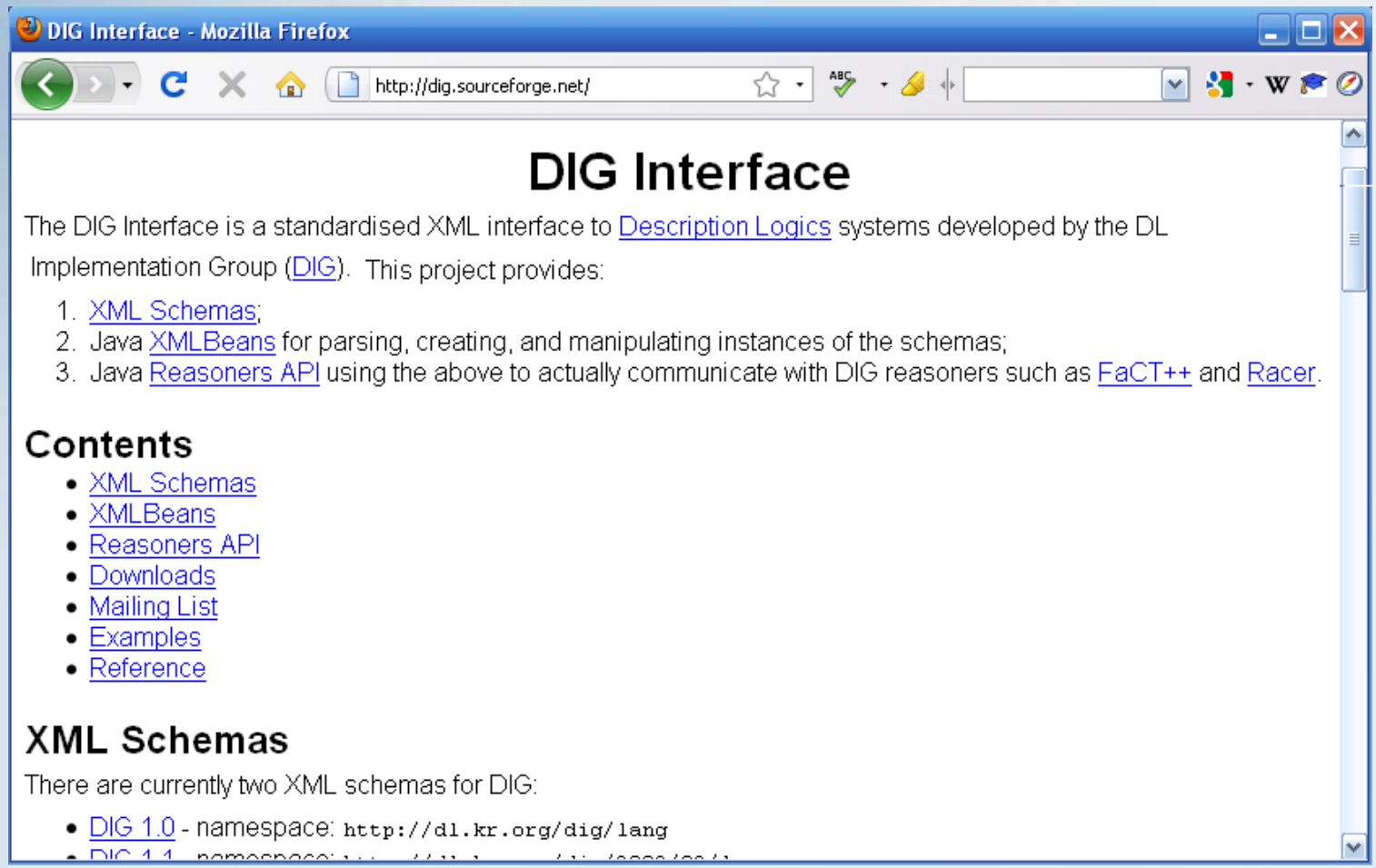
- For example:

HermiT Racer FaCT++ pellet

- 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: Driver ⊑ Person ⊓ ∃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	<defconcept name="CN"/> <defrole name="CN"/> <deffeature name="CN"/> <defattribute name="CN"/> <defindividual name="CN"/>
Concept Axioms	<impliesc>C1 C2</impliesc> <equalc>C1 C2</equalc> <disjoint>C1... Cn</disjoint>
Role Axioms	<impliesr>R1 R2</impliesc> <equalr>R1 R2</equalr> <domain>R E</domain> <range>R E</range> <rangeint>R</rangeint> <rangestring>R</rangestring> <transitive>R</transitive> <functional>R</functional>
Individual Axioms	<instanceof>I C</instanceof> <related>I1 R I2</related> <value>I A V</value>

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

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">
    <satisfiable id="q1">
      <catom name="Vehicle"/>
    </satisfiable>
    <descendants id="q2">
      <and>
        <catom name="person"/>
        <some>
          <ratom name="drives"/>
          <catom name="vehicle"/>
        </some>
      </and>
    </descendants>
    <types id="q3">
      <individual name="JohnSmith"></individual>
    </types>
</asks>
```

$KB \models Vehicle$

asks about satisfiability of the Vehicle concept

$a \mid \Sigma \models \text{Peron(a)} \sqcap \exists \text{Drives.Vehicle}$

asks for all those concepts subsumed by the description given, i.e. all the drivers

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

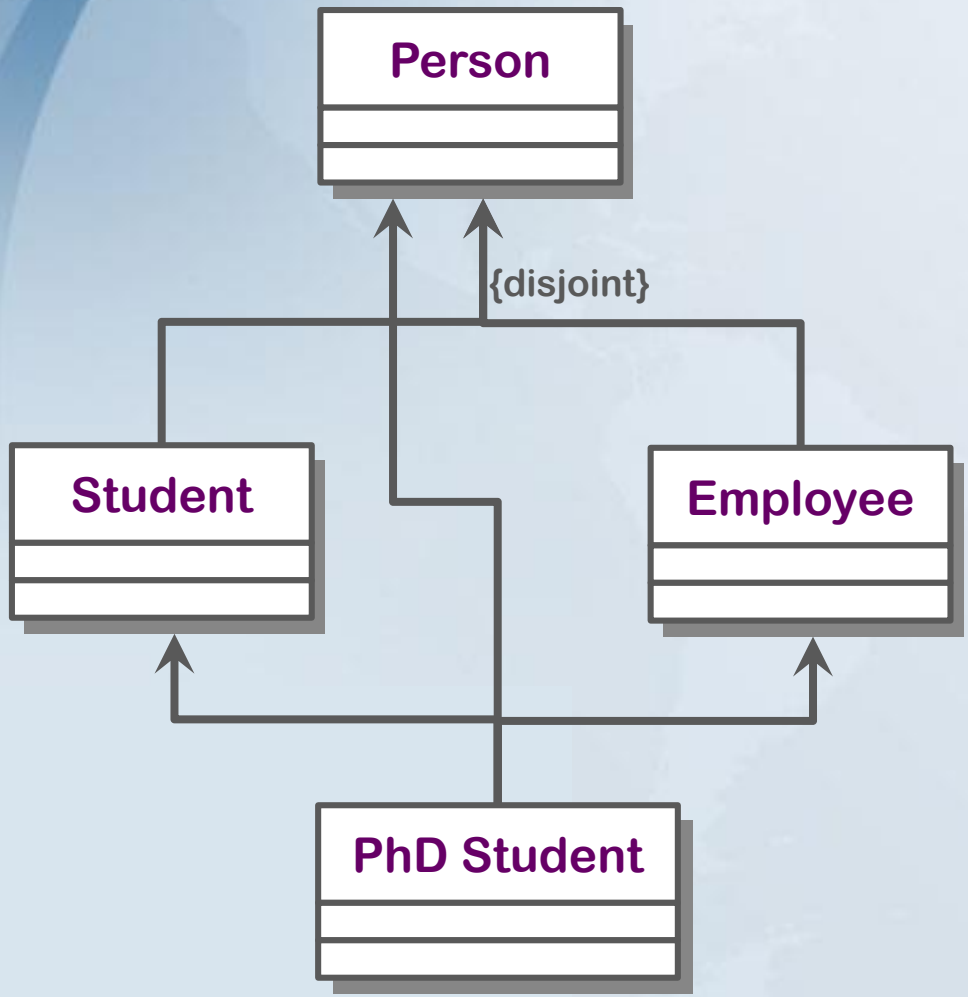
asks for the known types of the given individual

Ask Syntax

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

- Examples of Using Description Logic in Conceptual Modeling and Business rules

UML Class diagram
(with a contradiction and an implication)



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

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

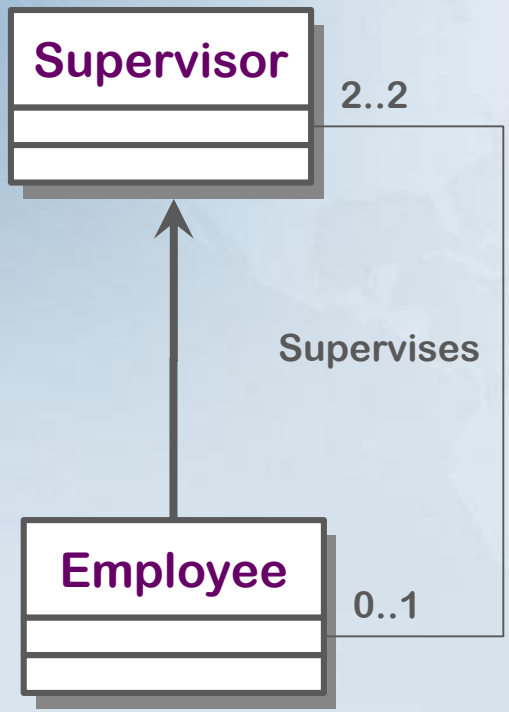
$\text{Employee} \sqsubseteq \text{Person}$

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

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

Infinite Domain: the democratic company

Based on [3]



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

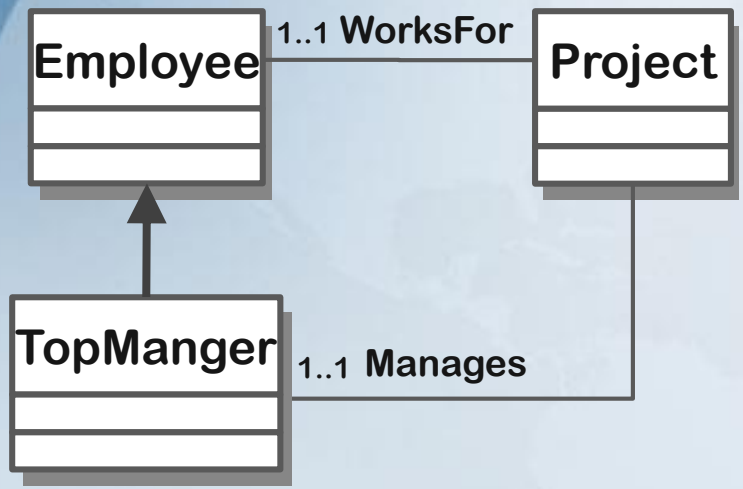
$$\text{Employee} \sqsubseteq \text{Supervisor}$$

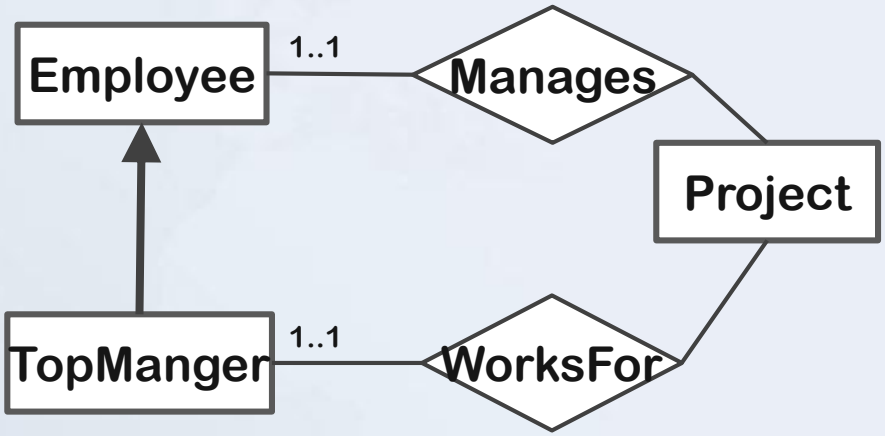
implies

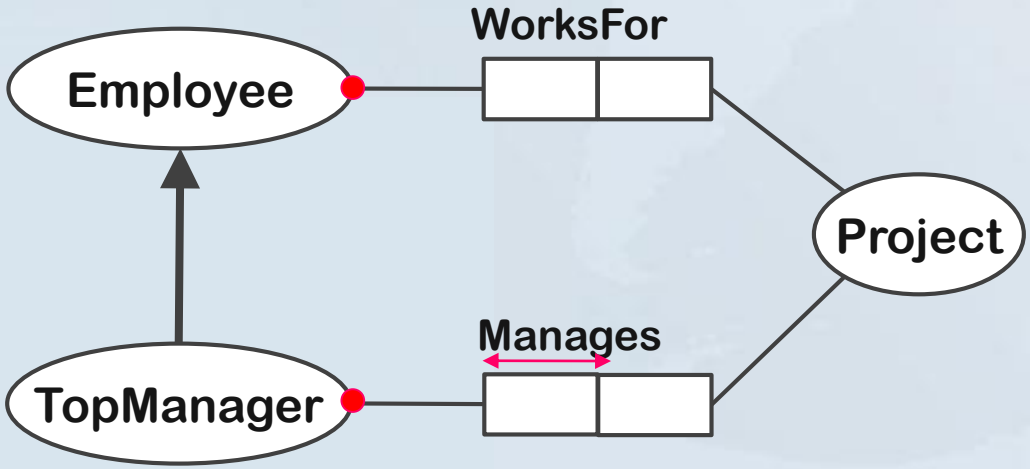
"the classes Employee and Supervisor necessarily contain an infinite number of instances".

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

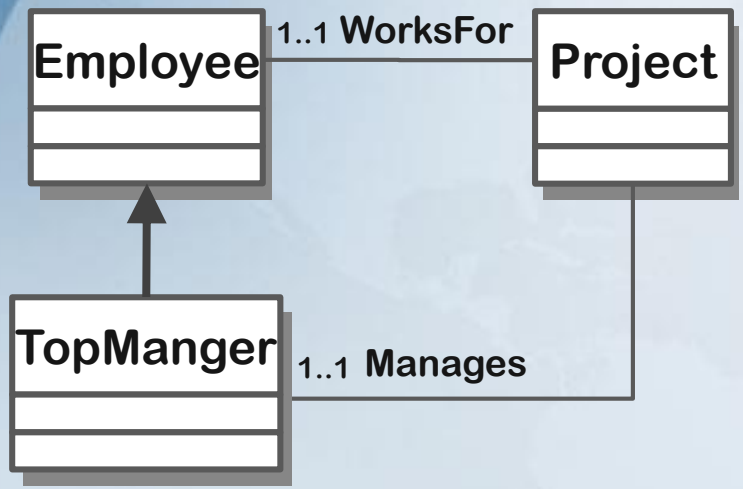
Example (in UML, EER and ORM)







Example (in UML, EER and ORM)

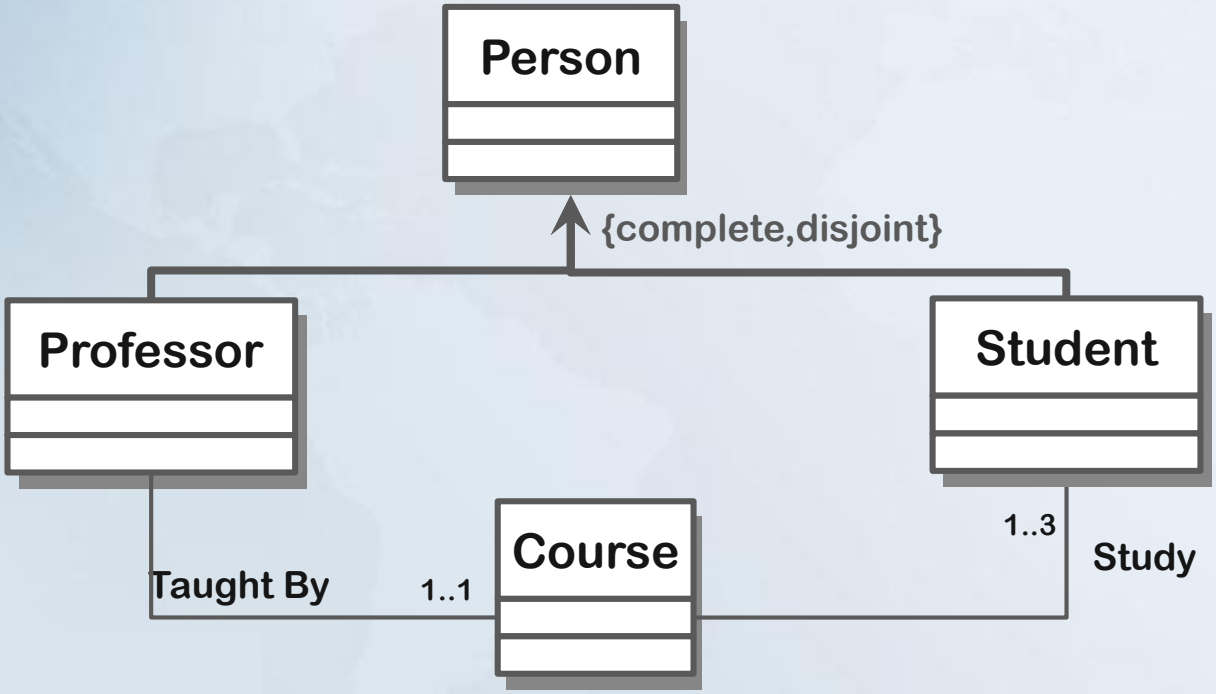


$\text{Employee} \sqsubseteq \exists^{=1} \text{WorksFor.Project}$

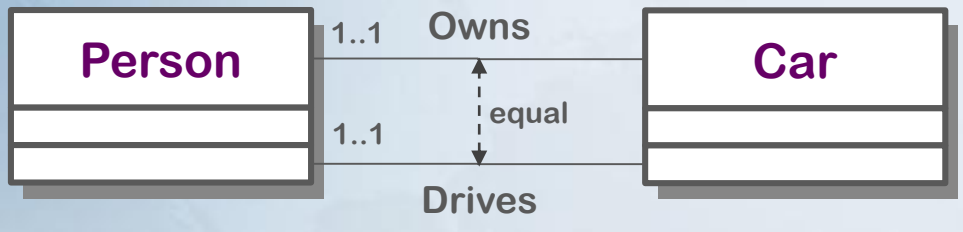
$\text{TopManager} \sqsubseteq \text{Employee} \sqcap \exists^{=1} \text{Manages.Project}$

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

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 owns one car.
- The second1..1 cardinality constraint means that every person must drives 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 that others can use it a reference to meaning of a terminology (ontology).

➔This will be the topic of the coming lectures.