

### Conceptual Modelling in Description Logic

INFS4206/INFS7206

School of Information Technology and Electrical Engineering

Semester 2, 2006



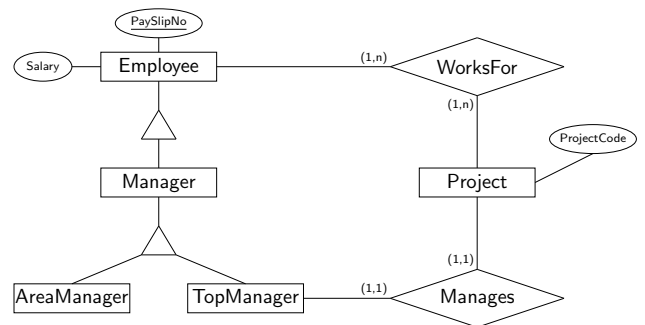
### What is an ontology

- ▶ An ontology is a formal conceptualisation of the world.
- ▶ An ontology specifies a set of **constraints**, which declare what should necessarily hold in any possible worlds.
- ▶ any possible world should conform to the constraints expressed in the ontology.
- ▶ Given an ontology, a *legal world description* is a possible world satisfying the constraints.

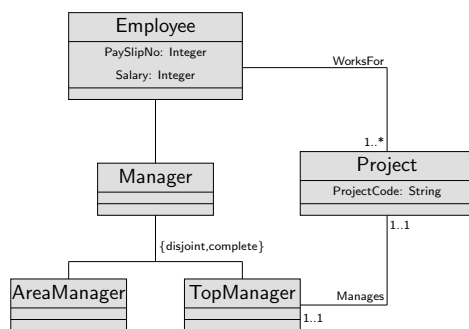
### Ontologies languages

- ▶ An ontology language usually introduces **concepts** (classes, entities), **properties** of concepts (slots, attributes, roles), **relationships** between concepts (associations), and additional **concepts**.
- ▶ Ontology languages may be simple (e.g., having only concepts), frame-based (having only concepts and properties), or logic-based (e.g. Ontolingua and OWL).
- ▶ Ontology languages are typically expressed by means of diagrams.
- ▶ The Entity-Relationship conceptual data model and UML Class Diagrams can be considered as ontology languages.

### Entity-Relationship Diagram

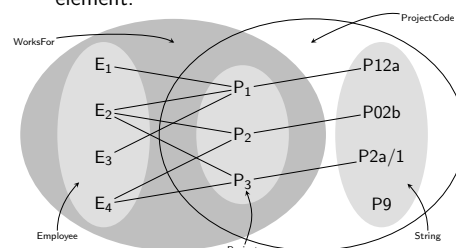


### UML Class diagram

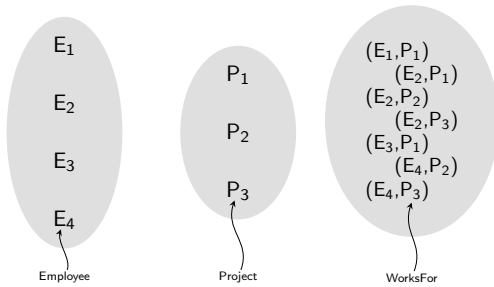


### Meaning of basic constructs

- ▶ An entity is a set of instances.
- ▶ An association (n-ary relationship) is a set of pair (n-tuples) of instances.
- ▶ An attribute is a set of pairs of an instance and a domain element.



A world is described by sets of instances



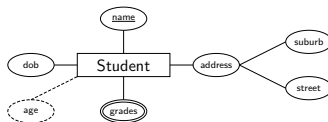
The relational representation

Employee	Project	ProjectCode	
employeeld	projectld	projectld	pcode
E <sub>1</sub>	P <sub>1</sub>	P <sub>1</sub>	"P12a"
E <sub>2</sub>	P <sub>2</sub>	P <sub>2</sub>	"P02b"
E <sub>3</sub>	P <sub>3</sub>	P <sub>3</sub>	"P2a/1"
E <sub>4</sub>			

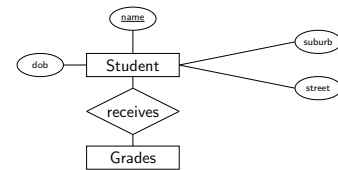
  

WorksFor	
projectld	employeeld
E <sub>1</sub>	P <sub>1</sub>
E <sub>2</sub>	P <sub>1</sub>
E <sub>2</sub>	P <sub>2</sub>
E <sub>2</sub>	P <sub>3</sub>
E <sub>3</sub>	P <sub>1</sub>
E <sub>4</sub>	P <sub>2</sub>
E <sub>4</sub>	P <sub>3</sub>

Meaning of Entities and Attributes



Meaning of Entities and Attributes



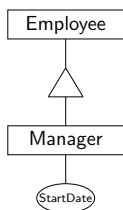
- ▶ An entity is a set of instances.
  - ▶ The key attribute is unique and it is used to identify the distinct instances of the entity
- ▶ An attribute is a set of pairs of an instance and a domain element.
  - ▶ every attribute corresponds to a concept and the association between the attribute and the entity is a role

$Student \sqsubseteq \exists hasStreet.Street \sqcap \exists hasDob.Dob \sqcap \exists hasSuburb.Suburb$

Meaning of ISA

An ISA describes a relationship between two entities/concepts

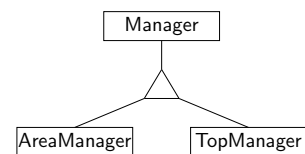
$Manager \sqsubseteq Employee$



Every attribute of the superclass is also an attribute of the subclass, but not the other way around

$Manager \sqsubseteq Employee \sqcap \exists hasStartDate.StartDate$

Meaning of disjoint and total constraints



- ISA:  $AreaManager \sqsubseteq Manager$
- ISA:  $TopManager \sqsubseteq Manager$
- Disjoint:  $AreaManager \sqcap TopManager \equiv \perp$
- Total:  $Manager \sqsubseteq AreaManager \sqcap TopManager$

### Meaning of Relationships



An association (n-ary relationship) is a set of pair (n-tuples) of instances.

$$WorksFor^I \subseteq Employee^I \times Project^I$$

Hence a binary relationship corresponds to a role, and we have referential integrity constraints from the relationship and the entities involved in it.

$$Employee \sqsubseteq \forall worksFor. Project$$

$$Project \sqsubseteq \forall workFor^{-}. Employee$$

### Meaning of Cardinality Constraints

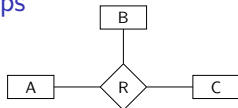


$$Employee^I \subseteq \{e \mid \min \leq \#(worksFor^I \cap (\{e\} \times \Delta)) \leq \max\}$$

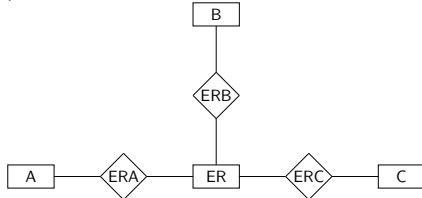
$$Employee^I \subseteq \{e \mid \min \leq \#\{p \mid (e, p) \in workFor^I \wedge p \in Project^I\}\}$$

if  $\min = 0$   $Employee \sqsubseteq \forall worksFor. Project$   
 if  $\min > 0$   $Employee \sqsubseteq \exists^{\min} \leq worksFor. Project$   
 if  $\max = n$   $Employee \sqsubseteq \exists^{\max} \geq worksFor. Project$

### n-ary Relationships



Roles are binary relations. Thus we have to **reify** the relationship (i.e., reduce n-ary relationships to binary relations)



### Reasoning

Given an ontology – seen as a collection of constraints – it is possible that additional constraints can be inferred.

- ▶ An entity is inconsistent if it denotes always the empty set.
- ▶ An entity is a sub-entity of another entity if the former denotes a subset of the set denoted by the latter.
- ▶ Two entities are equivalent if they denote the same set.
- ▶ ...

### Building Knowledge Bases

In order to build good KBs some choices must be done during its design. It is important to well understand some subtle distinctions:

- ▶ Primitive vs. Defined.
- ▶ Definitional vs. Incidental.
- ▶ Concept vs. Individual.
- ▶ Concept vs. Role.

### When to Use Primitive Concepts?

- ▶ some concepts can not be completely defined (e.g., natural kinds);
- ▶ it can be not convenient/useful to completely define a concept;
- ▶ sooner or later we must end up with something not completely defined (*encyclopedic knowledge* cannot be given).

Thus, primitive concepts must be used when:

- ▶ there is no other way;
- ▶ even if it were defined, no (automatic) classification below it will be never required by the application.

Typically primitive concepts lie in the top region of the taxonomy.

### When to use defined concepts?

- ▶ ontological reason: it is easy and natural (in the context of the application) to give a complete definition of the concept;
- ▶ organisation of the antecedents of rules;
- ▶ capturing complete descriptions used by rules for populating primitive concepts.

### Definitional vs. Incidental

Are **incidental** all the properties that are contingent features for a concept, and thus must not be part of its definition.

$\forall intelligence.Stupid$

is **incidental** for **Chicken** while

$\forall reproducesWith.Egg$

is **definitional**.

### Concept vs. Individual

- ▶ the set of individuals is a countable, discrete set;
- ▶ the concept space is ideally continuous and infinite;
- ▶ each individual has a clear identity: even if two individual have the same properties, they are distinct;
- ▶ if two concepts have equivalent descriptions, they denote the same concept;
- ▶ individual descriptions can be modified;
- ▶ concept descriptions can not be modified;
- ▶ individual update does not (usually) change the concept hierarchy;
- ▶ rules applies only to individuals.

### Concept vs. Individual, cont.

Nevertheless, it is not always easy to decide whether an object should be a concept or an individual. The main issue to deal with is the “granularity” level.

#### Example

Consider the KB describing courses in ITEE: is “Distributed Enterprise Computing (INFS4206)” course a concept or an individual?

A key to solve the problem could be asking the domain/application expert: “how many courses do you have?”, in order to understand the needed granularity.

### Concept vs. Role

Is not always easy to decide what must be a concept and what a role.

- ▶ *Person*: it is a concept.
- ▶ *MOTHER*:
  - ▶ consider “Sue is a new mother” and “Sue is the mother of Tom”.
  - ▶ *Mother* as a concept does not exist if we don’t consider the “role she plays” in a parental relation, i.e., if “Sue is a new mother” she must be the *mother* of somebody!

$Mother \equiv Woman \sqcap \exists mother.Person$

### Concept vs. Role

- ▶ But when the role is not the only important component of the definition, this dual use is not so neat.
- ▶ Another problem is *the reading direction*: in the above example the role could be, *mother* or *child*.
- ▶ A clear convention must be stated, possibly creating long, non ambiguous names for roles, as, e.g.: *hasChild*, *isTheParentOf*

---

---

---

---

---

---

## How to design a KB in steps

1. **Enumerate Objects.** As a bare list of elements of the KB; they will become individuals, concepts, or role.
2. **Distinguish Concepts from Roles.** Make a first decision about what object must be considered role; remember that some could have a "natural" concept associated. The remaining objects will be concepts (or maybe individuals). Also, try to distinguish roles from attributes.
3. **Develop Concept Taxonomy.** Try to decide a classification of all the concepts, imagining their extensions. This taxonomy will be used as a first reference, and could be revised when definition will be given. It will be used also to check if definition meet our expectations (sometime, interesting, unforeseen (re)classifications are found).

---

---

---

---

---

---

## How to design a KB in steps

4. **Devise partitions.** Try to make explicit all the disjointness and covering constraints among classes, and reclassify the concepts.
5. **Individuals.** Try to list as many as possible generally useful individuals. Some could have been already listed in step 1. Try to describe them (classify).
6. **Properties and Parts.** Begin to define the internal structure of concepts (this process will continue in the next steps). For each concept list:
  - ▶ *intrinsic* properties, that are part of the very nature of the concept;
  - ▶ *extrinsic* properties, that are contingent or external properties of the object; they can sometime change during the time;
  - ▶ *parts*, in the case of structured or collective objects. They can be physical (e.g., "the components of a car", "a students of a class", "the members of a group") or abstract (e.g., "the courses of a meal", "the lessons of a course", "the topics of a lesson" ).

---

---

---

---

---

---

## How to design a KB in steps

- ▶ In some cases some relationships between individuals of classes can be considered too accidental to be listed above.
- ▶ In general, the above distinctions depend on the level of detail adopted.
- ▶ Some of the listed roles will be later considered definitional, and some incidental.
- ▶ After this and the next steps check/revision of the taxonomy could be necessary.

---

---

---

---

---

---

## How to design a KB in steps

7. **Cardinality Restrictions.** For the relevant roles for each concept.
8. **Value Restriction.** As above. Also, chose the right restriction.
9. **Propagate Value Restrictions.** If some value restrictions stated in the previous step does not correspond to already existing concepts, they must be defined.
10. **Inter-role Relationship.** Even if hardly definable in DL, they can be useful during the populating and debugging phases.
11. **Definitional and Incidental.** It is important distinguish between definitional and incidental properties, w.r.t. to the particular application.
12. **Primitive and Defined.** As above.