

Introduction to Semantic Web Ontology Languages

Formalising Ontologies

Enrico Franconi

`franconi@inf.unibz.it`

`http://www.inf.unibz.it/~franconi`

Faculty of Computer Science, Free University of Bozen-Bolzano, Italy

Ambiguities: classes and instances

Person : AGE : Number,
SEX : *M*, *F*,
HEIGHT : Number,
WIFE : Person.

john : AGE : 29,
SEX : *M*,
HEIGHT : 76,
WIFE : *mary*.

Ambiguities: incomplete information

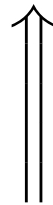
29'er : AGE : 29,
SEX : *M*,
HEIGHT : Number,
WIFE : Person.

john : AGE : 29,
SEX : *M*,
HEIGHT : Number,
WIFE : Person.

Ambiguities: is-a

Sub-class:

Person : AGE : Number,
SEX : *M, F*,
HEIGHT : Number,
WIFE : Person.



Male : AGE : Number,
SEX : *M*,
HEIGHT : Number,
WIFE : Female.

Ambiguities: is-a

Instance-of:

Male : AGE : Number,
SEX : *M*,
HEIGHT : Number,
WIFE : Female.



john : AGE : 35,
SEX : *M*,
HEIGHT : 76,
WIFE : *mary*.

Ambiguities: is-a

Instance-of:

29'er : AGE : 29,
SEX : *M*,
HEIGHT : Number,
WIFE : Person.



john : AGE : 29,
SEX : *M*,
HEIGHT : Number,
WIFE : Person.

Ambiguities: relations

Implicit relation:

john : AGE : 35,
SEX : *M*,
HEIGHT : 76,
WIFE : *mary*.

mary : AGE : 32,
SEX : *F*,
HEIGHT : 59,
HUSBAND : *john*.

Ambiguities: relations

Explicit relation:

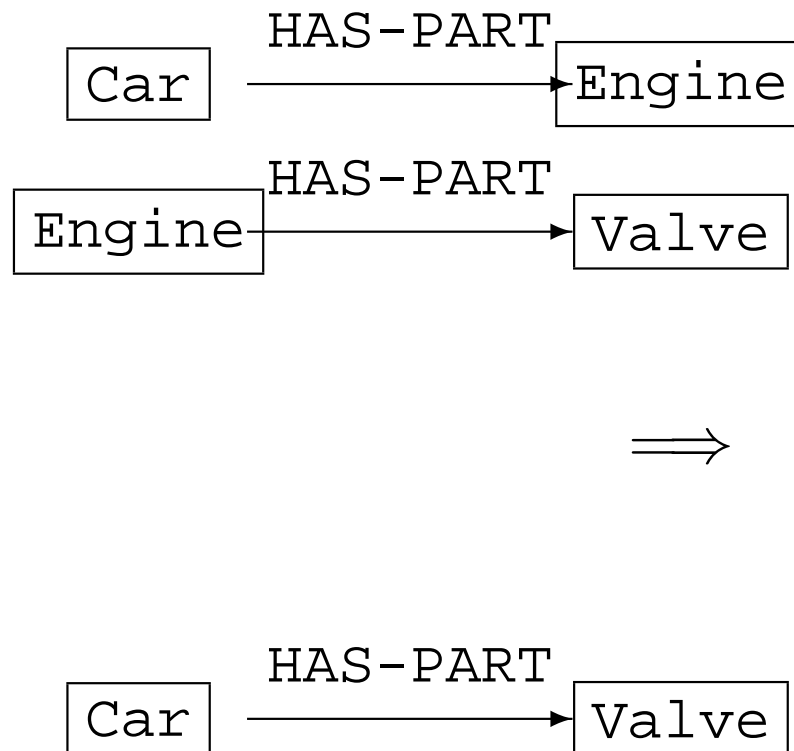
john : AGE : 35,
SEX : *M*,
HEIGHT : 76.

mary : AGE : 32,
SEX : *F*,
HEIGHT : 59.

m-j-family : WIFE : *mary*,
HUSBAND : *john*.

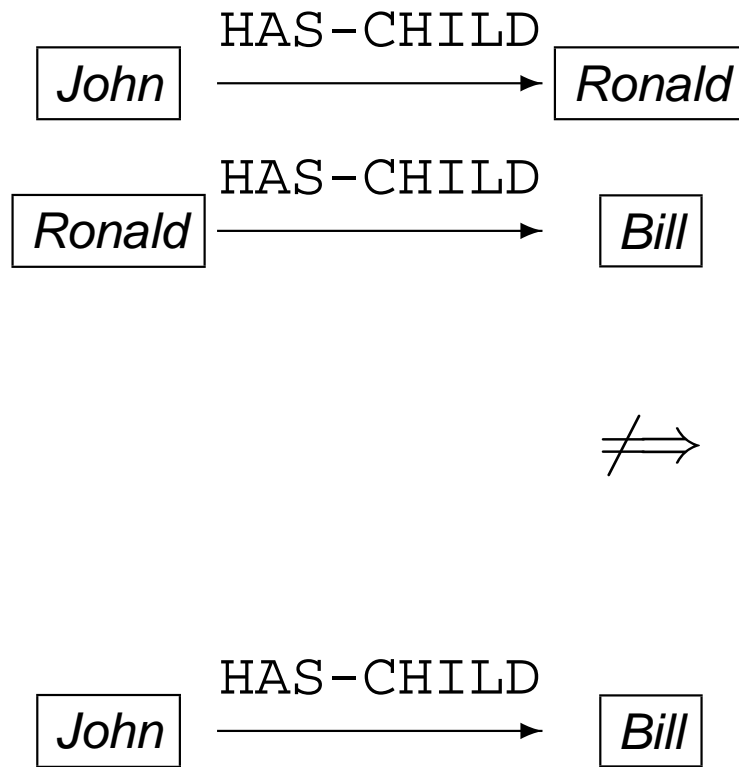
Ambiguities: relations

Special relation:



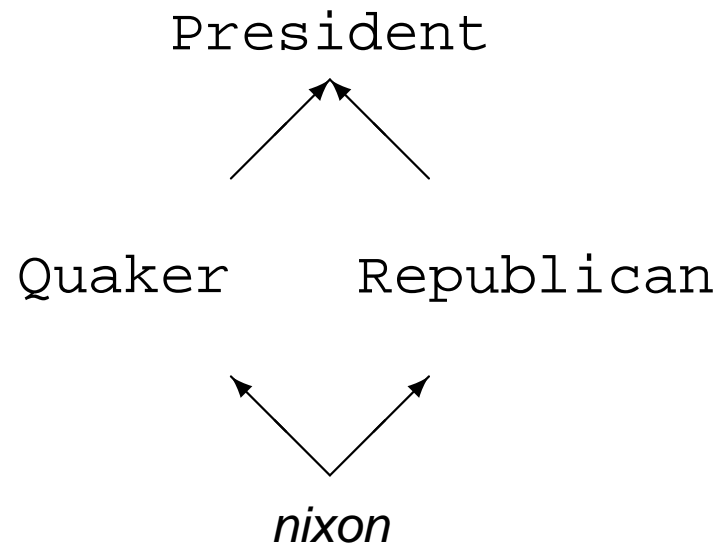
Ambiguities: relations

Normal relation:



Ambiguities: default

The *Nixon* diamond:

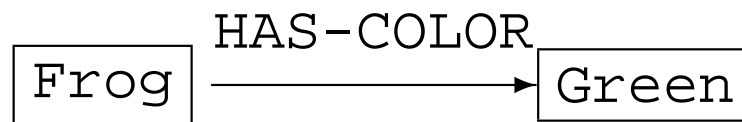


Quakers are pacifist, Republicans are not pacifist.

⇒ Is Nixon pacifist or not pacifist?

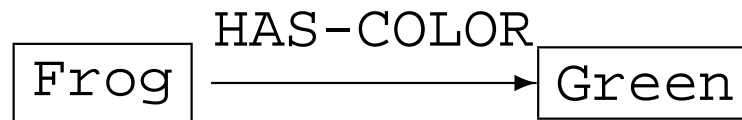
Ambiguities: quantification

What is the exact meaning of:



Ambiguities: quantification

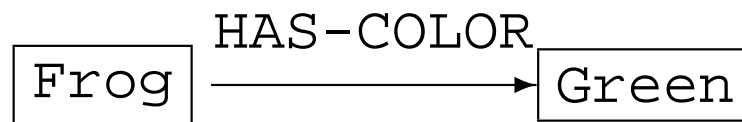
What is the exact meaning of:



- Every frog is just green

Ambiguities: quantification

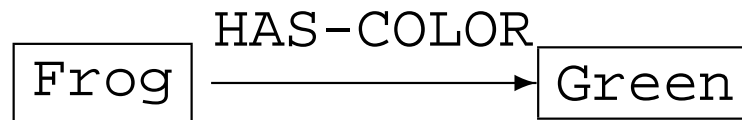
What is the exact meaning of:



- Every frog is just green
- Every frog is also green

Ambiguities: quantification

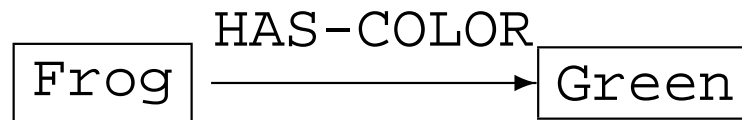
What is the exact meaning of:



- Every frog is just green
- Every frog is also green
- Every frog is of some green

Ambiguities: quantification

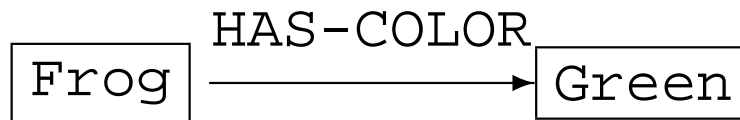
What is the exact meaning of:



- Every frog is just green
- Every frog is also green
- Every frog is of some green
- There is a frog, which is just green

Ambiguities: quantification

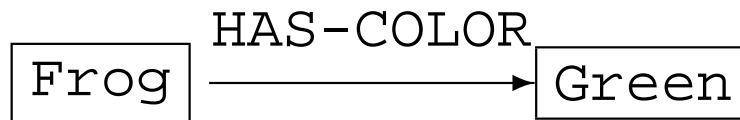
What is the exact meaning of:



- Every frog is just green
- Every frog is also green
- Every frog is of some green
- There is a frog, which is just green
- The *collection* of all frogs is green
- ...

Ambiguities: quantification

What is the exact meaning of:



- Every frog is just green
- Every frog is also green
- Every frog is of some green
- There is a frog, which is just green
- The *collection* of all frogs is green
- ...
- Frogs are typically green, but there may be exceptions

False friends

- The meaning of object-oriented representations is logically very ambiguous.
- The appeal of the graphical nature of object-oriented representation tools has led to forms of reasoning that do not fall into standard logical categories, and are not yet very well understood.
- It is unfortunately much easier to develop some algorithm that appears to reason over structures of a certain kind, than to *justify* its reasoning by explaining what the structures are saying about the domain.

Processing Knowledge = “Reasoning”

Representation alone is not useful.

We want to be able to access represented knowledge and to *process* it.

- access alone is, in general, insufficient
- *implicit* knowledge has to be made explicit

~> *deduction methods*

- the results should only depend on the semantics . . .
- and not on accidental syntactic differences in representations

Logic

A logic allows the axiomatisation of the domain information, and the drawing of conclusions from that information.

- Syntax
- Semantics
- Logical inference = *reasoning*

Important Questions

- **Expressive Power** of representation language

~> able to *represent* the problem

- **Correctness** of entailment procedure

~> *no false* conclusions are drawn

- **Completeness** of entailment procedure

~> *all correct* conclusions are drawn

- **Decidability** of entailment problem

~> there exists a (terminating) algorithm to compute entailment

- **Complexity**

~> resources needed for computing the solution

What is a Logic

Clearly distinguish the definitions of:

- the *formal language*
 - Syntax
 - Semantics
 - Expressive Power
- the *reasoning problem* (e.g., entailment)
 - Decidability
 - Computational Complexity
- the *problem solving procedure*
 - Soundness and Completeness
 - (Asymptotic) Complexity

The ideal Logic

- Expressive
- With decidable reasoning problems
- With sound and complete reasoning procedures
- With efficient reasoning procedures – possibly sub-optimal

Goals of research in the field

- Study how **declarative knowledge** can be *formally defined* using a logic-based approach.
- Give a *computational* account to it, in order to reproduce it in a computing device.

Formal Semantics

- *Formal semantics* describes precisely the meaning of knowledge.
- “Precisely” here means that the semantics does not refer to subjective intuitions, nor is it open to different interpretations by different persons (or machines).
- The importance of formal semantics is well-established in the domain of mathematical logic.
- In the context of ontology languages, the semantics enforces the meaning of the expressed knowledge as a set of constraints over the domain.
- Any possible instantiation of the domain should necessarily conform to the constraints expressed by the ontology.

Models of an ontology

- Given a **statement** in an ontology, the role of the semantics is to devise precisely which are the **models** of the statement, i.e.,
 - **all the possible instantiations of the domain that are compatible with the statement.**
- We say that a statement is **true** in an instantiation of the domain if this instantiation is compatible with the statement;
- the instantiation of the domain in which a statement is true is of course a model of the statement, and vice-versa.
- So, an ontology will itself devise a set of models, which is the intersection among all the models of each statement in the ontology.
- **The models of an ontology represent the only possible realisable situations.**

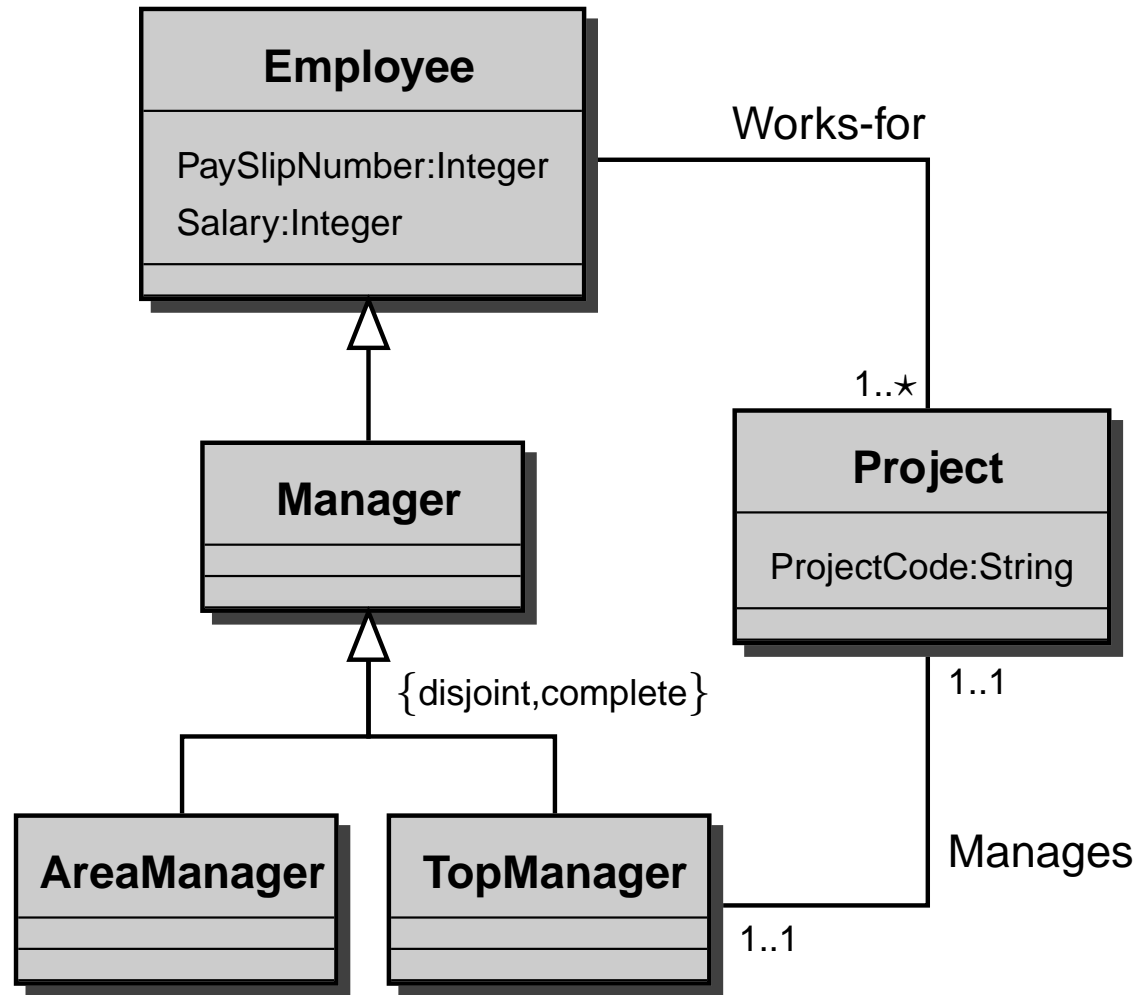
Example

- If an ontology states that professor is a subclass of academic staff member
 - (i.e., in any possible situation, each professor is also an academic staff member),
- and if it is known that Michael Maher is a professor
 - (i.e., Michael Maher is an instance of the professor class),
- then in any possible situation it is necessarily true that Michael Maher is an academic staff member,
 - since the situation in which he would not be an academic staff member is incompatible with the constraints expressed in the ontology.

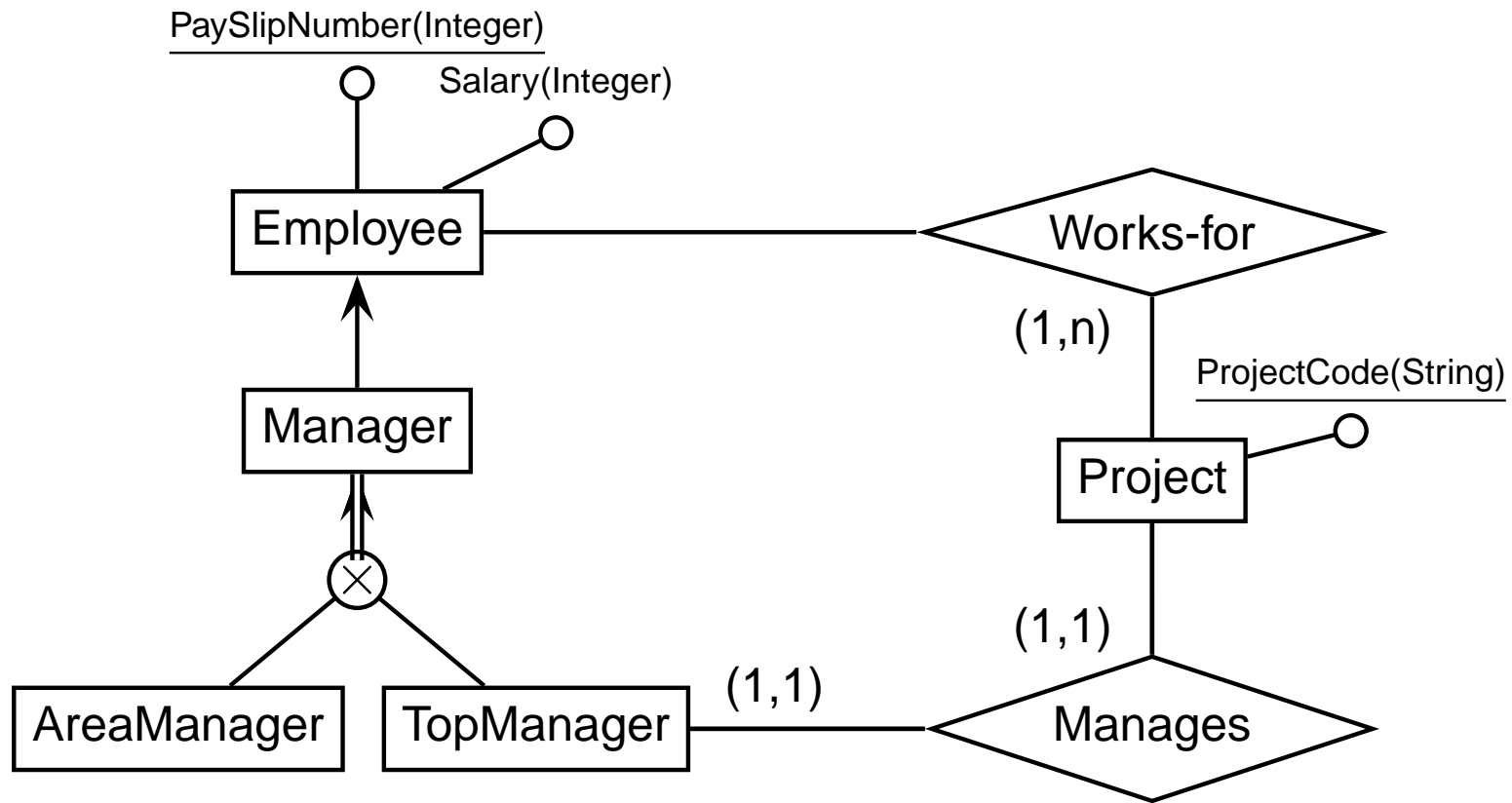
What are Models

- An ontology language talks basically about classes, properties and objects of a domain.
- A model (i.e., a specific instantiation of the domain) is nothing else than the precise characterisation for each objects of the classes it is instance of, and of the properties it participates to.
- So, in the above example, in any model of the ontology Michael Maher should be an instance of the academic staff member class.

An ontology as a UML Class Diagram



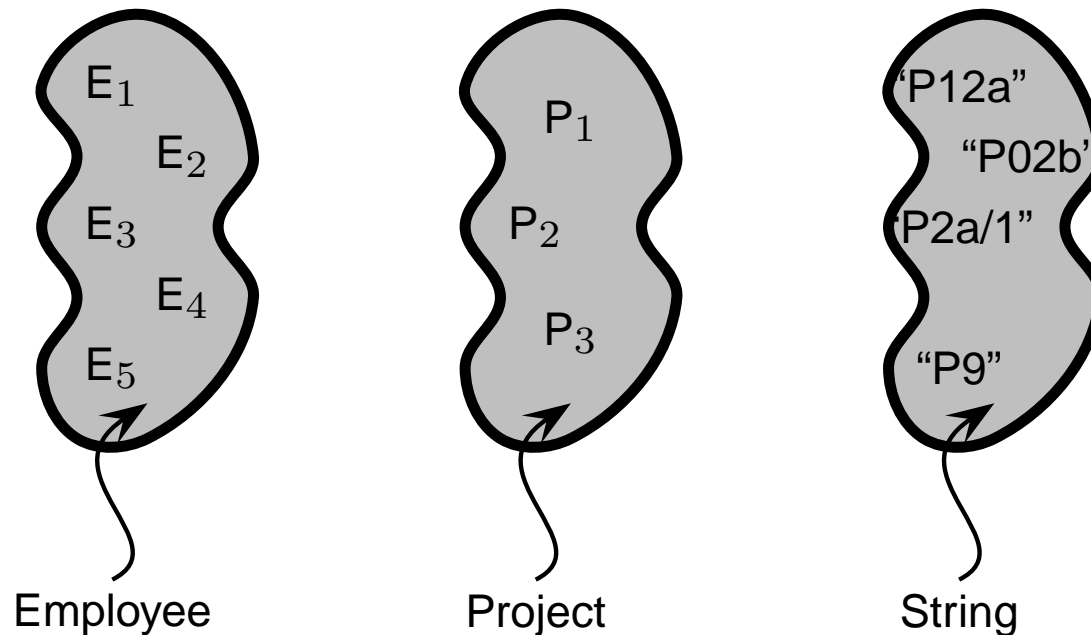
(by the way: Entity-Relationship Schema)



Semantics

In a specific model (i.e., world):

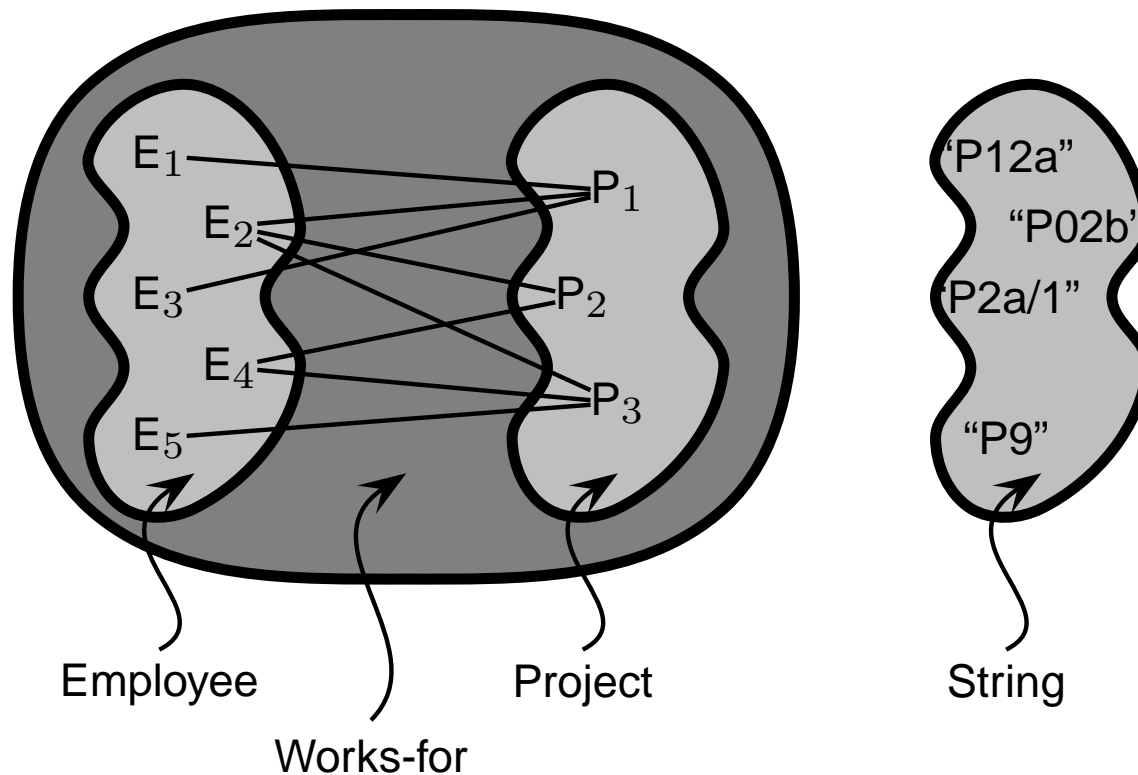
- A class is a **set of instances**;
- a property is a **set of pairs of instances**;
- an attribute (a property with literal value) is a **set of pairs of an instance and a domain element**.



Semantics

In a specific model (i.e., world):

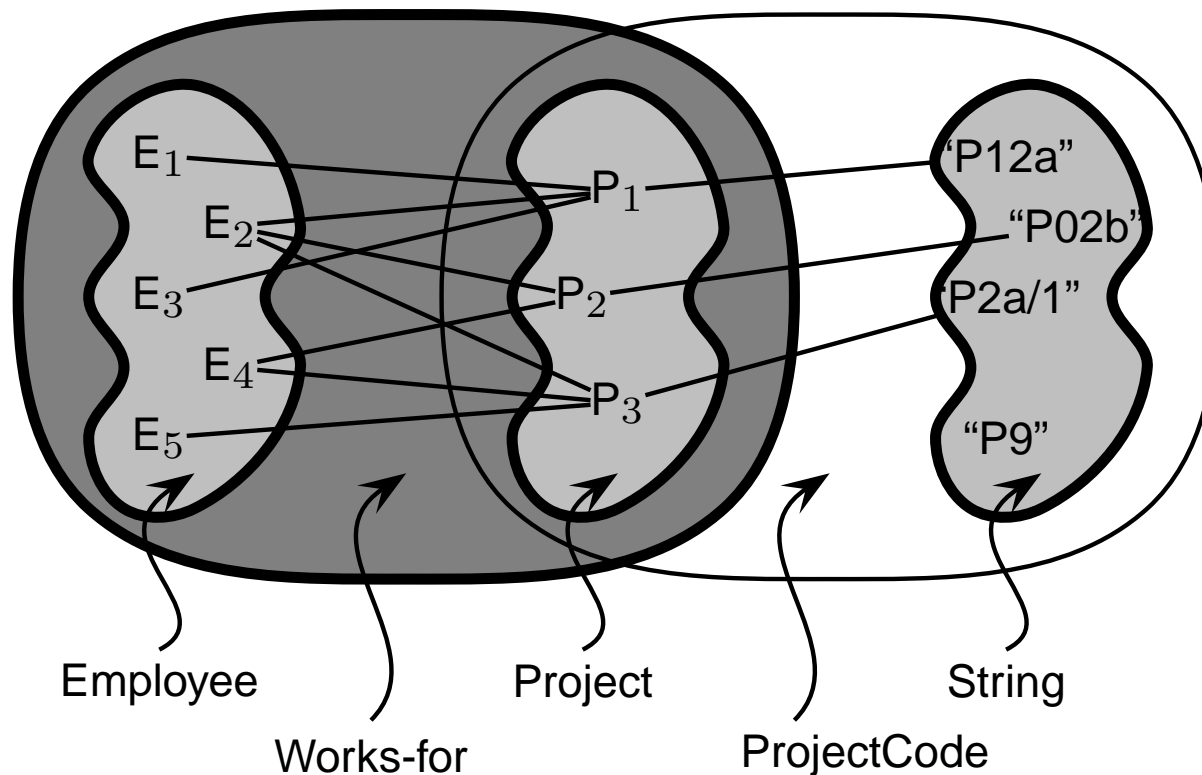
- A class is a **set of instances**;
- a property is a **set of pairs of instances**;
- an attribute (a property with literal value) is a **set of pairs of an instance and a domain element**.



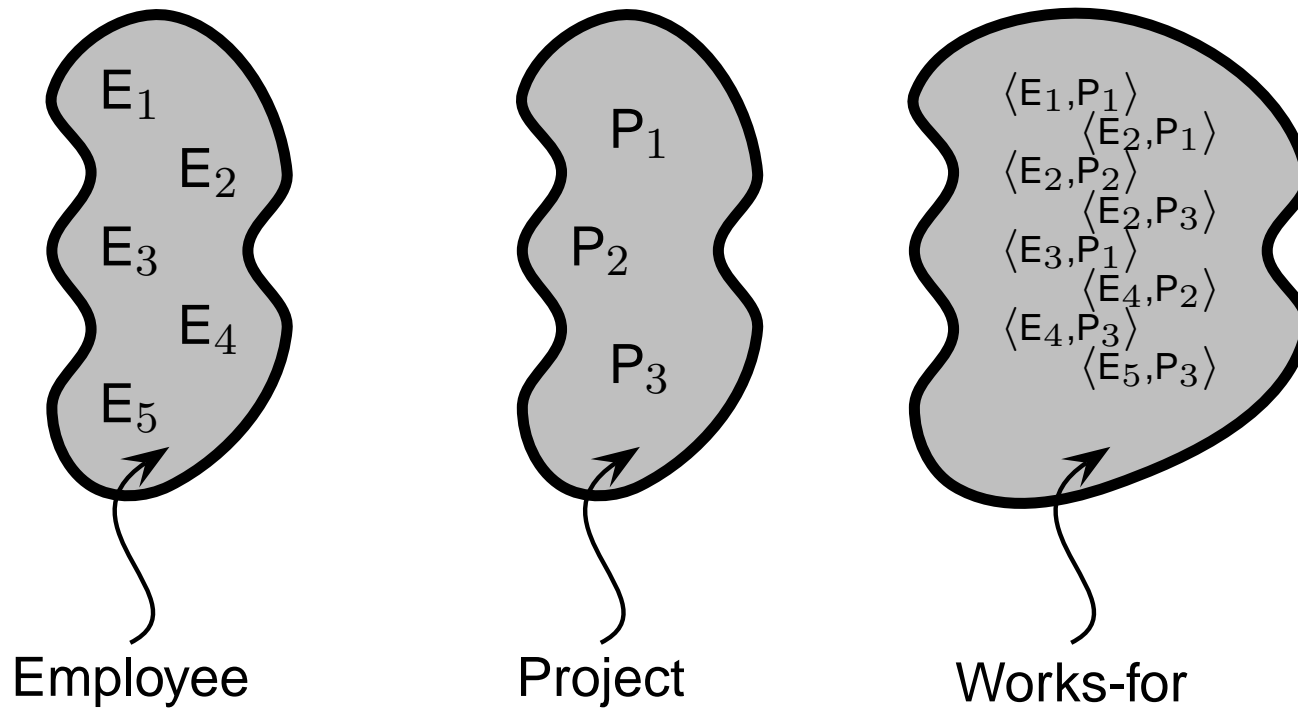
Semantics

In a specific model (i.e., world):

- A class is a **set of instances**;
- a property is a **set of pairs of instances**;
- an attribute (a property with literal value) is a **set of pairs of an instance and a domain element**.



A model is described by sets of instances



The relational representation of a model

Employee

<i>employeeId</i>
E ₁
E ₂
E ₃
E ₄
E ₅

Project

<i>projectId</i>
P ₁
P ₂
P ₃

String

<i>anystring</i>
"P12a"
"P02b"
"P2a/1"
"P9"
...

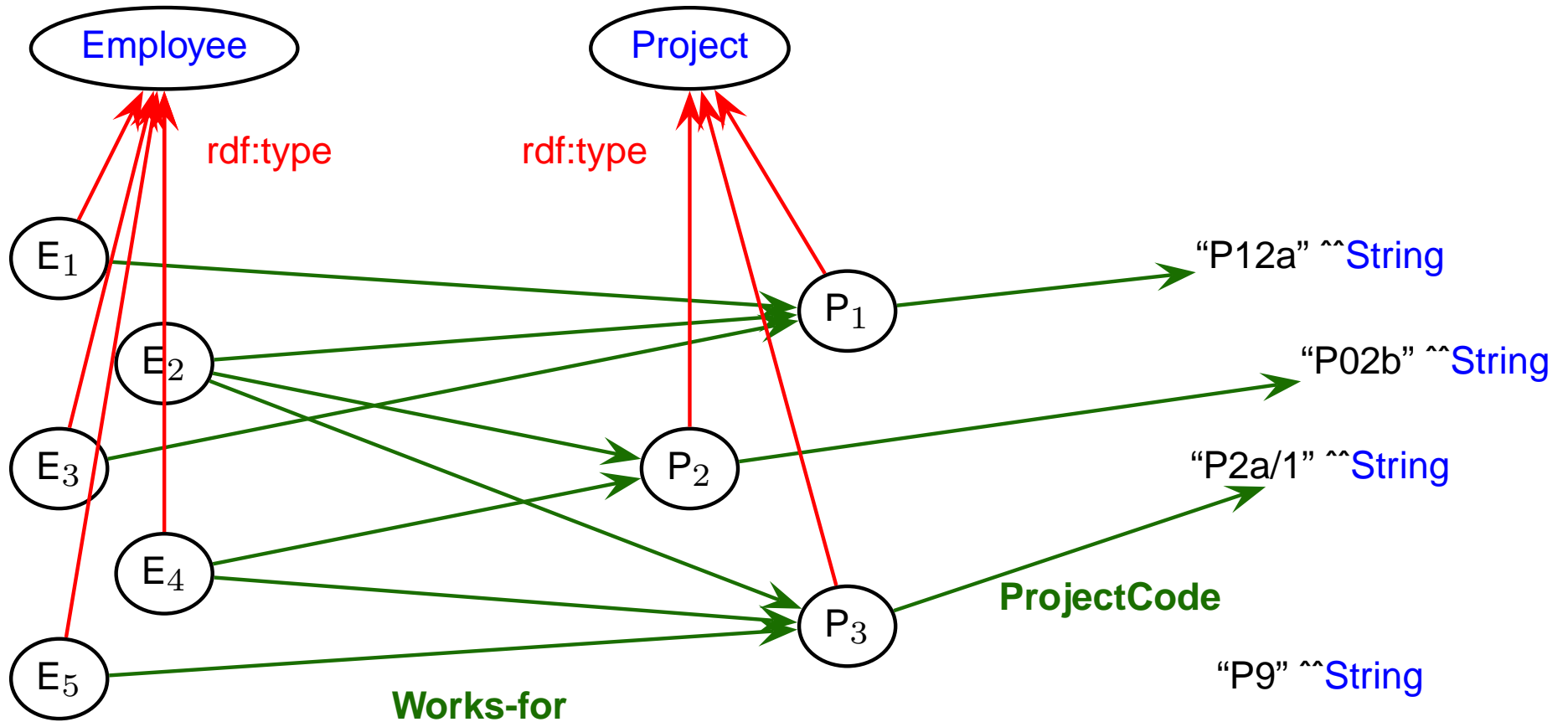
Works-for

<i>employeeId</i>	<i>projectId</i>
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

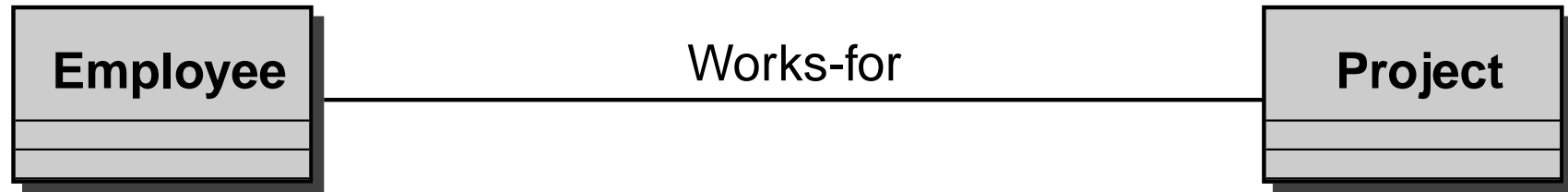
ProjectCode

<i>projectId</i>	<i>pcode</i>
P ₁	"P12a"
P ₂	"P02b"
P ₃	"P2a/1"

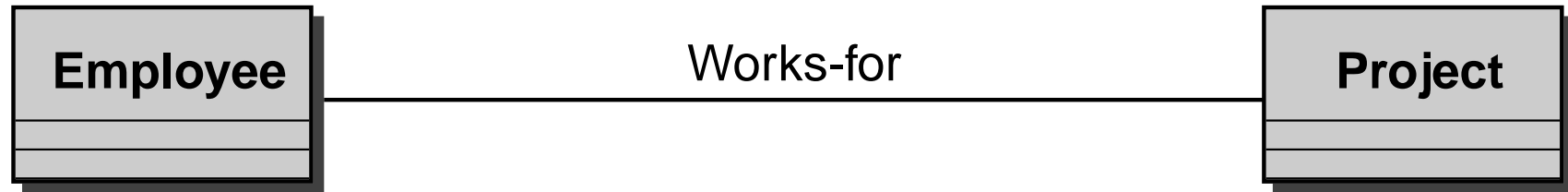
The graph representation of a model: RDF triples



Constraints introduced by Properties

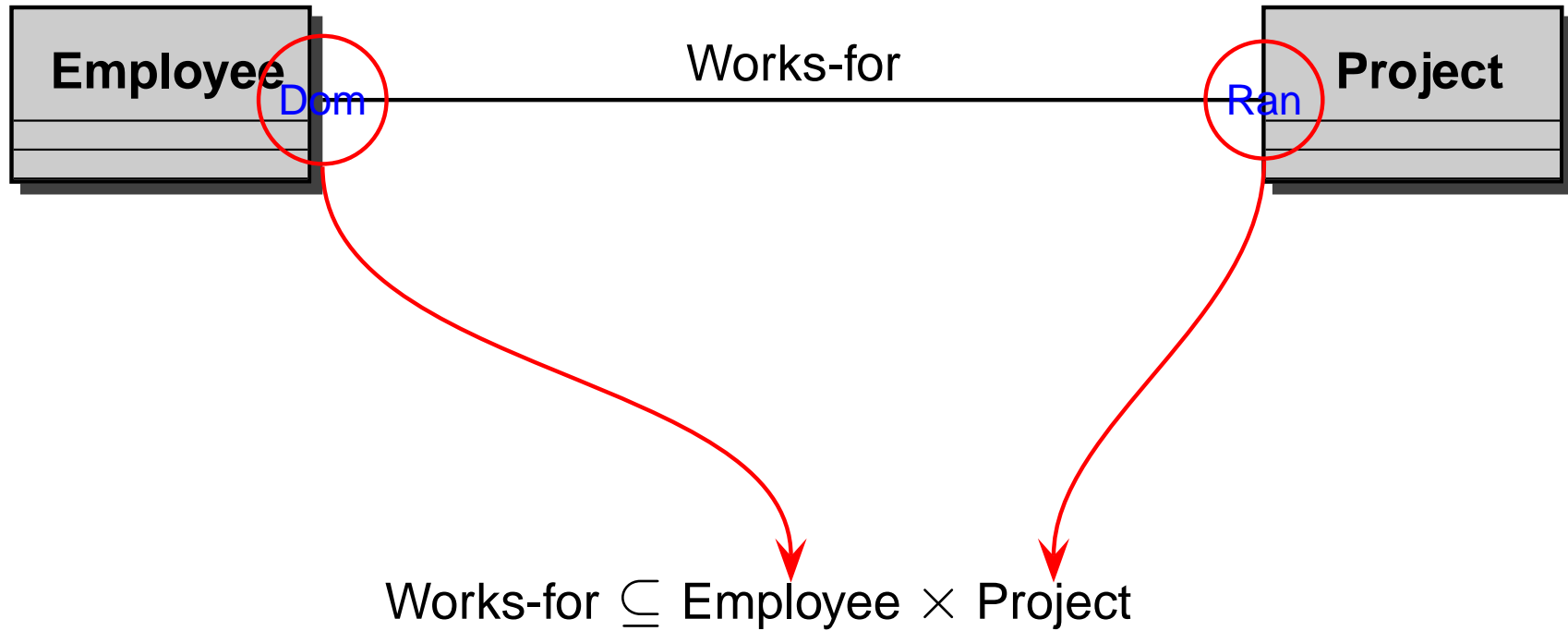


Constraints introduced by Properties



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

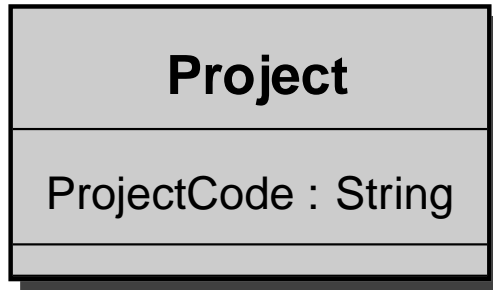
Constraints introduced by Properties



Constraints introduced by Attributes

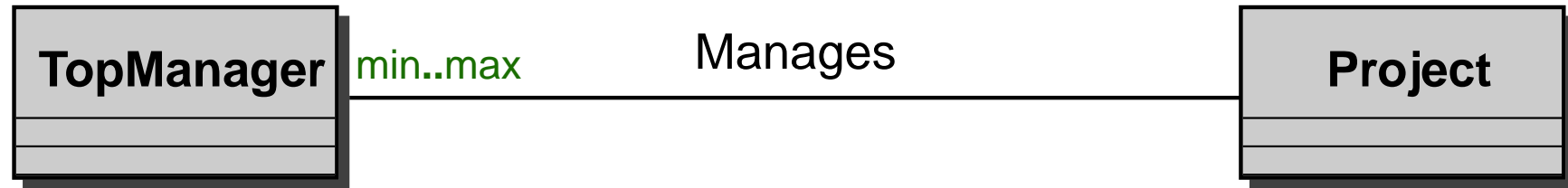


Constraints introduced by Attributes

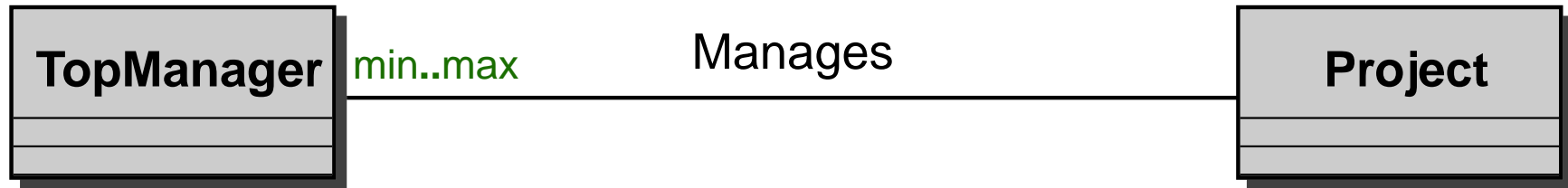


$$\text{Project} \subseteq \{p \mid \#(\text{ProjectCode} \cap (\{p\} \times \text{String})) \geq 1\}$$

Constraints introduced by Cardinality Constraints



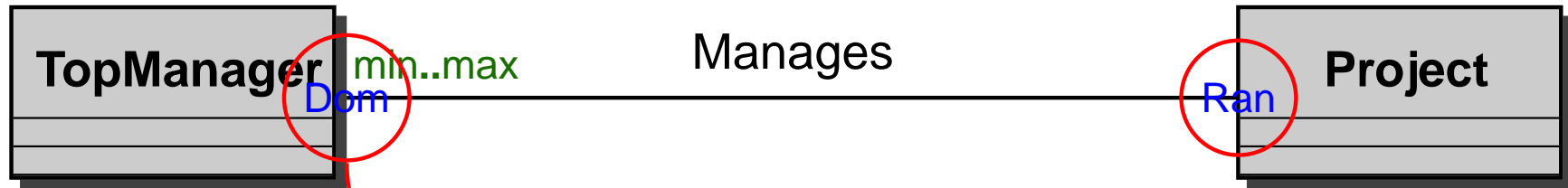
Constraints introduced by Cardinality Constraints



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

(where Ω is the set of all instances)

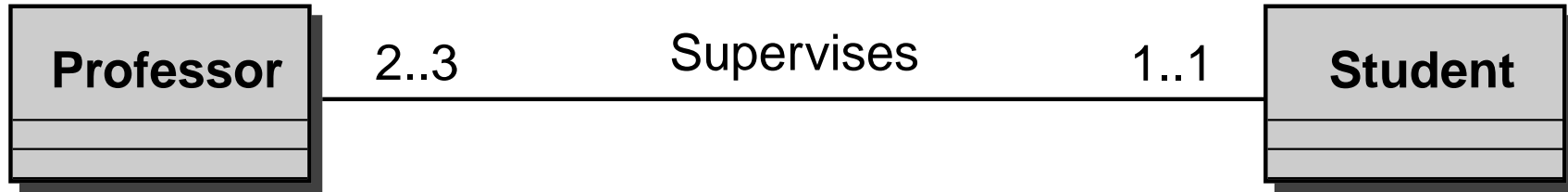
Constraints introduced by Cardinality Constraints



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

(where Ω is the set of all instances)

The Cardinality Construct: An Example



A possible model is:

Professor

<i>professorId</i>
Alex
Bob

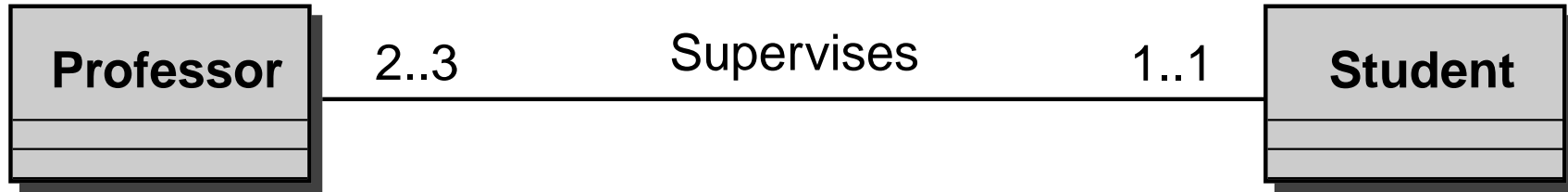
Student

<i>studentId</i>
John
Mary
Nick
Paul
Laura

Supervises

<i>professorId</i>	<i>studentId</i>
Alex	John
Bob	Laura
Alex	Mary
Bob	Nick
Alex	Paul

The Cardinality Construct: An Example



An impossible model is:

Professor

<i>professorId</i>
Alex
Bob

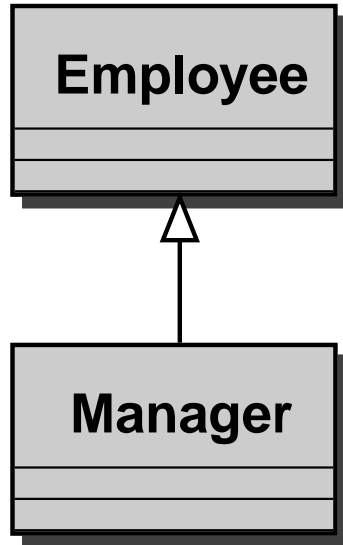
Student

<i>studentId</i>
John
Mary
Nick
Paul
Laura

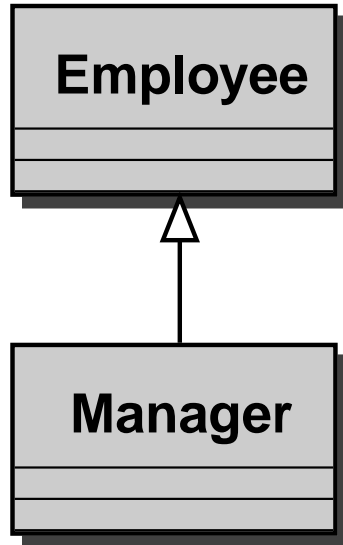
Supervises

<i>professorId</i>	<i>studentId</i>
Alex	John
Bob	Laura
Alex	Mary
Bob	Nick
Alex	Paul
Alex	Laura

Constraints introduced by ISA

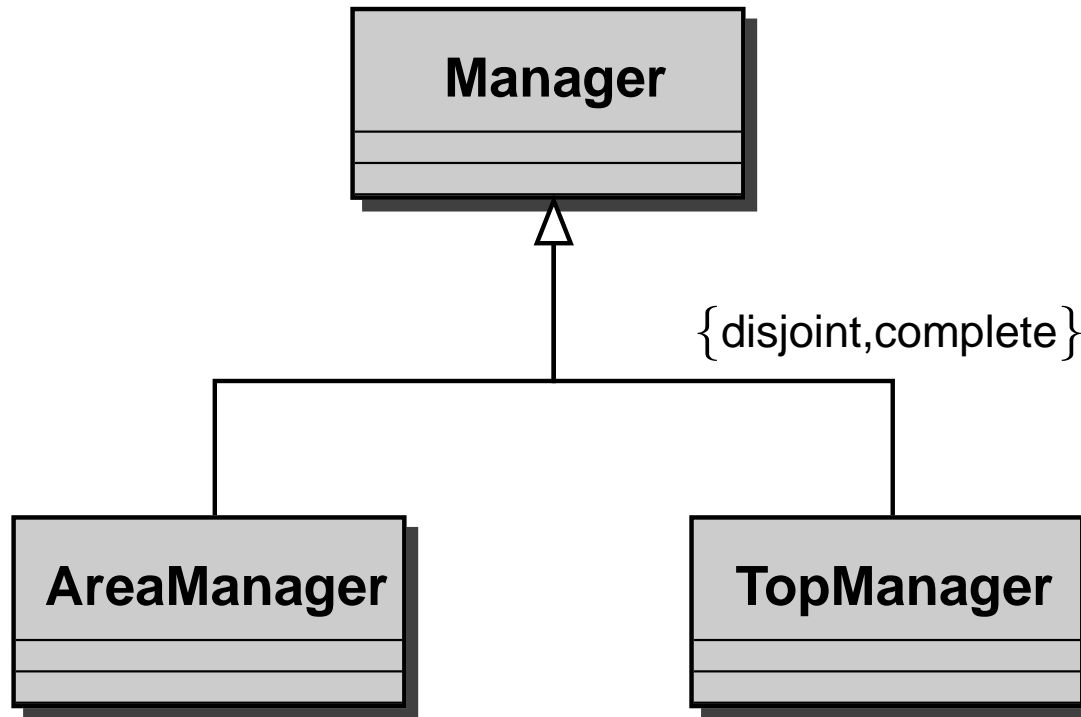


Constraints introduced by ISA

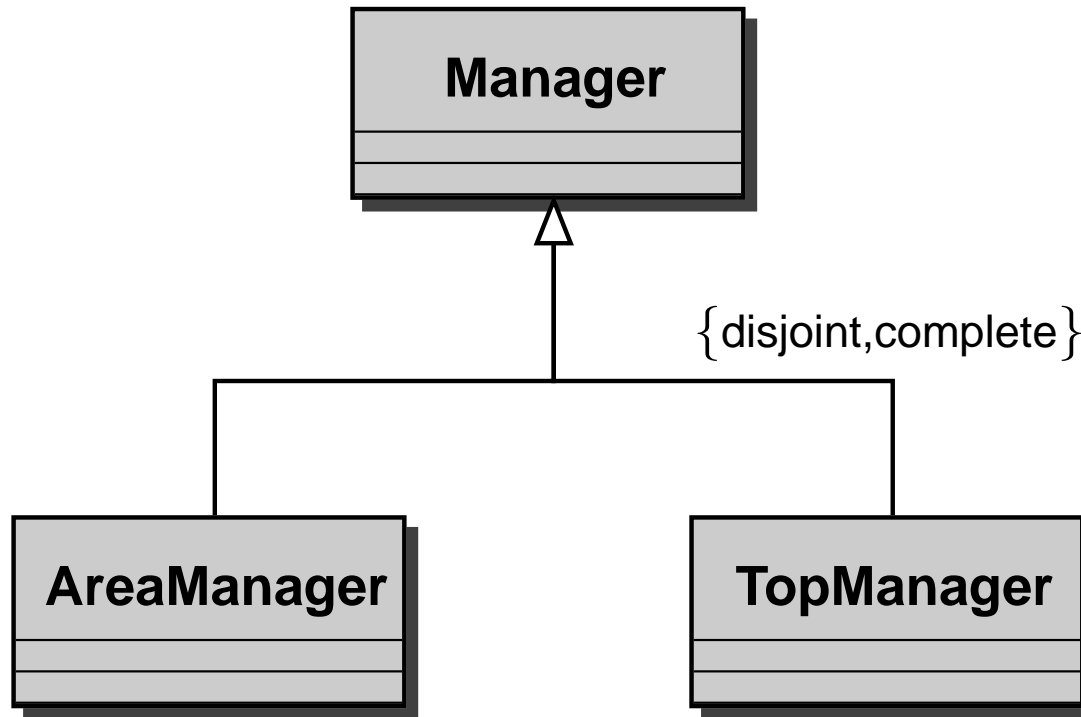


Manager \subseteq Employee

Disjoint and Total constraints



Disjoint and Total constraints



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

Constraints introduced by the initial diagram

Works-for \subseteq Employee \times Project

Manages \subseteq TopManager \times Project

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

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

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

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

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

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

Manager \subseteq Employee

AreaManager \subseteq Manager

TopManager \subseteq Manager

AreaManager \cap TopManager = \emptyset

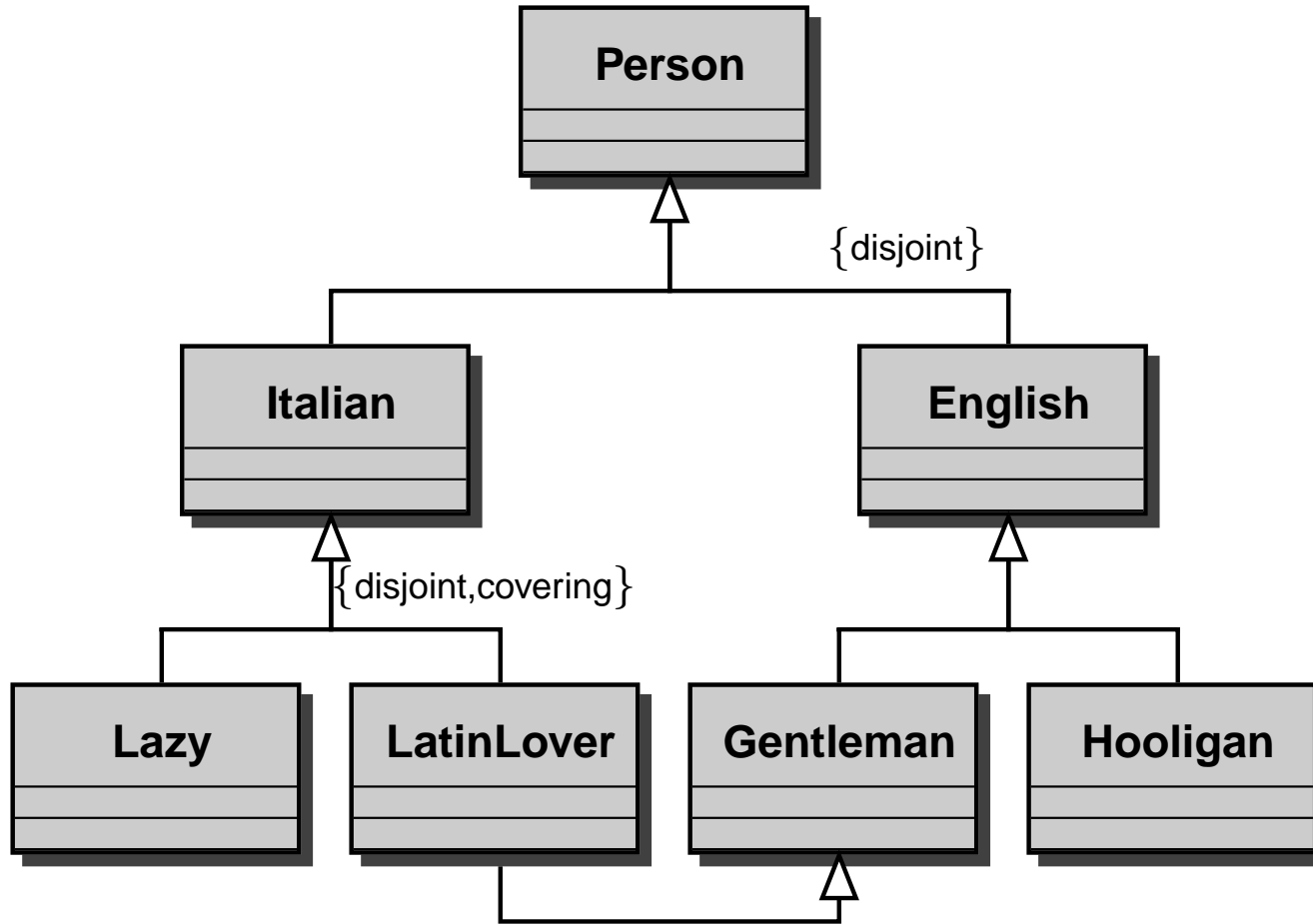
Manager \subseteq AreaManager \cup TopManager

Reasoning

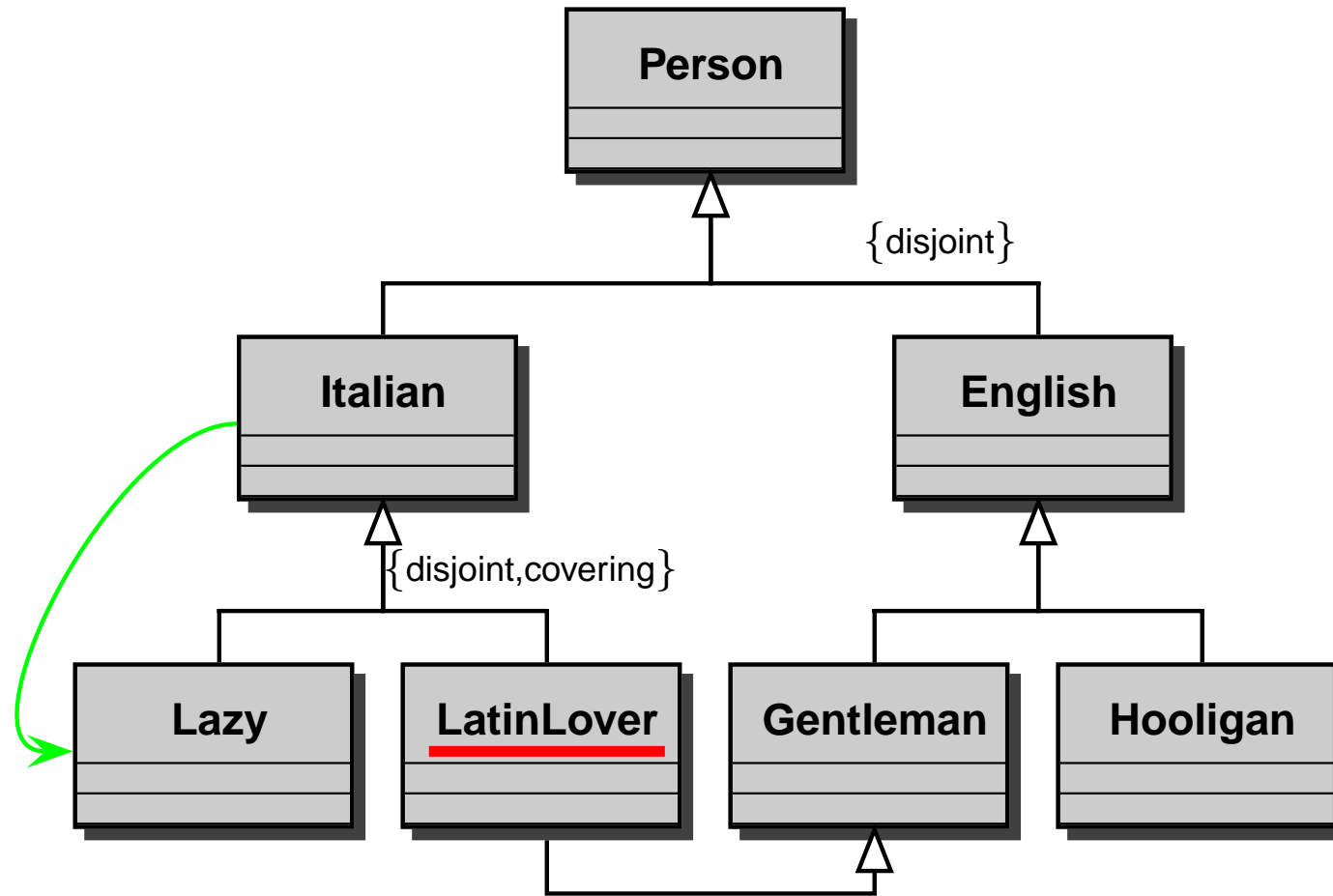
Given an ontology – seen as a collection of constraints – it is possible that additional constraints can be inferred.

- A class is **inconsistent** if it denotes the empty set in any legal world description.
- A class is a **subclass** of another class if the former denotes a subset of the set denoted by the latter in any legal world description.
- Two classes are **equivalent** if they denote the same set in any legal world description.
- A **stricter** constraint is inferred – e.g., a **cardinality** constraint – if it holds in in any legal world description.
- . . .

Simple reasoning example



Simple reasoning example

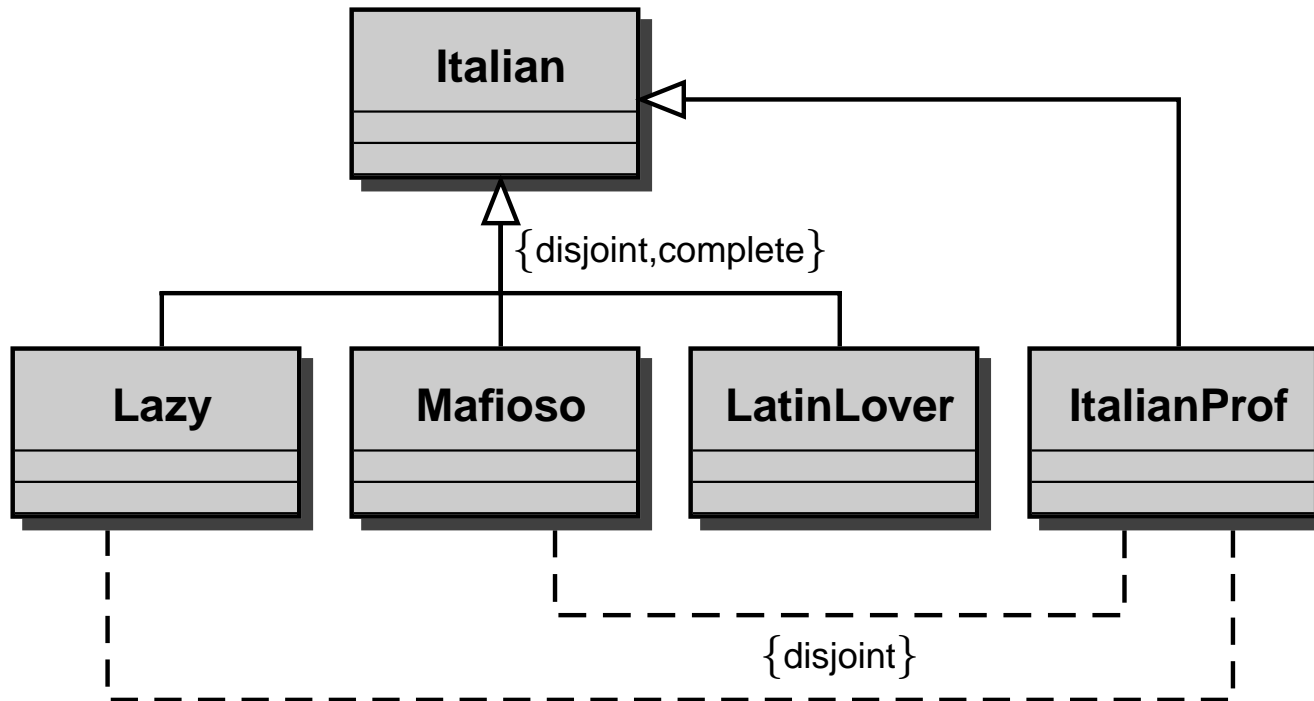


implies

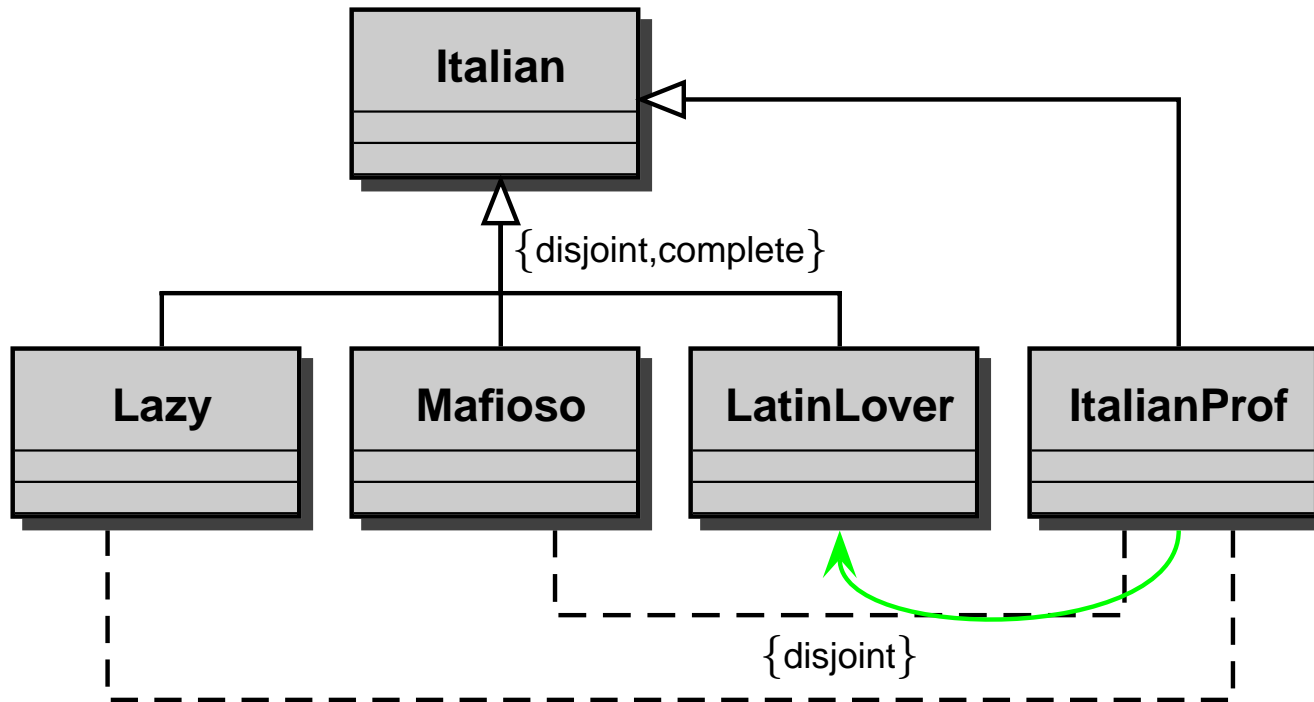
LatinLover = \emptyset

Italian \subseteq Lazy – Italian \equiv Lazy

Reasoning by cases



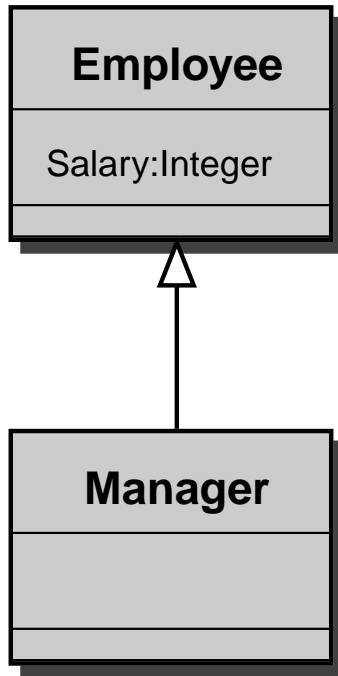
Reasoning by cases



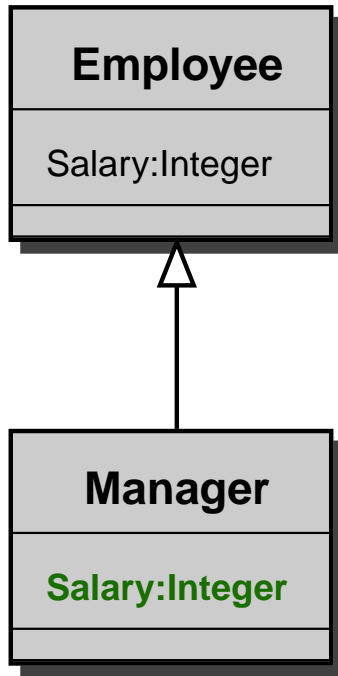
implies

ItalianProf \subseteq LatinLover

ISA and Inheritance



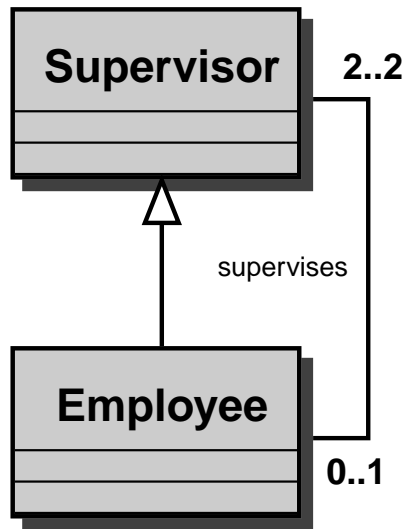
ISA and Inheritance



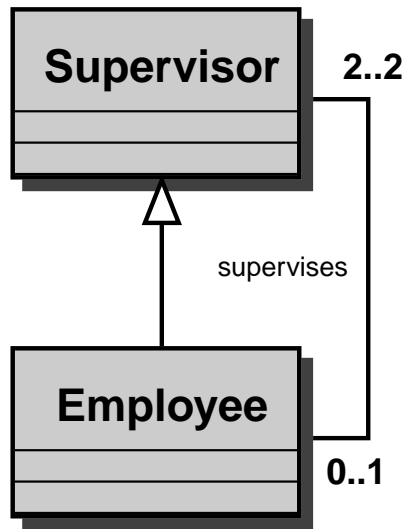
implies

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

Infinite worlds: the democratic company



Infinite worlds: the democratic company

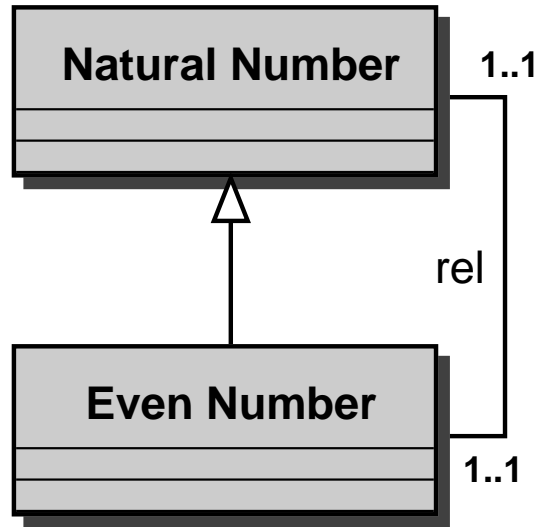


implies

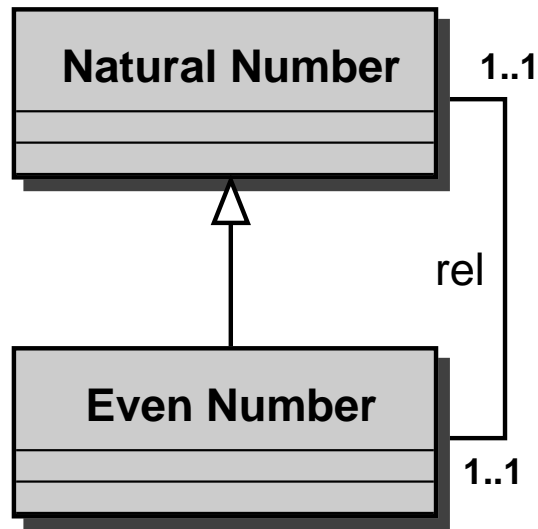
“the classes **Employee** and **Supervisor** contain an infinite number of instances”.

Therefore, the schema is inconsistent.

Bijection: how many numbers



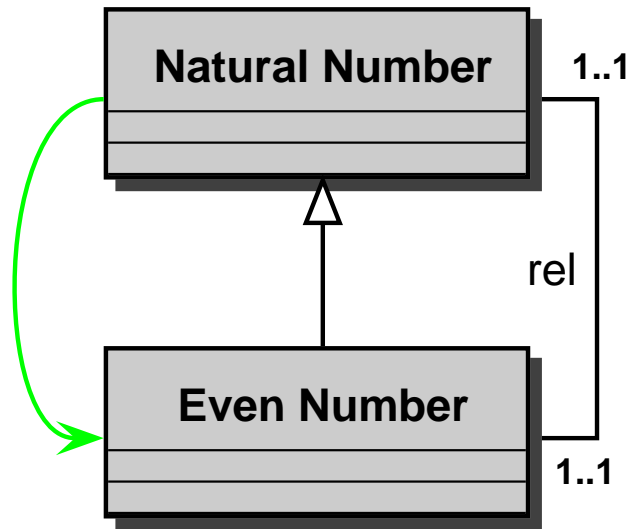
Bijection: how many numbers



implies

“the classes *'Natural Number'* and *'Even Number'* contain the same number of instances”.

Bijection: how many numbers



implies

“the classes *'Natural Number'* and *'Even Number'* contain the same number of instances”.

If the domain is finite: Natural Number \equiv Even Number

i●com: Intelligent Conceptual Modelling tool

- i●com allows for the specification of multiple ontologies diagrams and inter- and intra-schema constraints;
- Complete logical reasoning is employed by the tool using a hidden underlying (description logic) inference engine;
- i●com verifies the specification, infers implicit facts and stricter constraints, and manifests any inconsistencies during the conceptual modelling phase.
- `www.inf.unibz.it/~franconi/icom/`

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.
- An alternative way to 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 FOL theory associated to it.

FOL: The Alphabet

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

- 1-ary predicate symbols: E_1, E_2, \dots, E_n for each Class (Entity);
 D_1, D_2, \dots, D_m for each Basic Domain.
- binary predicate symbols: A_1, A_2, \dots, A_k for each Attribute.
- binary predicate symbols: P_1, R_2, \dots, P_p for each Property.

FOL Notation

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

$$\bar{x} = x_1, \dots, x_n$$

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

$$\exists^{\leq n} x. \varphi(x) \equiv$$

$$\begin{aligned} \forall x_1, \dots, x_n, x_{n+1}. \varphi(x_1) \wedge \dots \wedge \varphi(x_n) \wedge \varphi(x_{n+1}) \rightarrow \\ (x_1 = x_2) \vee \dots \vee (x_1 = x_n) \vee (x_1 = x_{n+1}) \vee \\ (x_2 = x_3) \vee \dots \vee (x_2 = x_n) \vee (x_2 = x_{n+1}) \vee \\ \dots \vee (x_n = x_{n+1}) \end{aligned}$$

$$\exists^{\geq n} x. \varphi(x) \equiv$$

$$\begin{aligned} \exists x_1, \dots, x_n. \varphi(x_1) \wedge \dots \wedge \varphi(x_n) \wedge \\ \neg(x_1 = x_2) \wedge \dots \wedge \neg(x_1 = x_n) \wedge \\ \neg(x_2 = x_3) \wedge \dots \wedge \neg(x_2 = x_n) \wedge \\ \dots \wedge (x_{n-1} = x_n) \end{aligned}$$

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:
 - $\mathcal{B} = \bigcup_{i=1}^m \mathcal{B}_{Di}$. \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:

$$D_i^{\mathcal{I}} = \mathcal{B}_{D_i} \quad (\text{e.g., } \text{String}^{\mathcal{I}} = \mathcal{B}_{\text{String}}).$$

- *Entity-set Predicates* to elements of the entity domain:

$$E_i^{\mathcal{I}} \subseteq \Omega.$$

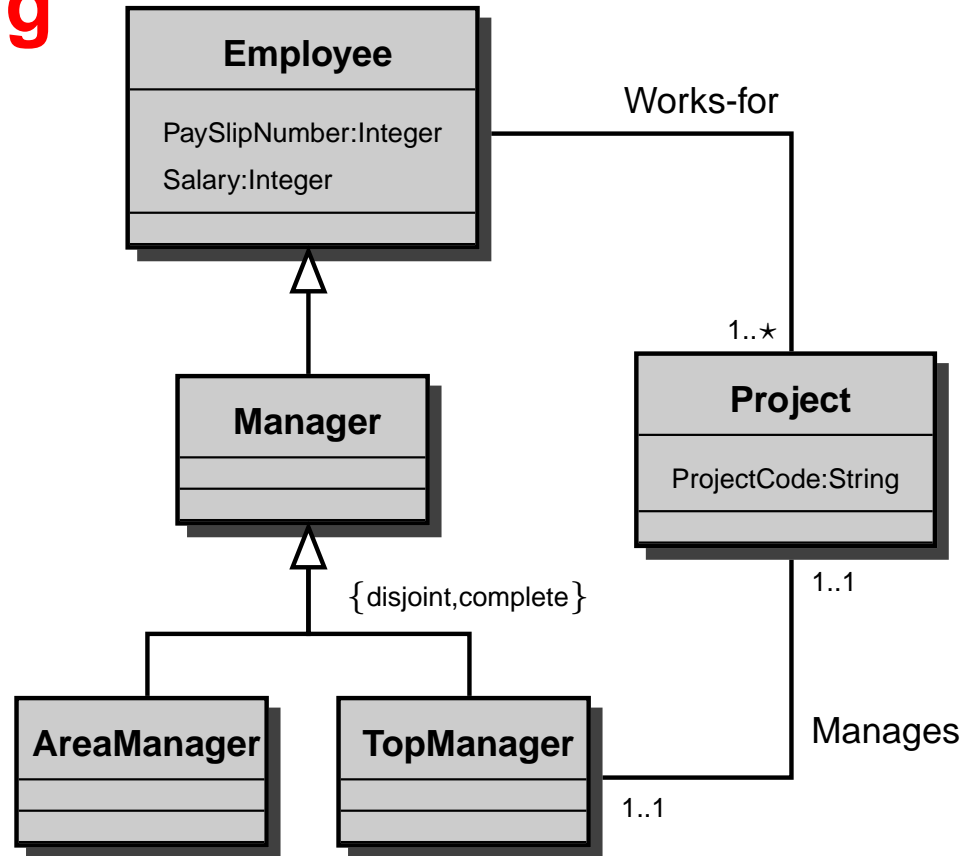
- *Attribute Predicates* to binary relations such that:

$$A_i^{\mathcal{I}} \subseteq \Omega \times \mathcal{B}.$$

- *Property Predicates* to binary relations over the entity domain:

$$R_i^{\mathcal{I}} \subseteq \Omega \times \Omega = \Omega^2.$$

FOL encoding



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