Description Logics

Logics and Ontologies

Enrico Franconi

`franconi@cs.man.ac.uk`

`http://www.cs.man.ac.uk/~franconi`

Department of Computer Science, University of Manchester

Summary

- What is an ontology
- Ontology languages
- Formalising ontologies with set theory
- Reasoning in ontologies
- Formalising ontologies with first order logic
- Integrity constraints
- The i●com ontology design tool

What is an Ontology

- An ontology is a formal conceptualisation of the world.

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 world.

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 world.
- Any possible world should conform to the constraints expressed by the ontology.

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 world.
- Any possible world should conform to the constraints expressed by the ontology.
- Given an ontology, a *legal world description* is a possible world satisfying the constraints.

Ontology languages

- An ontology language usually introduces **concepts** (aka classes, entities), **properties** of concepts (aka slots, attributes, roles), **relationships** between concepts (aka associations), and additional **constraints**.

Ontology languages

- An ontology language usually introduces **concepts** (aka classes, entities), **properties** of concepts (aka slots, attributes, roles), **relationships** between concepts (aka associations), and additional **constraints**.
- Ontology languages may be simple (e.g., having only concepts), frame-based (having only concepts and properties), or logic-based (e.g. Ontolingua and DAML+OIL).

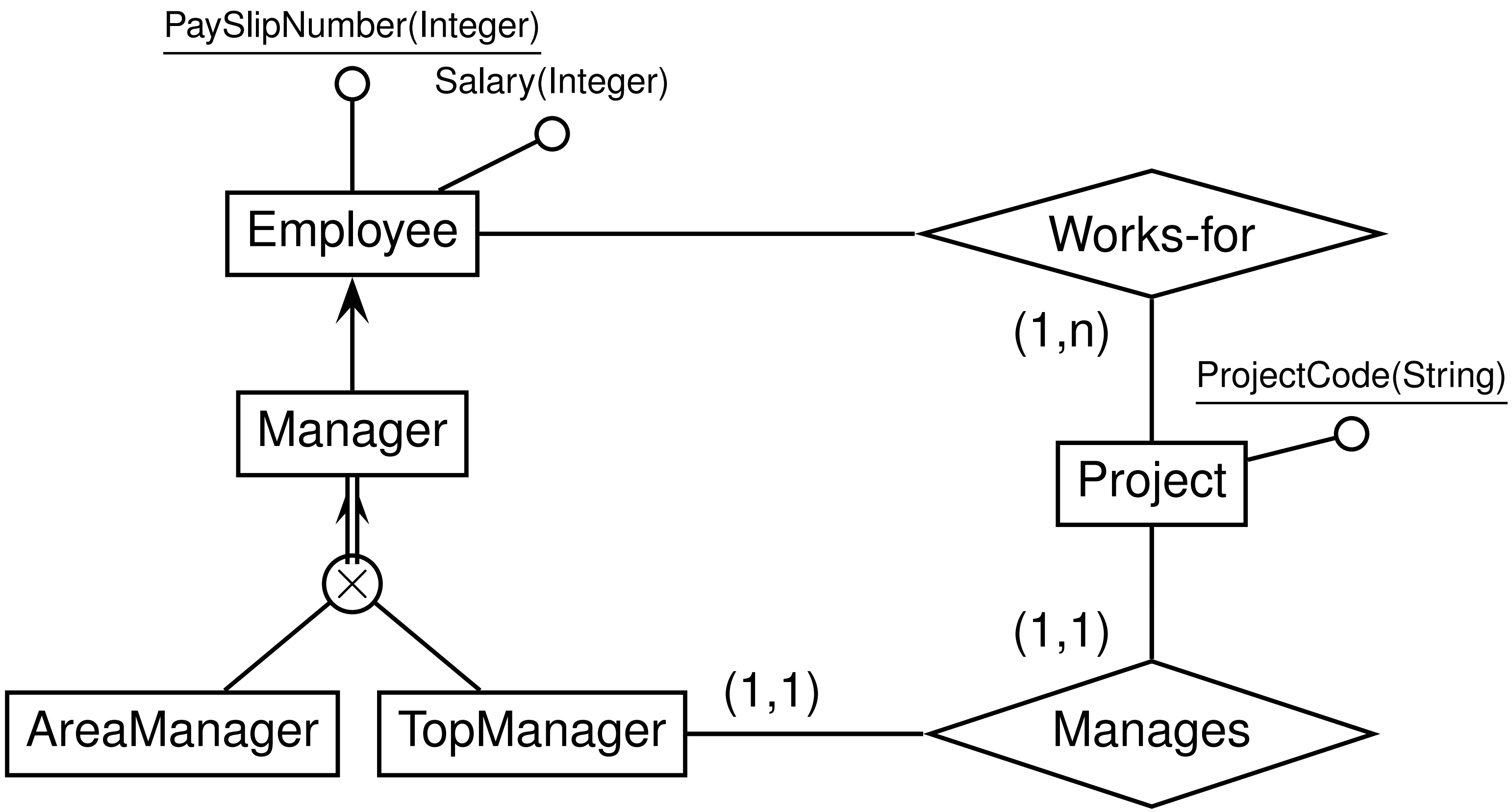
Ontology languages

- An ontology language usually introduces **concepts** (aka classes, entities), **properties** of concepts (aka slots, attributes, roles), **relationships** between concepts (aka associations), and additional **constraints**.
- Ontology languages may be simple (e.g., having only concepts), frame-based (having only concepts and properties), or logic-based (e.g. Ontolingua and DAML+OIL).
- Ontology languages are typically expressed by means of diagrams.

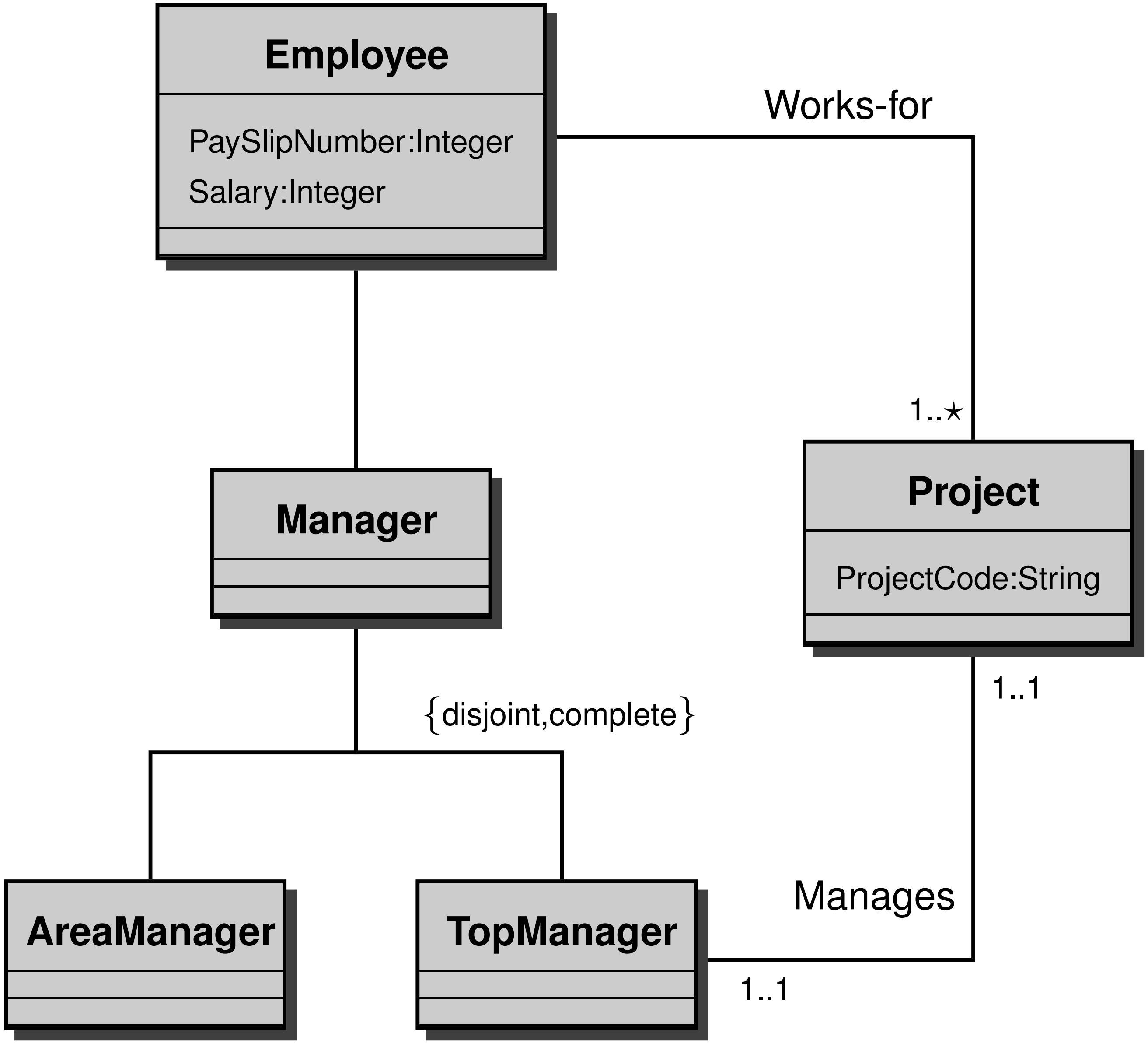
Ontology languages

- An ontology language usually introduces **concepts** (aka classes, entities), **properties** of concepts (aka slots, attributes, roles), **relationships** between concepts (aka associations), and additional **constraints**.
- Ontology languages may be simple (e.g., having only concepts), frame-based (having only concepts and properties), or logic-based (e.g. Ontolingua and DAML+OIL).
- 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 Schema

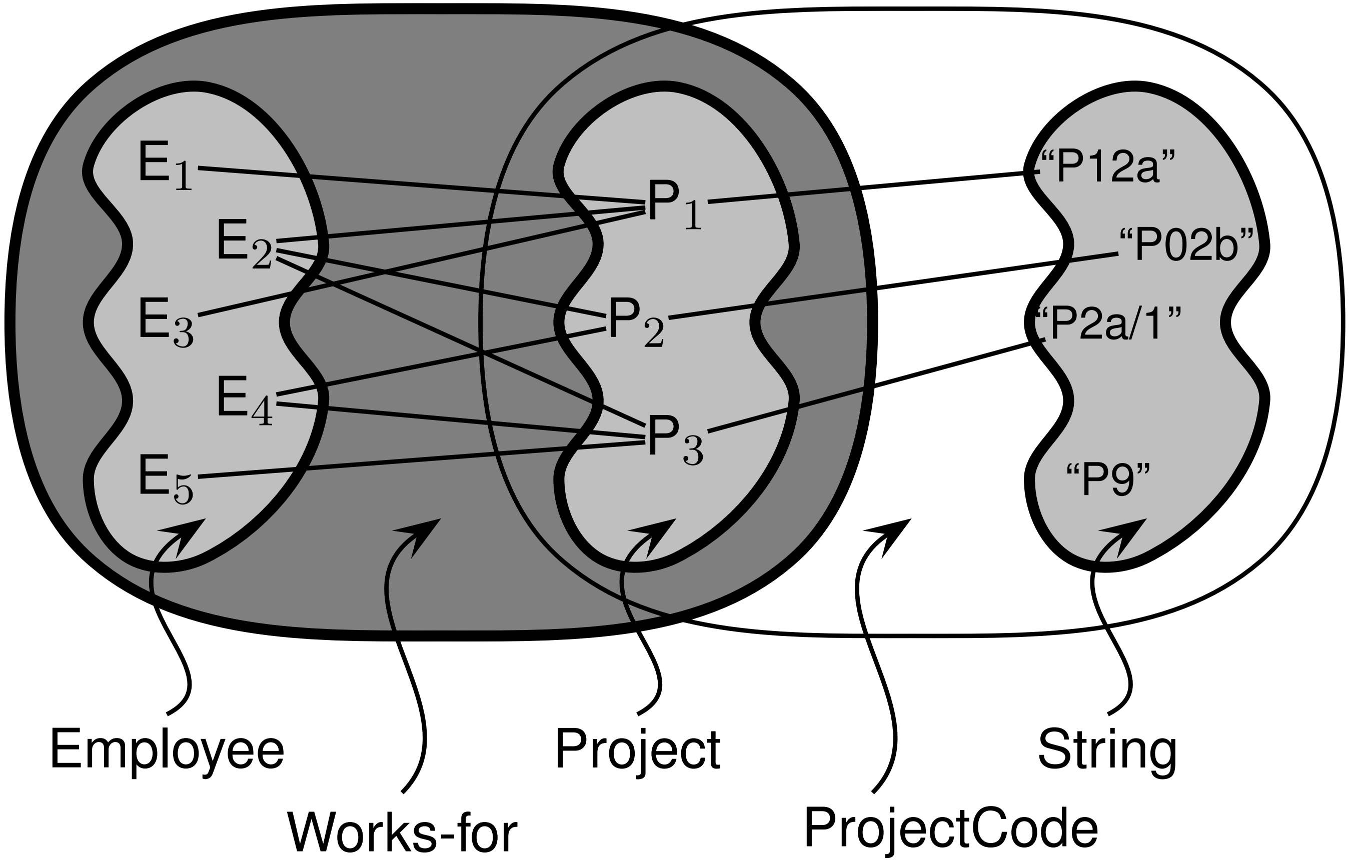


UML Class Diagram

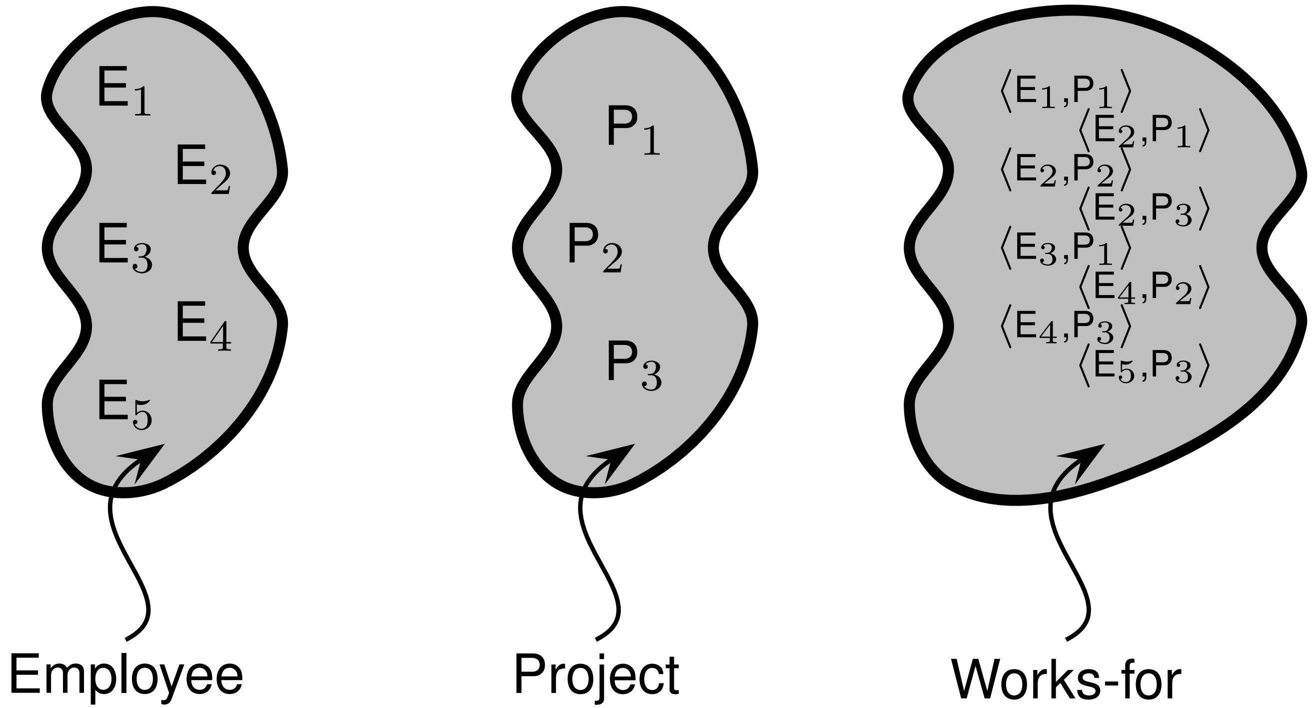


Meaning of basic constructs

- An entity/class is a ***set of instances***;
- an association (n-ary relationship) is a ***set of pairs (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

employeeId
E_1
E_2
E_3
E_4
E_5

Project

projectId
P_1
P_2
P_3

String

anystring
"P12a"
"P02b"
"P2a/1"
"P9"
...

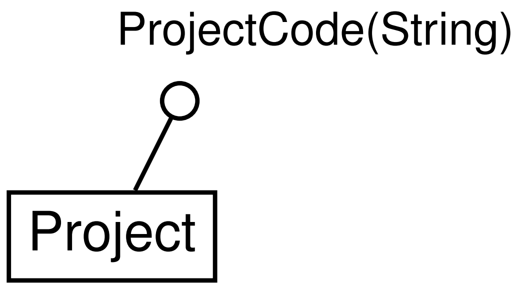
Works-for

employeeId	*projectId*
E_1	P_1
E_2	P_1
E_2	P_2
E_2	P_3
E_3	P_1
E_4	P_2
E_4	P_3
E_5	P_3

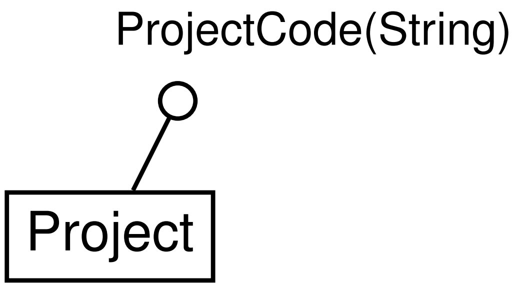
ProjectCode

projectId	*pcode*
P_1	"P12a"
P_2	"P02b"
P_3	"P2a/1"

Meaning of Attributes

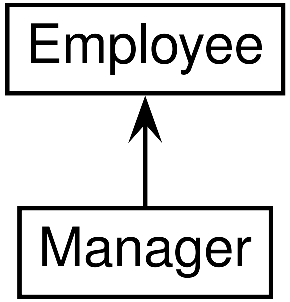


Meaning of Attributes

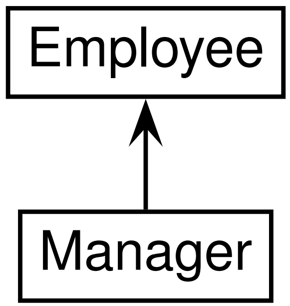


$$\text{Project} \subseteq \{p \mid \sharp(\text{ProjectCode} \cap (\{p\} \times \texttt{String})) \geq 1\}$$

Meaning of ISA

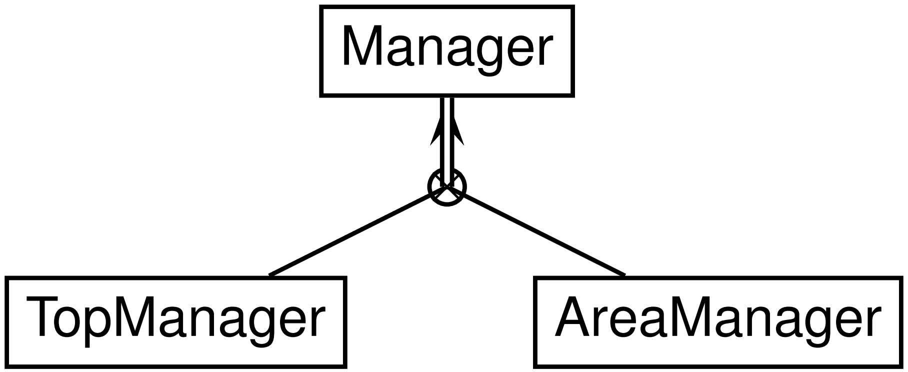


Meaning of ISA

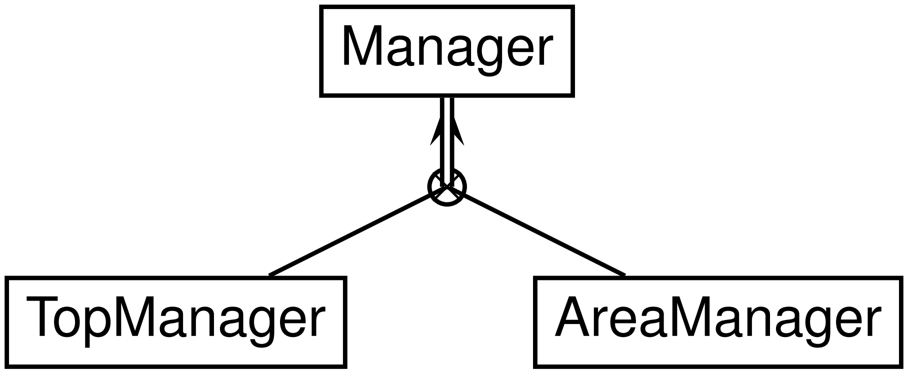


Manager $\subseteq$ Employee

Meaning of *disjoint* and *total* constraints

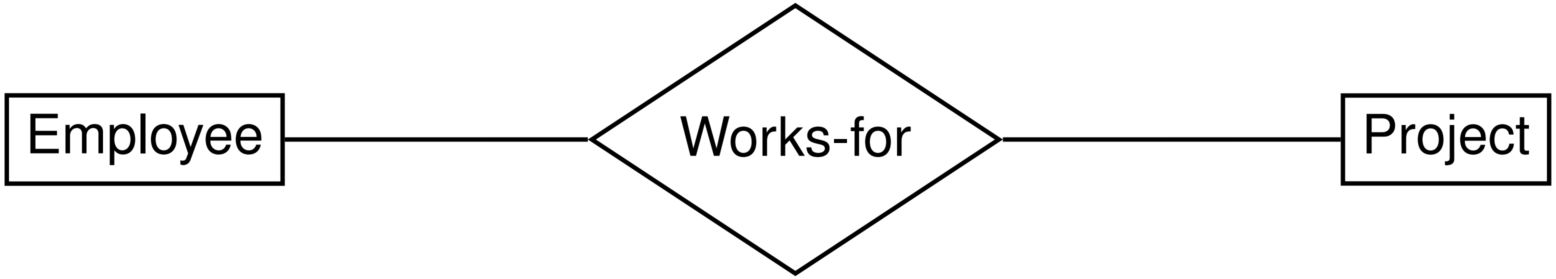


Meaning of *disjoint* and *total* constraints

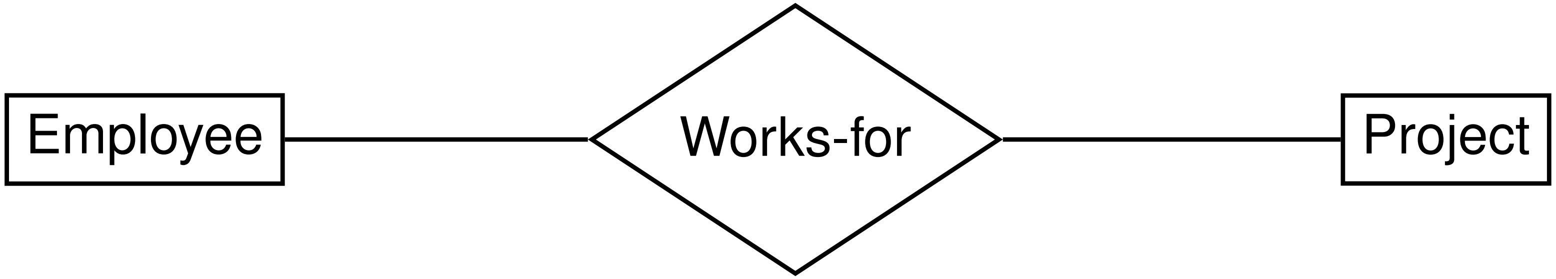


- *ISA:* AreaManager $\subseteq$ Manager
- *ISA:* TopManager $\subseteq$ Manager
- *disjoint:* AreaManager $\cap$ TopManager $= \emptyset$
- *total* Manager $\subseteq$ AreaManager $\cup$ TopManager

Meaning of Associations and Relationships

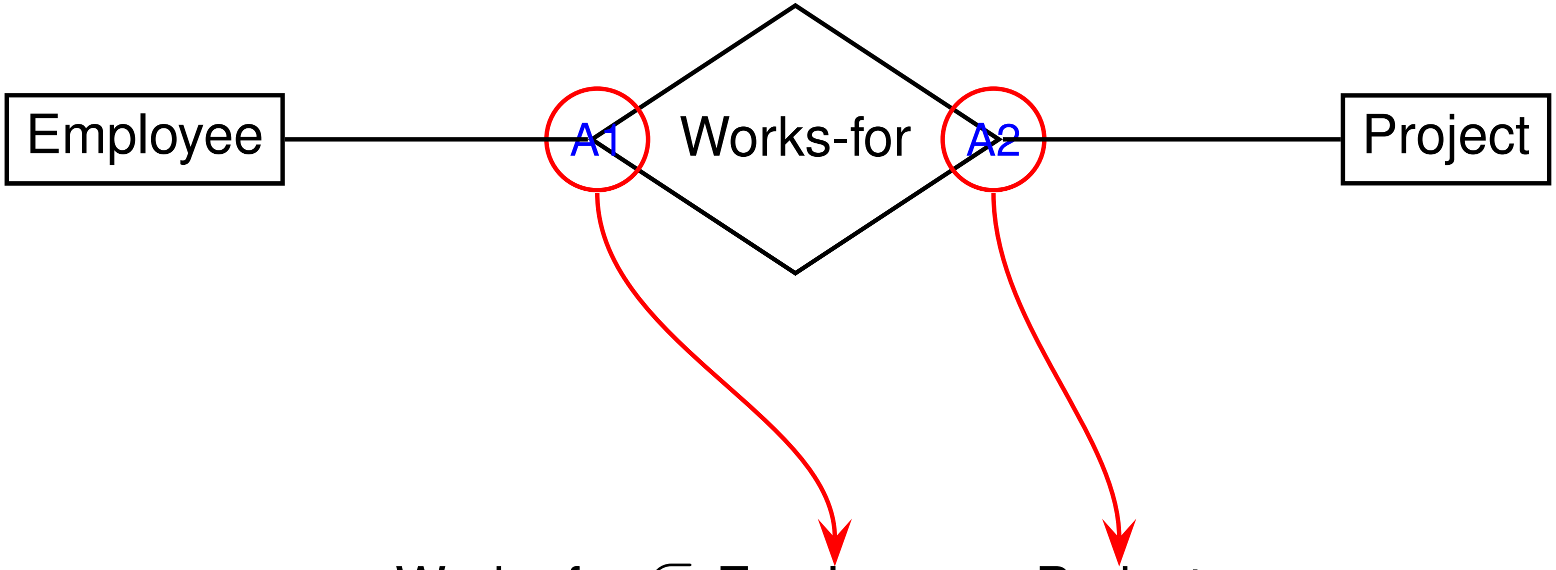


Meaning of Associations and Relationships



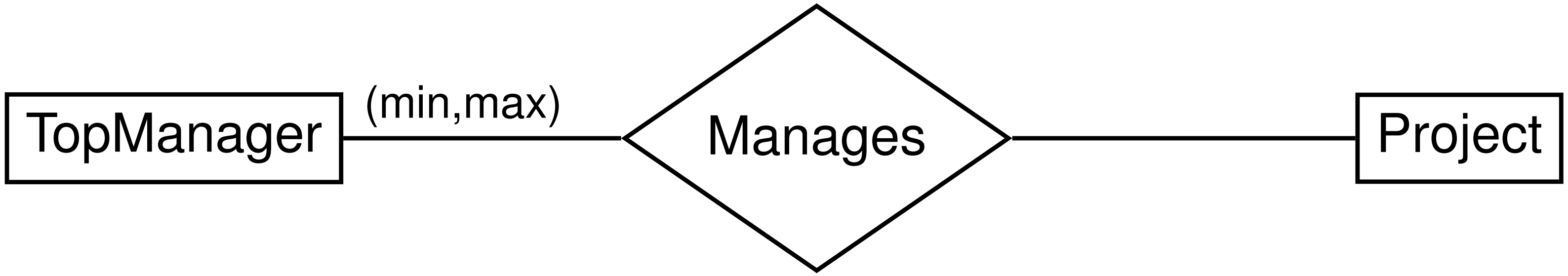
$$\text{Works-for} \subseteq \text{Employee} \times \text{Project}$$

Meaning of Associations and Relationships

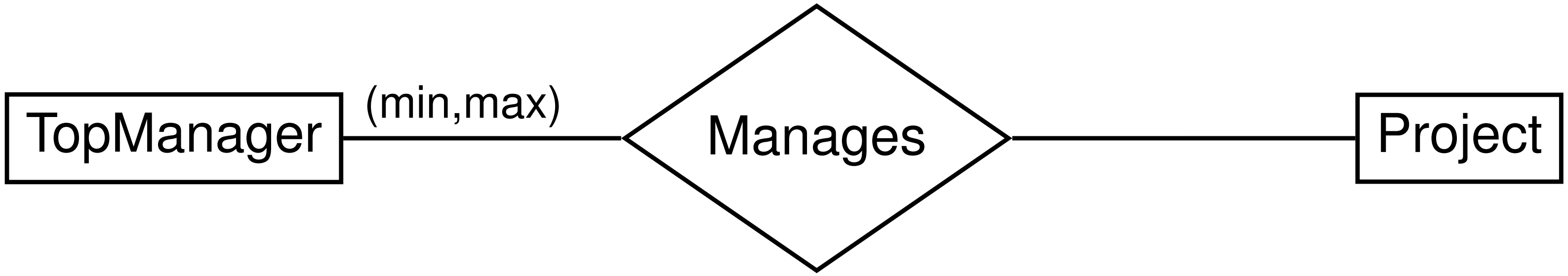


Works-for $\subseteq$ Employee $\times$ Project

Meaning of Cardinality Constraints



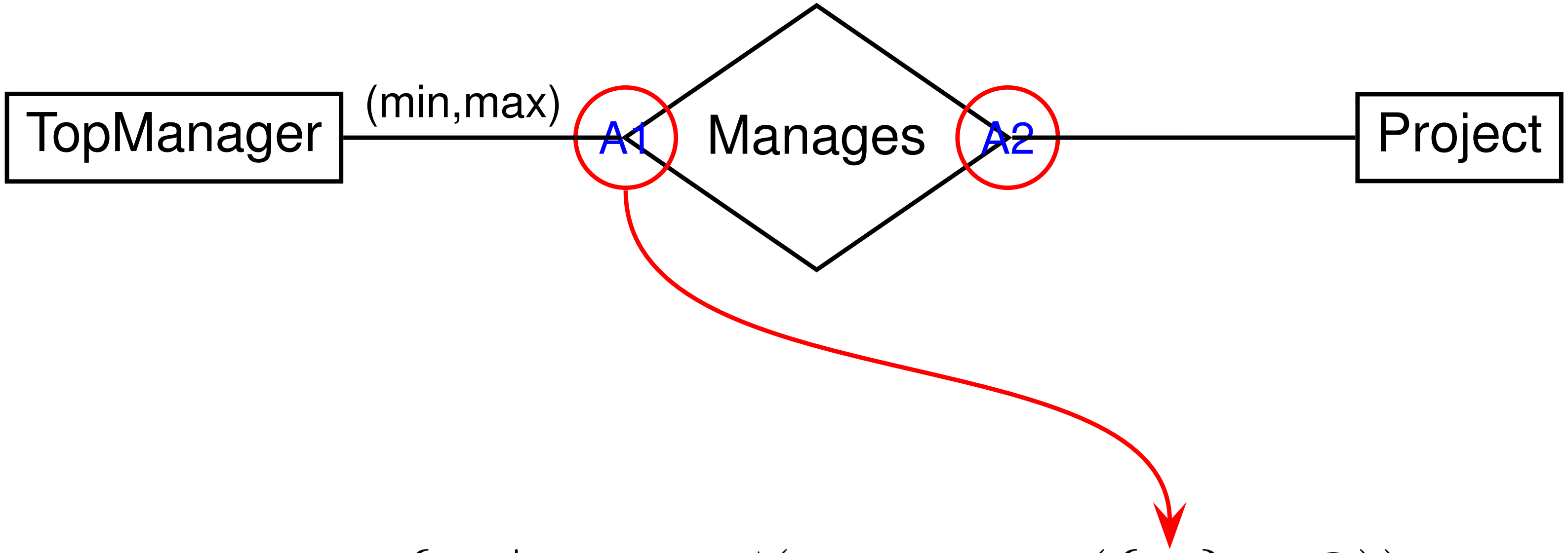
Meaning of Cardinality Constraints



$$\text{TopManager} \subseteq \{m \mid \text{max} \geq \sharp(\text{Manages} \cap (\{m\} \times \Omega)) \geq \text{min}\}$$

(where Ω is the set of all instances)

Meaning of Cardinality Constraints



$$\text{TopManager} \subseteq \{m \mid \text{max} \geq \sharp(\text{Manages} \cap (\{m\} \times \Omega)) \geq \text{min}\}$$

(where Ω is the set of all instances)

Meaning of the initial diagram

Works-for $\subseteq$ Employee $\times$ Project

Manages $\subseteq$ TopManager $\times$ Project

Employee $\subseteq \{e \mid \sharp(\text{PaySlipNumber} \cap (\{e\} \times \texttt{Integer})) \geq 1\}$

Employee $\subseteq \{e \mid \sharp(\text{Salary} \cap (\{e\} \times \texttt{Integer})) \geq 1\}$

Project $\subseteq \{p \mid \sharp(\text{ProjectCode} \cap (\{p\} \times \texttt{String})) \geq 1\}$

TopManager $\subseteq \{m \mid 1 \geq \sharp(\text{Manages} \cap (\{m\} \times \Omega)) \geq 1\}$

Project $\subseteq \{p \mid 1 \geq \sharp(\text{Manages} \cap (\Omega \times \{p\})) \geq 1\}$

Project $\subseteq \{p \mid \sharp(\text{Works-for} \cap (\Omega \times \{p\})) \geq 1\}$

Manager $\subseteq$ Employee

AreaManager $\subseteq$ Manager

TopManager $\subseteq$ Manager

AreaManager $\cap$ TopManager $= \emptyset$

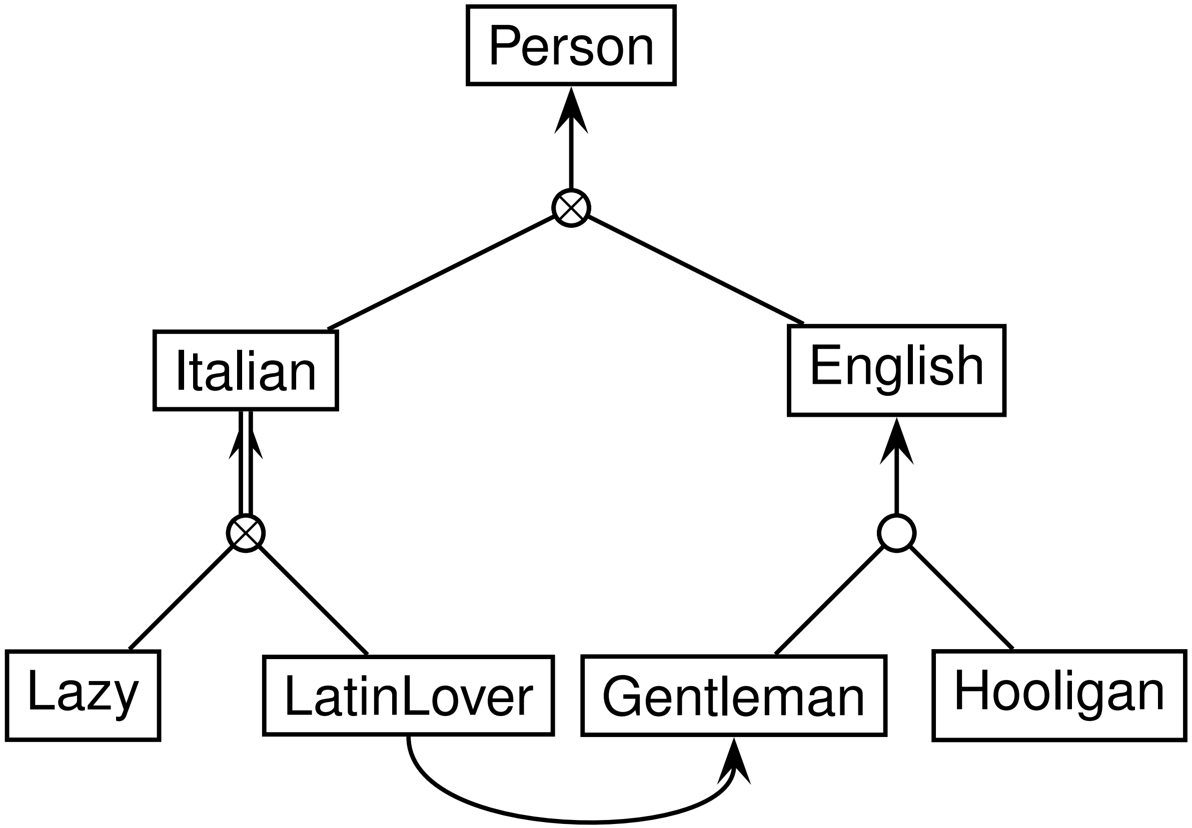
Manager $\subseteq$ AreaManager $\cup$ TopManager

Reasoning

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

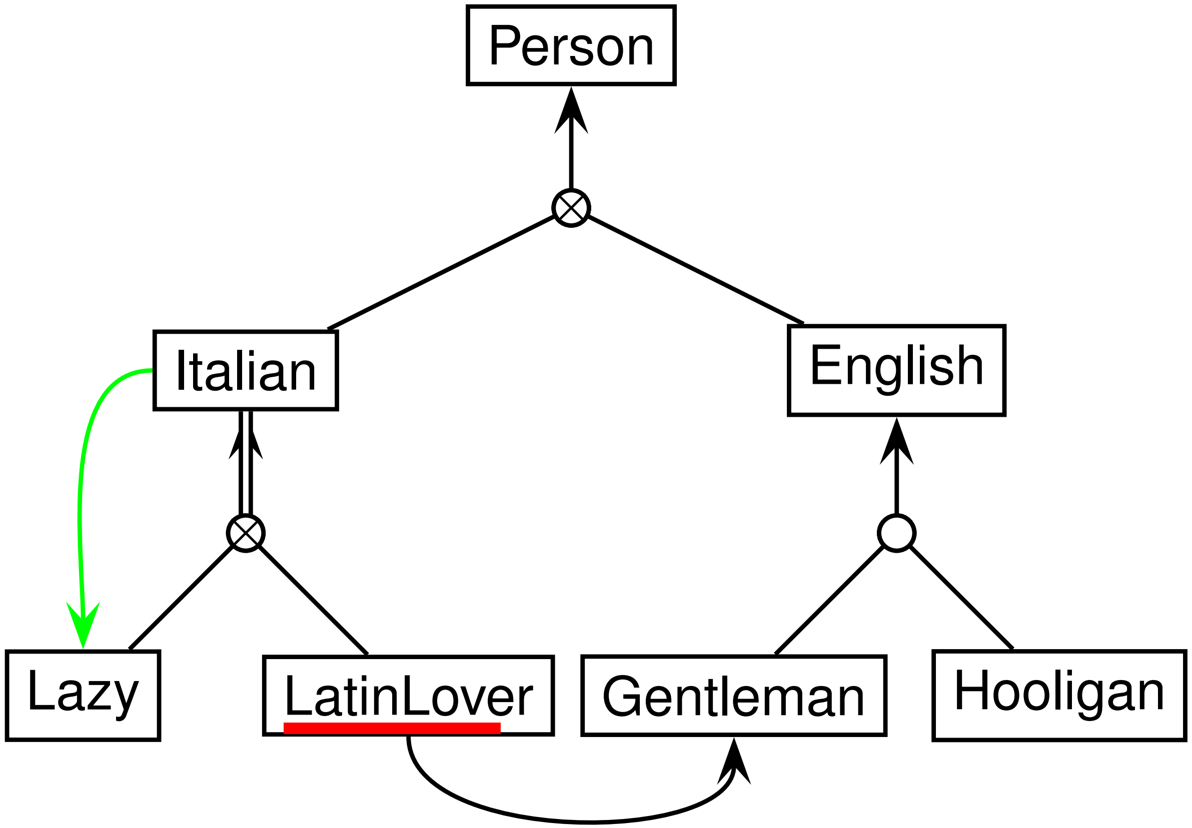
- An entity is inconsistent if it denotes always the empty set.
- An entity is a sub-entity of another entity if the former denotes a subset of the set denoted by the latter.
- Two entities are equivalent if they denote the same set.
- . . .

Reasoning



Person
Italian
English
Lazy
LatinLover
Gentleman
Hooligan

Reasoning



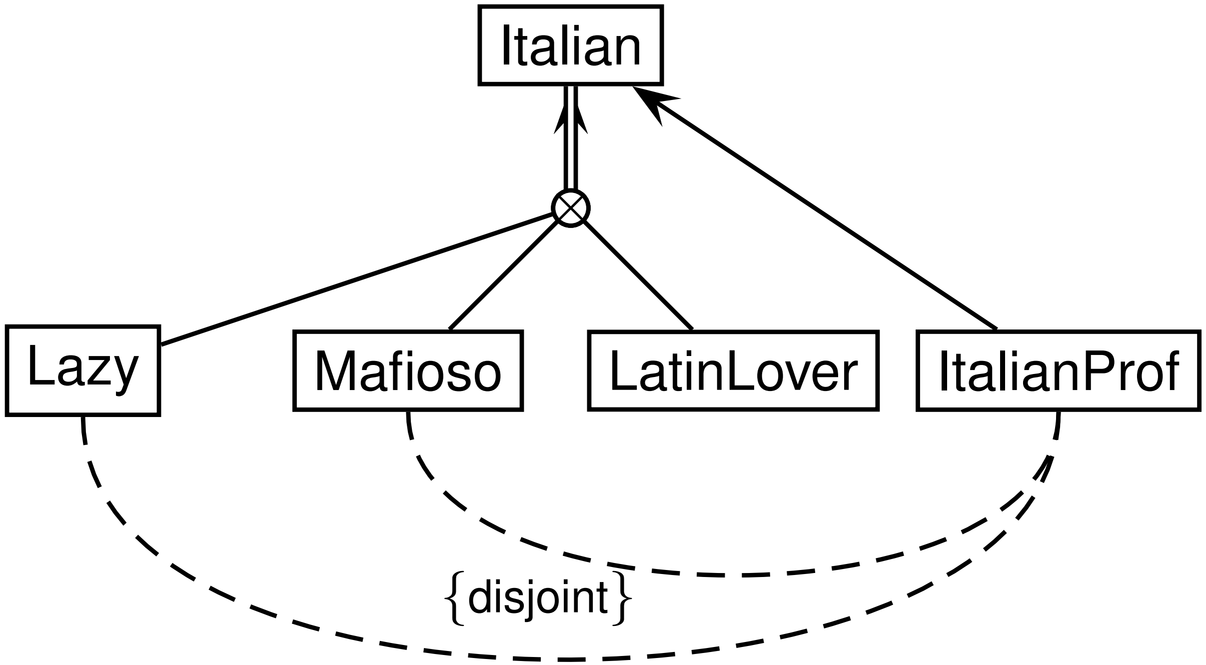
implies

LatinLover $= \emptyset$

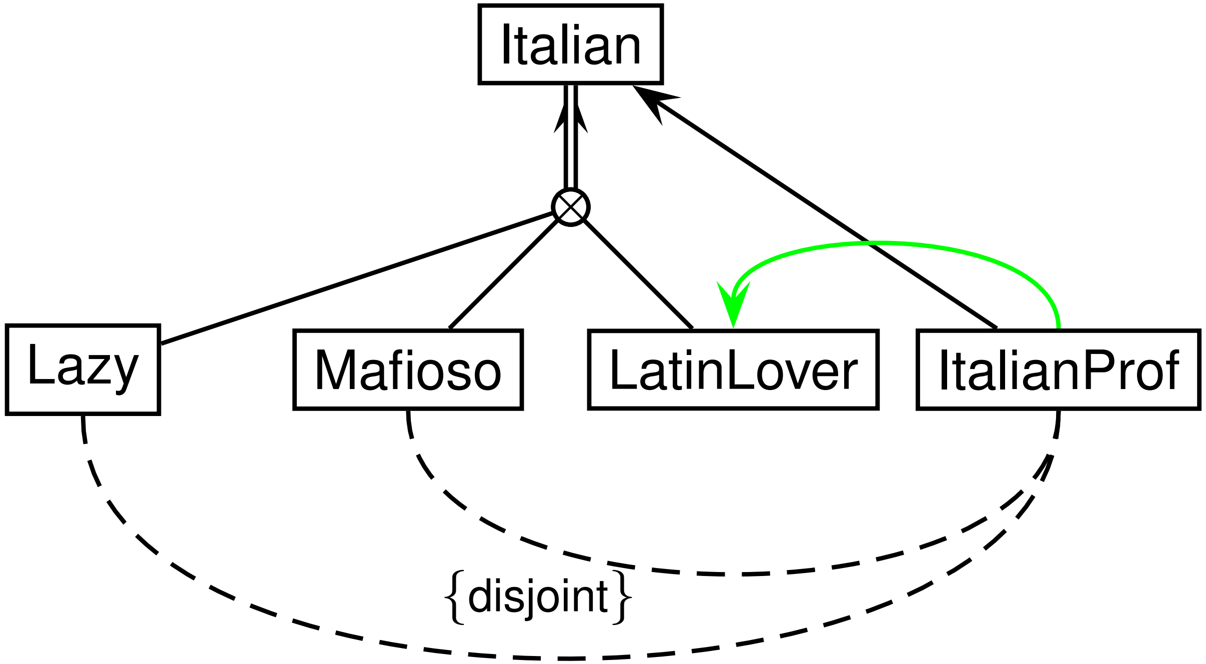
Italian $\subseteq$ Lazy

Italian $\equiv$ Lazy

Reasoning by cases



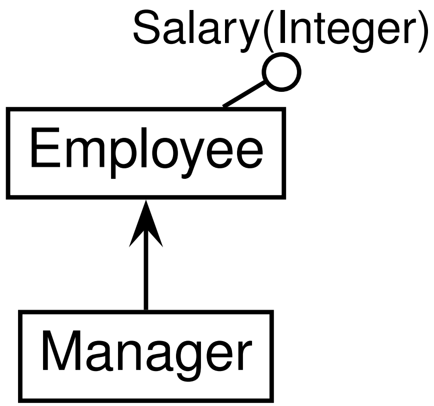
Reasoning by cases



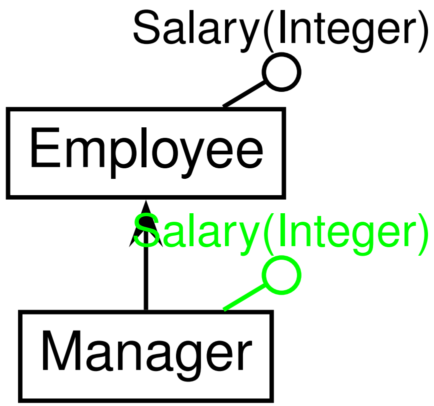
implies

ItalianProf $\subseteq$ LatinLover

ISA and Inheritance



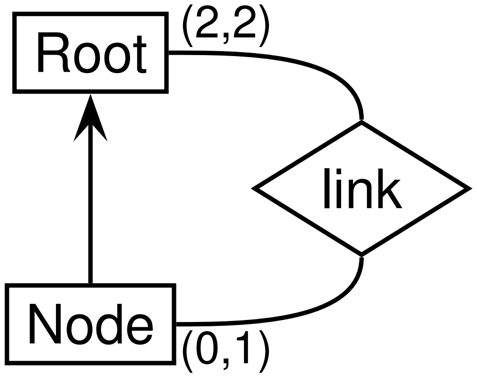
ISA and Inheritance



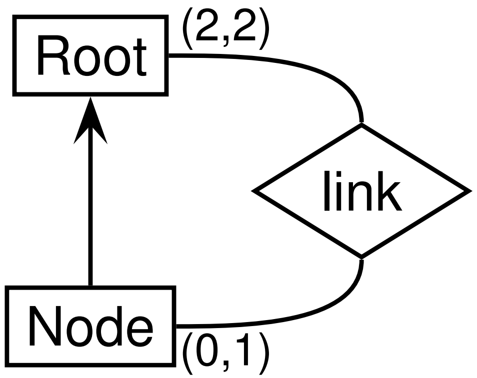
implies

$$\text{Manager} \subseteq \{m \mid \sharp(\text{Salary} \cap (\{m\} \times \texttt{Integer})) \geq 1\}$$

Infinite worlds



Infinite worlds



implies

“the classes Root and Node contain an infinite number of instances”.

Ontologies in First Order Logic

- We have introduced ontology languages that specify a set of constraints that should be satisfied by the world of interest.
- The *interpretation* of an ontology is therefore defined as the collection of all the *legal world descriptions* – i.e., all the (finite) relational structures which conform to the constraints imposed by the ontology.
- In order to formally define the interpretation, an ontology is mapped into a set of *First Order Logic* (FOL) formulas.
- The legal world descriptions (i.e., the interpretation) of an ontology are all the models of the translated FOL theory.

FOL alphabet

The Alphabet of the FOL language will have the following set of *Predicate* symbols:

- 1-ary predicate symbols: $E_1, E_2, \ldots, E_n$ for each Entity-set; $D_1, D_2, \ldots, D_m$ for each Basic Domain.
- binary predicate symbols: $A_1, A_2, \ldots, A_k$ for each Attribute.
- n-ary predicate symbols: $R_1, R_2, \ldots, R_p$ for each Relationship-set.

FOL Notation

- *Vector variables* indicated as $\overline{x}$ stand for an n-tuple of variables:

  $\overline{x} = x_1, \ldots, x_n$

- *Counting existential quantifier* indicated as $\exists^{\leq n}$ or $\exists^{\geq n}$.

  $$\exists^{\leq n} x.\, R(x,y) \equiv$$
  $$\forall x_1, \ldots, x_n, x_{n+1}.\, R(x_1,y) \wedge \ldots \wedge R(x_n,y) \wedge R(x_{n+1},y) \rightarrow$$
  $$(x_1 = x_2) \vee \ldots \vee (x_1 = x_n) \vee (x_1 = x_{n+1}) \vee$$
  $$(x_2 = x_3) \vee \ldots \vee (x_2 = x_n) \vee (x_2 = x_{n+1}) \vee$$
  $$\ldots \ldots \vee (x_n = x_{n+1})$$

  $$\exists^{\geq n} x.\, R(x,y) \equiv$$
  $$\exists x_1, \ldots, x_n.\, R(x_1,y) \wedge \ldots \wedge R(x_n,y) \wedge R(x_{n+1},y) \wedge$$
  $$\neg(x_1 = x_2) \wedge \ldots \wedge \neg(x_1 = x_n) \wedge$$
  $$\neg(x_2 = x_3) \wedge \ldots \wedge \neg(x_2 = x_n) \wedge$$
  $$\ldots \ldots \wedge (x_{n-1} = x_n)$$

The Interpretation function

Interpretation: $\mathcal{I} = \langle \mathbf{D}, \cdot^{\mathcal{I}} \rangle$, where $\mathbf{D}$ is an arbitrary non-empty set such that:

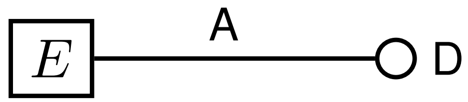
- $\mathbf{D} = \Omega \cup \mathcal{B}$, where:
  - $\mathcal{B} = \cup_{i=1}^{m} \mathcal{B}_{Di}$. $\mathcal{B}_{Di}$ is the set of values associated with each basic domain (i.e., integer, string, etc.); and $\mathcal{B}_{Di} \cap \mathcal{B}_{Dj} = \emptyset, \forall i, j.\, i \neq j$
  - Ω is the abstract entity domain such that $\mathcal{B} \cap \Omega = \emptyset$.

The Formal Semantics for the Atoms

$\mathcal{I}$ is the interpretation function that maps:

- *Basic Domain Predicates* to elements of the relative basic domain:
  $D_i^{\mathcal{I}} = \mathcal{B}_{Di}$ (e.g., $String^{\mathcal{I}} = \mathcal{B}_{String}$).
- *Entity-set Predicates* to elements of the entity domain:
  $E_i^{\mathcal{I}} \subseteq \Omega$.
- *Attribute Predicates* to binary relations such that:
  $A_i^{\mathcal{I}} \subseteq \Omega \times \mathcal{B}$.
- *Relationship-set Predicates* to n-ary relations over the entity domain:
  $R_i^{\mathcal{I}} \subseteq \Omega \times \Omega \ldots \times \Omega = \Omega^n$.

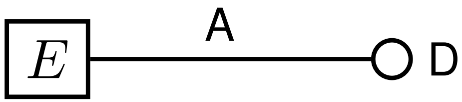
The Attribute Construct



- The meaning of this constraint is:

$$E^{\mathcal{I}} \subseteq \{e \in \Omega \mid \sharp(A^{\mathcal{I}} \cap (\{e\} \times \mathcal{B}_D)) \geq 1\}$$

The Attribute Construct



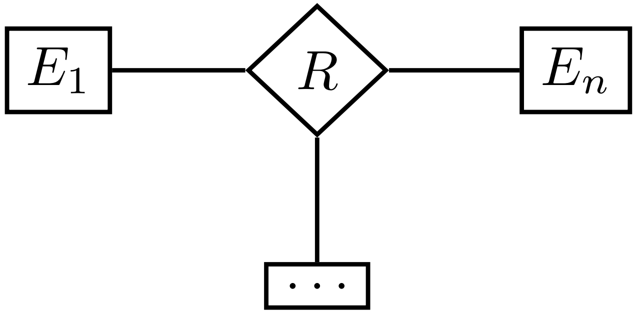
- The meaning of this constraint is:

$$E^{\mathcal{I}} \subseteq \{e \in \Omega \mid \sharp(A^{\mathcal{I}} \cap (\{e\} \times \mathcal{B}_D)) \geq 1\}$$

- The FOL translation is the formula:

$$\forall x. E(x) \rightarrow \exists y. A(x, y) \wedge D(y)$$

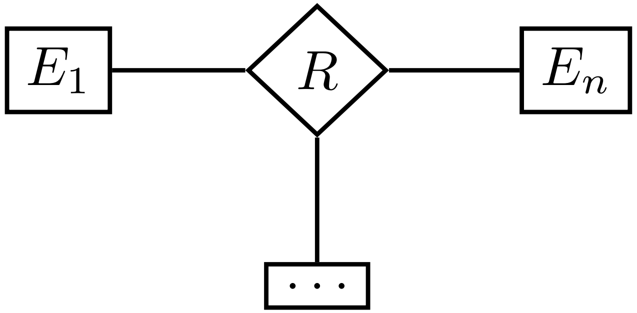
The Relationship Construct



- The meaning of this constraint is:

$$R^{\mathcal{I}} \subseteq {E_1}^{\mathcal{I}} \times \ldots \times {E_n}^{\mathcal{I}}$$

The Relationship Construct



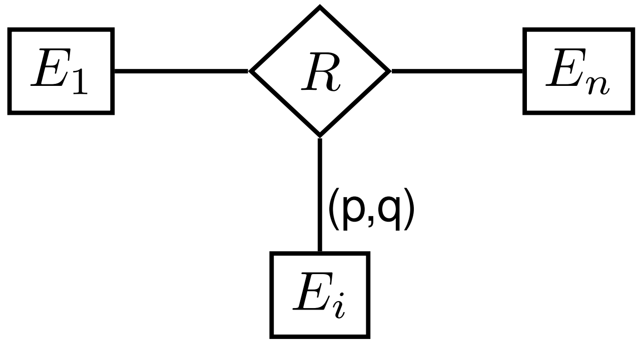
- The meaning of this constraint is:

$$R^{\mathcal{I}} \subseteq {E_1}^{\mathcal{I}} \times \ldots \times {E_n}^{\mathcal{I}}$$

- The FOL translation is the formula:

$$\forall x_1, \ldots, x_n.\, R(x_1, \ldots, x_n) \rightarrow E_1(x_1) \wedge \ldots \wedge E_n(x_n)$$

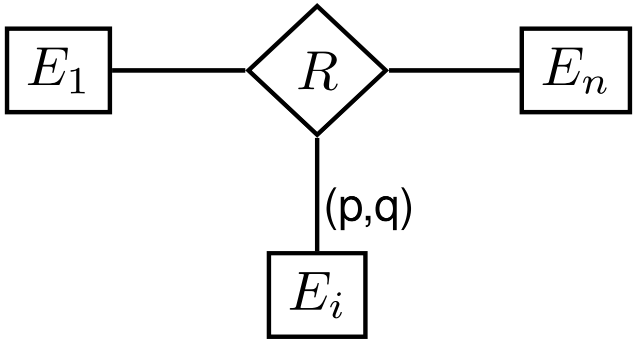
The Cardinality Construct



- The meaning of this constraint is:

$$E_i^{\mathcal{I}} \subseteq \{e_i \in \Omega \mid p \leq \sharp(R^{\mathcal{I}} \cap (\Omega \times \{e_i\} \times \Omega)) \leq q\}$$

The Cardinality Construct



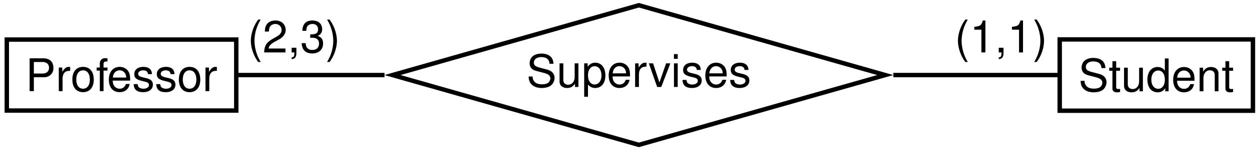
- The meaning of this constraint is:

$$E_i^{\mathcal{I}} \subseteq \{e_i \in \Omega \mid p \leq \sharp(R^{\mathcal{I}} \cap (\Omega \times \{e_i\} \times \Omega)) \leq q\}$$

- The FOL translation is the formula:

$$\begin{aligned}\forall x_i.\, E(x_i) \rightarrow\ & \exists^{\geq p} x_1, \ldots, x_{i-1}, x_{i+1}, \ldots x_n.\, R(x_1, \ldots, x_n) \wedge \\ & \exists^{\leq q} x_1, \ldots, x_{i-1}, x_{i+1}, \ldots x_n.\, R(x_1, \ldots, x_n)\end{aligned}$$

The Cardinality Construct: An Example



A valid world description is:

Professor

professorId
Alex
Bob

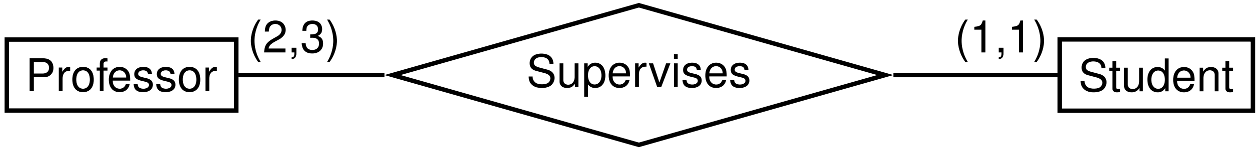
Student

studentId
John
Mary
Nick
Paul
Laura

Supervises

professorId	*studentId*
Alex	John
Bob	Laura
Alex	Mary
Bob	Nick
Alex	Paul

The Cardinality Construct: An Example



An invalid world description is:

Professor

professorId
Alex
Bob

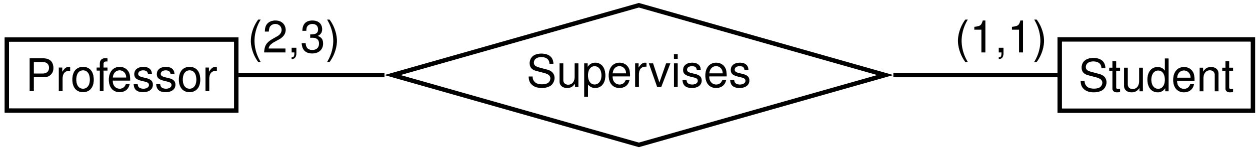
Student

studentId
John
Mary
Nick
Paul
Laura

Supervises

professorId	*studentId*
Alex	John
Bob	Laura
Alex	Mary
Bob	Nick
Alex	Paul
Alex	Laura

The Cardinality Construct: An Example



- The FOL translation is:

$$\forall x, y. \texttt{Supervises}(x, y) \rightarrow \texttt{Professor}(x) \wedge \texttt{Student}(y)$$

$$\forall x. \texttt{Professor}(x) \rightarrow \exists^{\geq 2} y. \texttt{Supervises}(x, y) \wedge \exists^{\leq 3} y. \texttt{Supervises}(x, y)$$

$$\forall y. \texttt{Student}(y) \rightarrow \exists^{=1} x. \texttt{Supervises}(x, y)$$

ISA Relations

The **ISA** relation is a constraint that specifies *sub-entity sets*.

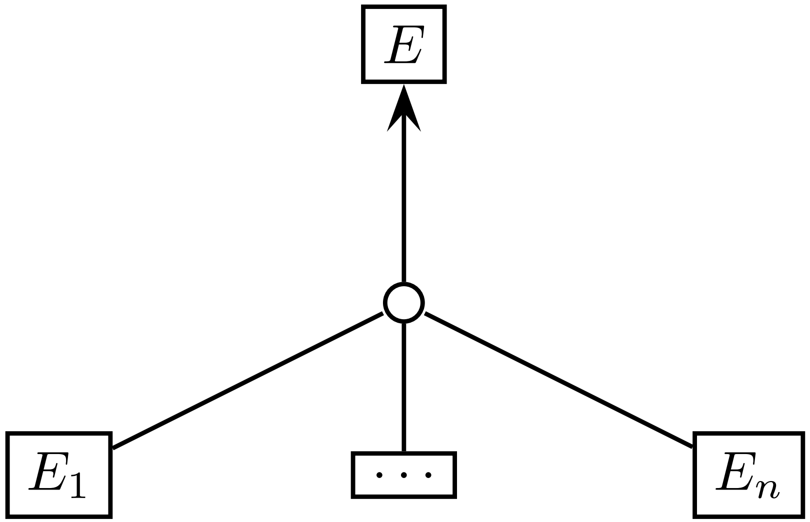
Sub-entity-set = contains entities with more properties – both more attributes and different participation in relationships – not pertinent to the Super-entity-set.

A Sub-entity-set *inherits* all the properties of its Sub-entity-sets.

We distinguish between the following different ISA relations:

- Overlapping Partial;
- Overlapping Total;
- Disjoint Partial;
- Disjoint Total.

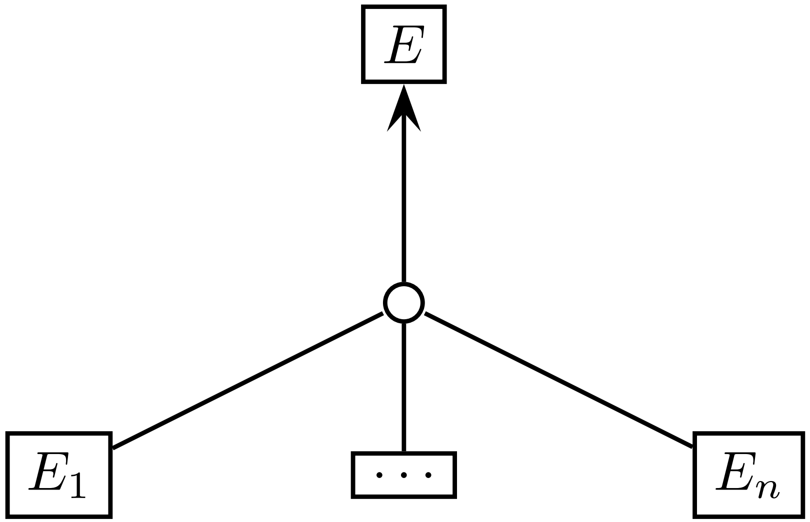
The Overlapping Partial Construct



- The meaning of this constraint is:

$$E_i^{\mathcal{I}} \subseteq E^{\mathcal{I}}, \text{ for all } i = 1, \ldots, n.$$

The Overlapping Partial Construct



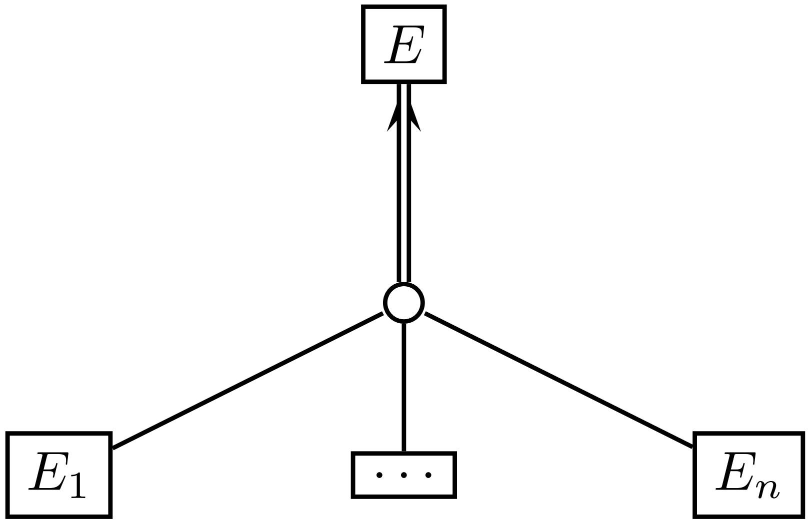
- The meaning of this constraint is:

$$E_i^{\mathcal{I}} \subseteq E^{\mathcal{I}}, \text{ for all } i = 1, \ldots, n.$$

- The FOL translation is the formula:

$$\forall x. E_i(x) \rightarrow E(x), \text{ for all } i = 1, \ldots, n.$$

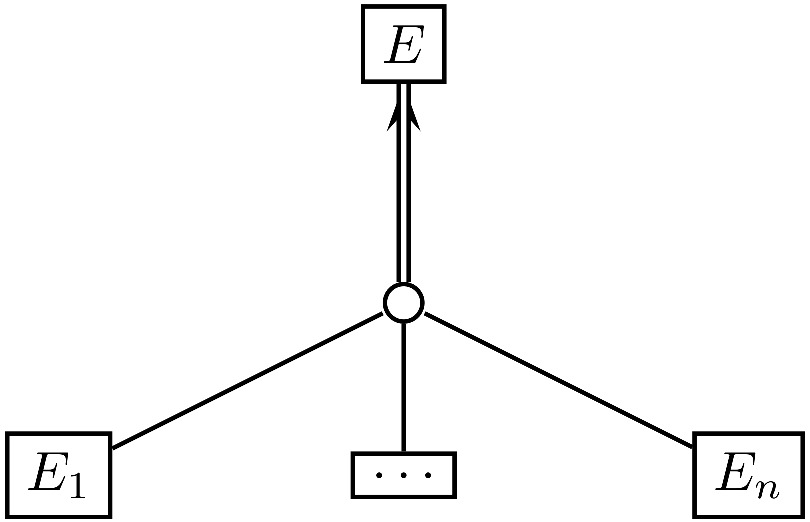
The Overlapping Total Construct



- The meaning of this constraint is:

$$E_i^{\mathcal{I}} \subseteq E^{\mathcal{I}}, \text{ for all } i = 1, \ldots, n$$
$$E^{\mathcal{I}} \subseteq E_1{}^{\mathcal{I}} \cup \ldots \cup E_n{}^{\mathcal{I}}$$

The Overlapping Total Construct



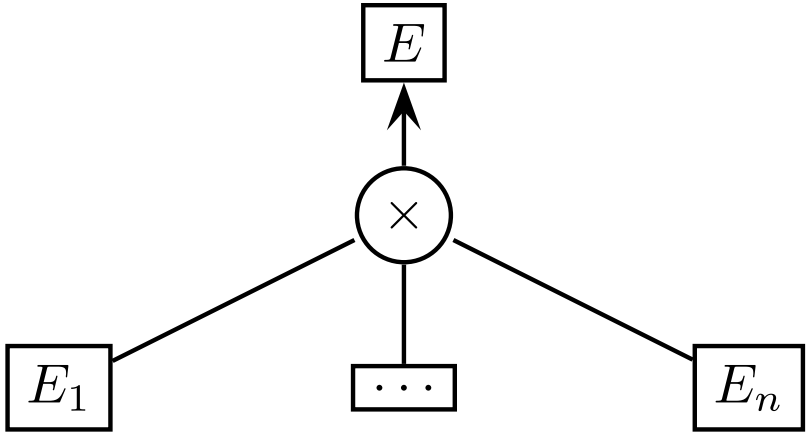
- The meaning of this constraint is:

$$E_i^{\mathcal{I}} \subseteq E^{\mathcal{I}}, \text{ for all } i = 1, \ldots, n$$
$$E^{\mathcal{I}} \subseteq E_1^{\mathcal{I}} \cup \ldots \cup E_n^{\mathcal{I}}$$

- The FOL translation is the set of formulas:

$$\forall x. E_i(x) \rightarrow E(x), \text{ for all } i = 1, \ldots, n$$
$$\forall x. E(x) \rightarrow E_1(x) \vee \ldots \vee E_n$$

The Disjoint Partial Construct

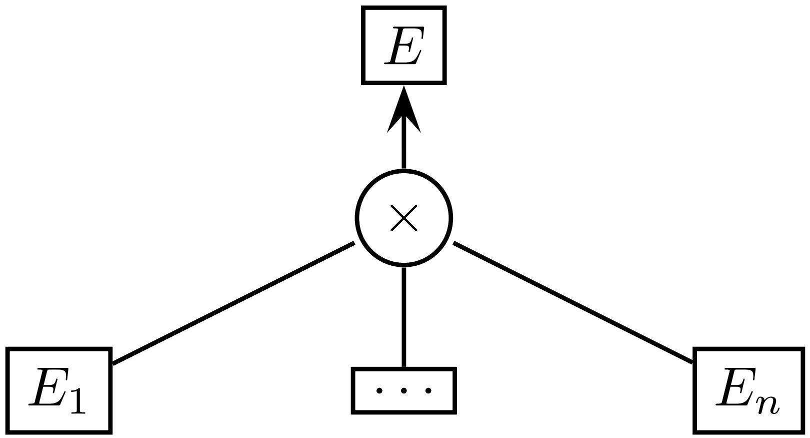


- The meaning of this constraint is:

$$E_i^{\mathcal{I}} \subseteq E^{\mathcal{I}} \qquad \text{for all } i = 1, \ldots, n$$

$$E_i^{\mathcal{I}} \cap E_j^{\mathcal{I}} = \emptyset \qquad \text{for all } i \neq j$$

The Disjoint Partial Construct

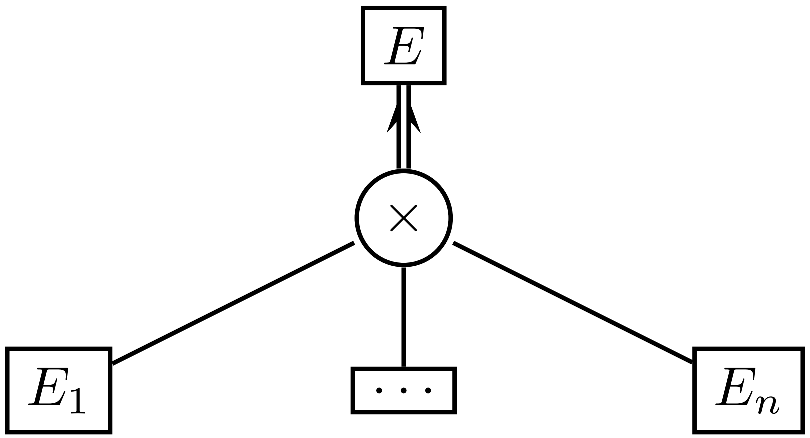


- The meaning of this constraint is:
$$E_i^{\mathcal{I}} \subseteq E^{\mathcal{I}} \qquad \text{for all } i = 1, \ldots, n$$
$$E_i^{\mathcal{I}} \cap E_j^{\mathcal{I}} = \emptyset \qquad \text{for all } i \neq j$$

- The FOL translation is the set of formulas:

$$\begin{array}{lcl}
\forall x.\, E_1(x) & \rightarrow & E(x) \wedge \neg E_2(x) \wedge \ldots \wedge \neg E_n(x) \\
\forall x.\, E_2(x) & \rightarrow & E(x) \wedge \neg E_3(x) \wedge \ldots \wedge \neg E_n(x) \\
\forall x.\, E_{n-1}(x) & \rightarrow & E(x) \wedge \neg E_n(x) \\
\forall x.\, E_n(x) & \rightarrow & E(x)
\end{array}$$

The Disjoint Total Construct



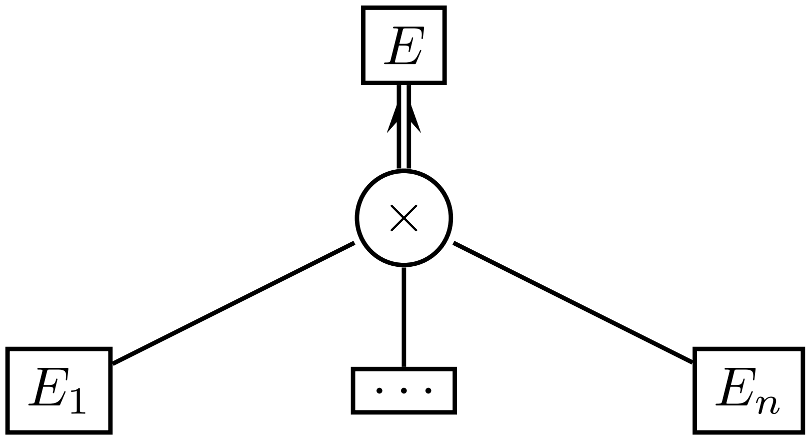
- The meaning of this constraint is:

  ${E_i}^{\mathcal{I}} \subseteq E^{\mathcal{I}}$ for all $i = 1, \dots, n$

  ${E_i}^{\mathcal{I}} \cap {E_j}^{\mathcal{I}} = \emptyset$ for all $i \neq j$

  $E^{\mathcal{I}} \subseteq {E_1}^{\mathcal{I}} \cup \dots \cup {E_n}^{\mathcal{I}}$

The Disjoint Total Construct



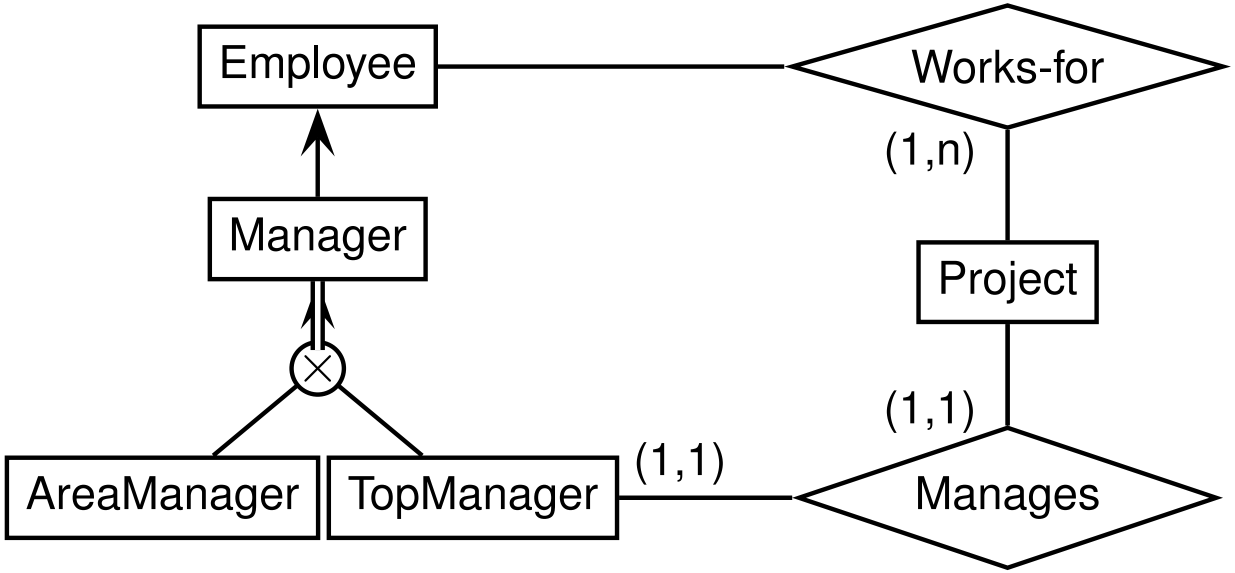
- The meaning of this constraint is:

$$E_i{}^{\mathcal{I}} \subseteq E^{\mathcal{I}} \qquad \text{for all } i = 1, \ldots, n$$
$$E_i{}^{\mathcal{I}} \cap E_j{}^{\mathcal{I}} = \emptyset \qquad \text{for all } i \neq j$$
$$E^{\mathcal{I}} \subseteq E_1{}^{\mathcal{I}} \cup \ldots \cup E_n{}^{\mathcal{I}}$$

- The FOL translation is the set of formulas:

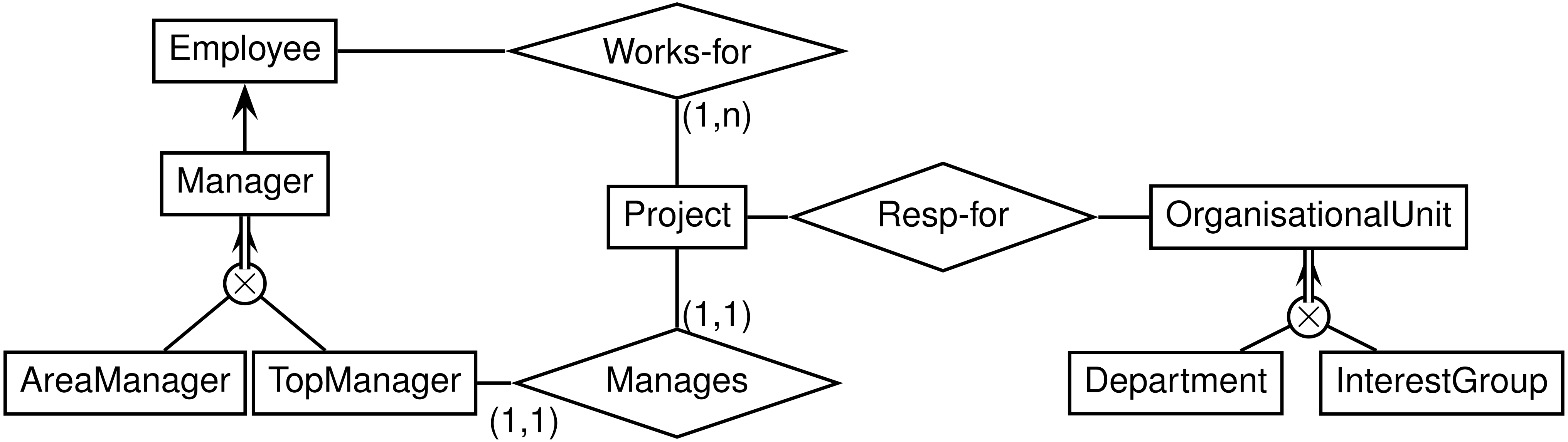
$$\forall x. E(x) \rightarrow E_1(x) \vee \ldots \vee E_n$$
$$\forall x. E_1(x) \rightarrow E(x) \wedge \neg E_2(x) \wedge \ldots \wedge \neg E_n(x)$$
$$\forall x. E_2(x) \rightarrow E(x) \wedge \neg E_3(x) \wedge \ldots \wedge \neg E_n(x)$$
$$\forall x. E_{n-1}(x) \rightarrow E(x) \wedge \neg E_n(x)$$
$$\forall x. E_n(x) \rightarrow E(x)$$

FOL Translation: An Example



$$
\begin{array}{lcl}
\forall x, y. \texttt{Works-for}(x, y) & \rightarrow & \texttt{Employee}(x) \wedge \texttt{Project}(y) \\
\forall x, y. \texttt{Manages}(x, y) & \rightarrow & \texttt{Top-Manager}(x) \wedge \texttt{Project}(y) \\
\forall y. \texttt{Project}(y) & \rightarrow & \exists x. \texttt{Works-for}(x, y) \\
\forall y. \texttt{Project}(y) & \rightarrow & \exists^{=1} x. \texttt{Manages}(x, y) \\
\forall x. \texttt{Top-Manager}(x) & \rightarrow & \exists^{=1} y. \texttt{Manages}(x, y) \\
\forall x. \texttt{Manager}(x) & \rightarrow & \texttt{Employee}(x) \\
\forall x. \texttt{Manager}(x) & \rightarrow & \texttt{Area-Manager}(x) \vee \texttt{Top-Manager}(x) \\
\forall x. \texttt{Area-Manager}(x) & \rightarrow & \texttt{Manager}(x) \wedge \neg \texttt{Top-Manager}(x) \\
\forall x. \texttt{Top-Manager}(x) & \rightarrow & \texttt{Manager}(x)
\end{array}
$$

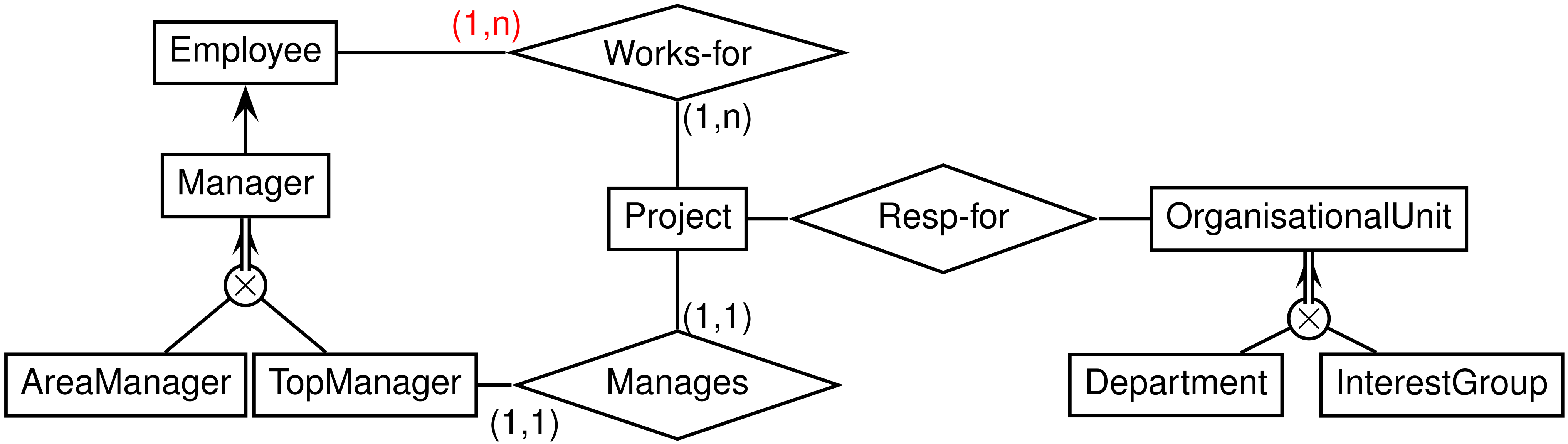
Additional (integrity) constraints



- Managers do not work for a project (she/he just manages it).

$$\forall x.\, \texttt{Manager}(x) \rightarrow \forall y.\, \neg \texttt{WORKS-FOR}(x, y)$$

Additional (integrity) constraints

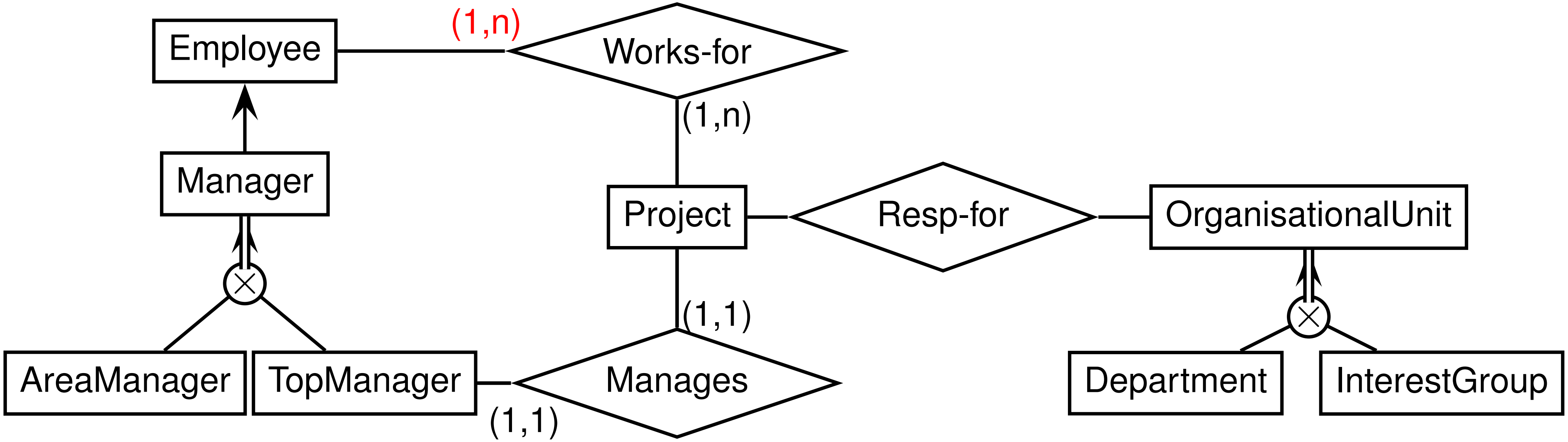


- Managers do not work for a project (she/he just manages it).

$$\forall x. \texttt{Manager}(x) \rightarrow \forall y. \neg \texttt{WORKS-FOR}(x, y)$$

- If the minimum cardinality for the participation of employees to the *works-for* relationship is increased, then . . .

Additional (integrity) constraints



- Managers do not work for a project (she/he just manages it).

$$\forall x.\,\texttt{Manager}(x) \rightarrow \forall y.\,\neg\texttt{WORKS-FOR}(x, y)$$

- If the minimum cardinality for the participation of employees to the *works-for* relationship is increased, then . . .
- If an ISA link is added stating that Interest Groups are Departments, then . . .