UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Conceptual Design and the Entity-Relationship Model

Basi di Dati

Bachelor's Degree in Computer Engineering
Academic Year 2024/2025

Stefano Marchesin

Intelligent Interactive Information Access (IIIA) Hub
Department of Information Engineering
University of Padua

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

Outline

- Entity-relationship constructs

- Constraints in the entity-relationship model

- Documentation

The Entity-Relationship (ER) Model

- The **Entity-Relationship** (**ER**) **model** provides several constructs that impact on both the **intensional level** (**schema**) and the **extensional level** (**instance**)
  - Entity
  - Attribute of entity
  - Relationship
  - Role
  - Attribute of relationship
  - Generalization [not covered in the lectures]
  - Identification constraints
  - Cardinality constraints
  - Other constraints
- The ER model allows us to express the **conceptual schema** representing the mini-world in **graphical form**

Entity

Entity

An entity is a class of objects of the real world (facts, people, things) which exist autonomously and have shared properties

Student

Vehicle

Course

Department

- Each entity has a **name** which **univocally identify** it in the schema
- In the ER diagram, an entity is represented by a **rectangle** which contains the name of the entity
- We use the **classification abstraction** to create entities

Entity: Extensional Semantics

- At **extensional** level an entity consist of a set of objects called **instances** (or **occurrences**) of the entity

- Therefore, if in a schema **S** an entity **e** is defined, in each instance **i** of the schema **S**, the entity **e** is associated with a **set of objects**, also called the **extension** of **e** in the instance **i** of **S**

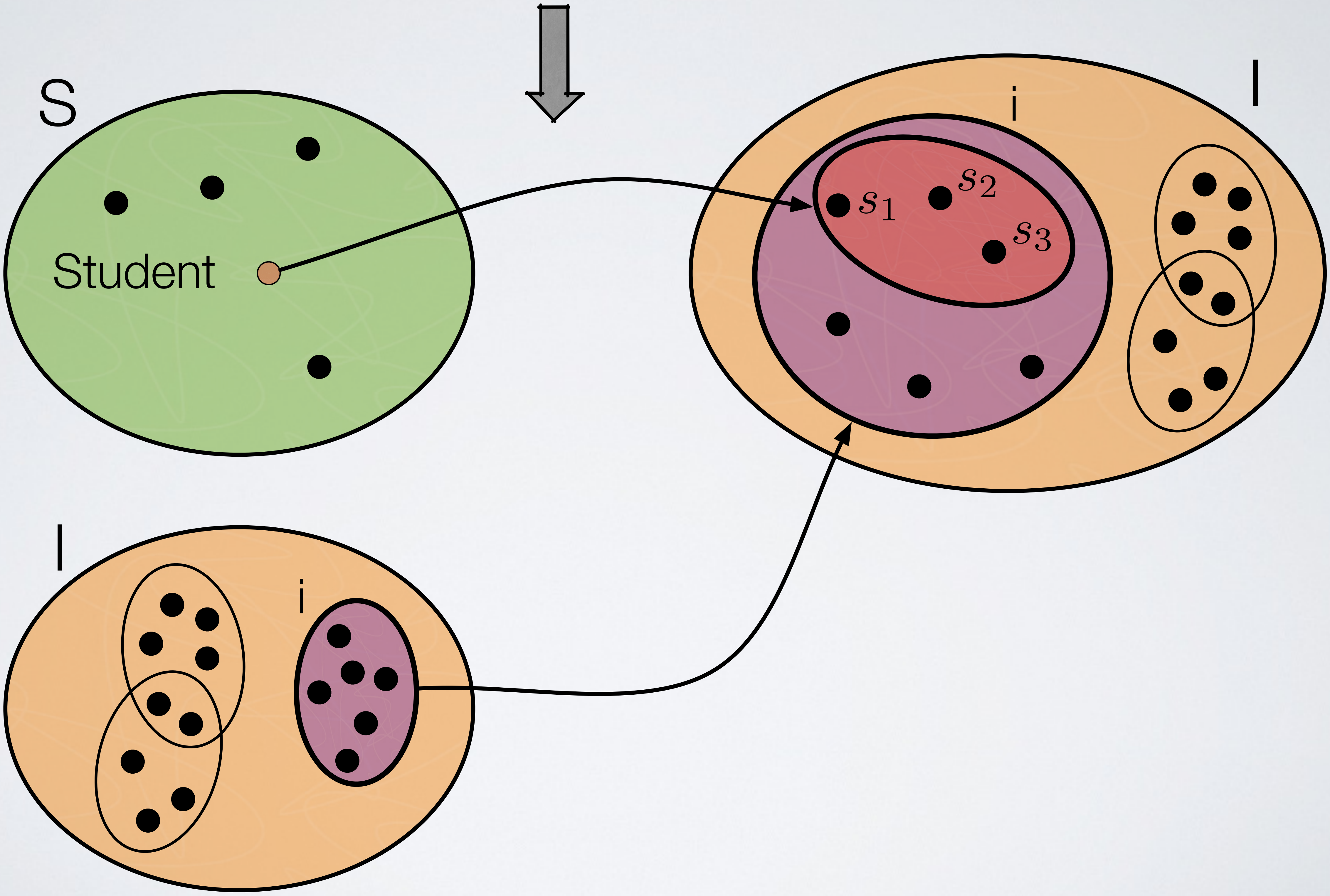
$$\begin{aligned} \text{instance}: \quad & \mathbf{I} \times \mathbf{S} \rightarrow \mathbf{I} \\ & (\mathsf{i}, \mathsf{e}) \mapsto \text{instance}(\mathsf{i}, \mathsf{e}) = \{e_1, e_2, \ldots, e_n\} \end{aligned}$$

- An **instance** e_i of an entity is not a value that identifies an object but it is the **object itself**

- In the conceptual schema we represent only the entities and not their instances

Entity: Example of Extensional Semantics

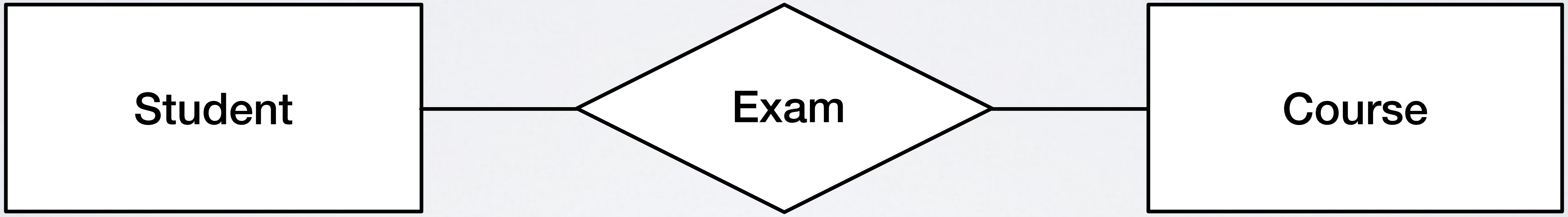
$$\text{instance}(\mathsf{i}, \mathsf{Student}) = \{s_1, s_2, s_3\}$$



Relationship

Relationship

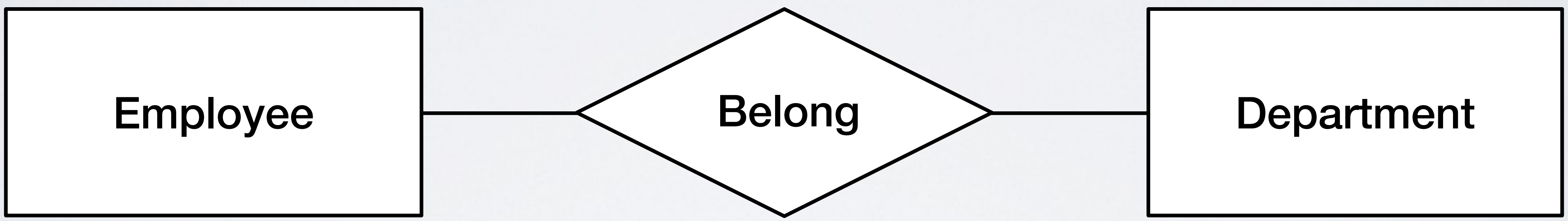
A relationship represents an association among two or more entities. The number of entities participating in a relationship is called the degree of the relationship.



- Each relationship has a **name** which **univocally identify** it in the schema
- In the ER diagram, a relationship is represented by a **rhombus** which contains the name of the relationship
- We use the **association abstraction** to create relationships

Relationship

A **relationship** represents an **association** among **two or more entities**. The number of entities participating in a relationship is called the **degree** of the relationship.



- Each relationship has a **name** which **univocally identify** it in the schema
- In the ER diagram, a relationship is represented by a **rhombus** which contains the name of the relationship
- We use the **association abstraction** to create relationships

Binary Relationship: Extensional Semantics

- At **extensional** level a binary relationship **r** between entities **e** and **f** consists of the pairs $r_k = (e_i, f_j)$ such that e_i is an instance of **e** and f_j is an instance of **f**. Each pair is an instance (or occurrence) of the relationship **r**

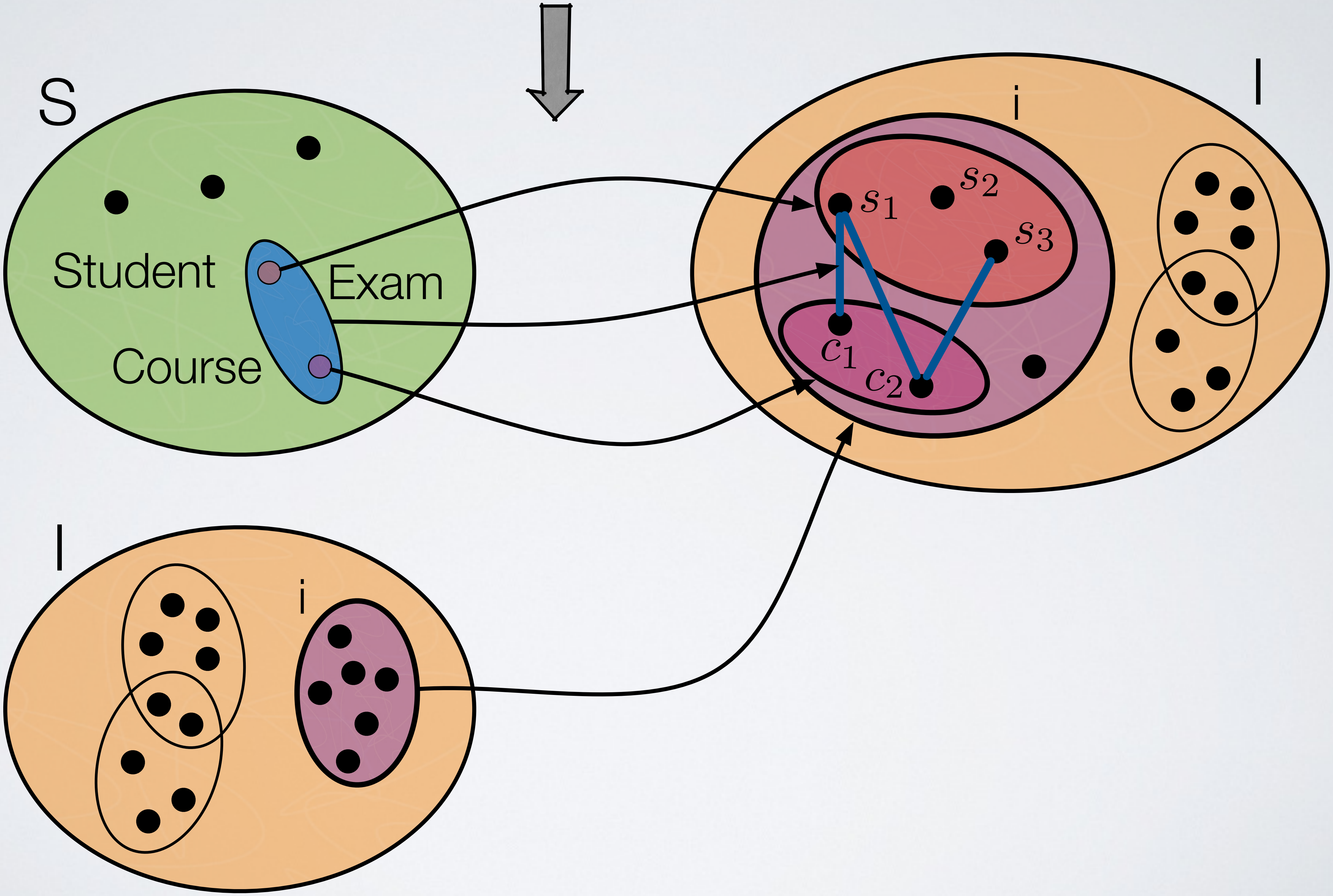
- Therefore, if in a schema **S** a relationship **r** is defined between the entities **e** and **f**, in each instance **i** of the schema **S**, the relationship **r** is associated with a **set of pairs**, also called the **extension** of **r** in the instance **i** of **S**

$$
\begin{aligned}
\text{instance}: \quad & \mathsf{I} \times \mathsf{S} \to \mathsf{I} \times \mathsf{I} \\
& (\mathsf{i}, \mathsf{r}) \mapsto \text{instance}(\mathsf{i}, \mathsf{r}) \subseteq \text{instance}(\mathsf{i}, \mathsf{e}) \times \text{instance}(\mathsf{i}, \mathsf{f}) \\
& \qquad = \{(e_1, f_1), (e_2, f_2) \ldots, (e_n, f_n)\}
\end{aligned}
$$

- In the conceptual schema we represent only the relationships and not their instances

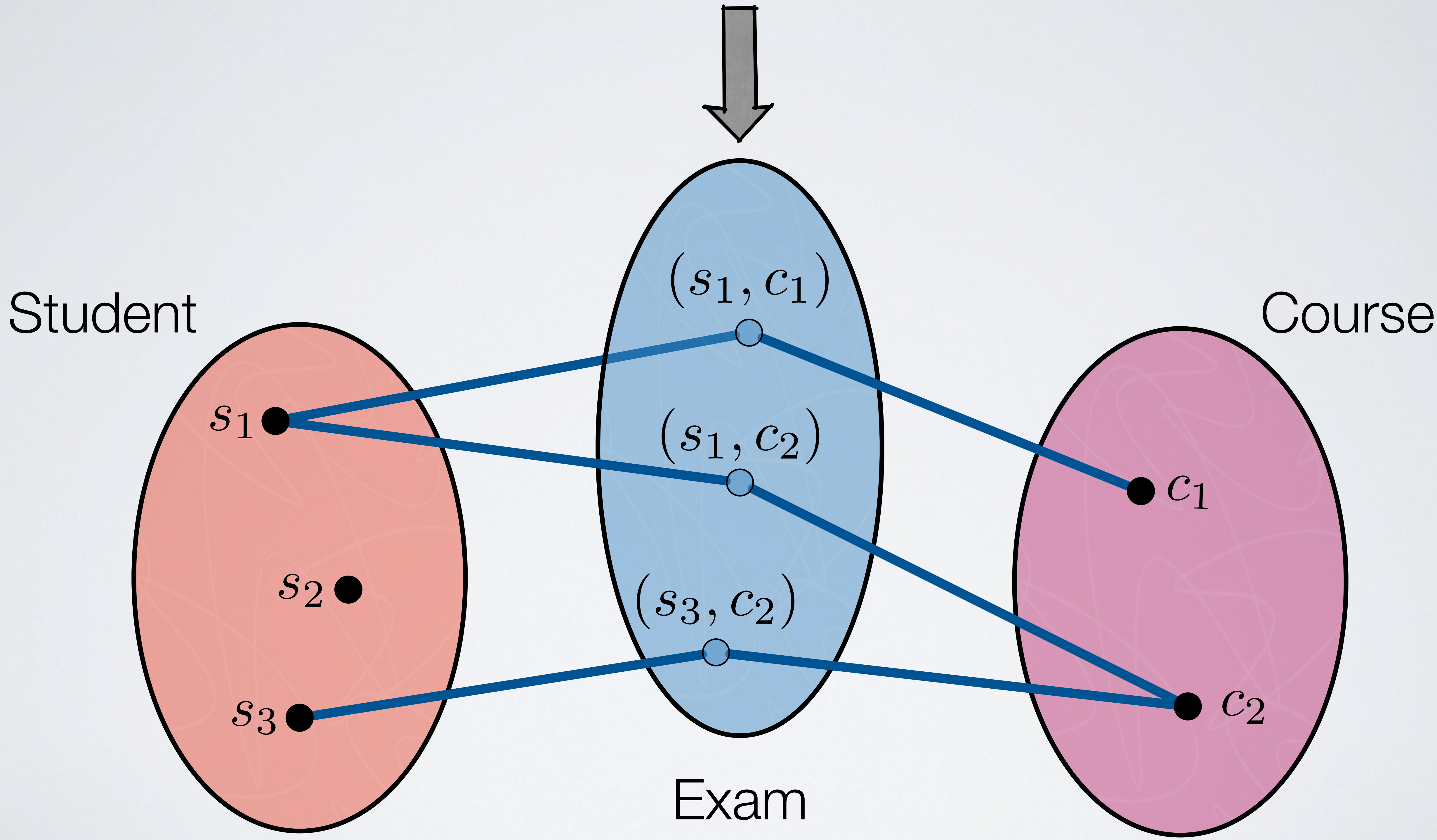
Binary Relationship: Example of Extensional Semantics

$$\text{instance}(\mathsf{i}, \mathsf{Exam}) = \{(s_1, c_1), (s_1, c_2), (s_3, c_2)\}$$



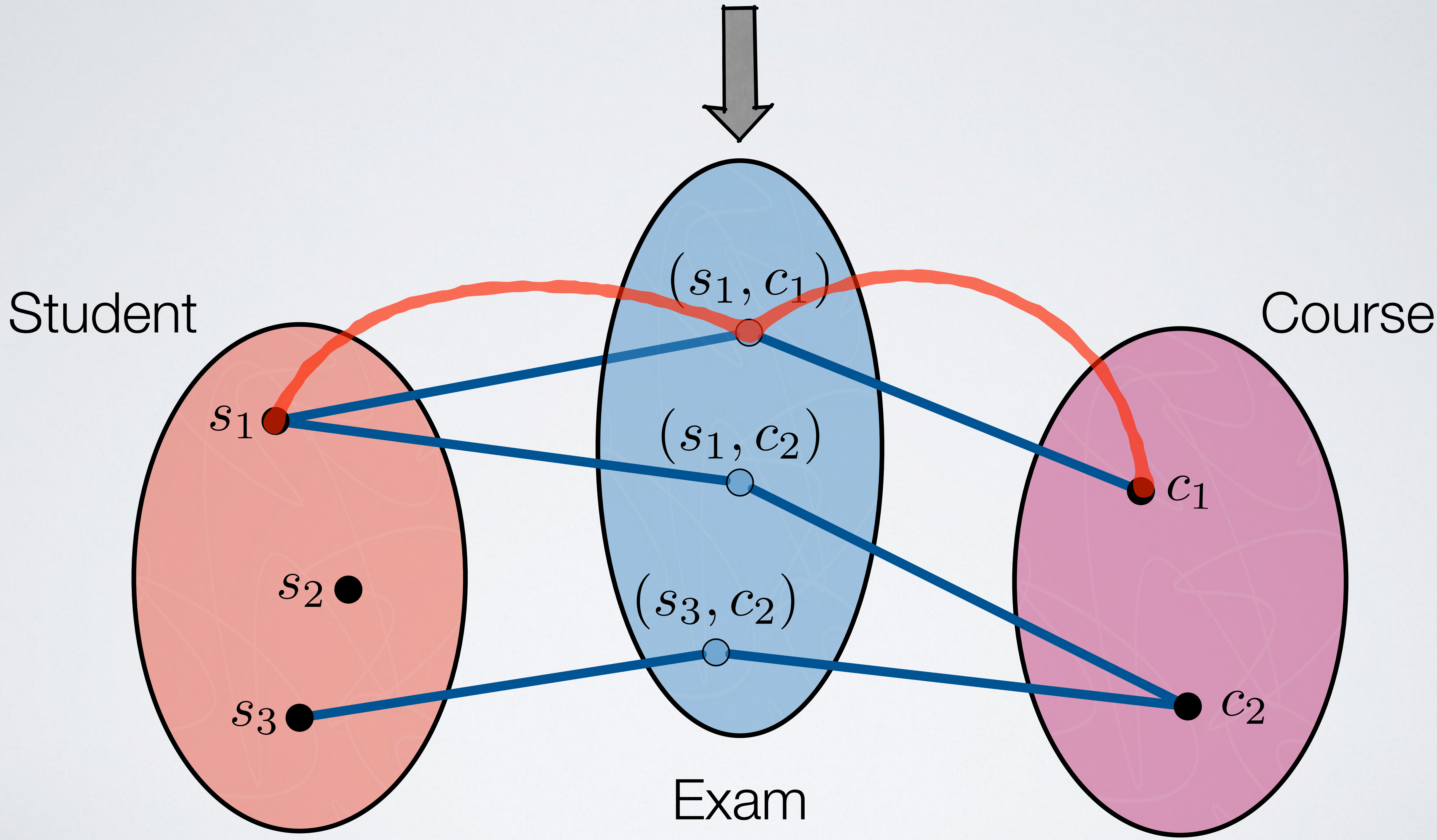
Binary Relationship: Example of Extensional Semantics

$$\text{instance}(\mathsf{i}, \mathsf{Exam}) = \{(s_1, c_1), (s_1, c_2), (s_3, c_2)\}$$



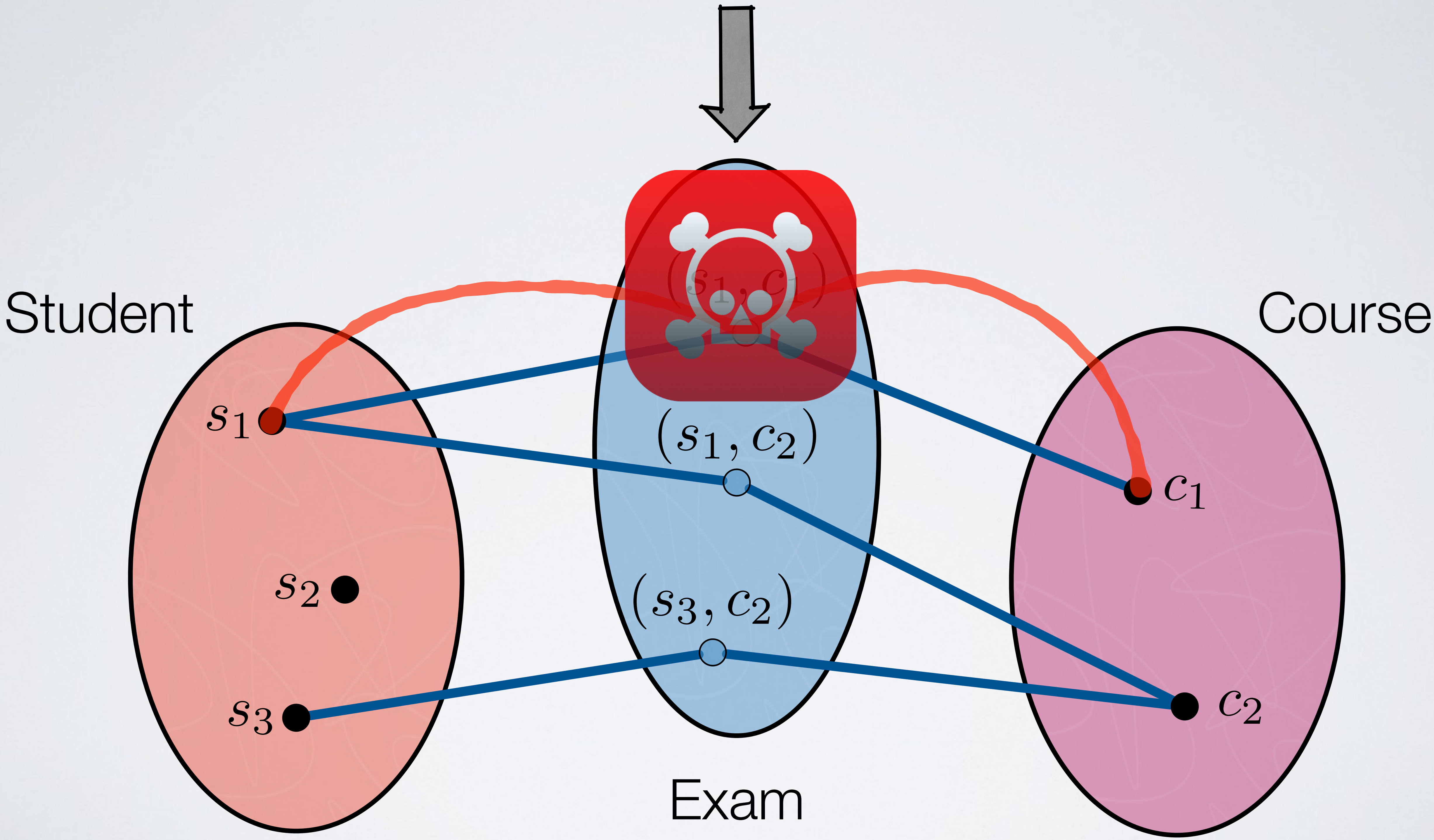
Binary Relationship: Example of Extensional Semantics

$$\text{instance}(\mathsf{i}, \mathsf{Exam}) = \{(s_1, c_1), (s_1, c_2), (s_3, c_2)\}$$

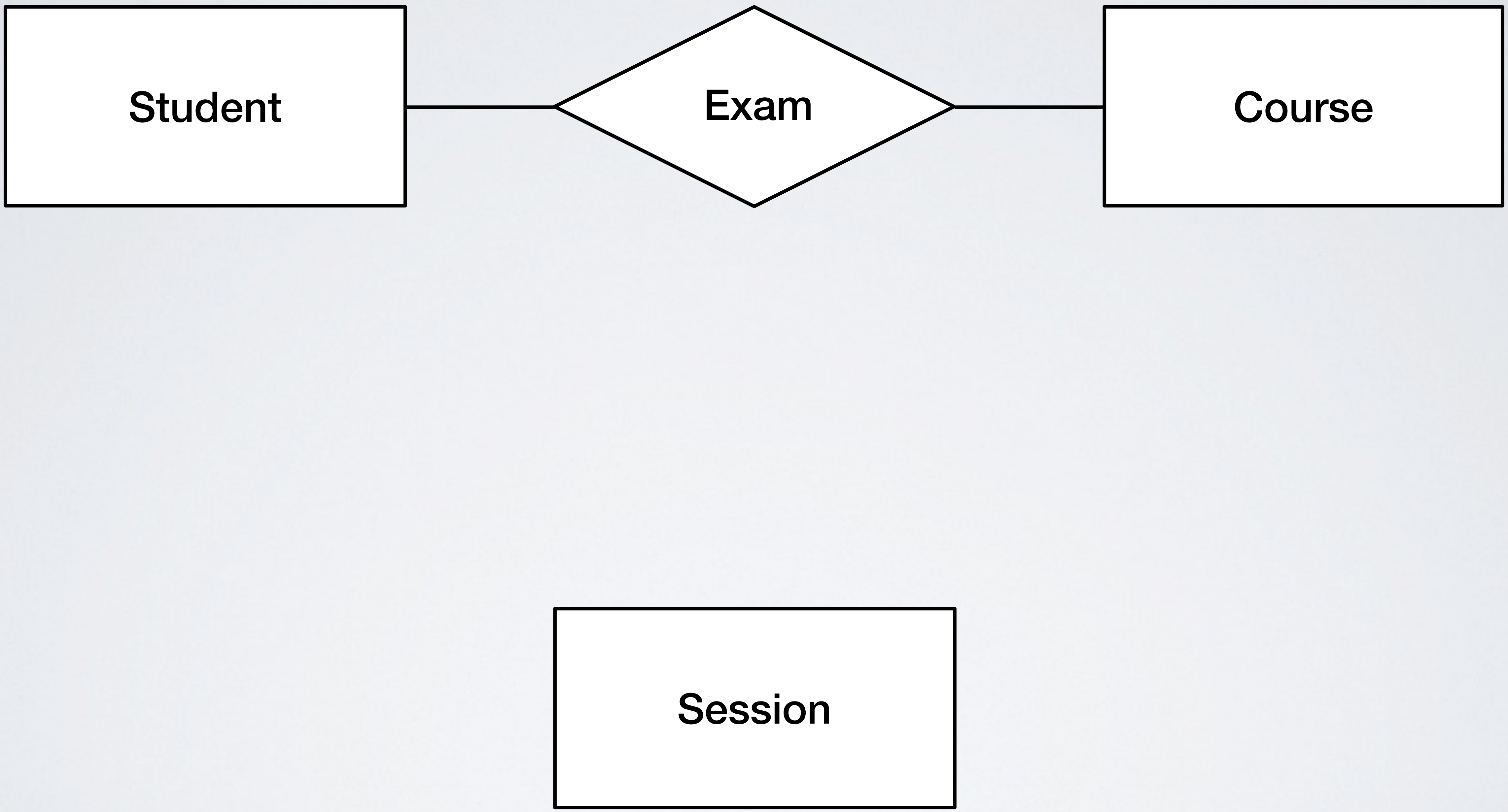


Binary Relationship: Example of Extensional Semantics

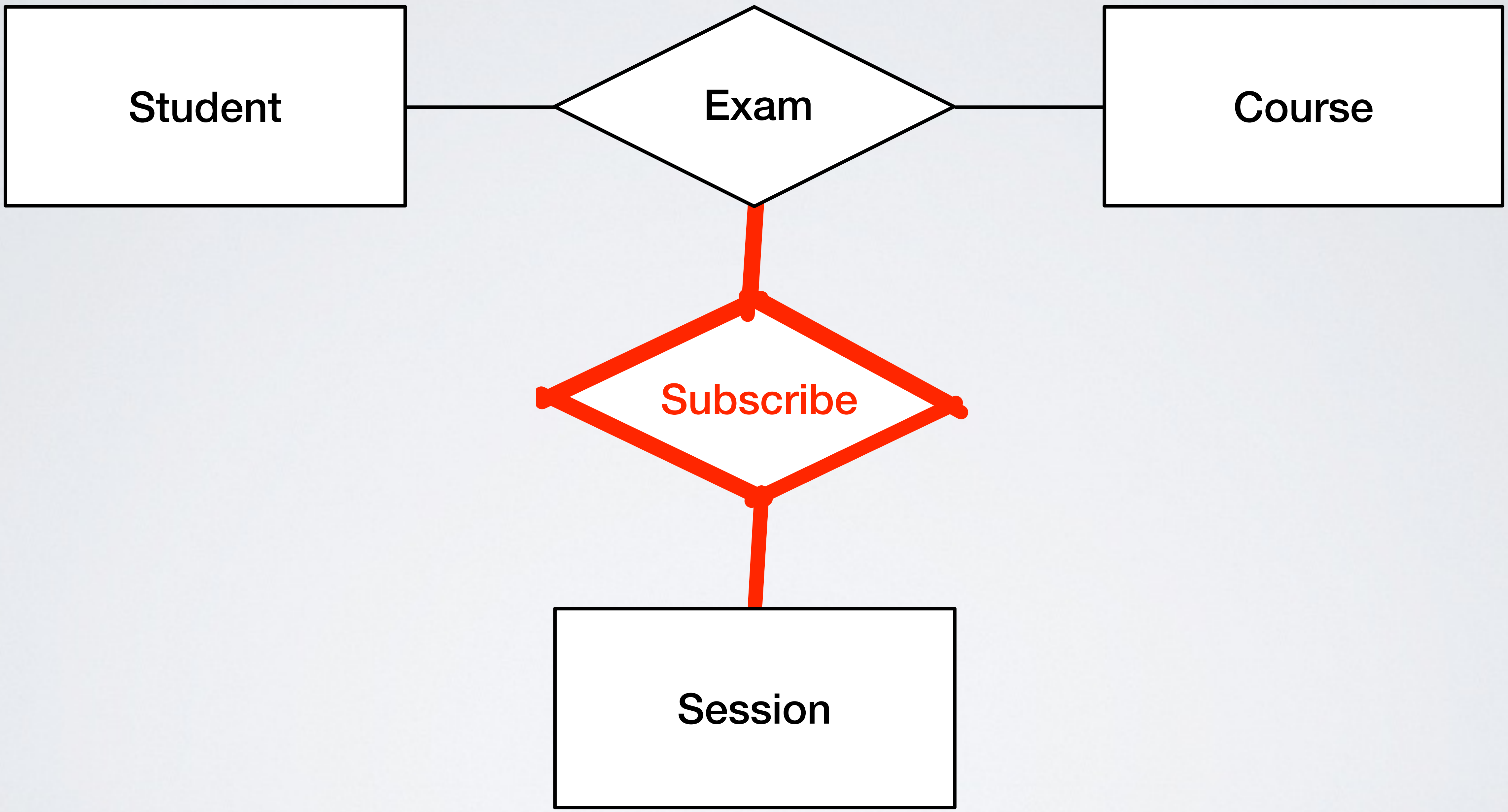
$$\text{instance}(\mathsf{i}, \mathsf{Exam}) = \{(s_1, c_1), (s_1, c_2), (s_3, c_2)\}$$



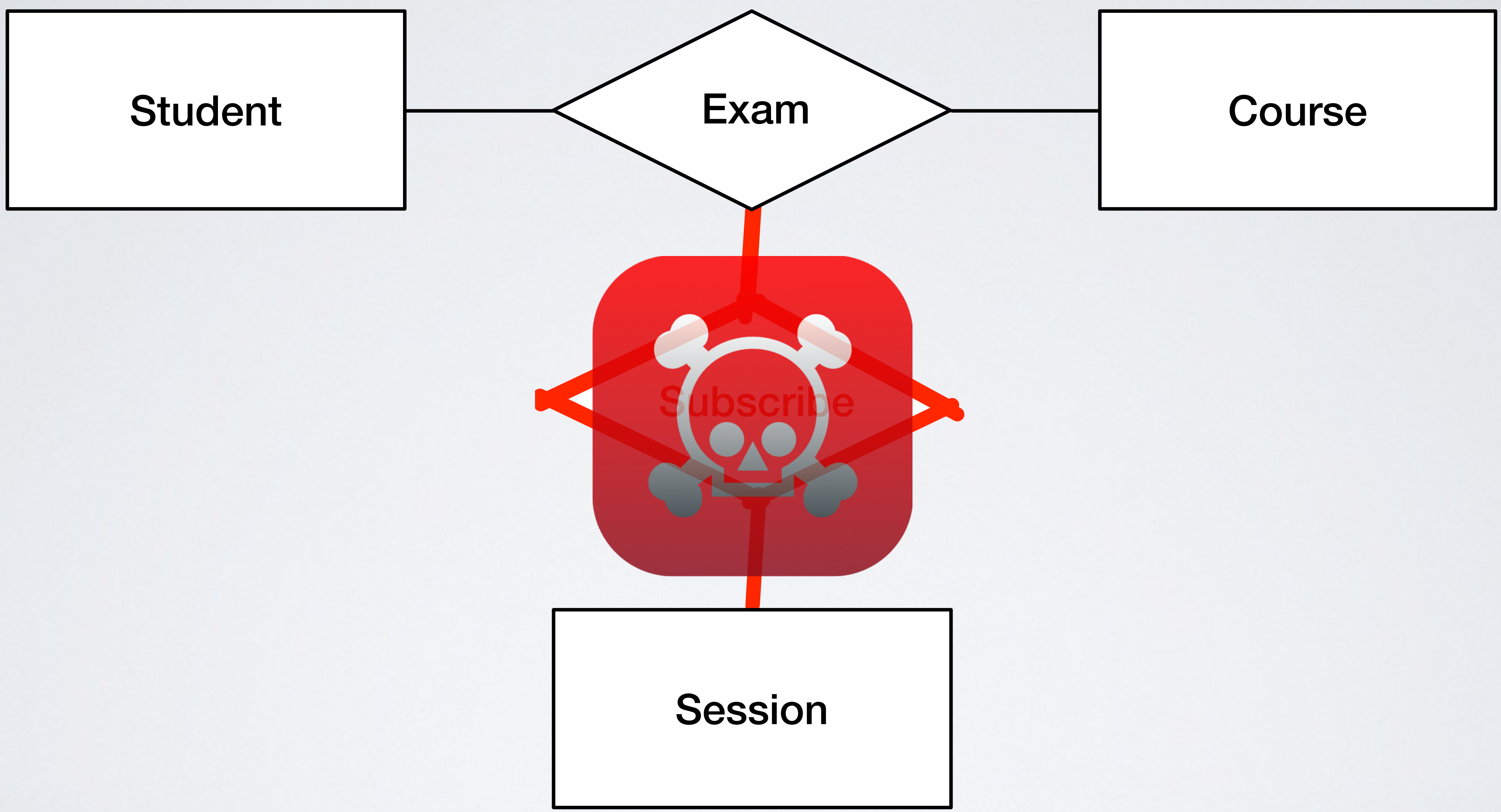
(Binary) Relationships: Beware!



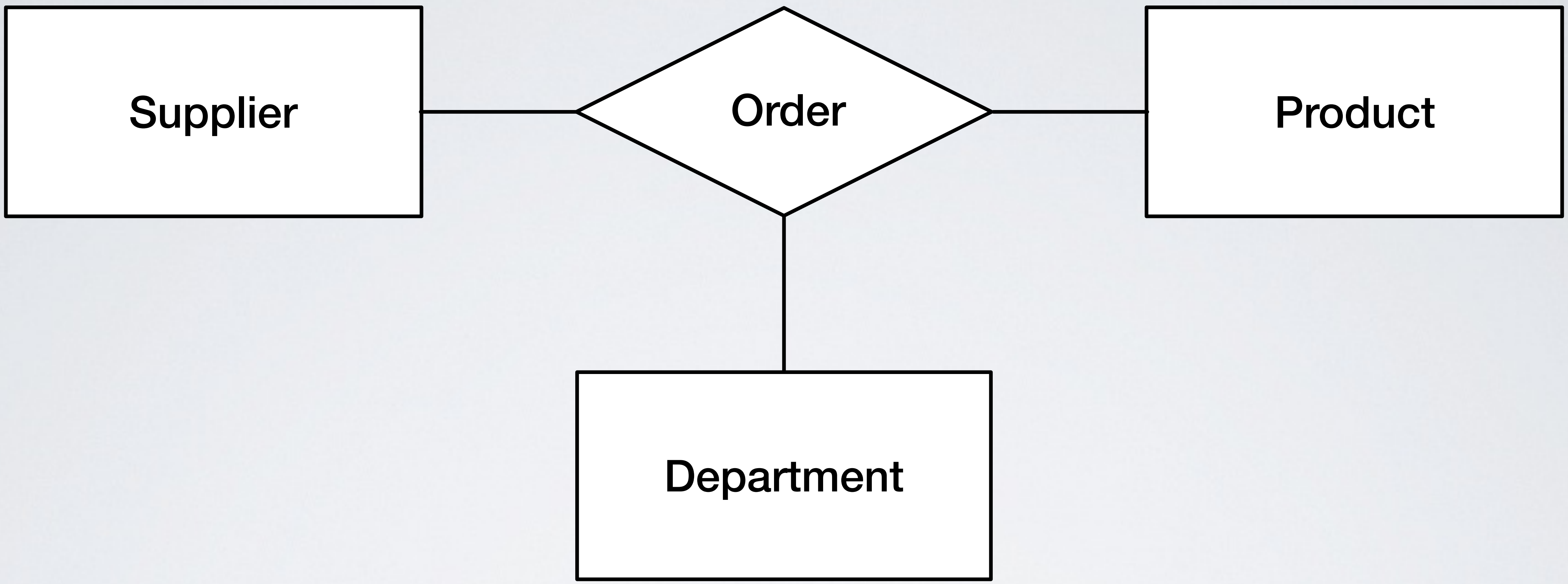
(Binary) Relationships: Beware!



(Binary) Relationships: Beware!



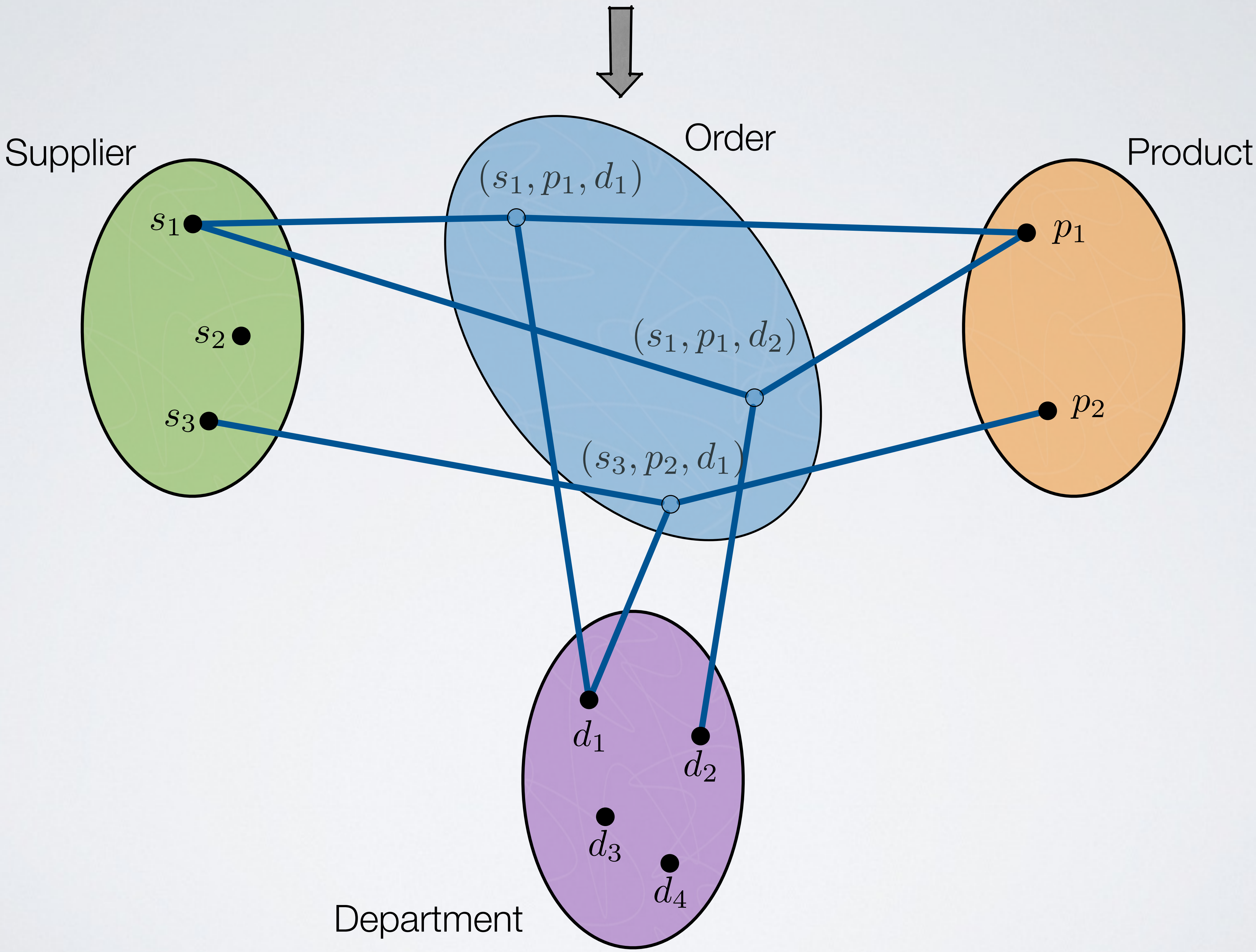
N-ary Relationship



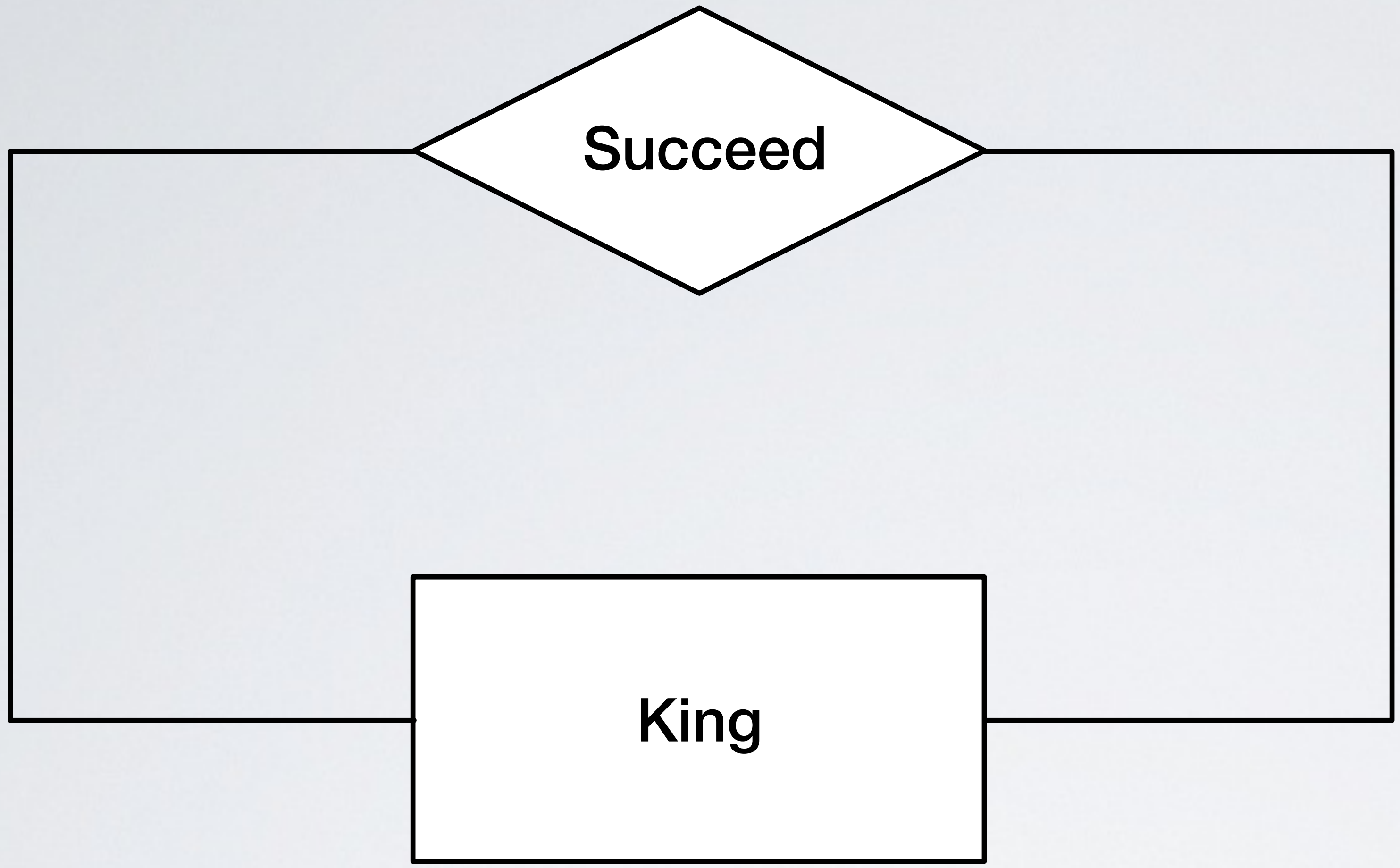
- A relationship with degree higher than 2 is called **n-ary**
- Order is a **ternary** relationship among Supplier, Product, and Department

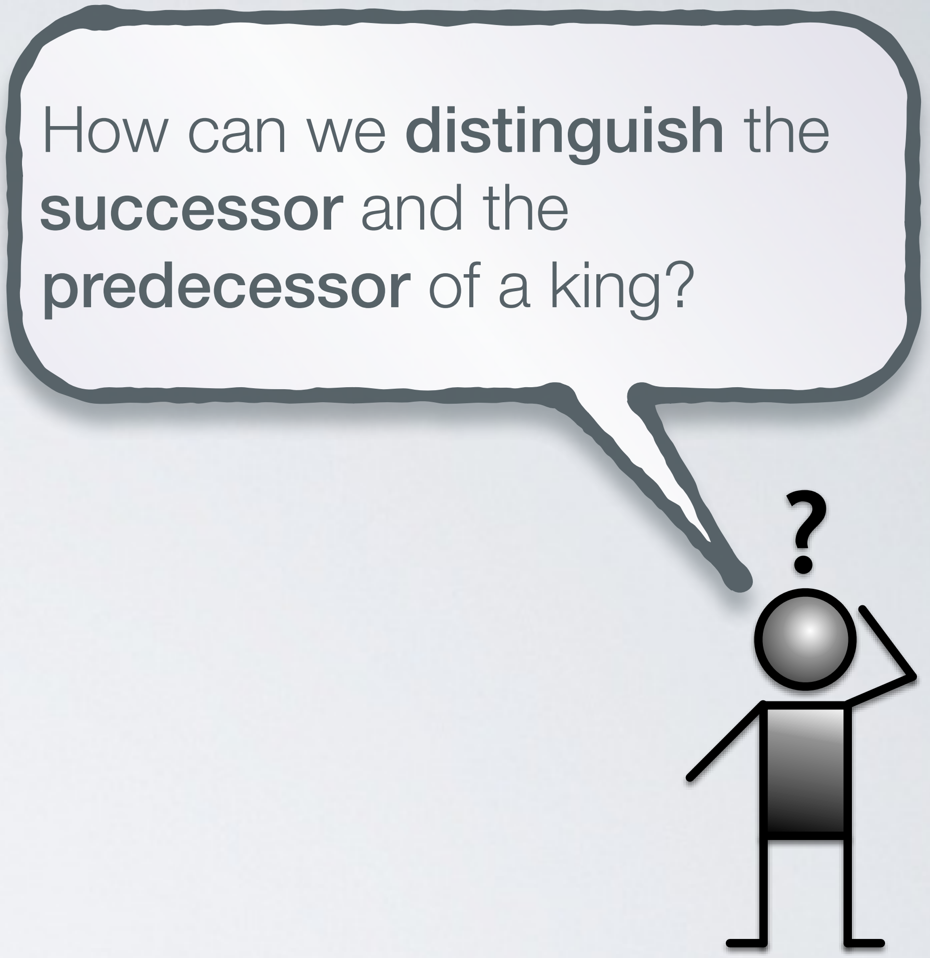
N-ary Relationship: Example of Extensional Semantics

$$\text{instance}(\mathsf{i}, \mathsf{Order}) = \{(s_1, p_1, d_1), (s_1, p_1, d_2), (s_3, p_2, d_1)\}$$

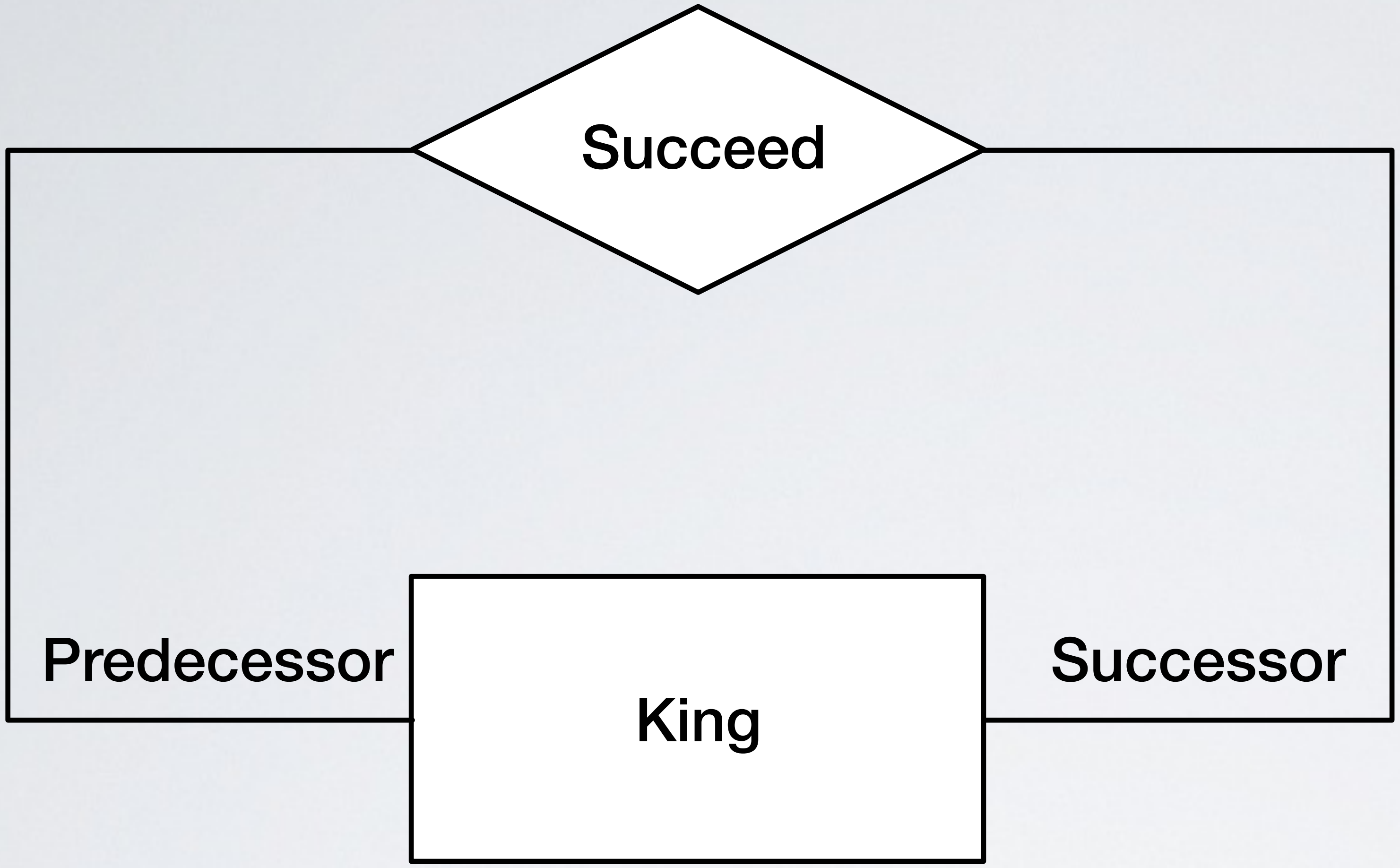


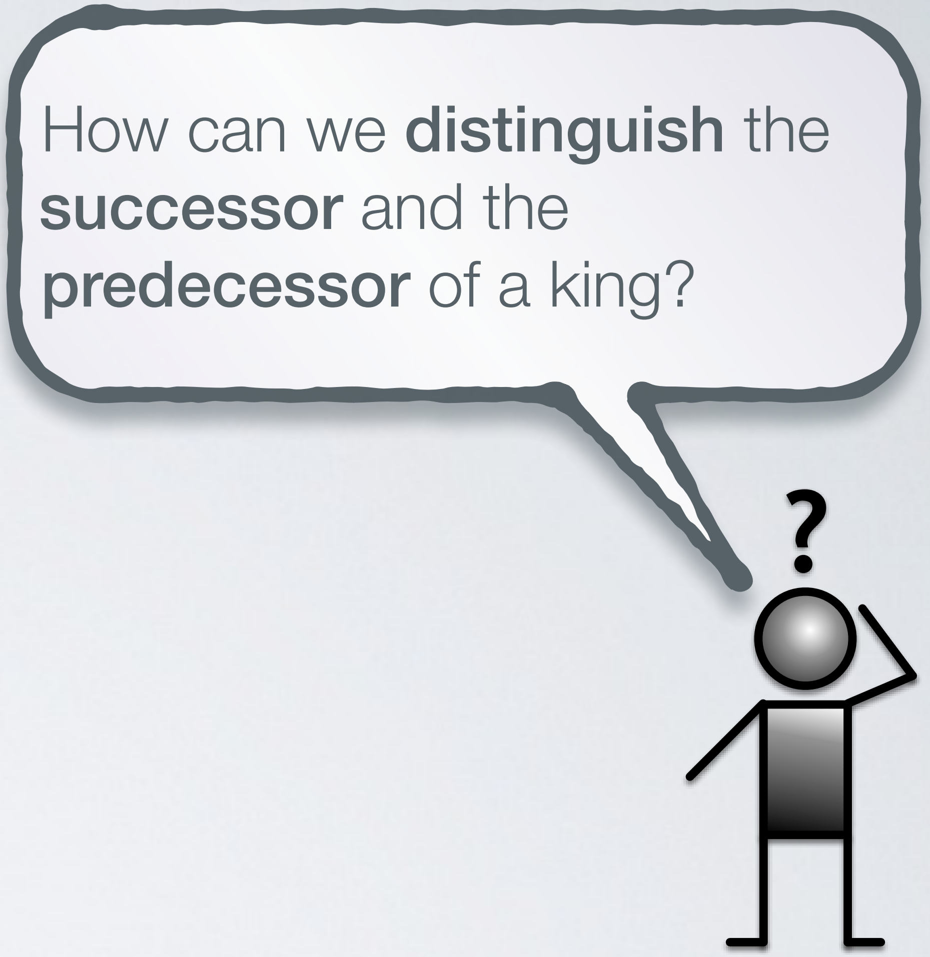
Recursive Relationship





Recursive Relationship





- We specify the **role** played by an entity when it participates more than once to a relationship

Relationship and Role: Extensional Semantics

- At **extensional** level an n-ary relationship **r** among entities $\mathbf{e_1}$, $\mathbf{e_2}$, ..., $\mathbf{e_n}$ (not necessarily distinct) with roles $\mathbf{u_1}$, $\mathbf{u_2}$, ..., $\mathbf{u_n}$ (all distinct), respectively, consists of a set of labeled n-uples

$$r_k = (\mathsf{u}_1 : e_{1,i}, \mathsf{u}_2 : e_{2,j}, \ldots, \mathsf{u}_n : e_{n,h})$$

such that $e_{1,i}$ is an instance of $\mathbf{e_1}$, $e_{2,j}$ is an instance of $\mathbf{e_2}$, and $e_{n,h}$ is an instance of $\mathbf{e_n}$. Each n-uple is an instance of the relationship **r**

- The role of an entity must be explicitly specified in the ER diagram when it participates more than once in a relationship. When it is not explicitly specified, i.e. when the entity participates only once to the relationship, we implicitly assume that the role coincides with the name of the entity

Attribute

Entity Attribute

An **attribute of an entity** is a **local property** of that entity, relevant for the application. An attribute **maps** each **instance of an entity** to a **value** belonging to a set called **domain**

- Each attribute has a **name** which **univocally identify** it in the entity
- In the ER diagram, an attribute is represented by a **circle with the name of the attribute**, connected to the entity the attribute belongs to
  - the domain of the attribute is typically omitted in the ER diagram but it is reported in the data dictionary (see later on)
- We use the **aggregation abstraction** to create attributes

Entity Attribute: Extensional Semantics

- At **extensional** level an **attribute** is a (total) **function** which maps **each instance** of the entity **e** to a **value** belonging to a domain **D**.

- Therefore, if in a schema **S** an entity **e** and an attribute **a** of **e** on a domain **D** are defined, in each instance **i** of the schema **S**, for each instance e_i of the entity **e**, there exists a unique value v_j in the domain **D** associated to e_i

$$
\begin{aligned}
\text{instance}: \quad & \mathsf{I} \times \mathsf{S} \to \mathsf{I} \times \mathsf{D} \\
& (\mathsf{i}, \mathsf{a}) \mapsto \text{instance}(\mathsf{i}, \mathsf{a}) \quad \subseteq \text{instance}(\mathsf{i}, \mathsf{e}) \times \mathsf{D} \\
& \qquad\qquad\qquad\qquad\quad = \{(e_1, v_1), (e_2, v_2) \ldots, (e_n, v_n)\}
\end{aligned}
$$

$$
\forall e_i \in \text{instance}(\mathsf{i}, \mathsf{e}), \exists! v_j \in \mathsf{D} | (e_i, v_j) \in \text{instance}(\mathsf{i}, \mathsf{a})
$$

Entity Attribute: Extensional Semantics

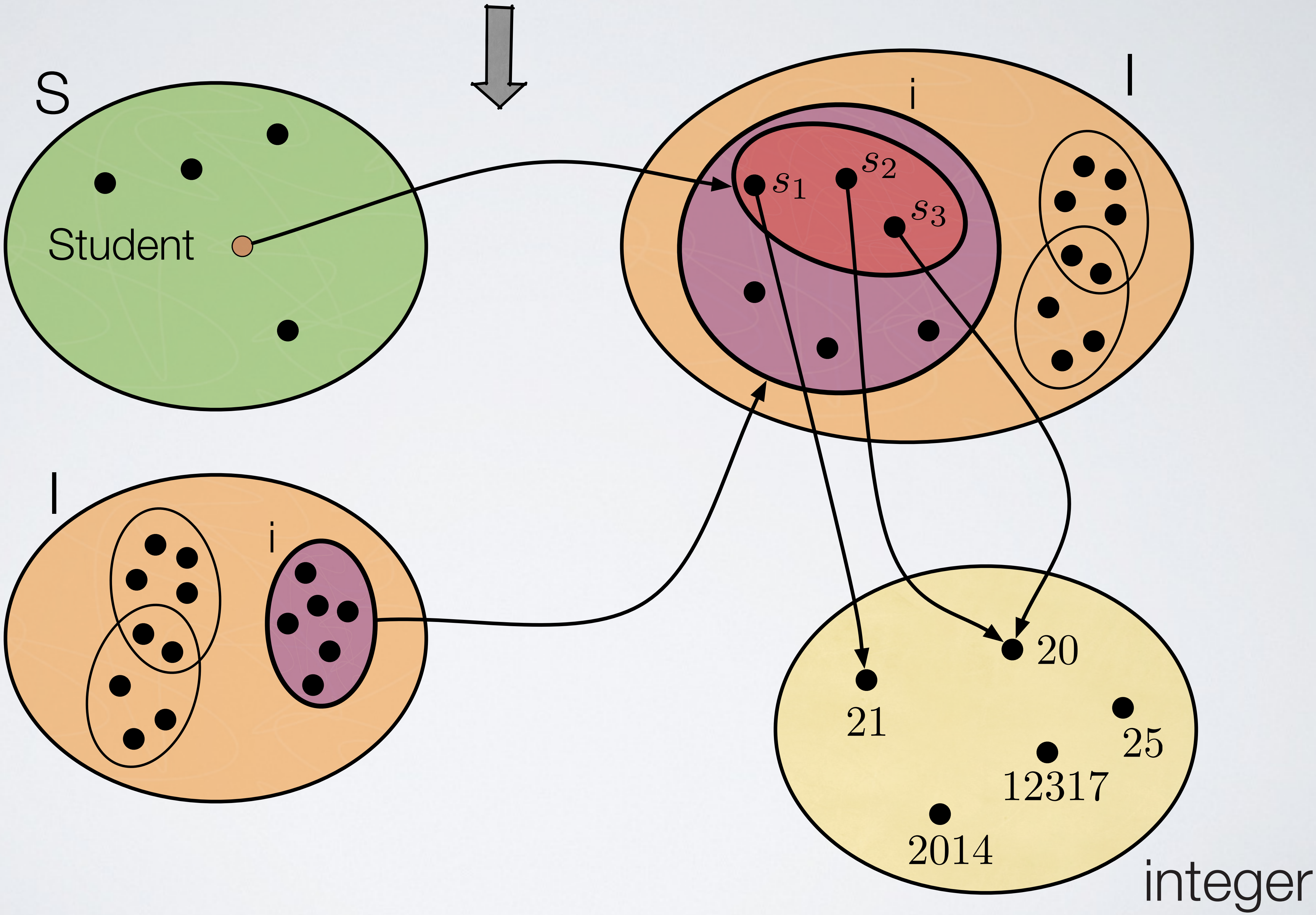
- At **extensional** level an **attribute** is a (total) **function** which maps **each instance** of the entity **e** to a **value** belonging to a domain **D**.

- Therefore, if in a schema **S** an entity **e** and an attribute **a** of **e** on a domain **D** are defined, in each instance **i** of the schema **S**, for each instance e_i of the entity **e**, there exists a unique value v_j in the domain **D** associated to e_i

$$\mathsf{a}: \quad \mathrm{instance}(\mathsf{i}, \mathsf{e}) \rightarrow \mathsf{D}$$
$$e_i \mapsto v_j$$

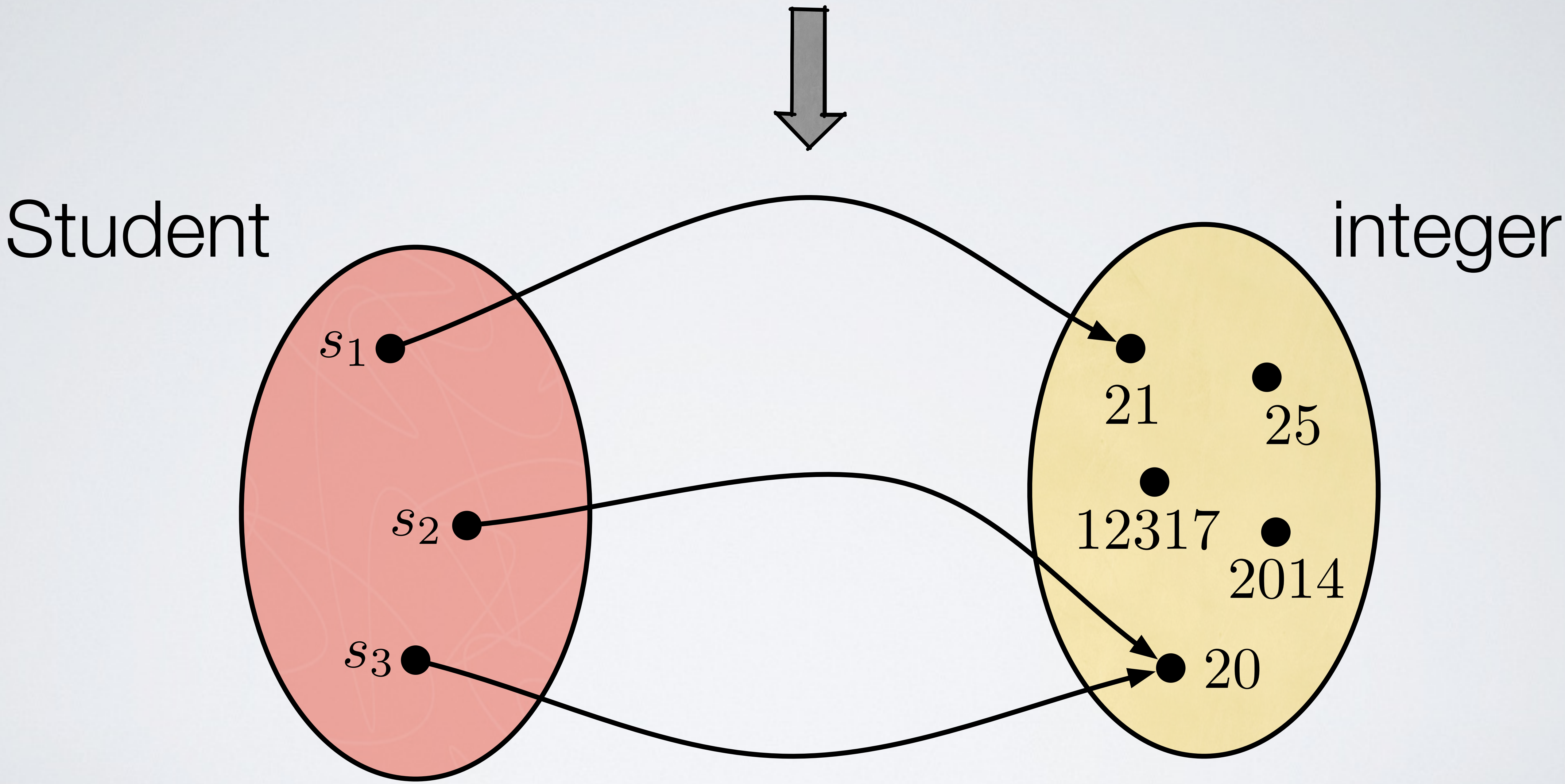
Attribute: Example of Extensional Semantics

$$\text{instance}(\mathsf{i}, \mathsf{Student.Age}) = \{(s_1, 21), (s_2, 20), (s_3, 20)\}$$



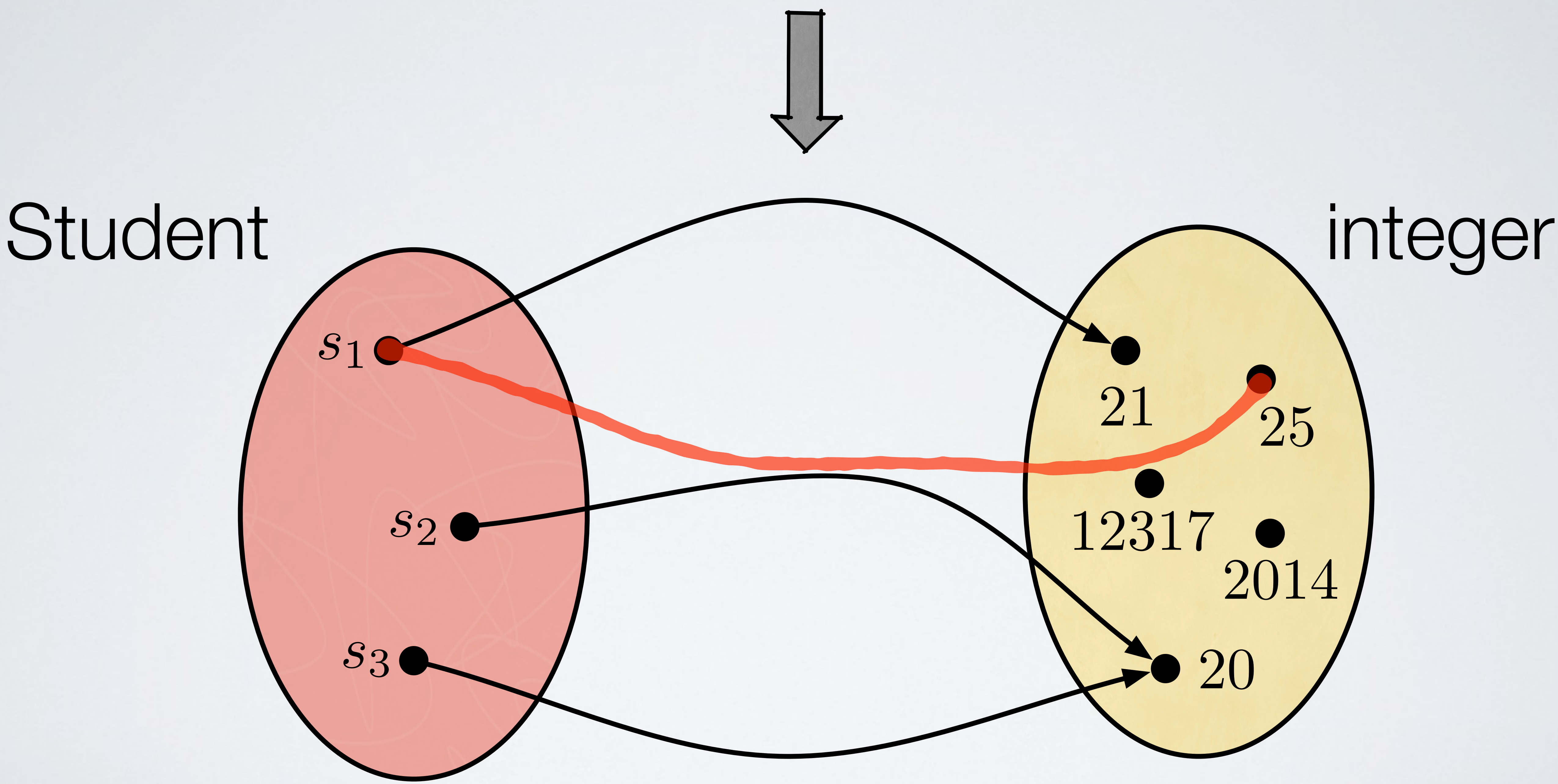
Attribute: Example of Extensional Semantics

$$\text{instance}(\mathsf{i}, \mathsf{Student.Age}) = \{(s_1, 21), (s_2, 20), (s_3, 20)\}$$



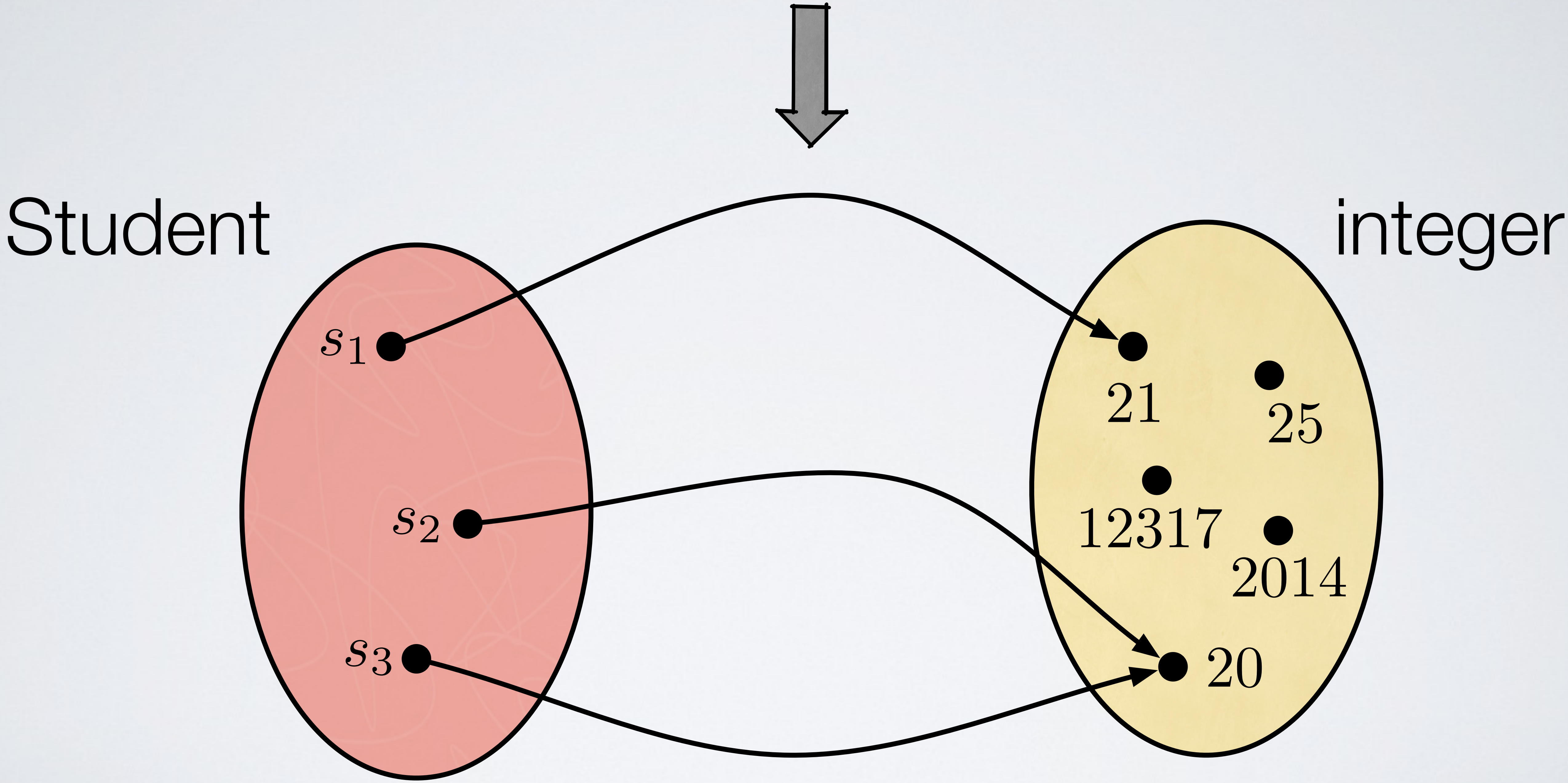
Attribute: Example of Extensional Semantics

$$\text{instance}(\mathsf{i}, \mathsf{Student.Age}) = \{(s_1, 21), (s_2, 20), (s_3, 20)\}$$

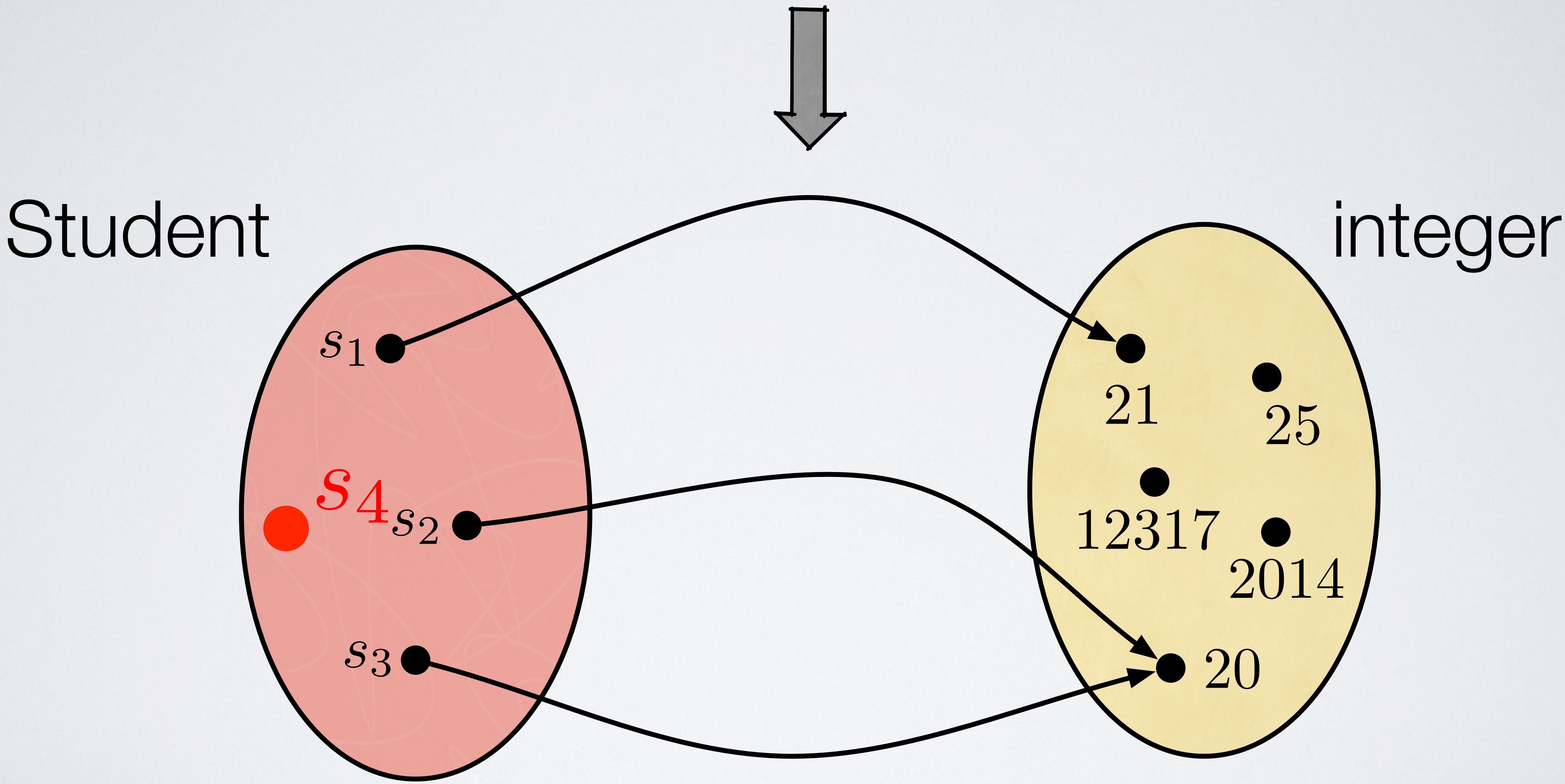


Attribute: Example of Extensional Semantics

$$\text{instance}(\mathsf{i}, \mathsf{Student.Age}) = \{(s_1, 21), (s_2, 20), (s_3, 20)\}$$

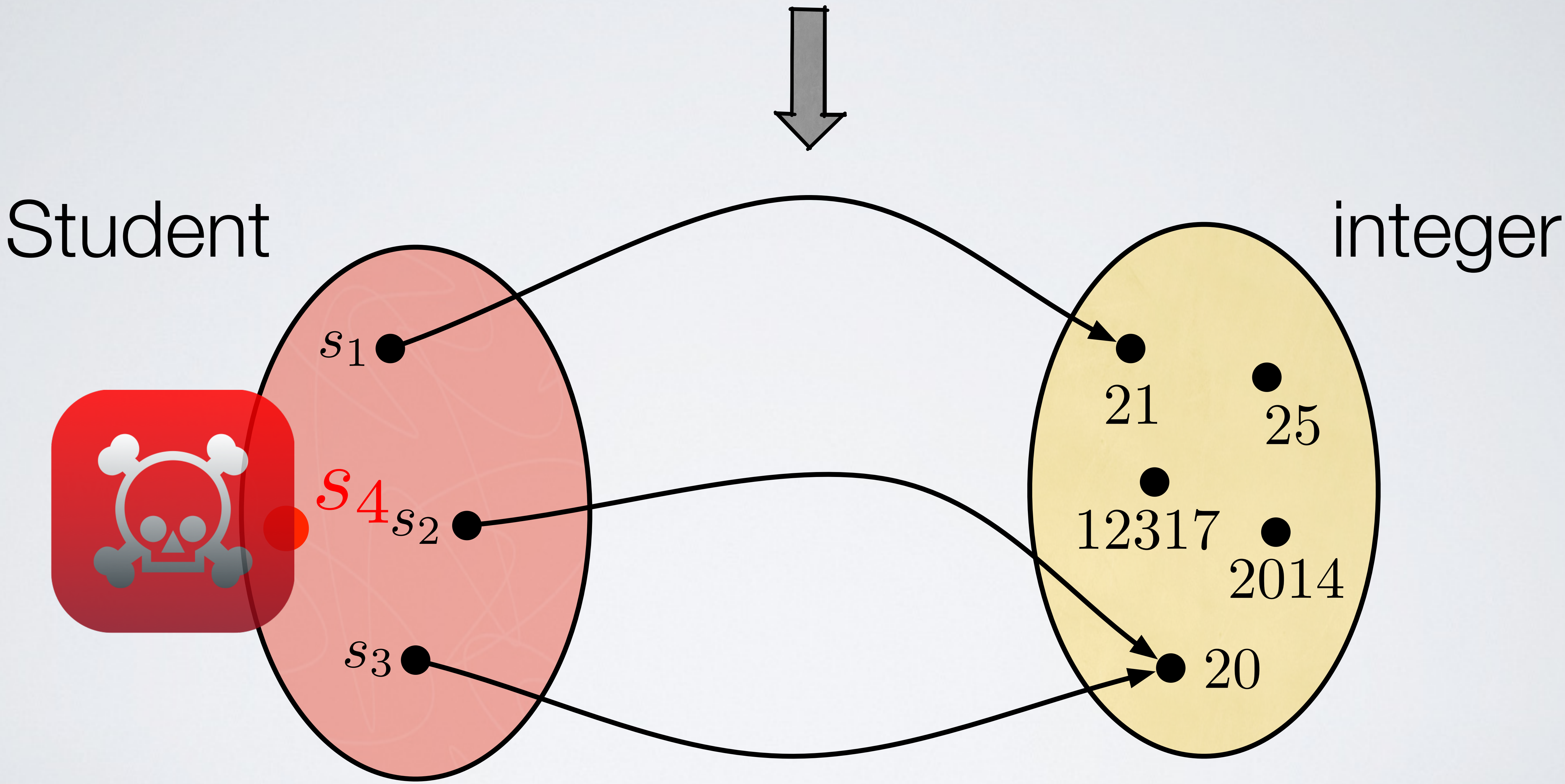


$$\text{instance}(\mathsf{i}, \mathsf{Student.Age}) = \{(s_1, 21), (s_2, 20), (s_3, 20)\}$$



Attribute: Example of Extensional Semantics

$$\text{instance}(\mathsf{i}, \mathsf{Student.Age}) = \{(s_1, 21), (s_2, 20), (s_3, 20)\}$$



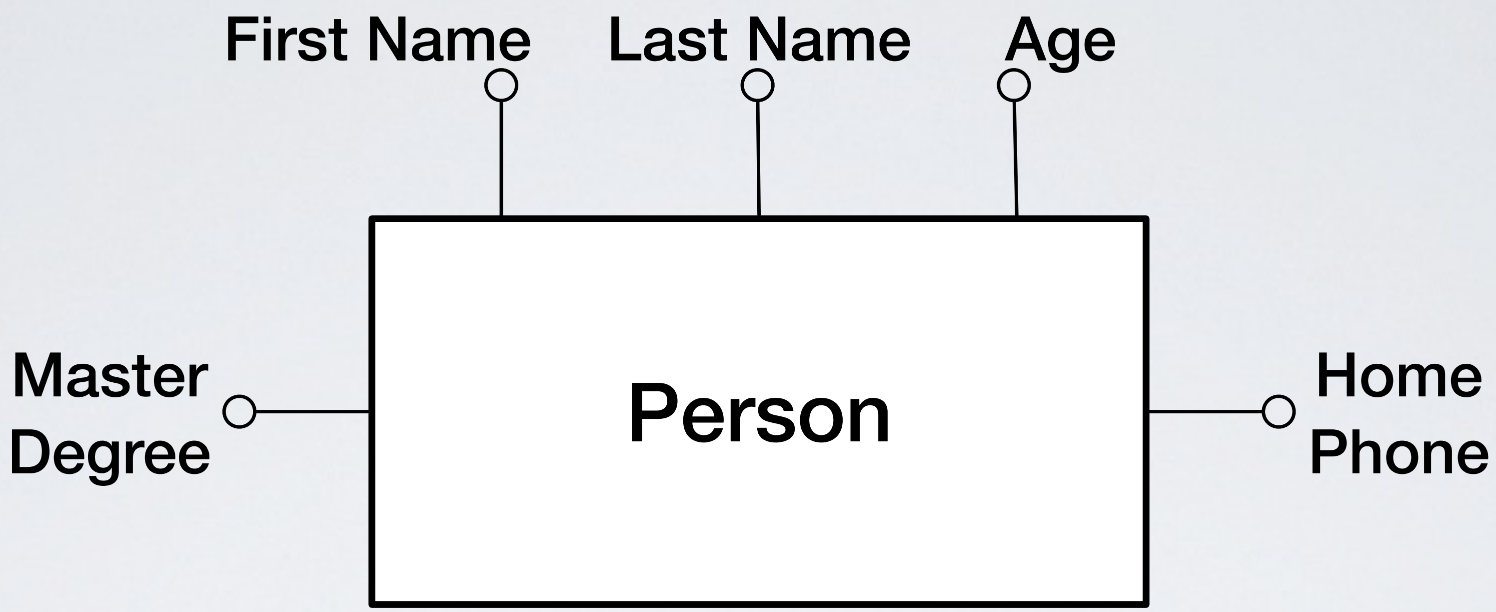
Incomplete Information: the NULL value

- It may happen that, for a given attribute, you cannot associate a value with an instance of an entity
  - but the definition of attribute requires to have a value associated with each instance of an entity

- To address this issue we define a special value, called **NULL**

- **NULL** is not one of the "ordinary" values of a domain **D**
  - "unused values" may not exist
  - "unused values" may change over time
  - you need to treat such "unused values" as special cases in the applications
  - "unused values" would be different from domain to domain

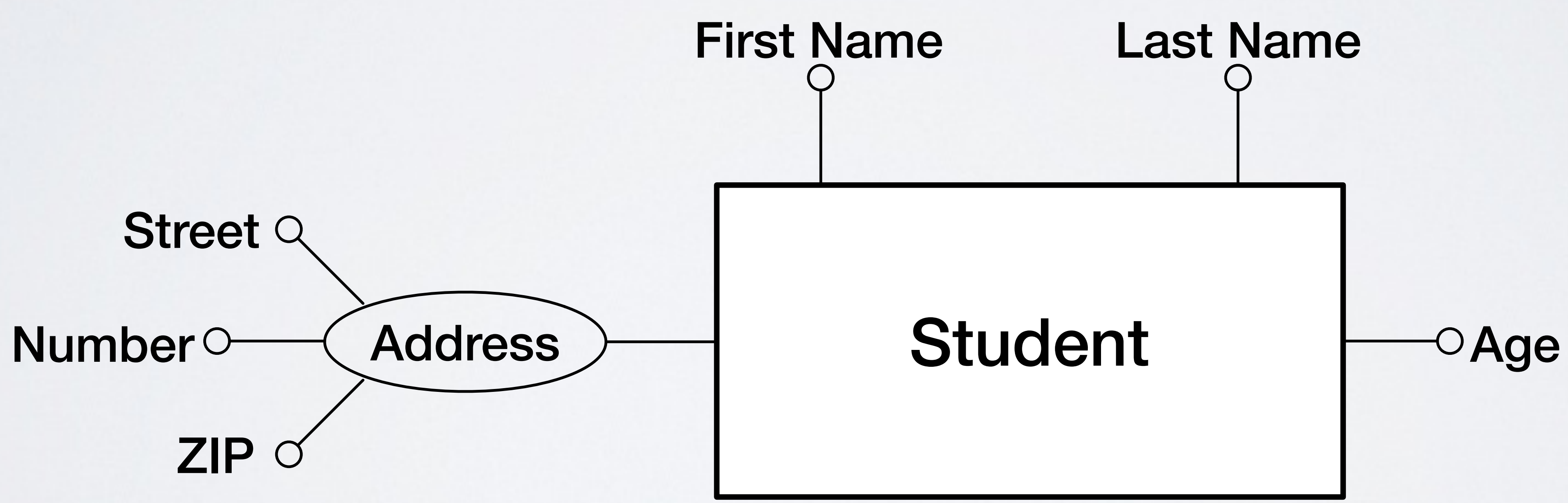
Meaning of the NULL value



- **Value undefined**: an appropriate value for a given instance does not exist
  - Master Degree does not apply to those people who have not attended university
- **Value not available**: an appropriate value for a given instance exists but it is not known in a given moment
  - You may not know the Age of a Person
- **Value unknown**: an appropriate value for a given instance may or may not exist
  - A Person may or may not have an home phone number

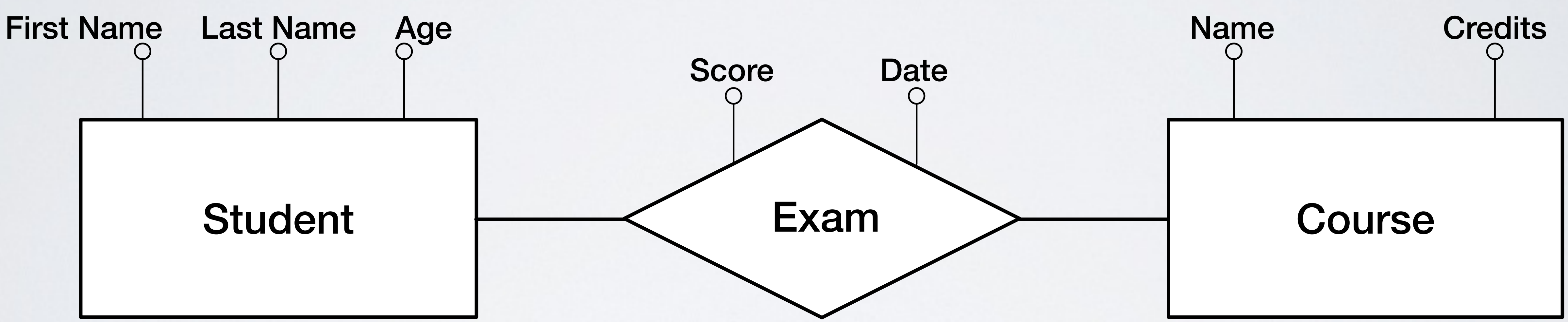
Composite Attribute

- Attributes can be defined also over **composite domains**



Relationship Attribute

An **attribute of a relationship** is a **local property** of that relation, relevant for the application. An attribute **maps** each **instance of a relationship** to a **value** belonging to a set called **domain**



- Each attribute has a **name** which **univocally identify** it in the relationship
- In the ER diagram, an attribute is represented by a **circle with the name of the attribute**, connected to the relationship the attribute belongs to
- An attribute of a relationship **r** among entities $\mathbf{e_1}$, $\mathbf{e_2}$, ... , $\mathbf{e_n}$ models a property **not** of $\mathbf{e_1}$, **not** of $\mathbf{e_2}$, …, **not** of $\mathbf{e_n}$ but of the **association** among $\mathbf{e_1}$, $\mathbf{e_2}$, ... , $\mathbf{e_n}$ as represented by **r**

Relationship Attribute: Extensional Semantics

- At **extensional** level an **attribute** is a (total) **function** which maps **each instance** of the relationship **r** to a **value** belonging to a domain **D**.

- Therefore, if in a schema **S** a relationship **r** and an attribute **a** of **r** on a domain **D** are defined, in each instance **i** of the schema **S**, for each instance $r_k = (e_{1,i}, e_{2,j}, \ldots, e_{n,h})$ of the relationship **r**, there exists a unique value v_j in the domain **D** associated with r_k

$$\begin{aligned} \text{instance}: \quad & \mathsf{I} \times \mathsf{S} \to \mathsf{I} \times \mathsf{D} \\ & (\mathsf{i}, \mathsf{a}) \mapsto \text{instance}(\mathsf{i}, \mathsf{a}) \quad \subseteq \text{instance}(\mathsf{i}, \mathsf{r}) \times \mathsf{D} \\ & \qquad = \{(r_1, v_1), (r_2, v_2) \ldots, (r_n, v_n)\} \end{aligned}$$

$$\forall r_i \in \text{instance}(\mathsf{i}, \mathsf{r}), \exists! v_j \in \mathsf{D} | (r_i, v_j) \in \text{instance}(\mathsf{i}, \mathsf{a})$$

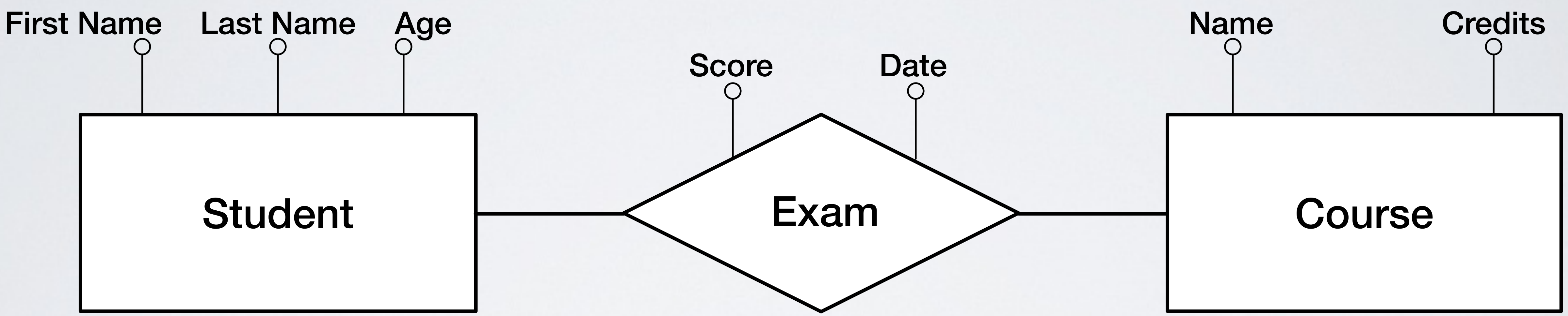
Relationship Attribute: Extensional Semantics

- At **extensional** level an **attribute** is a (total) **function** which maps **each instance** of the relationship **r** to a **value** belonging to a domain **D**.

- Therefore, if in a schema **S** a relationship **r** and an attribute **a** of **r** on a domain **D** are defined, in each instance **i** of the schema **S**, for each instance $r_k = (e_{1,i}, e_{2,j}, \ldots, e_{n,h})$ of the relationship **r**, there exists a unique value v_j in the domain **D** associated with r_k

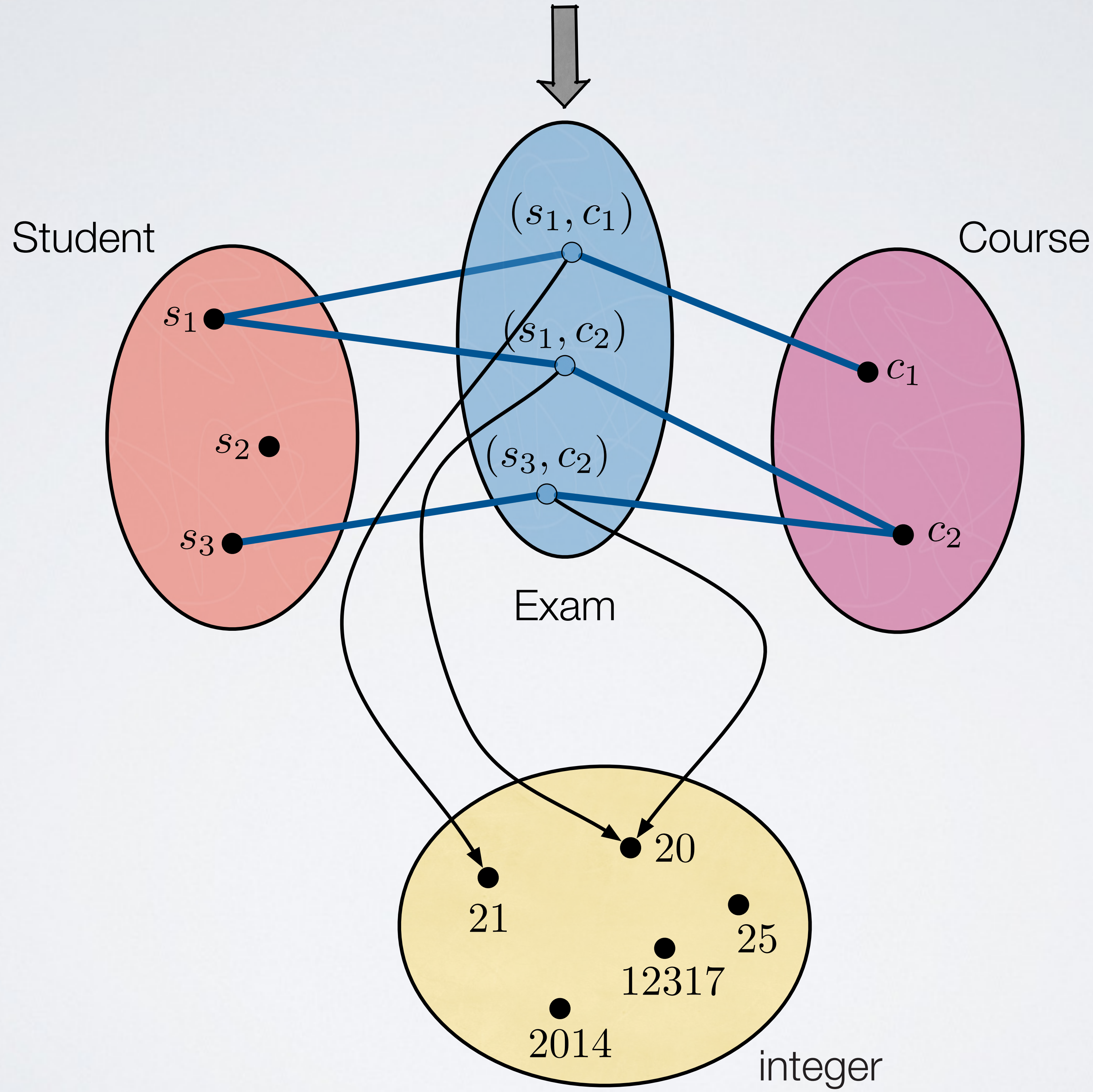
$$
\begin{aligned}
\mathsf{a}: \quad & \mathrm{instance}(\mathsf{i}, \mathsf{r}) \rightarrow \mathsf{D} \\
& \qquad\qquad r_i \mapsto v_j
\end{aligned}
$$

Binary Relationship Attribute: Example of Extensional Semantics



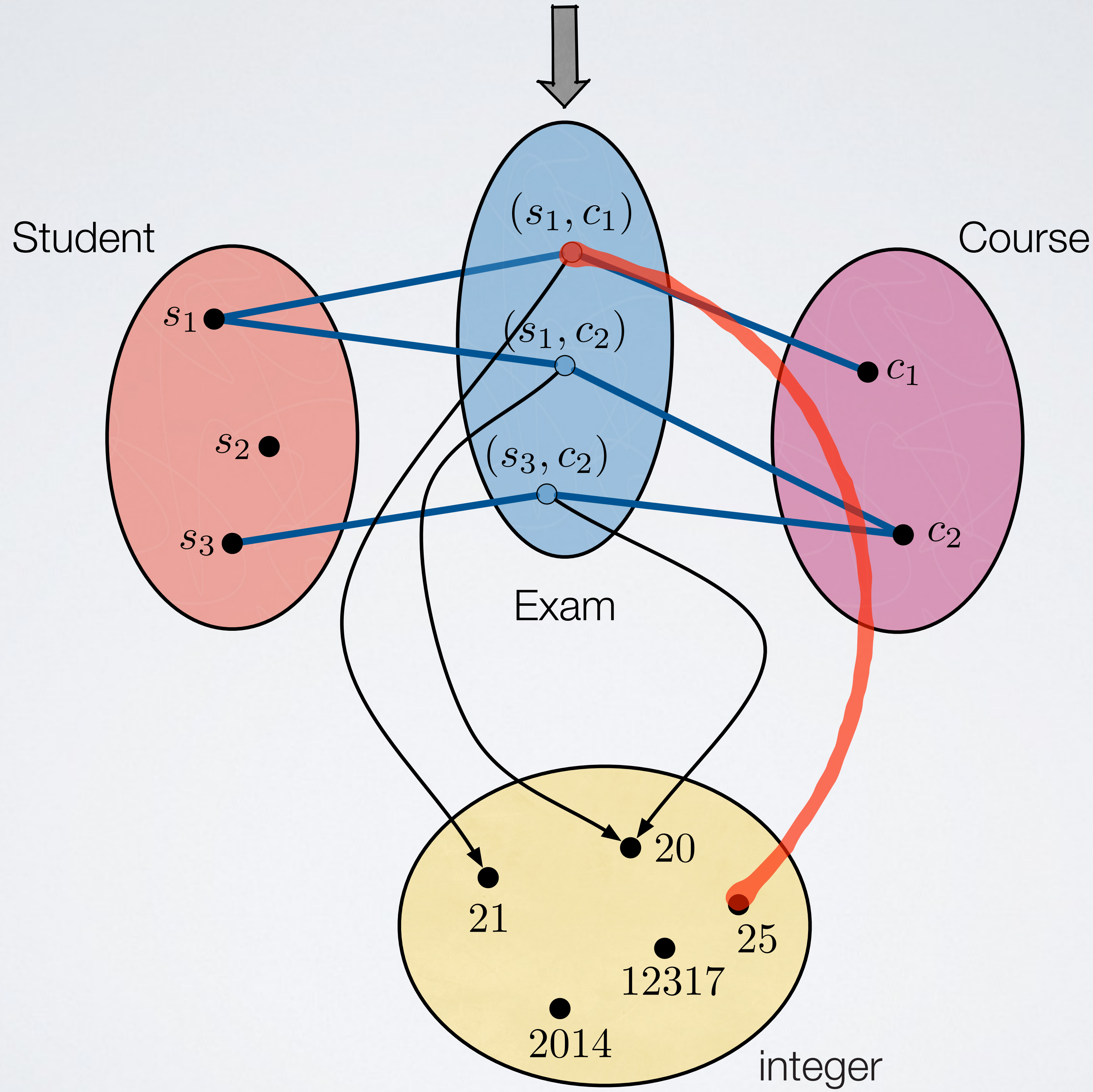
Binary Relationship Attribute: Example of Extensional Semantics

$$\text{instance}(\mathsf{i}, \mathsf{Exam.Score}) = \Big\{\big((s_1, c_1), 21\big), \big((s_1, c_2), 20\big), \big((s_3, c_2), 20\big)\Big\}$$



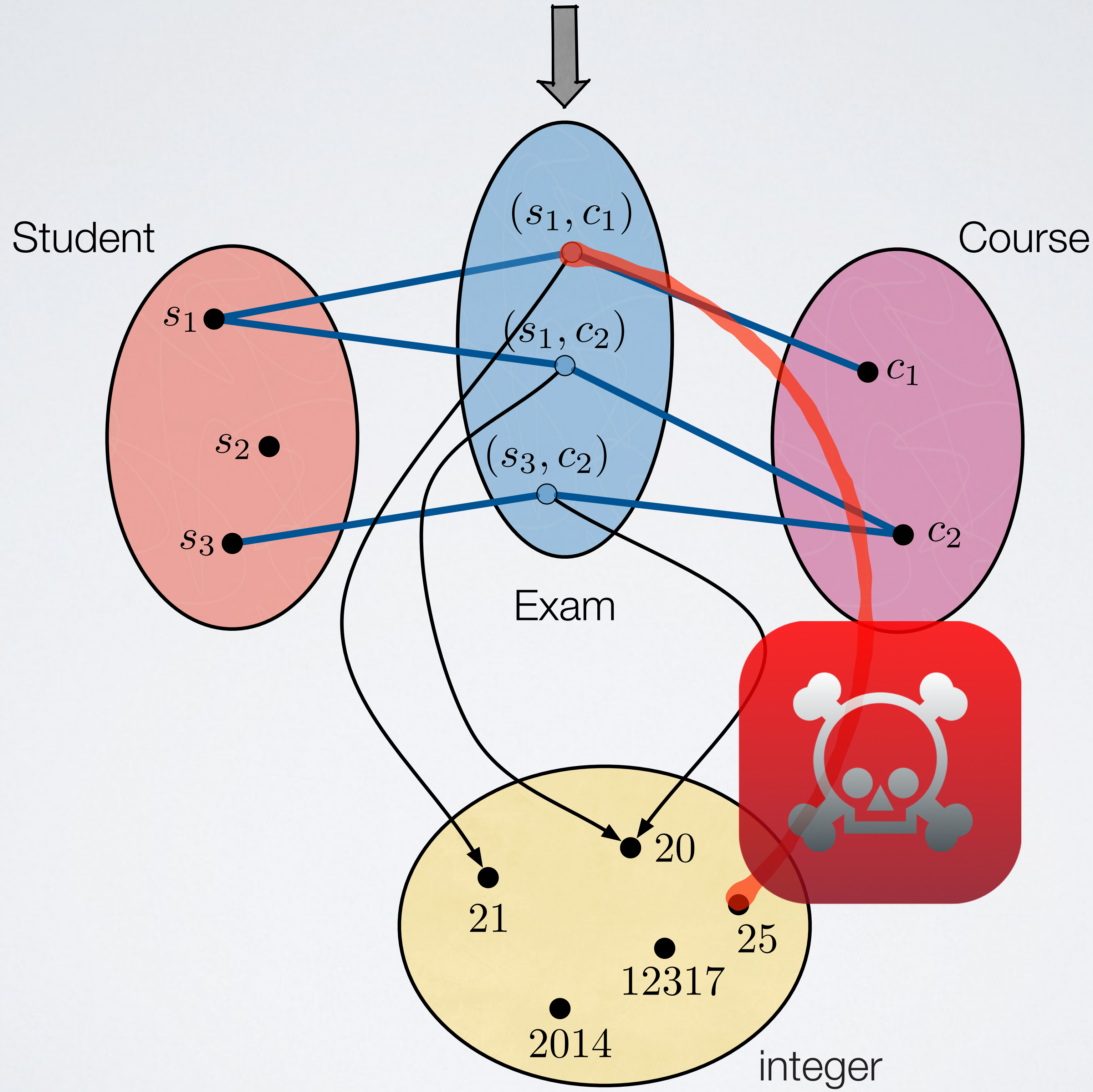
Binary Relationship Attribute: Example of Extensional Semantics

$$\text{instance}(\mathsf{i}, \mathsf{Exam.Score}) = \Big\{\big((s_1, c_1), 21\big), \big((s_1, c_2), 20\big), \big((s_3, c_2), 20\big)\Big\}$$



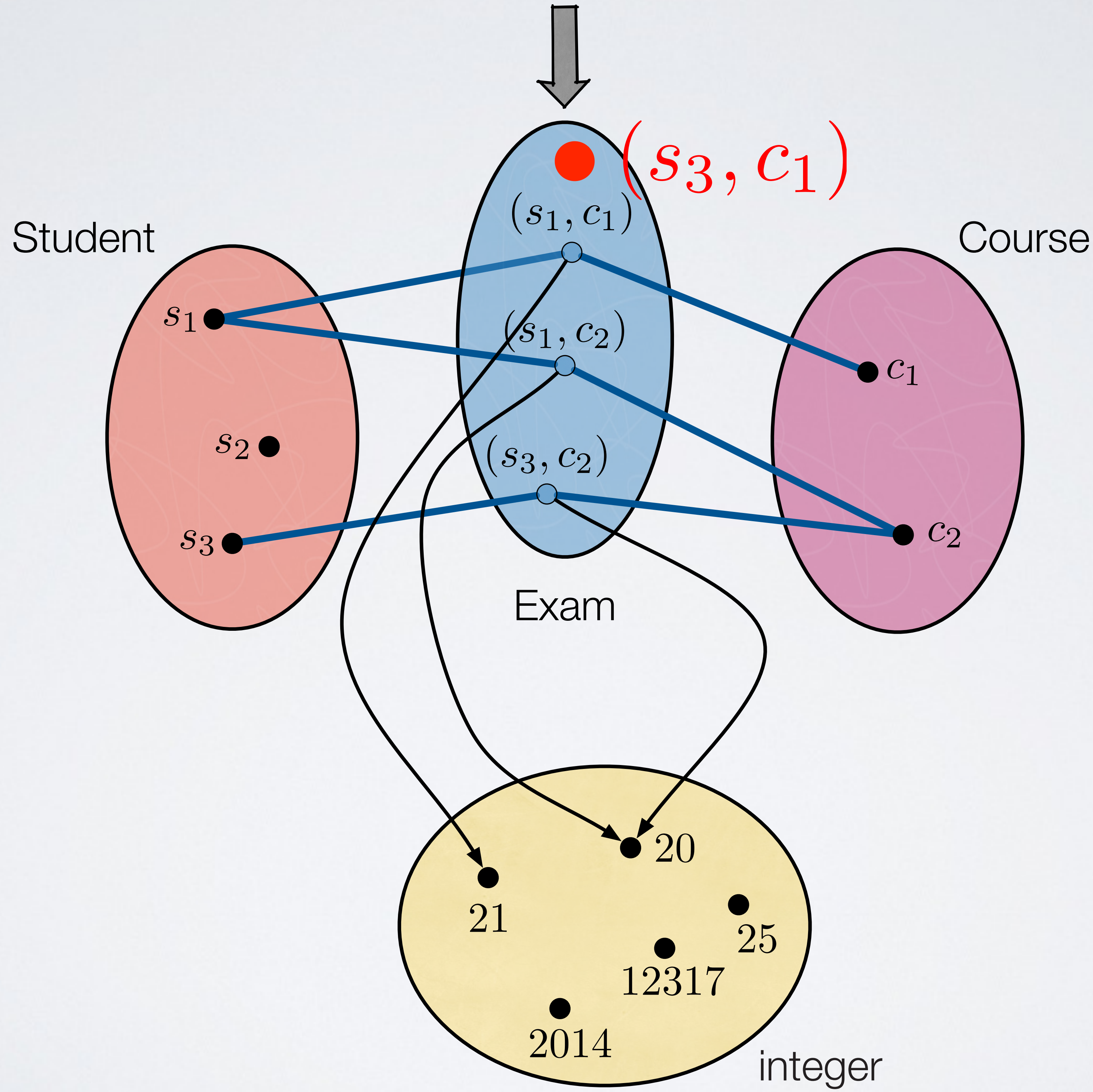
Binary Relationship Attribute: Example of Extensional Semantics

$$\text{instance}(\mathrm{i}, \mathsf{Exam.Score}) = \Big\{ \big(((s_1, c_1), 21\big), \big((s_1, c_2), 20\big), \big((s_3, c_2), 20\big) \Big\}$$



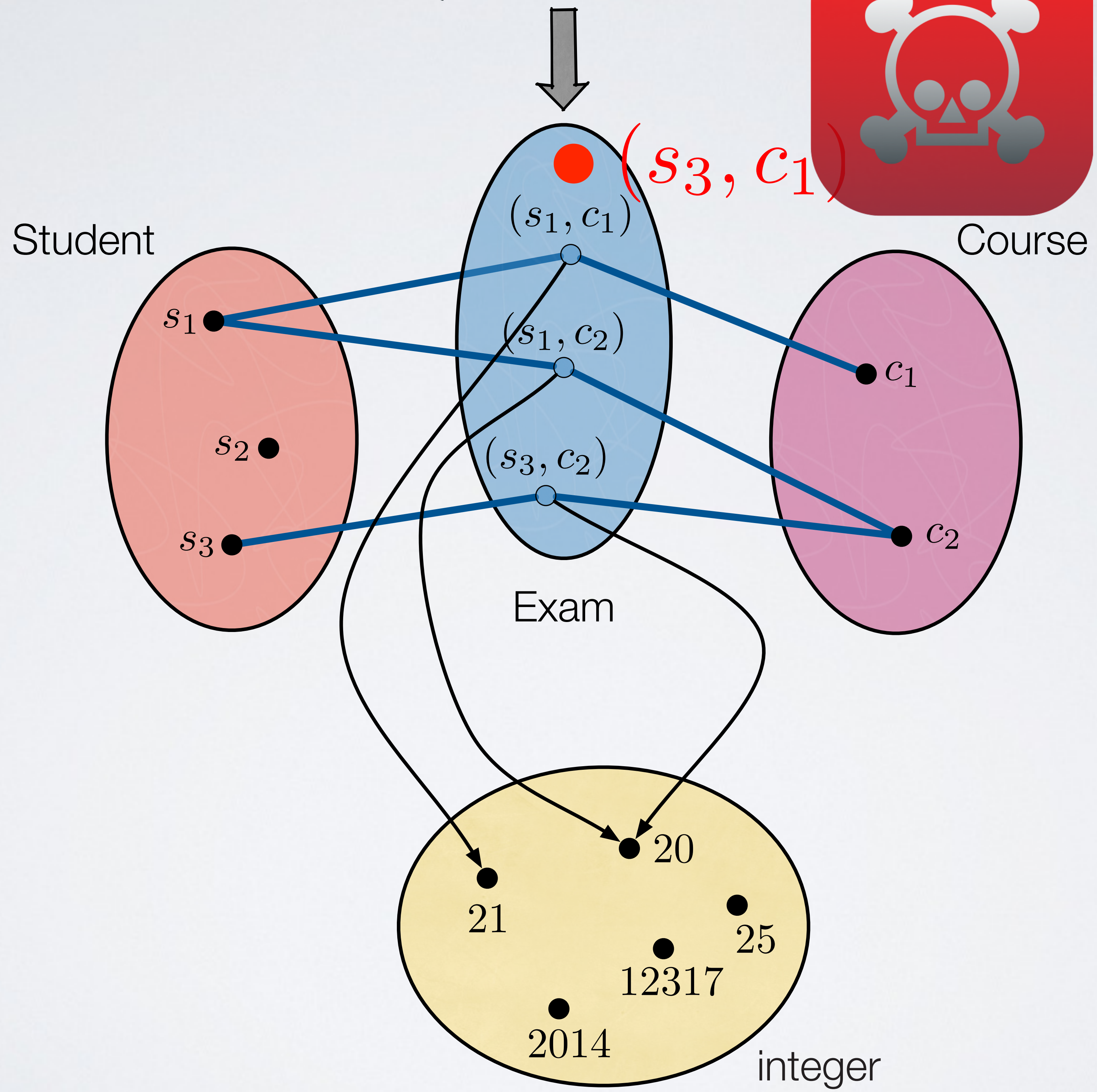
Binary Relationship Attribute: Example of Extensional Semantics

$$\mathrm{instance}(\mathsf{i}, \mathsf{Exam.Score}) = \Big\{\big((s_1, c_1), 21\big), \big((s_1, c_2), 20\big), \big((s_3, c_2), 20\big)\Big\}$$

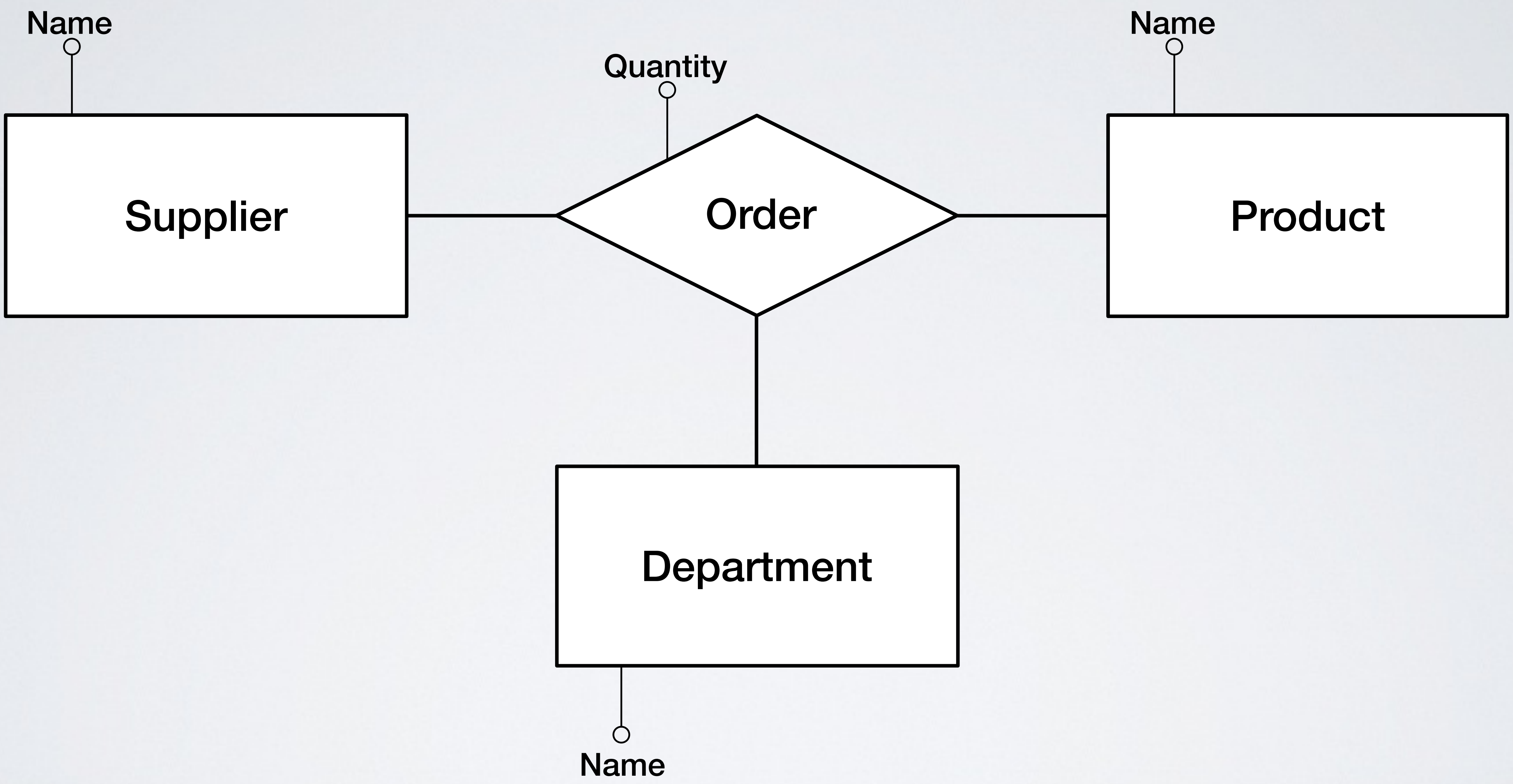


Binary Relationship Attribute: Example of Extensional Semantics

$$\text{instance}(\mathsf{i}, \mathsf{Exam.Score}) = \Big\{ \big((s_1, c_1), 21\big), \big((s_1, c_2), 20\big), \big((s_3, c_2), 20\big) \Big\}$$

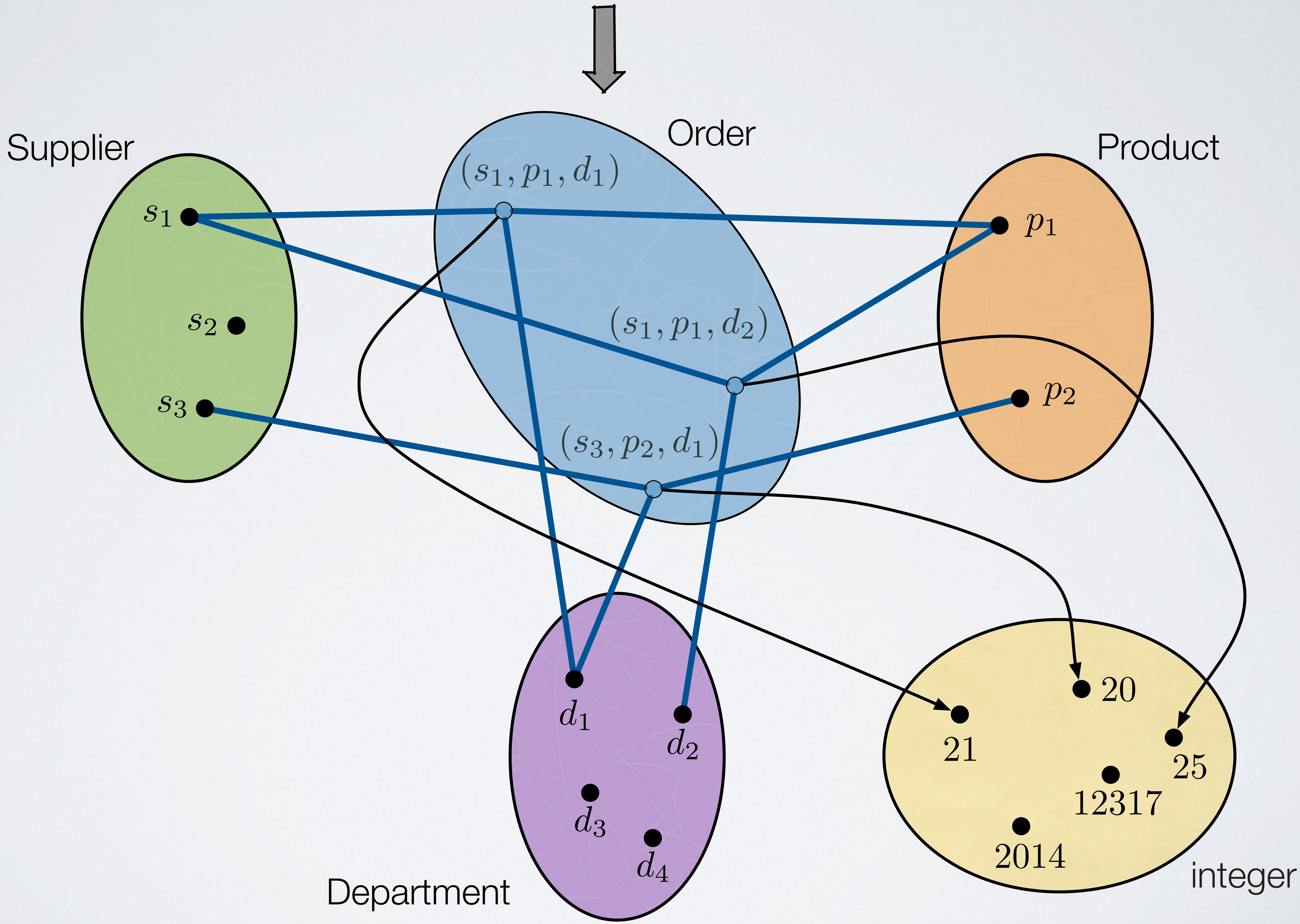


Attribute of N-ary Relationship: Example of Extensional Semantics



Attribute of N-ary Relationship: Example of Extensional Semantics

$$\text{instance}(\mathsf{i}, \mathsf{Order.Quantity}) = \Big\{\big((s_1, p_1, d_1), 21\big), \big((s_1, p_1, d_2), 25\big), \big((s_3, p_2, d_1), 20\big)\Big\}$$



Integrity Constraints

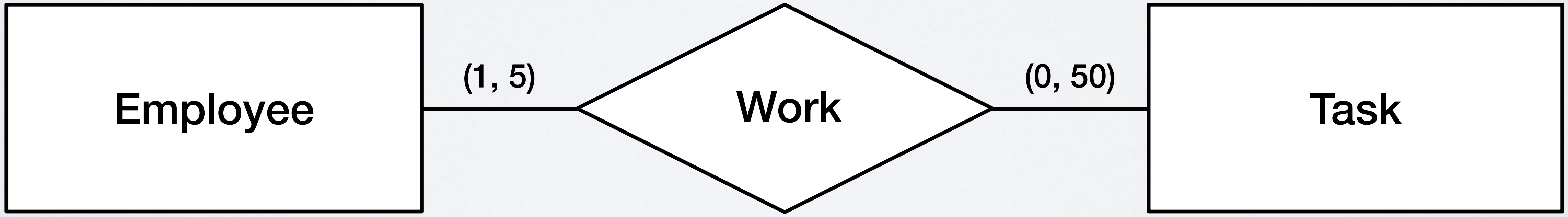
Integrity Constraint

An integrity constraint is a rule expressed on the schema (intensional level) that specifies a condition which must be met for each instance (extensional level) of the schema

- Cardinality constraints on relationships
- Cardinality constraints on attributes
- Identification constraints on entities
- Other constraints (external)

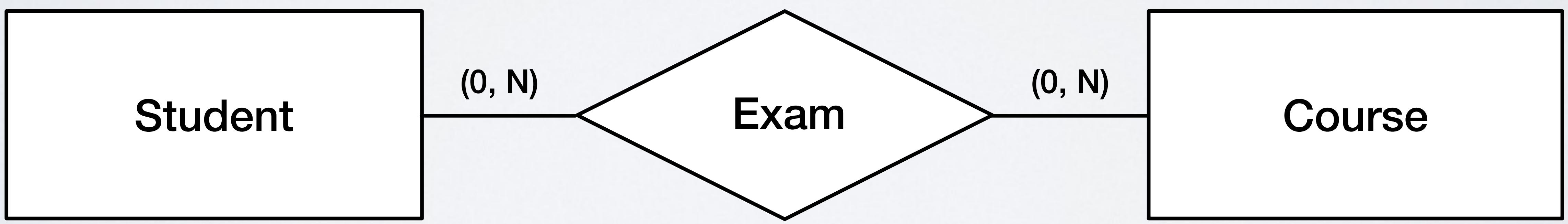
Cardinality Constraints on Relationships

A **cardinality constraint** refers to a role *u* of an entity *e* in a relationship *r* and it expresses a **lower bound** and an **upper bound** to the **number of instances of the relationship** *r* to which **each instance of the entity** *e* **can take part** with the **role** *u*



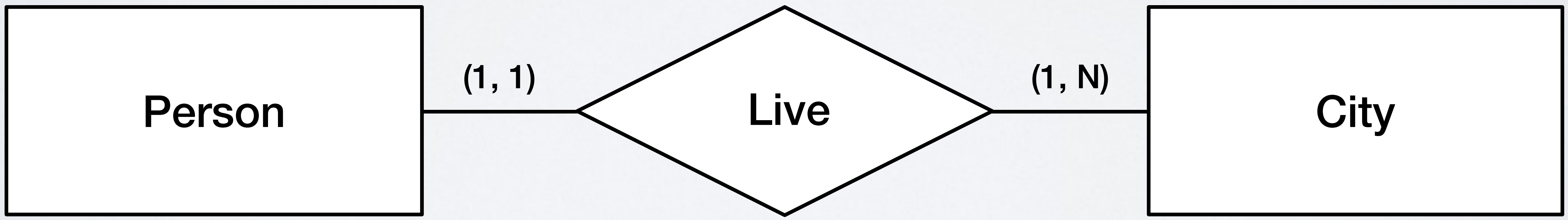
Cardinality Constraints on Relationships

A cardinality constraint refers to a role *u* of an entity *e* in a relationship *r* and it expresses a lower bound and an upper bound to the number of instances of the relationship *r* to which each instance of the entity *e* can take part with the role *u*



Cardinality Constraints on Relationships

A **cardinality constraint** refers to a role *u* of an entity *e* in a relationship *r* and it expresses a **lower bound** and an **upper bound** to the **number of instances of the relationship** *r* to which **each instance of the entity** *e* **can take part** with the **role** *u*



Notation for Cardinality Constraints

- Relevant cardinalities are: **0**, **1**, **N**

- Minimum cardinality:
  - **0** means “optional participation”
  - **1** means “mandatory participation”

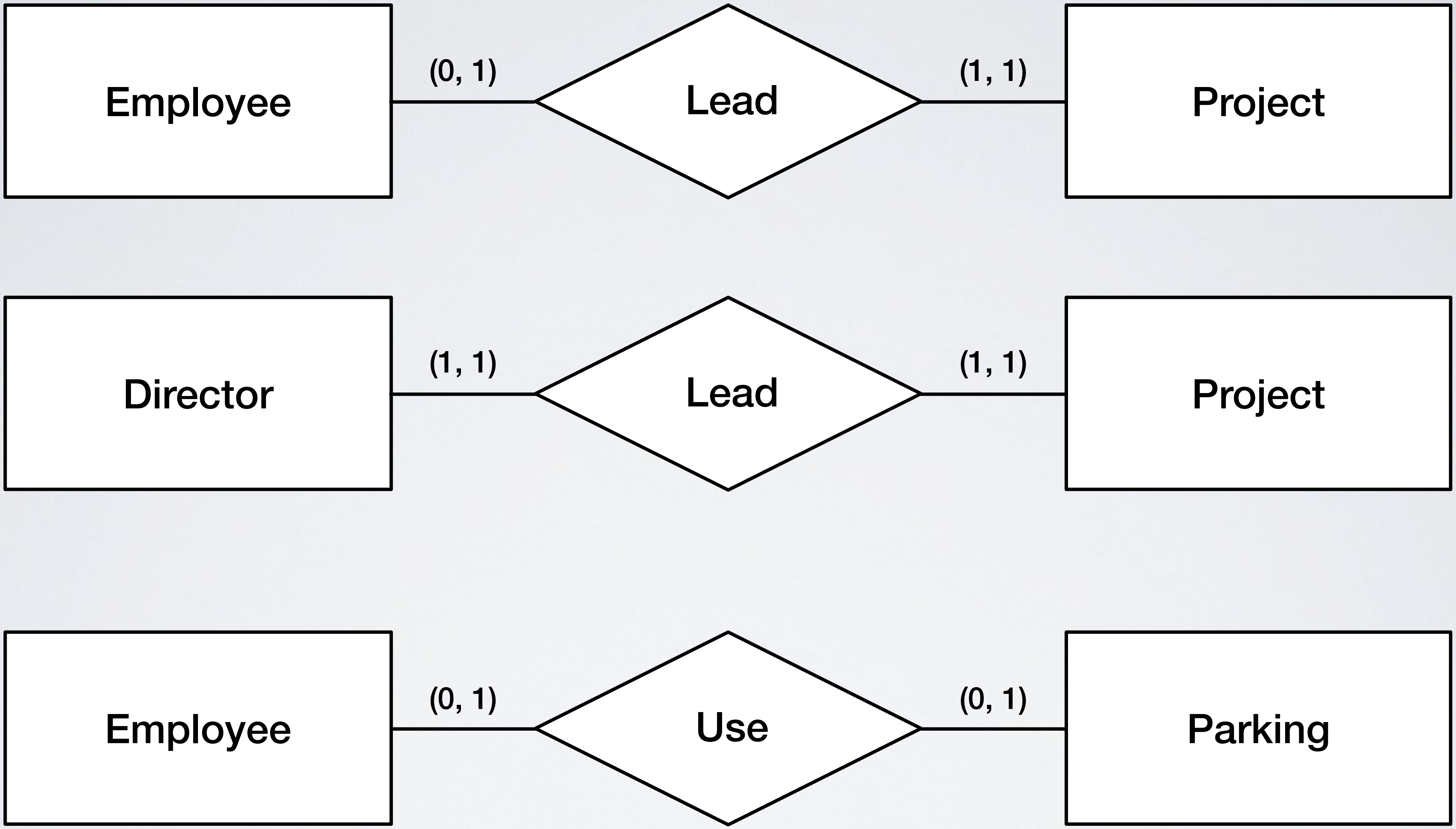
- Maximum cardinality:
  - **1** means: “the entity can participate at maximum once to the relationship”
  - **N** mean: “the entity can participate many times to the relationship”

Classification of Binary Relationships

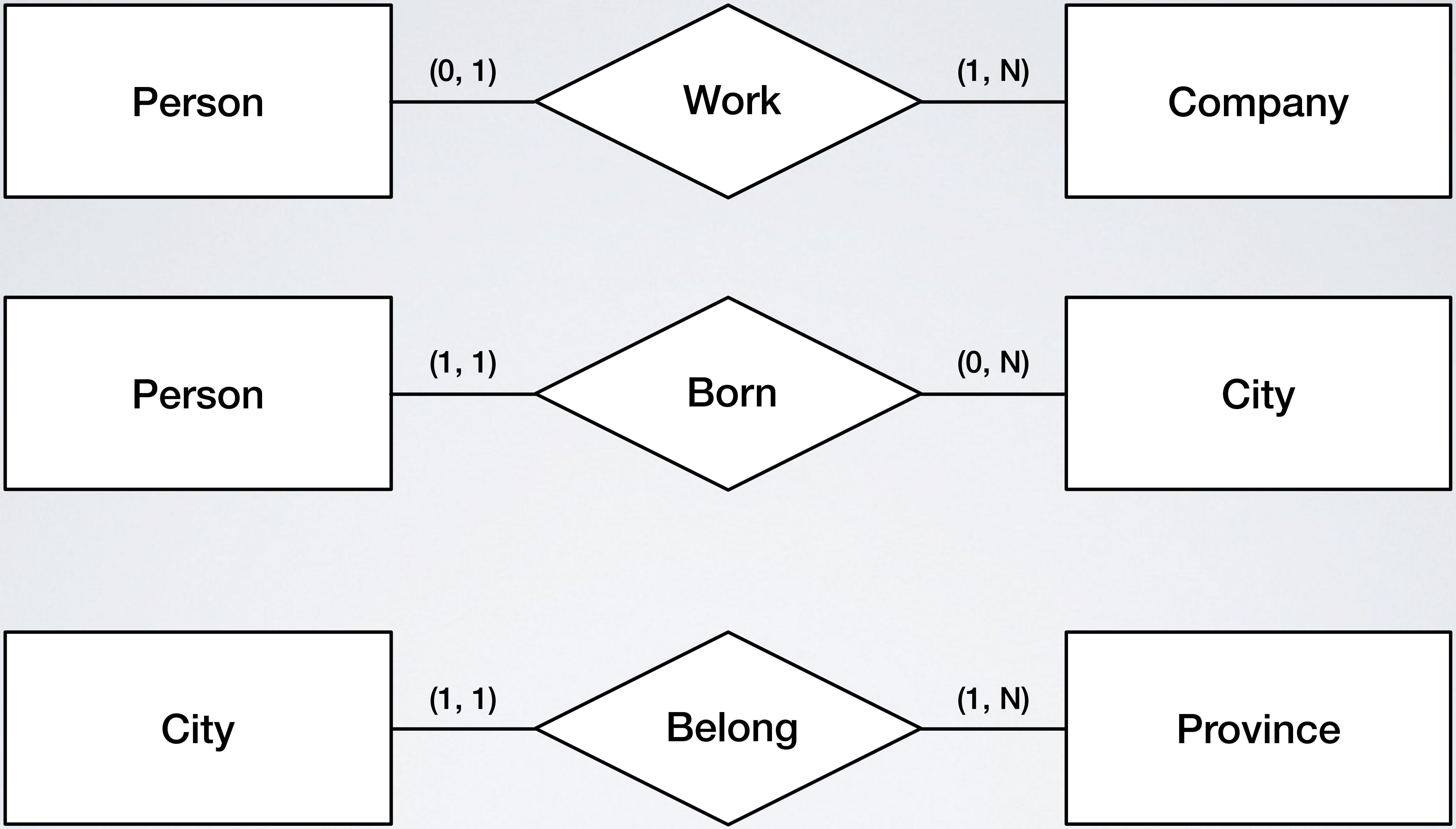
Binary relationships can be classified according to their cardinality constraints

- **One-to-One relationship**: each instance of an entity is associated with one and only one instance of another entity
- **One-to-Many relationship**: each instance of an entity is associated with one or more instances of another entity
- **Many-to-Many relationship**: one or more instances of an entity are associated with one or more instances of another entity

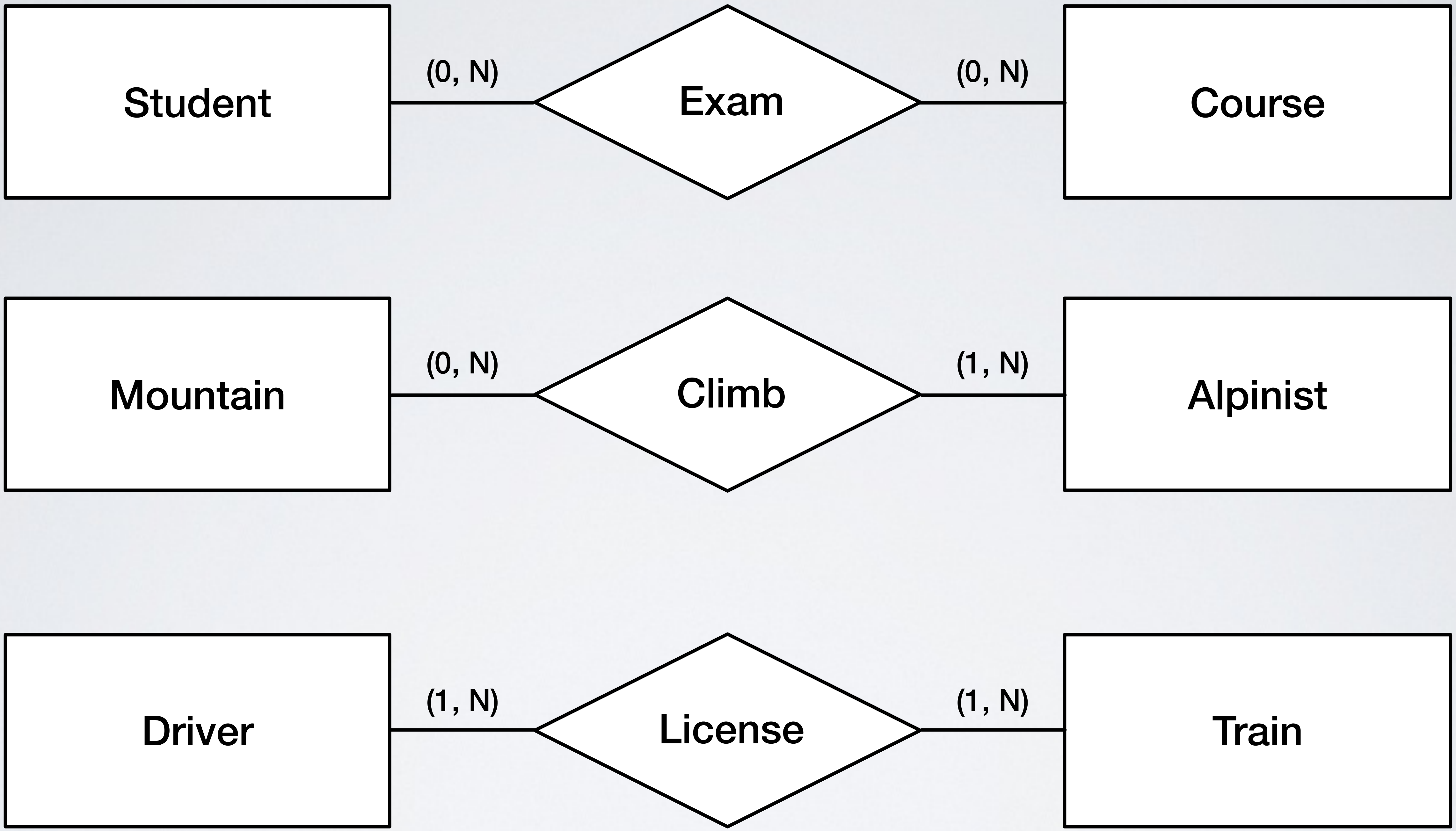
Example of One-to-One Relationships

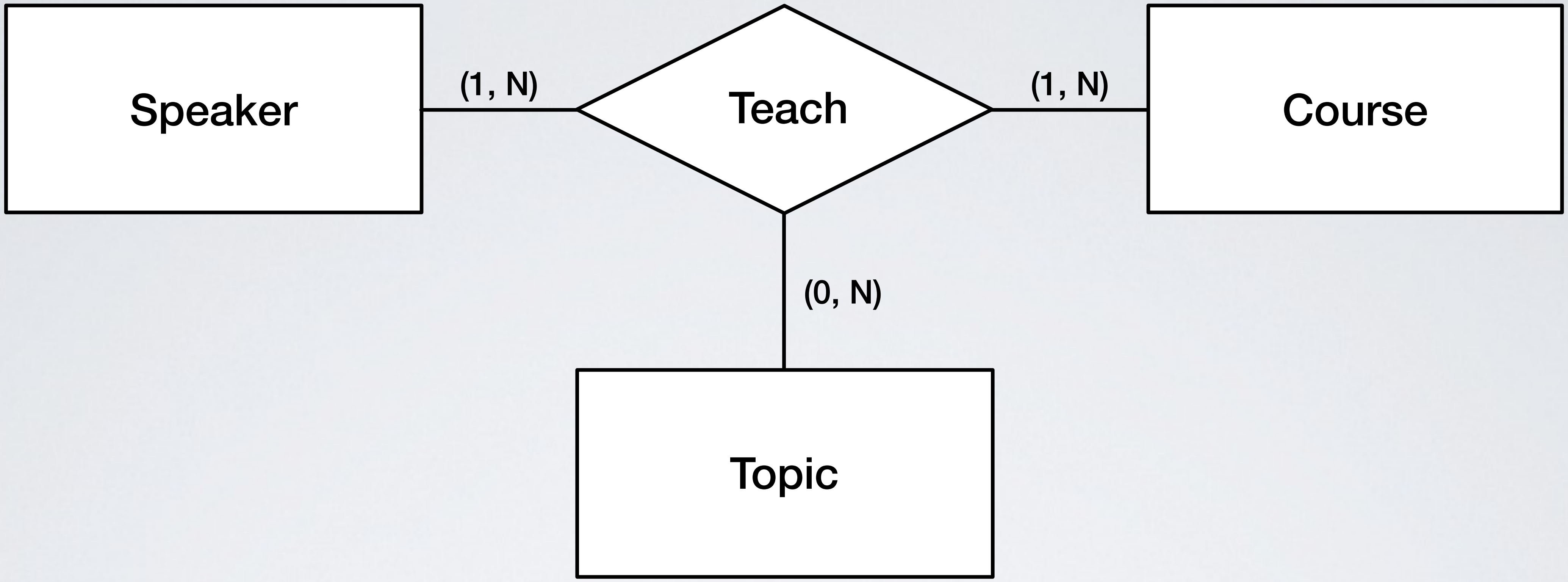


Example of One-to-Many Relationships



Example of Many-to-Many Relationships





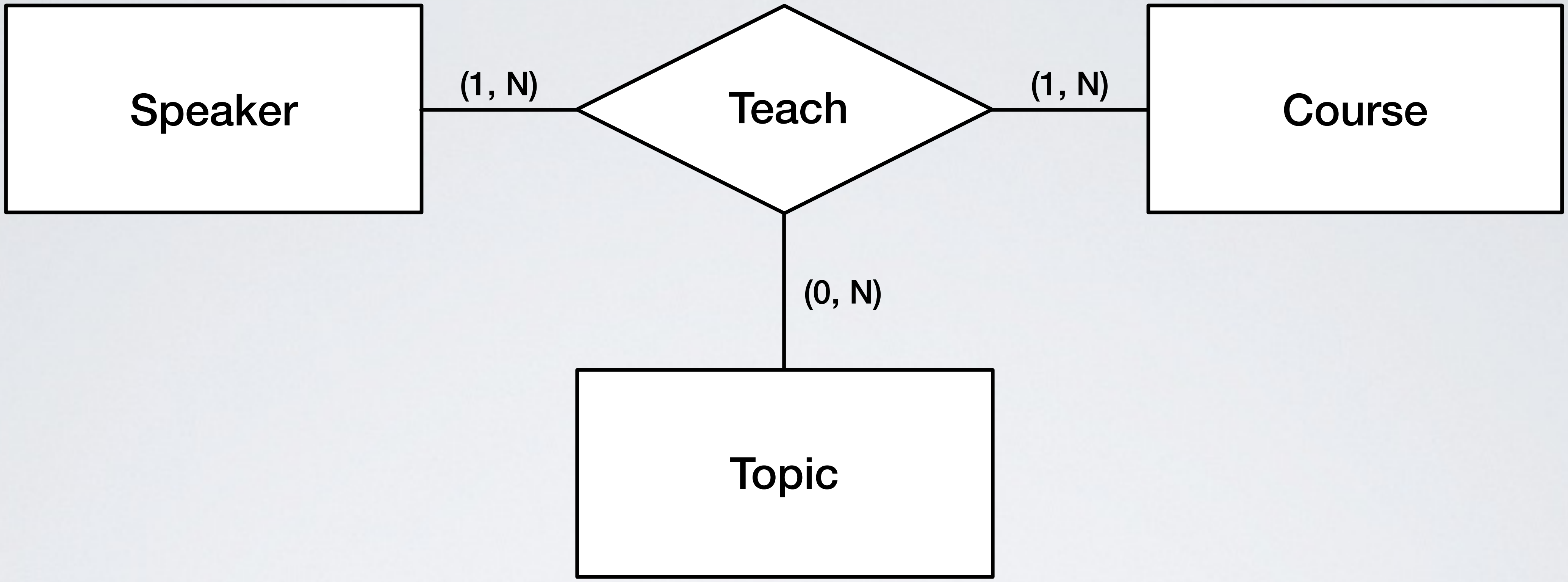
- By definition, an instance of the **Teach** relationship is a triple

$$t_k = (s_i, c_j, t_h) \in \text{instance}(\text{i, Speaker}) \times \text{instance}(\text{i, Course}) \times \text{instance}(\text{i, Topic})$$

- Even if the participation of the **Topic** entity is **optional**, it **cannot exist** an instance of the **Teach** relationship **without** an instance of the **Topic** entity

$$t_k = (s_i, c_j, \bullet)$$

- In other terms, you cannot use **Teach** to express the fact that a **Speaker** will give a **Course** but the **Topics** are not fixed yet



- By definition, an instance of the **Teach** relationship is a triple

$$t_k = (s_i, c_j, t_h) \in \text{instance}(\text{i, Speaker}) \times \text{instance}(\text{i, Course}) \times \text{instance}(\text{i, Topic})$$

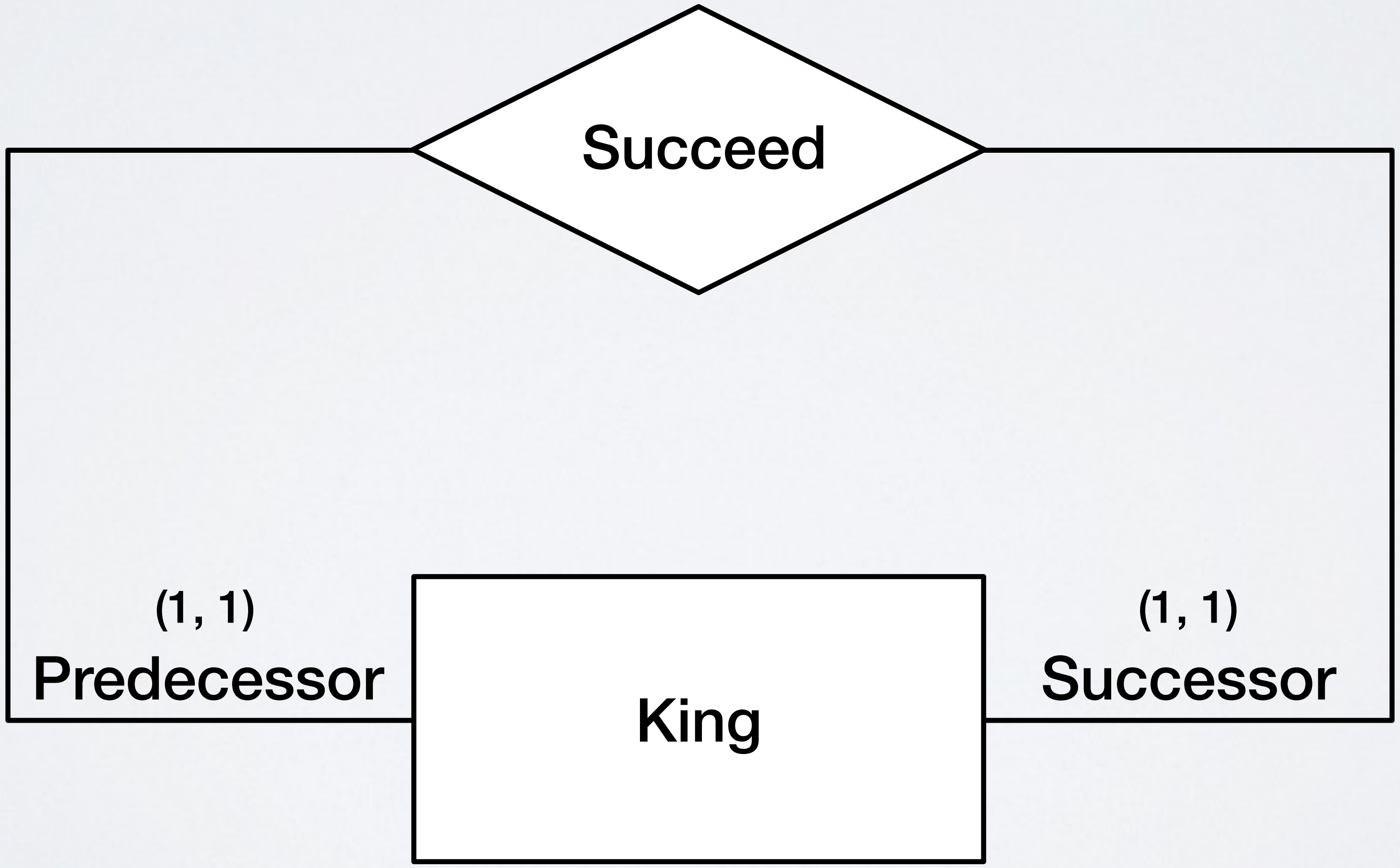
- Even if the participation of the **Topic** entity is **optional**, it **cannot exist** an instance of the **Teach** relationship **without** an instance of the **Topic** entity

$$t_k = (s_i, c_j, \bullet)$$

- In other terms, you cannot use **Teach** to express the fact that a **Speaker** will give a **Course** but the **Topics** are not fixed yet

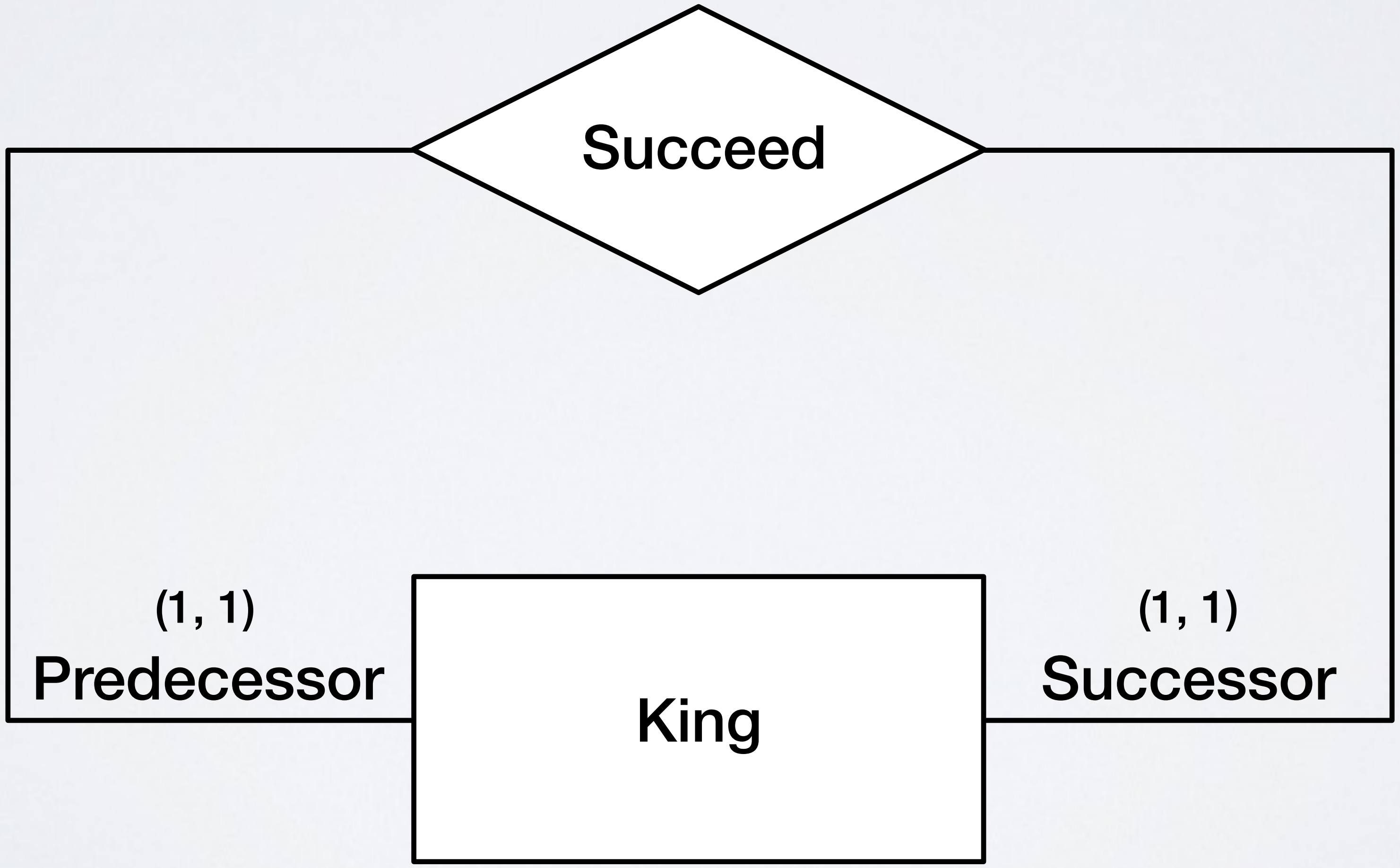
Recursive Relationship: Example of Cardinality

$$k_0 \rightarrow k_1 \rightarrow k_2 \rightarrow k_3 \rightarrow \ldots \rightarrow k_{n-1} \rightarrow k_n$$



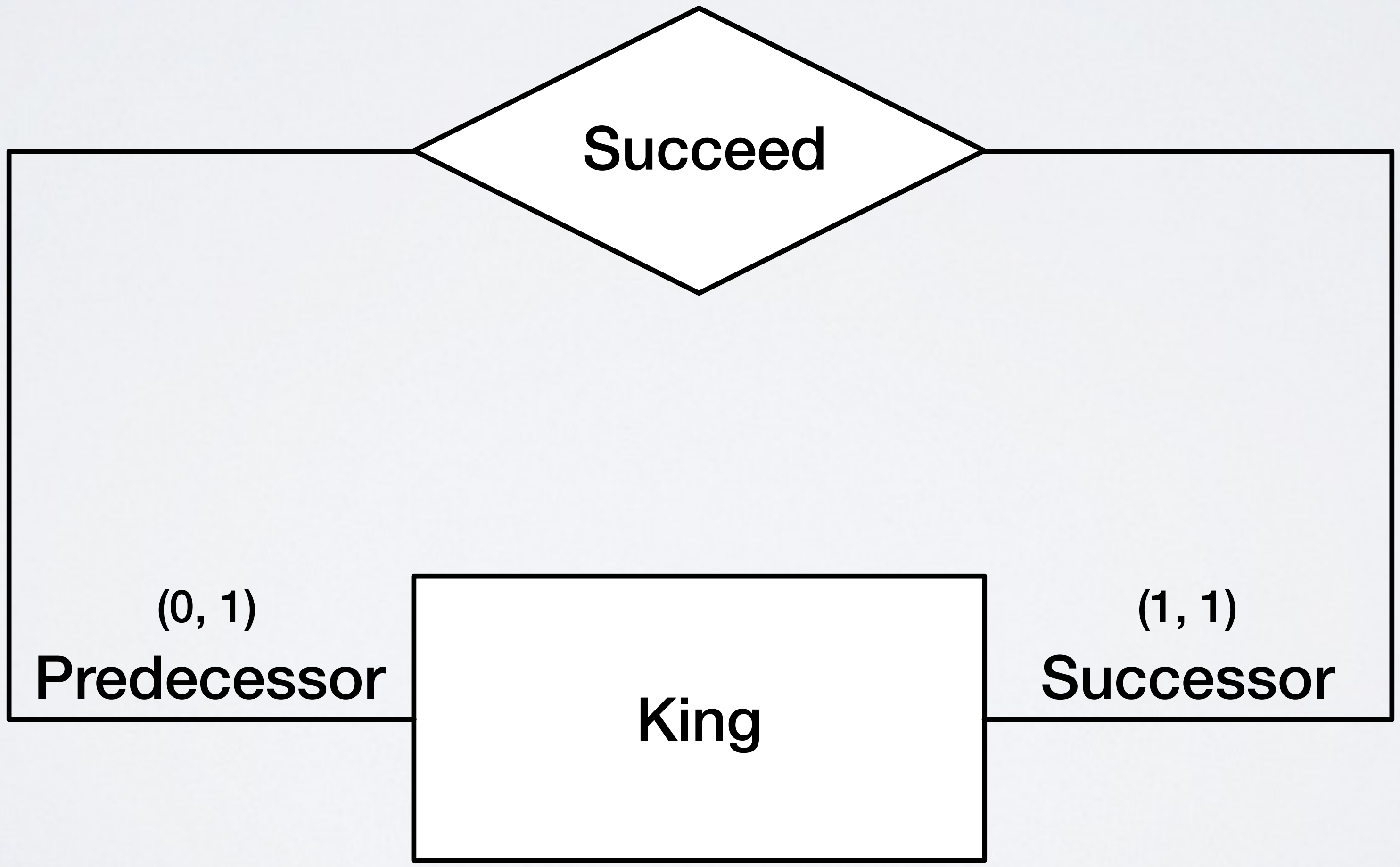
Recursive Relationship: Example of Cardinality

$$\boxed{k_0} \to \boxed{k_1 \to k_2 \to k_3 \to \ldots \to k_{n-1}} \to \boxed{k_n}$$

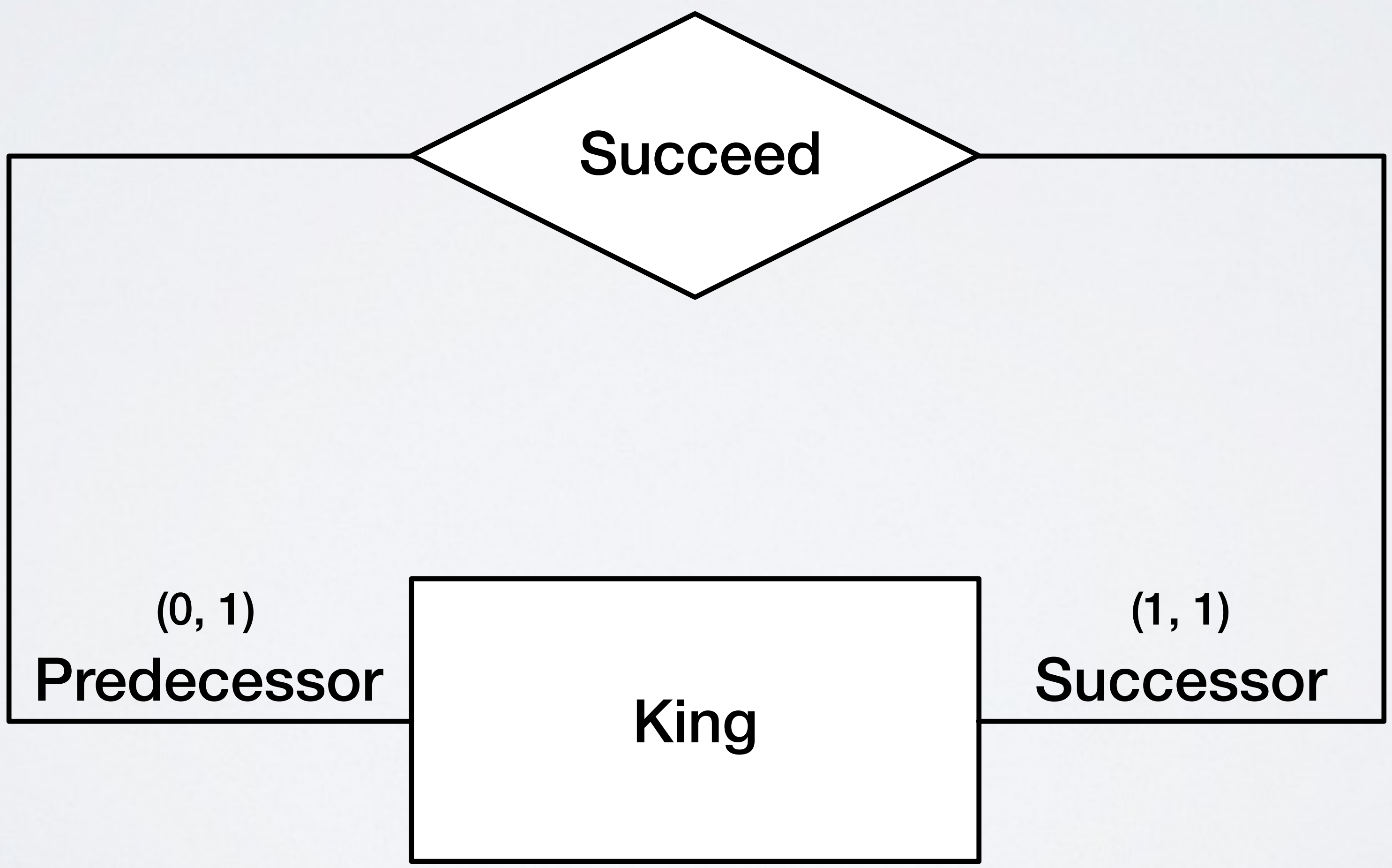


Recursive Relationship: Example of Cardinality

$$k_0 \rightarrow k_1 \rightarrow k_2 \rightarrow k_3 \rightarrow \ldots \rightarrow k_{n-1} \rightarrow k_n$$

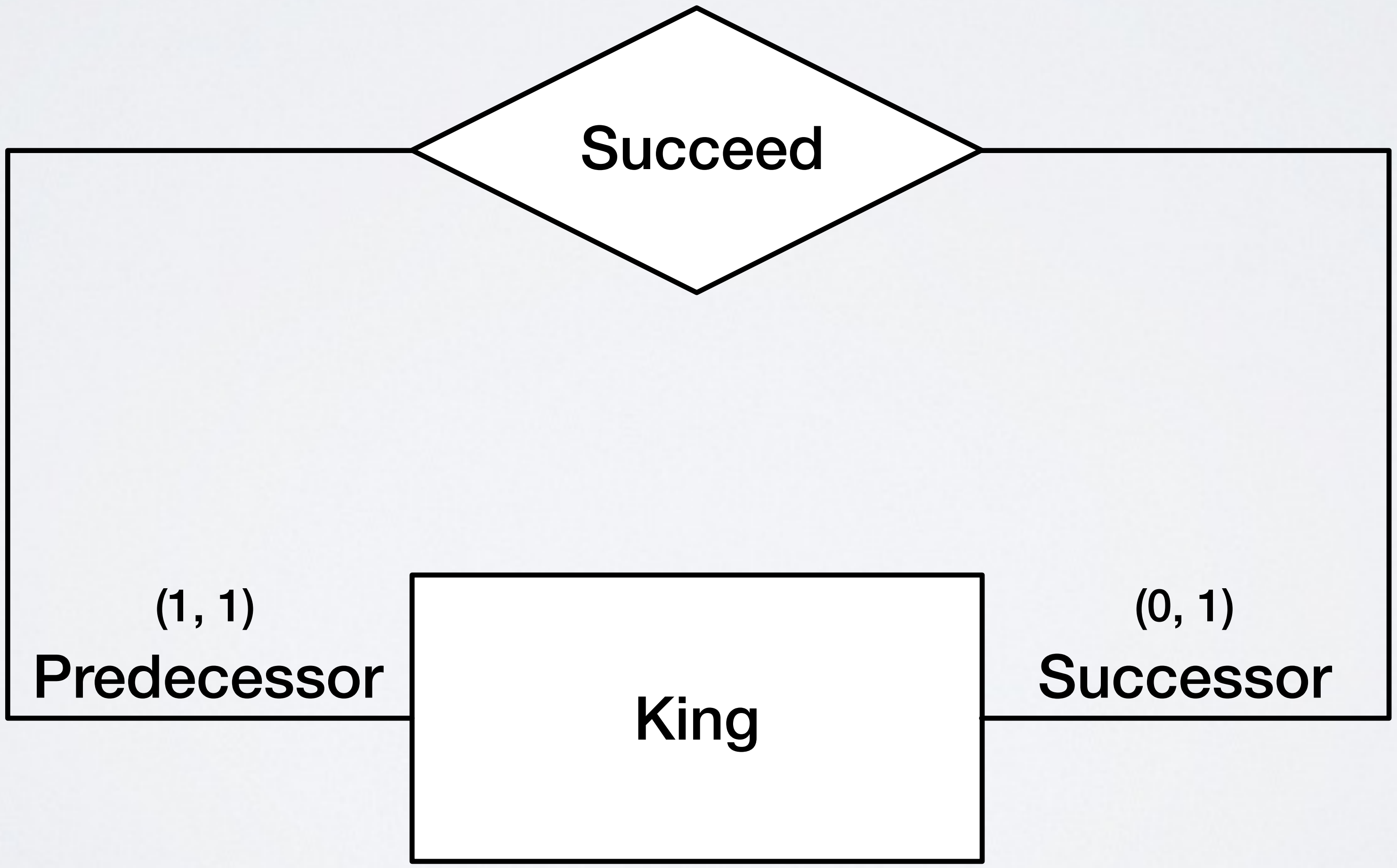


$$k_0 \rightarrow k_1 \rightarrow k_2 \rightarrow k_3 \rightarrow \ldots \rightarrow k_{n-1} \rightarrow k_n$$



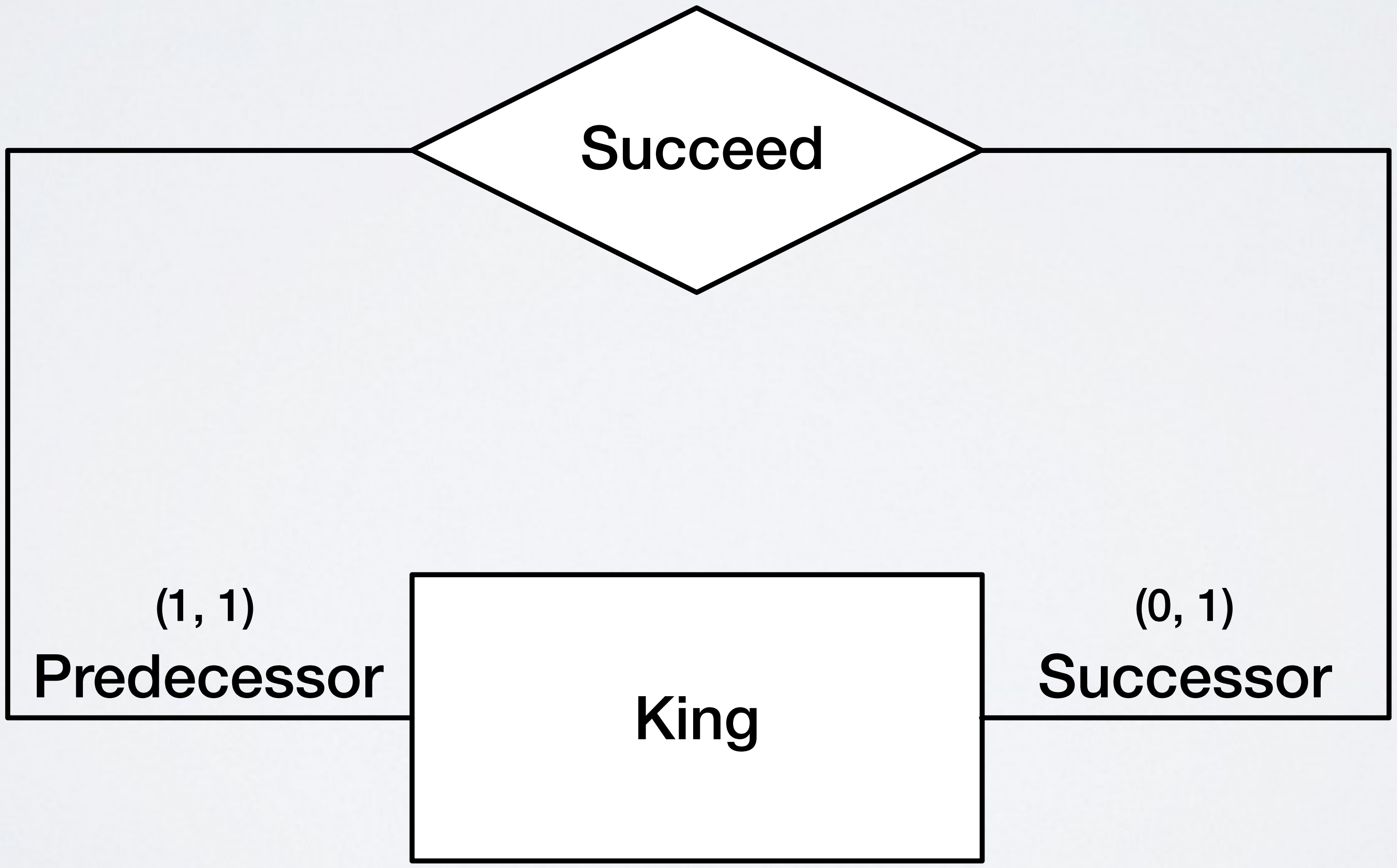
Recursive Relationship: Example of Cardinality

$$k_0 \rightarrow k_1 \rightarrow k_2 \rightarrow k_3 \rightarrow \ldots \rightarrow k_{n-1} \rightarrow k_n$$



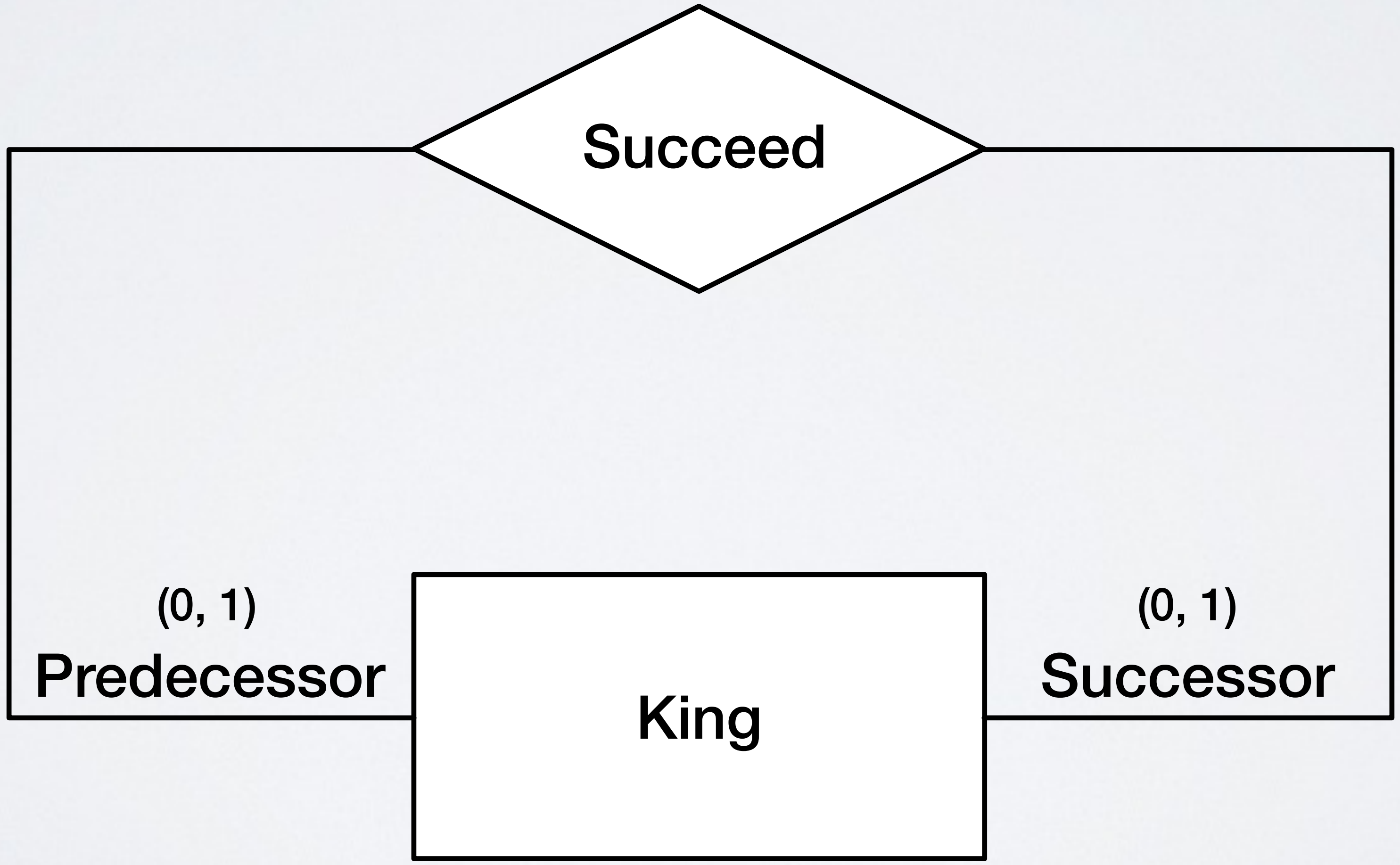
Recursive Relationship: Example of Cardinality

$$k_0 \rightarrow k_1 \rightarrow k_2 \rightarrow k_3 \rightarrow \ldots \rightarrow k_{n-1} \rightarrow k_n$$



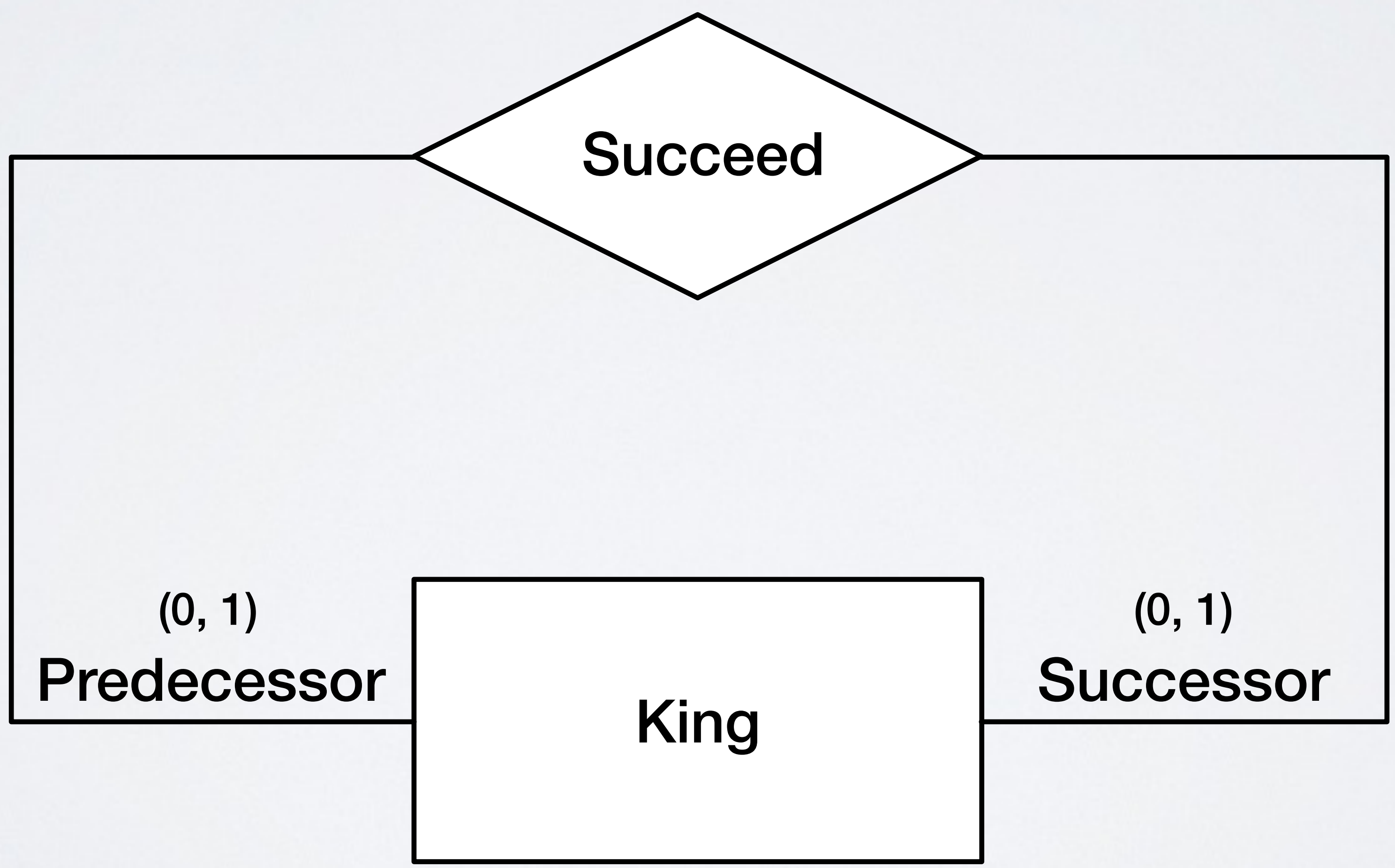
Recursive Relationship: Example of Cardinality

$$k_0 \rightarrow k_1 \rightarrow k_2 \rightarrow k_3 \rightarrow \ldots \rightarrow k_{n-1} \rightarrow k_n$$



Recursive Relationship: Example of Cardinality

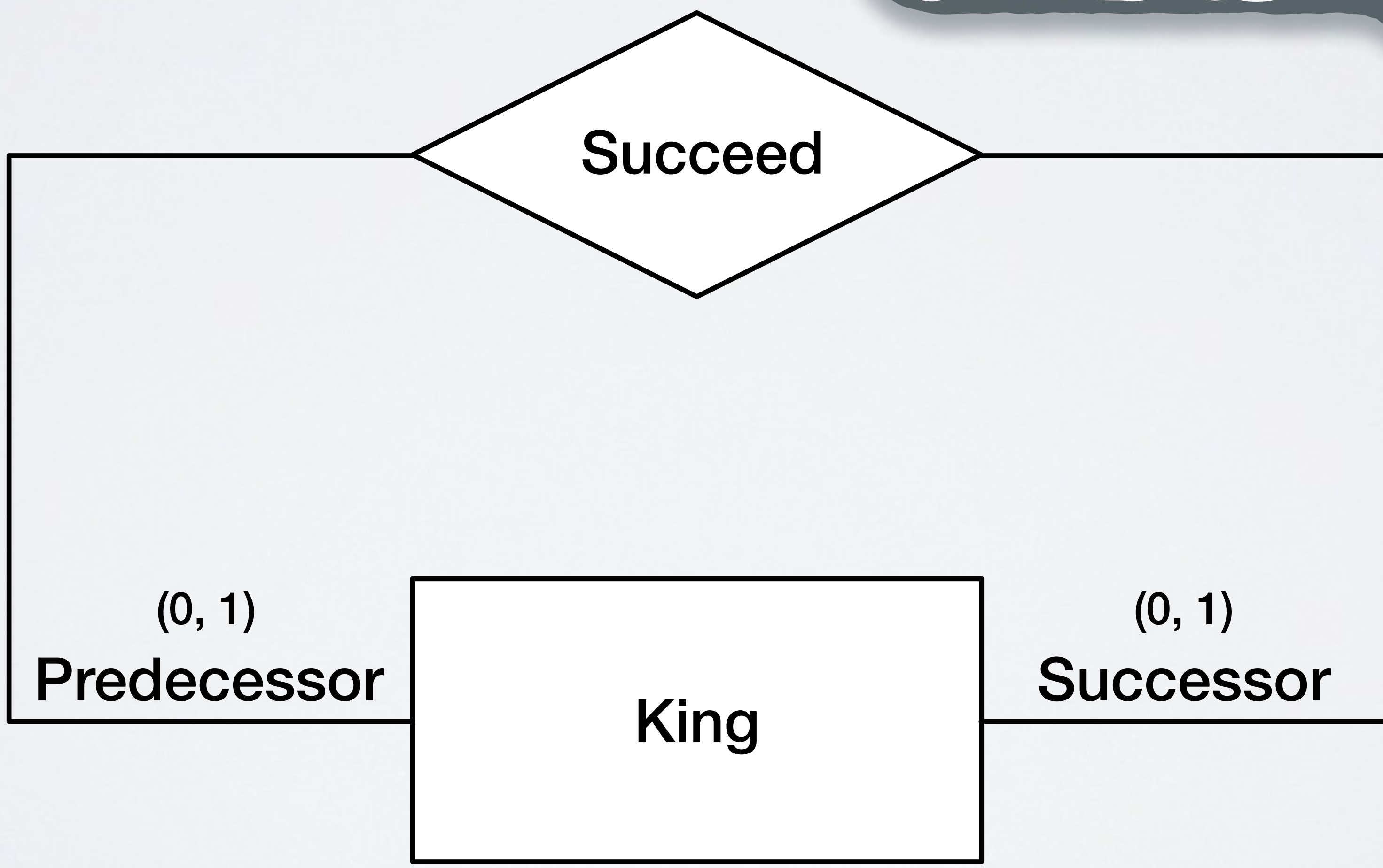
$$k_0 \rightarrow k_1 \rightarrow k_2 \rightarrow k_3 \rightarrow \ldots \rightarrow k_{n-1} \rightarrow k_n$$

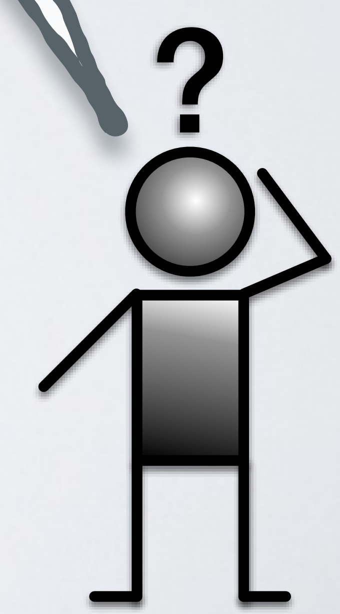


Recursive Relationship: Example of Cardinality

$$k_0 \rightarrow k_1 \rightarrow k_2 \rightarrow k_3 \rightarrow \ldots \rightarrow k_{n-1} \rightarrow k_n$$

How to **avoid** that the last king becomes the **predecessor** of the first one?

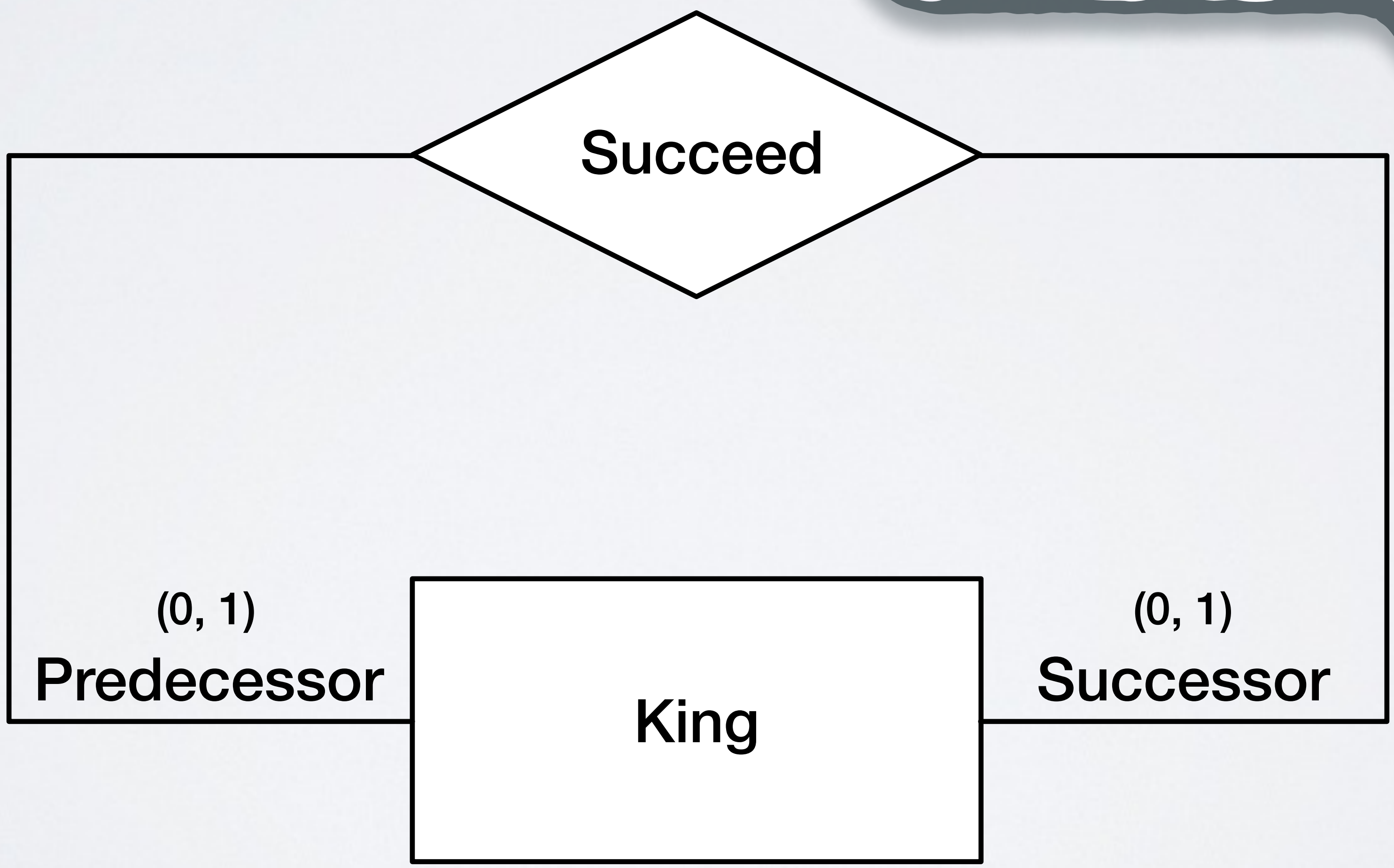


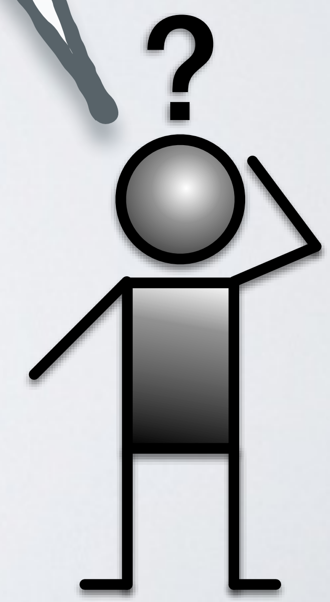


Recursive Relationship: Example of Cardinality

$$k_0 \rightarrow k_1 \rightarrow k_2 \rightarrow k_3 \rightarrow \ldots \rightarrow k_{n-1} \rightarrow k_n$$

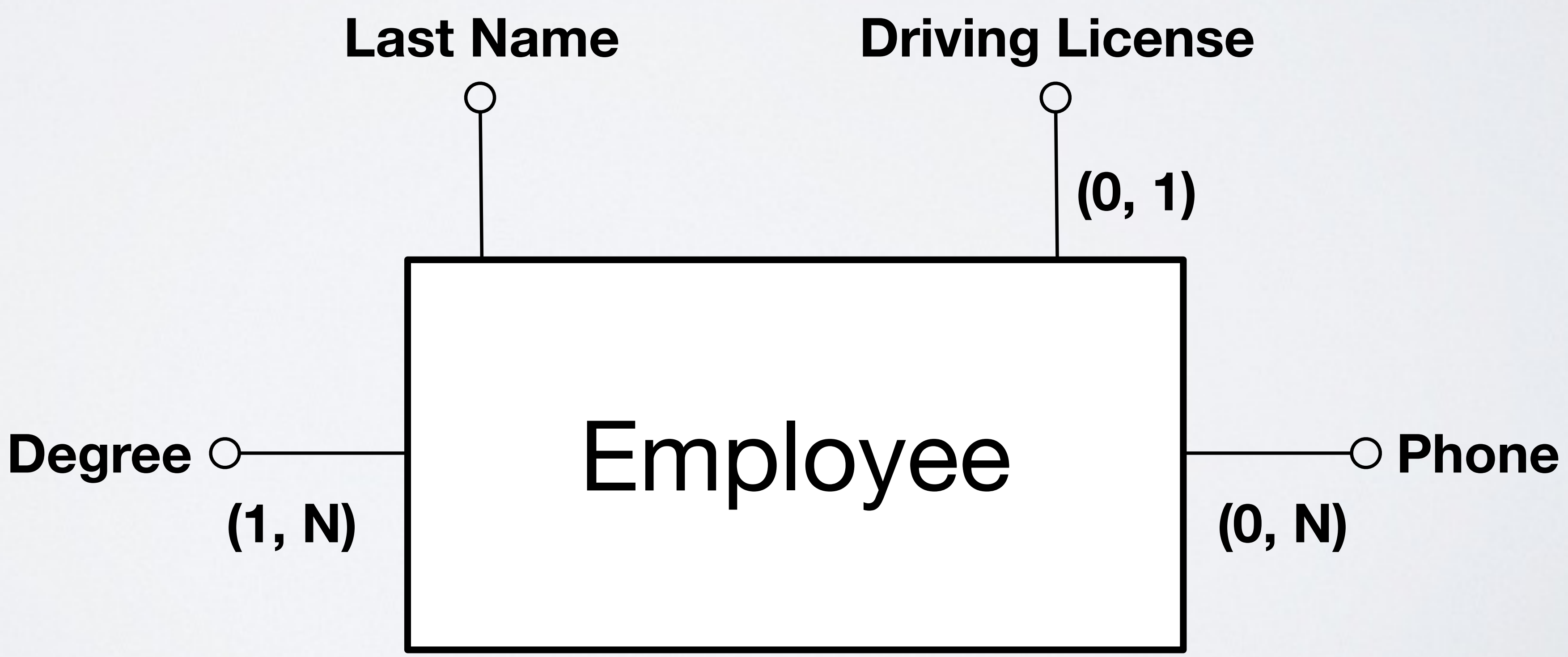
How to **avoid** that a king becomes **predecessor** and **successor** of herself/himself?





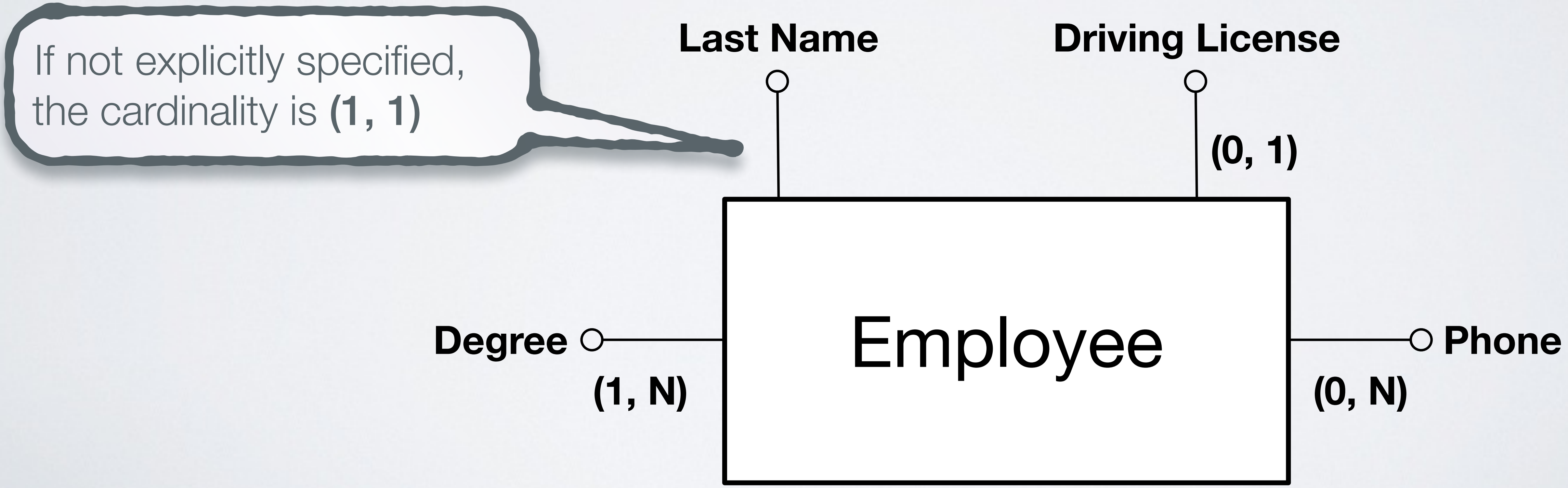
Cardinality Constraints on Attributes

A **cardinality constraint** on an attribute of an entity (relationship) expresses a **lower bound** and **upper bound** to the **number of values** associated with **each instance of the entity (relationship)**.



Cardinality Constraints on Attributes

A **cardinality constraint** on an attribute of an entity (relationship) expresses a **lower bound** and **upper bound** to the **number of values** associated with **each instance of the entity (relationship)**.



Notation for Cardinality Constraints on Attributes

- Relevant cardinalities are: **0**, **1**, **N**
- Minimum cardinality:
  - **0** means the attribute is **optional**
- Maximum cardinality:
  - **N** means the attribute is **multi-valued**
- If the constraint is not explicitly specified, the cardinality of the attribute is (1, 1), i.e. it is **mandatory**

Attribute: Extensional Semantics (Revisited)

- **Simple** attributes

$$D \subseteq A$$

where **A** is a domain of "atomic" values

- **Multi-valued** attributes

$$D \subseteq \mathcal{P}(D')$$

where **D'** is a domain of simple attributes and $\mathcal{P}(\bullet)$ is the power set

- **Optional** attributes

$$D \subseteq \mathcal{P}(D')$$

$$\text{NULL} \mapsto \varnothing$$

where **D'** and $\mathcal{P}(\bullet)$ are as above

Identification Constraints on Entities

An identification constraint for an entity *e* defines an identifier for the entity, that is a set of properties (attributes or relationships) which allow us to uniquely identify all the instances of an entity.

- There does not exist two different instances of the entity which have the same value on all the properties which constitute the identifier
- Any number of identification constraints can be defined on an entity but at least one

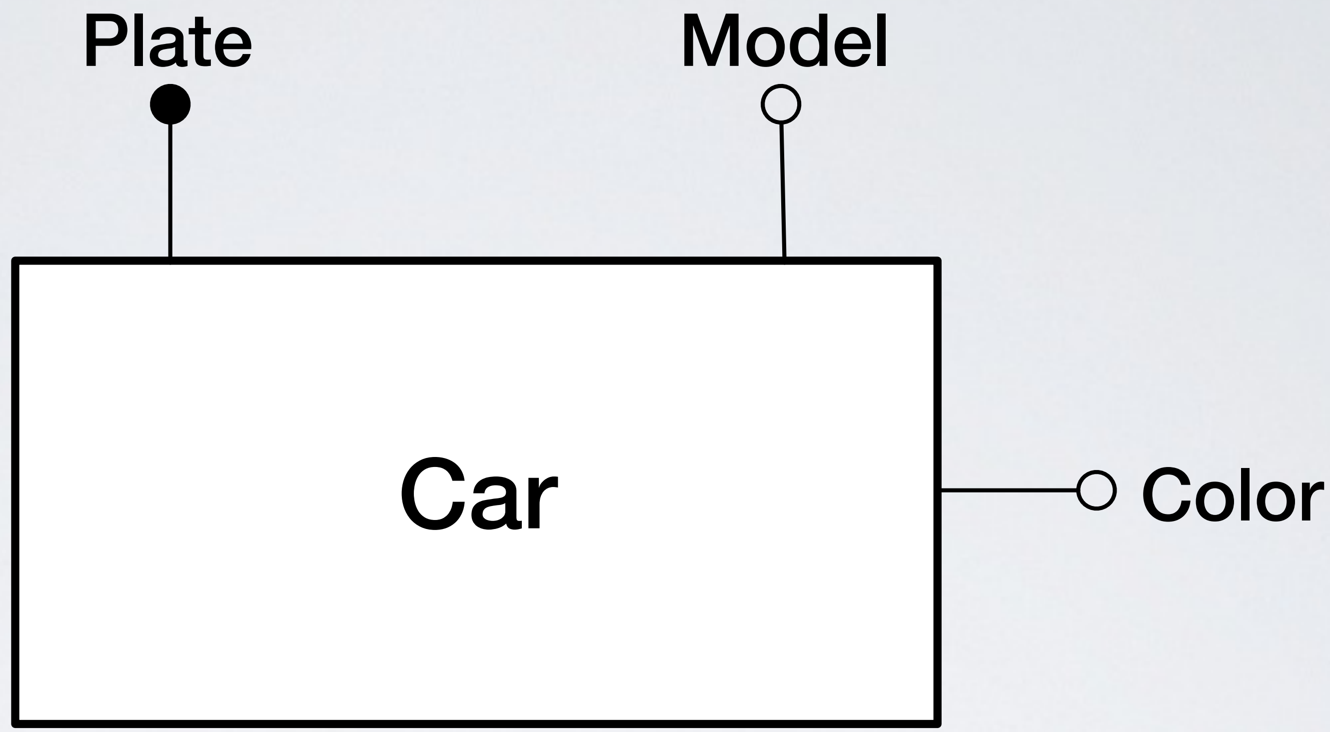
Types of Identifiers

- An **identifier** of an entity **e** can be
  - **Internal**, i.e. constituted by only attributes of **e** with cardinality (1, 1)
  - **External**, i.e. constituted by attributes of **e** AND by roles in relationships where **e** participates OR just by roles in relationships where **e** participates, provided that all the involved attributes and roles have cardinality (1,1)

- Notation for internal identifiers:
  - In case of identifier constituted by a single attribute, its circle is filled in
  - In case of identifier constituted by more attributes, a line with a black circle connects all the involved attributes

- Notation for external identifiers:
  - A line with a black circle connects all the involved attributes and/or roles

Example of Internal Identifier (1/2)



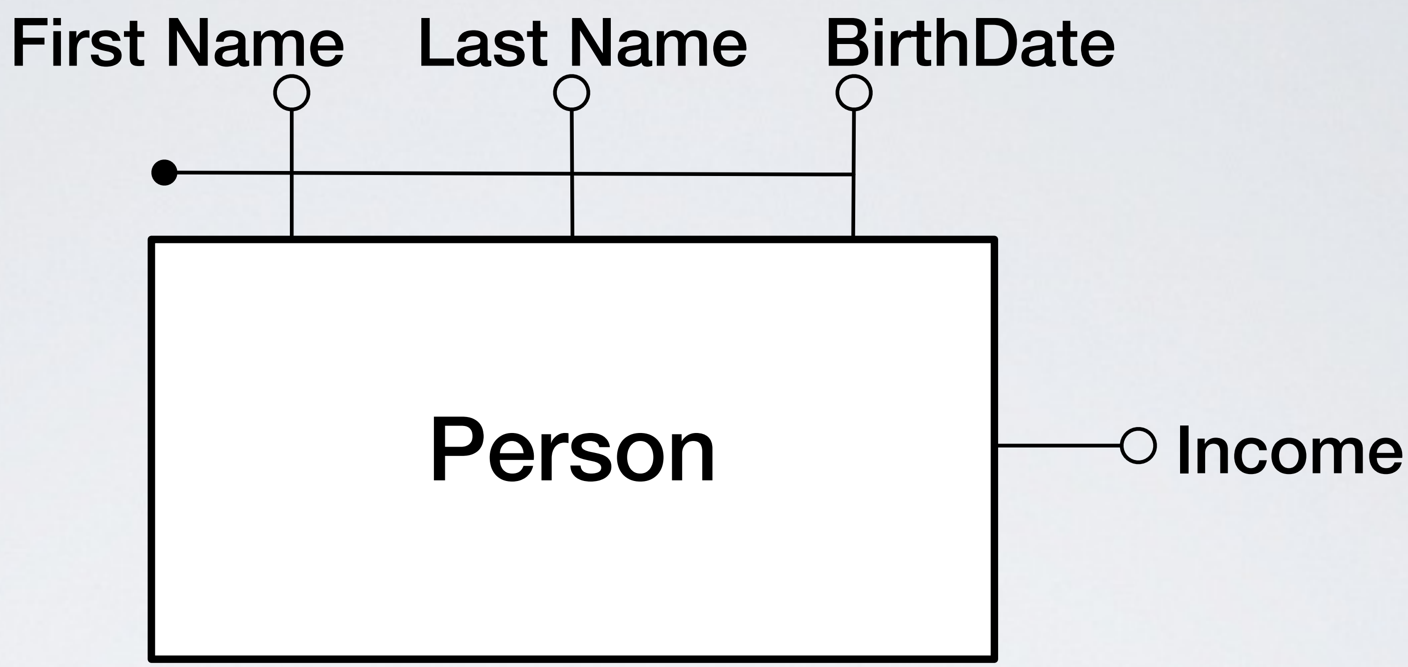
$$\text{instance}(\mathsf{i}, \mathsf{Car}) = \{c_1, c_2, c_3\}$$

$$\text{instance}(\mathsf{i}, \mathsf{Car.Plate}) = \{(c_1, \text{CZ 421 LK}), (c_2, \text{MK 9651 AB}), (c_3, \text{BG 529 BV})\}$$

$$\text{instance}(\mathsf{i}, \mathsf{Car.Model}) = \{(c_1, \text{Alfa MiTo}), (c_2, \text{Audi A3}), (c_3, \text{Citroen C3})\}$$

$$\text{instance}(\mathsf{i}, \mathsf{Car.Color}) = \{(c_1, \text{Red}), (c_2, \text{Red}), (c_3, \text{White})\}$$

Example of Internal Identifier (2/2)



$$\text{instance}(\mathsf{i}, \mathsf{Person}) = \{p_1, p_2, p_3\}$$

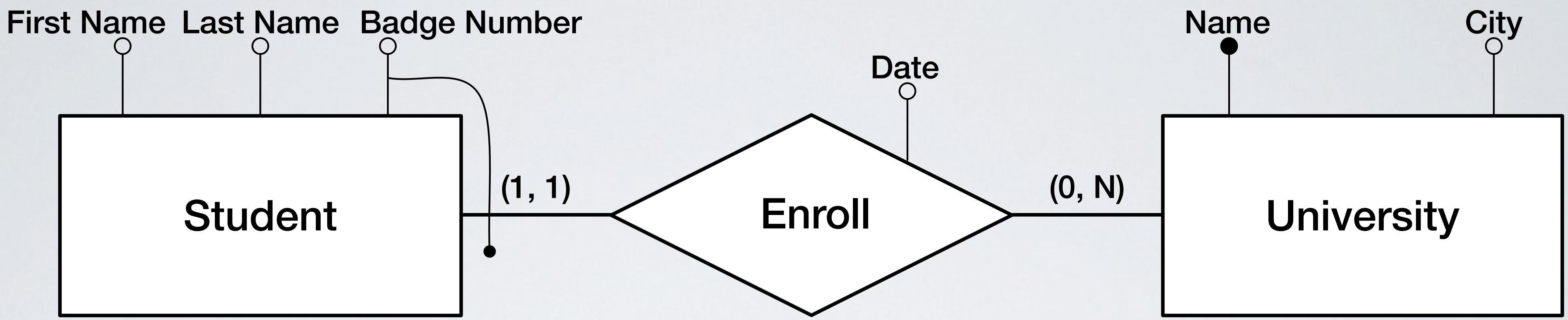
$$\text{instance}(\mathsf{i}, \mathsf{Person.FirstName}) = \{(p_1, \text{Mario}), (p_2, \text{Giovanni}), (p_3, \text{Mario})\}$$

$$\text{instance}(\mathsf{i}, \mathsf{Person.LastName}) = \{(p_1, \text{Rossi}), (p_2, \text{Bianchi}), (p_3, \text{Rossi})\}$$

$$\text{instance}(\mathsf{i}, \mathsf{Person.BirthDate}) = \{(p_1, \text{23-05-1975}), (p_2, \text{12-01-1967}), (a_3, \text{15-04-1984})\}$$

$$\text{instance}(\mathsf{i}, \mathsf{Person.Income}) = \{(p_1, 18.000), (p_2, 20.000), (p_3, 20.000)\}$$

Example of External Identifier



$$\text{instance(i, Student)} = \{s_1, s_2, s_3\}$$

$$\text{instance(i, University)} = \{u_1, u_2\}$$

$$\text{instance(i, Student.FirstName)} = \{(s_1, \text{Mario}), (s_2, \text{Giovanni}), (s_3, \text{Mario})\}$$

$$\text{instance(i, Student.LastName)} = \{(s_1, \text{Rossi}), (s_2, \text{Bianchi}), (s_3, \text{Rossi})\}$$

$$\text{instance(i, Student.BadgeNumber)} = \{(s_1, 419366), (p_2, 512344), (s_3, 419366)\}$$

$$\text{instance(i, University.Name)} = \{(u_1, \text{Università degli Studi di Padova}), (u_2, \text{Politecnico di Milano})\}$$

$$\text{instance(i, University.City)} = \{(u_1, \text{Padova}), (u_2, \text{Milano})\}$$

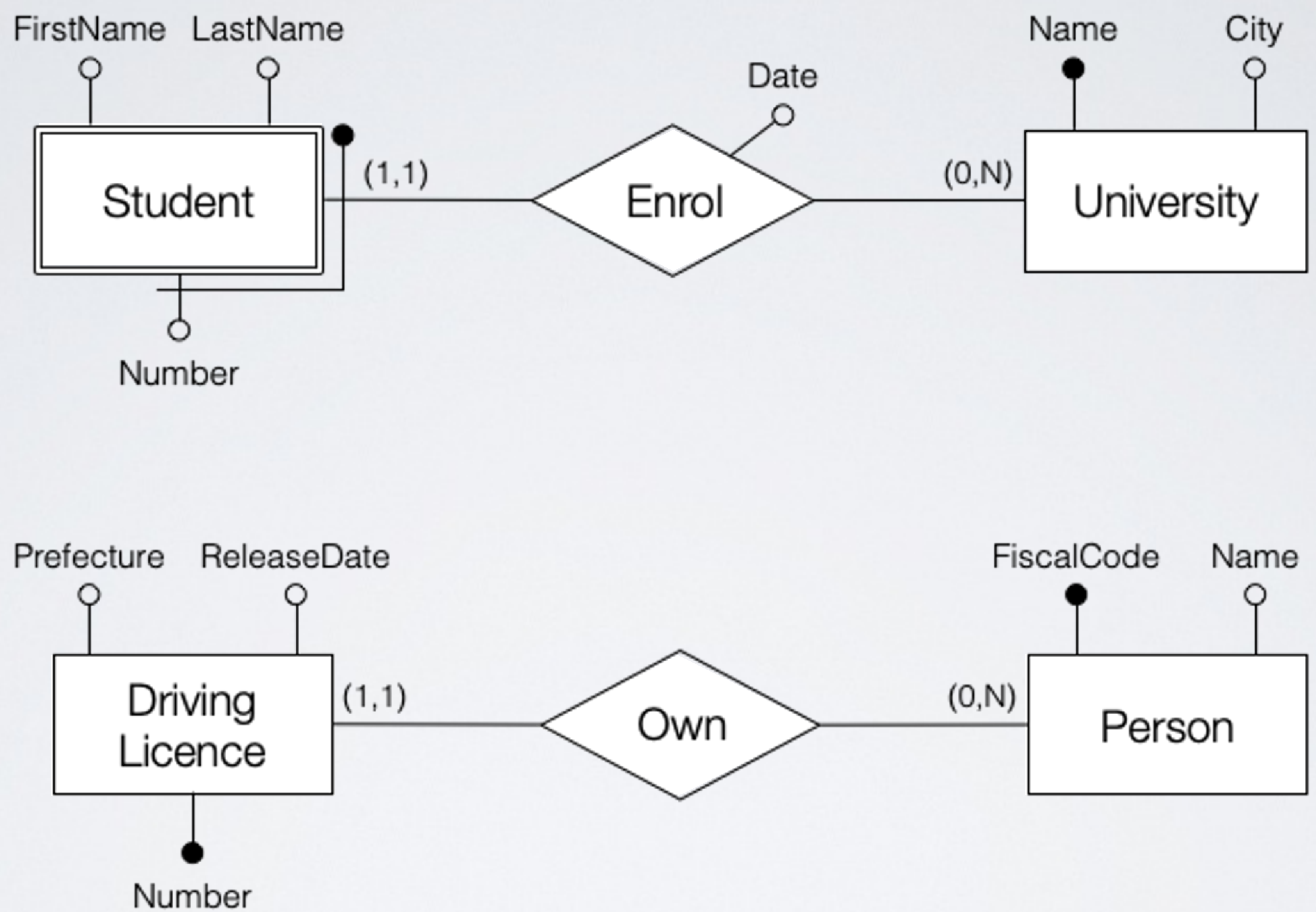
$$\text{instance(i, Enroll)} = \{(s_1, u_1), (s_2, u_1), (s_3, u_2)\}$$

$$\text{instance(i, Enroll.Date)} = \Big\{\big((s_1, u_1), \text{21-09-2010}\big), \big((s_2, u_1), \text{19-09-2011}\big), \big((s_3, u_2), \text{19-09-2011}\big)\Big\}$$

Weak Entity

- **Weak Entity:** entity types that do not have key attributes of their own
- Weak entities are identified by being related to specific entities from another entity type in combination with one of their attributes values (externally identified)
- Weak entities always have a total participation constraint with respect to their identifying relationship
- Not every existence dependency results in a weak entity

Weak Entity: Is it or Not?



External Constraints

- The ER schema allows us to grasp all the main types of relations and constraints among the data in the mini-world and provides us graphical means to express them

- However not all the constraints fall into the set of the pre-defined ones

- The documentation of an ER schema also contains all the additional constraints, called **external (to the ER diagram) constraints**, expressed as
  - mathematical notation
  - natural language assertion, as un-ambiguous as possible

Documentation

Documenting ER Diagrams

- In addition to the ER diagram, the conceptual schema is documented by the so-called **data dictionary**
- The data dictionary consists of a set of tables describing:
  - Entities
  - Relationships
  - Attributes
  - External Constraints

Data Dictionary: Entities

Entity	Description	Attributes	Identifiers
Name	Description	Attribute names, domain, description	Attribute names
…	…	…	…

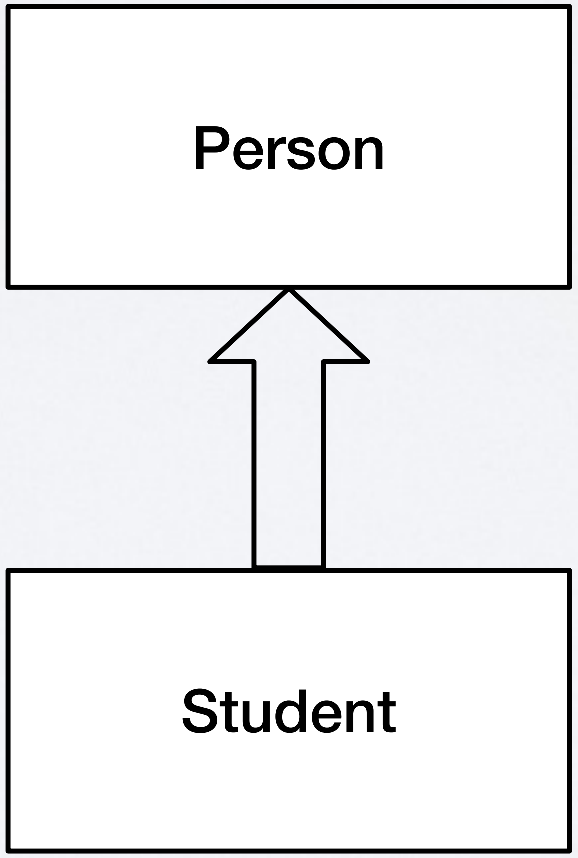
Data Dictionary: Relationships

Relationship	Description	Component Entities	Attributes
Name	Description	Entity Names	Attribute names, domain, description
…	…	…	…

Generalization and Specialization

Sub-classes and super-classes: the IS-A relation

The **IS-A relation (subset relation)** is defined between two entities, called **superclass** and **subclass**, and it means that **each instance of the subclass** is also an **instance of the superclass**.



Extensional Semantics of the IS-A Relation

- At extensional level, the **IS-A** relation requires that, in each instance **i** of a schema **S** such that

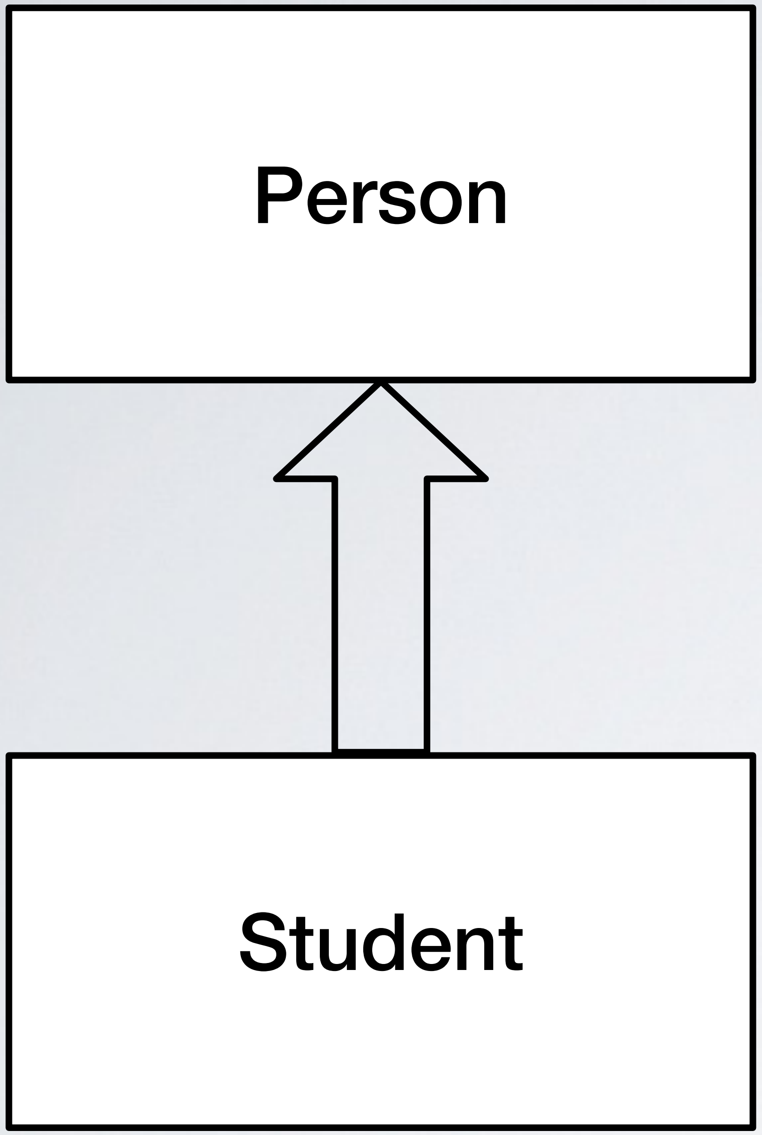
  $$\mathbf{e_1 \text{ IS-A } e_2}$$

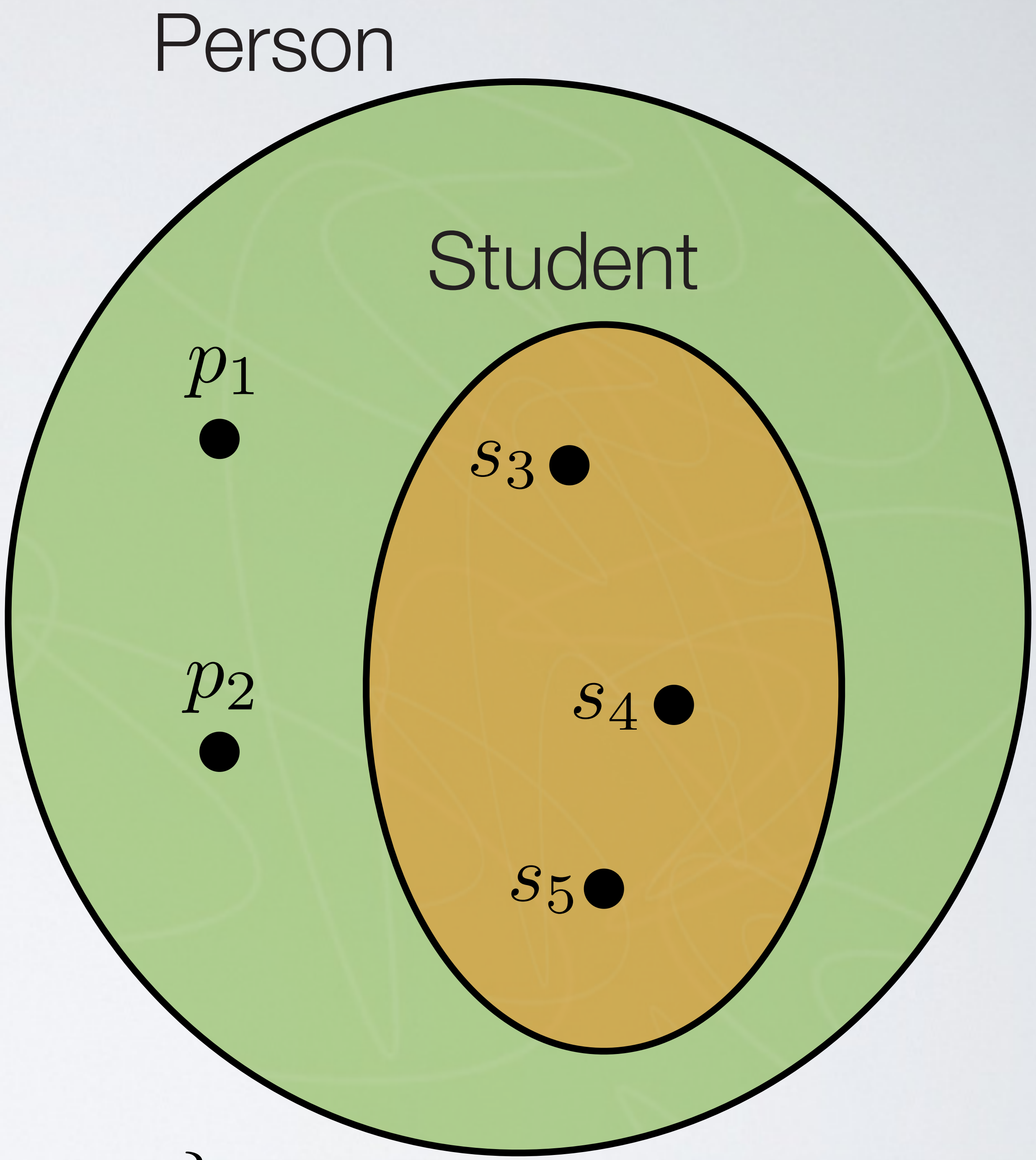
  the following constraint is complied with

  $$\text{instance}(\mathsf{i}, \mathsf{e}_1) \subseteq \text{instance}(\mathsf{i}, \mathsf{e}_2)$$

- From the definition, it follows that the IS-A relation is **reflexive** and **transitive** (but not symmetric)

Example of Extensional Semantics of the IS-A Relation

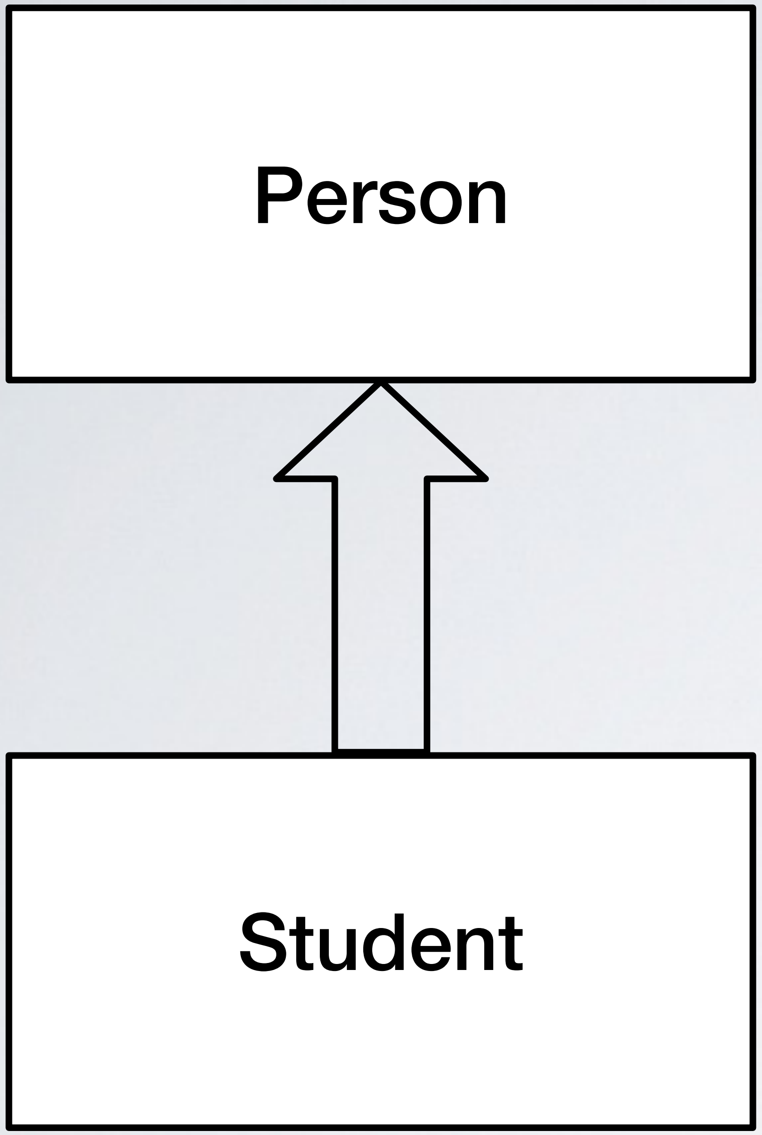


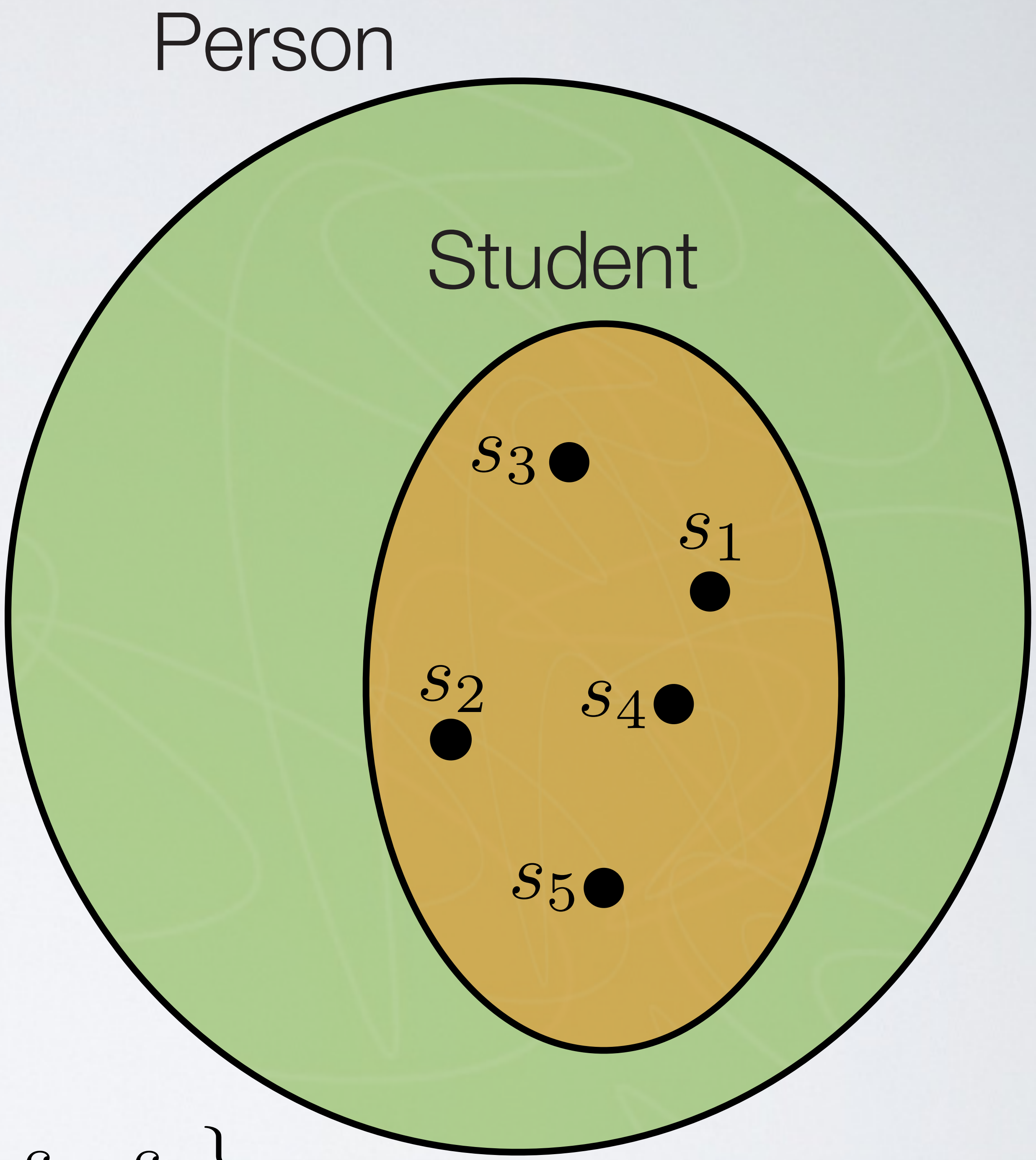


$$\text{instance(i, Person)} = \{p_1, p_2, s_3, s_4, s_5\}$$

$$\text{instance(i, Student)} = \{s_3, s_4, s_5\}$$

Example of Extensional Semantics of the IS-A Relation

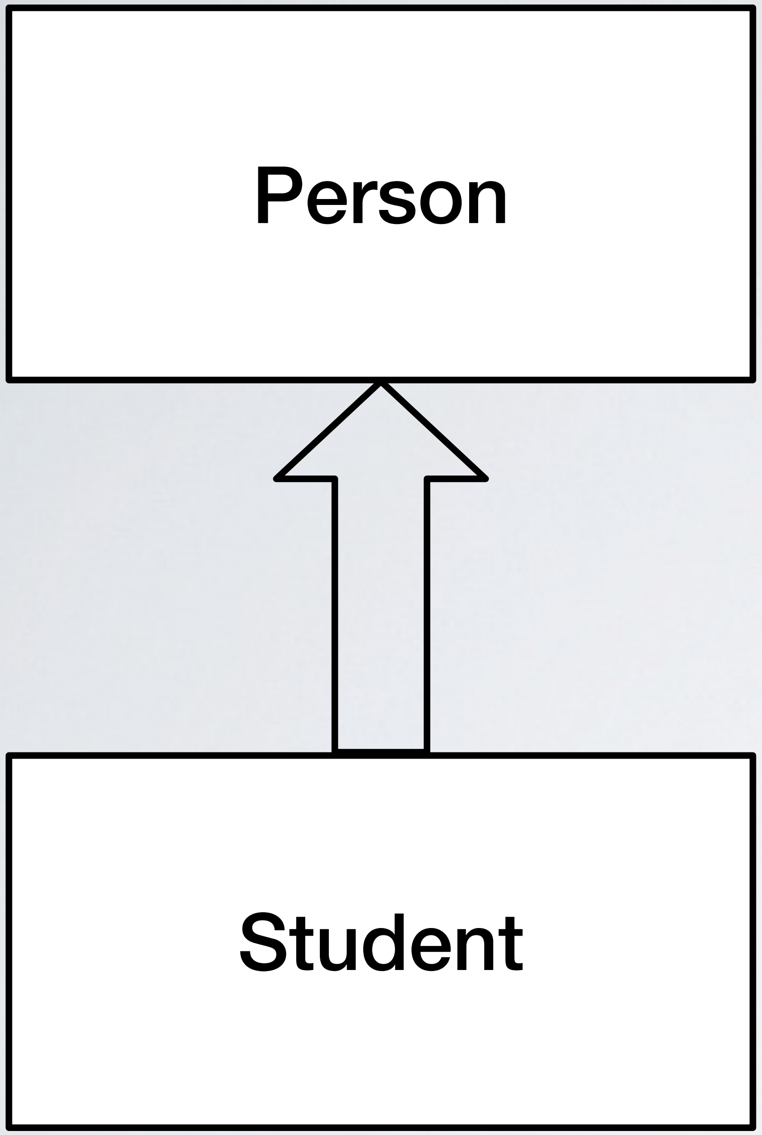


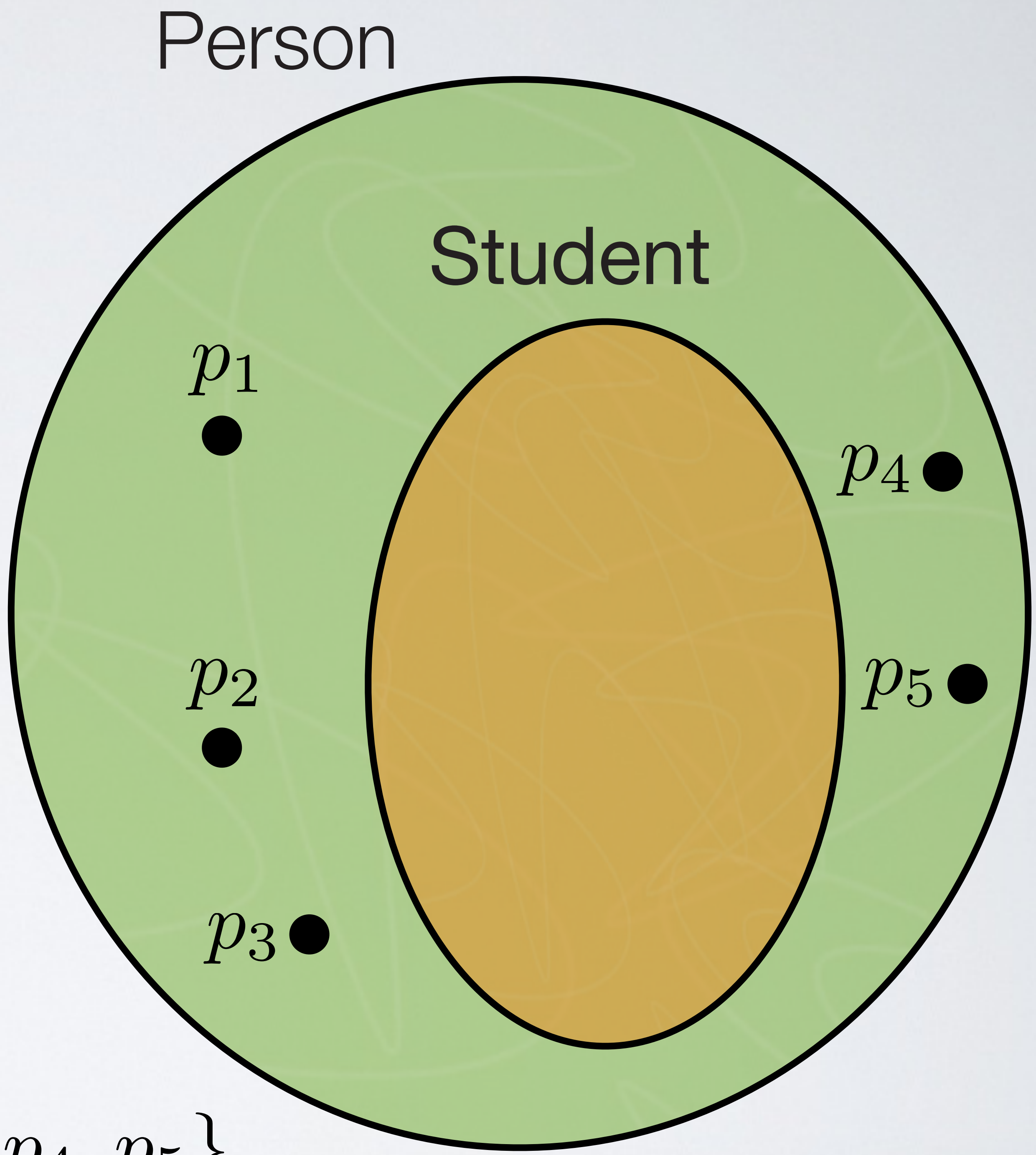


$$\text{instance}(\mathrm{i}, \mathrm{Person}) = \{s_1, s_2, s_3, s_4, s_5\}$$

$$\text{instance}(\mathrm{i}, \mathrm{Student}) = \{s_1, s_2, s_3, s_4, s_5\}$$

Example of Extensional Semantics of the IS-A Relation

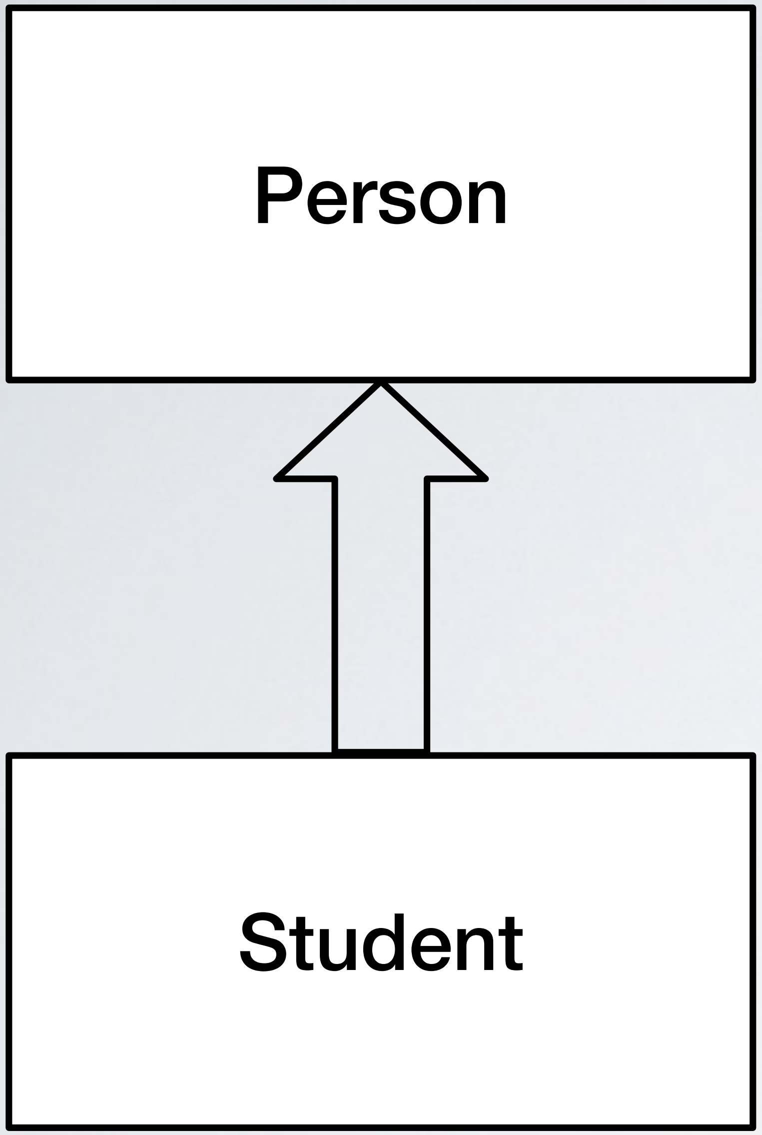


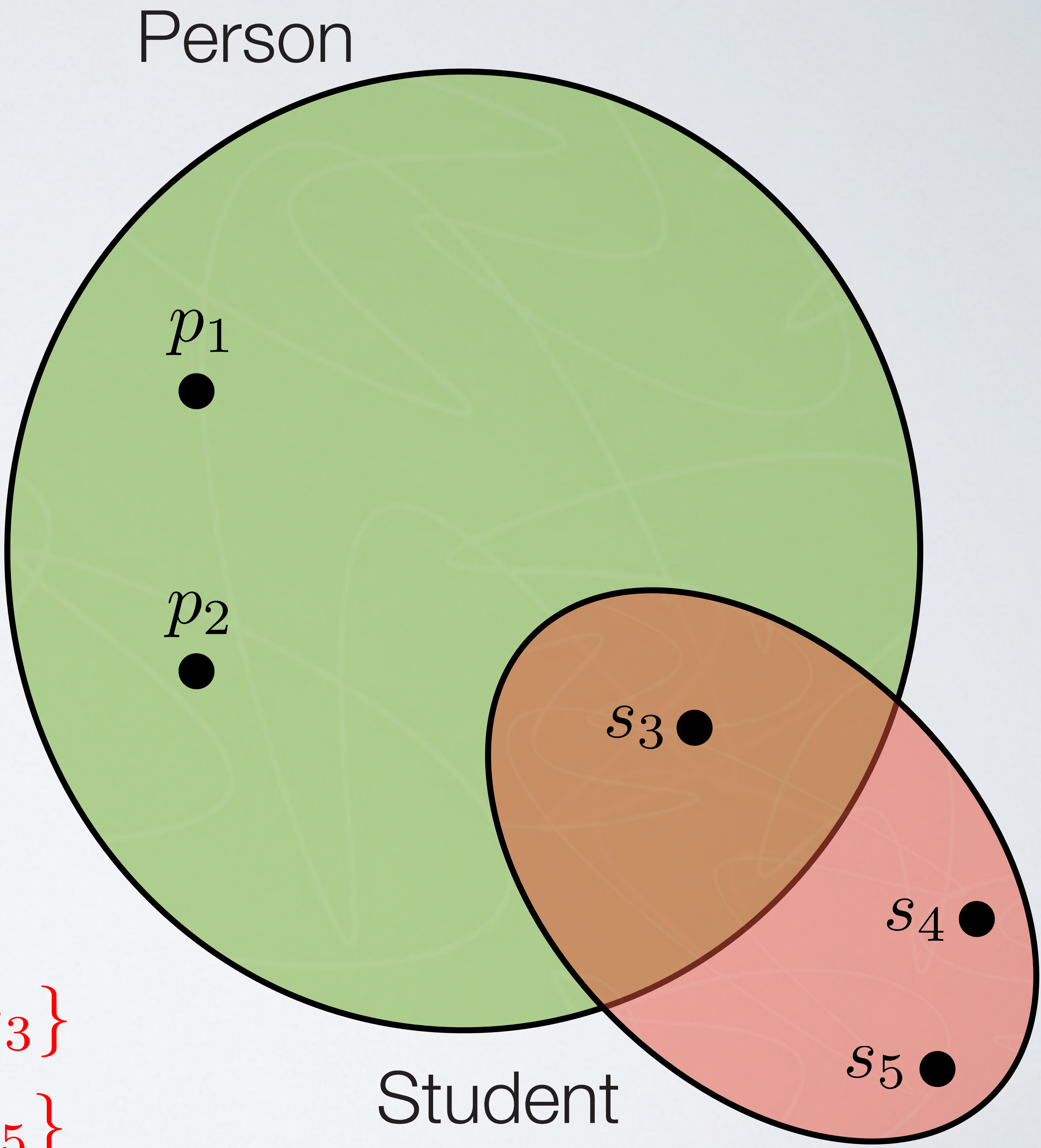


$$\text{instance}(\mathsf{i}, \mathsf{Person}) = \{p_1, p_2, p_3, p_4, p_5\}$$

$$\text{instance}(\mathsf{i}, \mathsf{Student}) = \{\}$$

Example of Extensional Semantics of the IS-A Relation

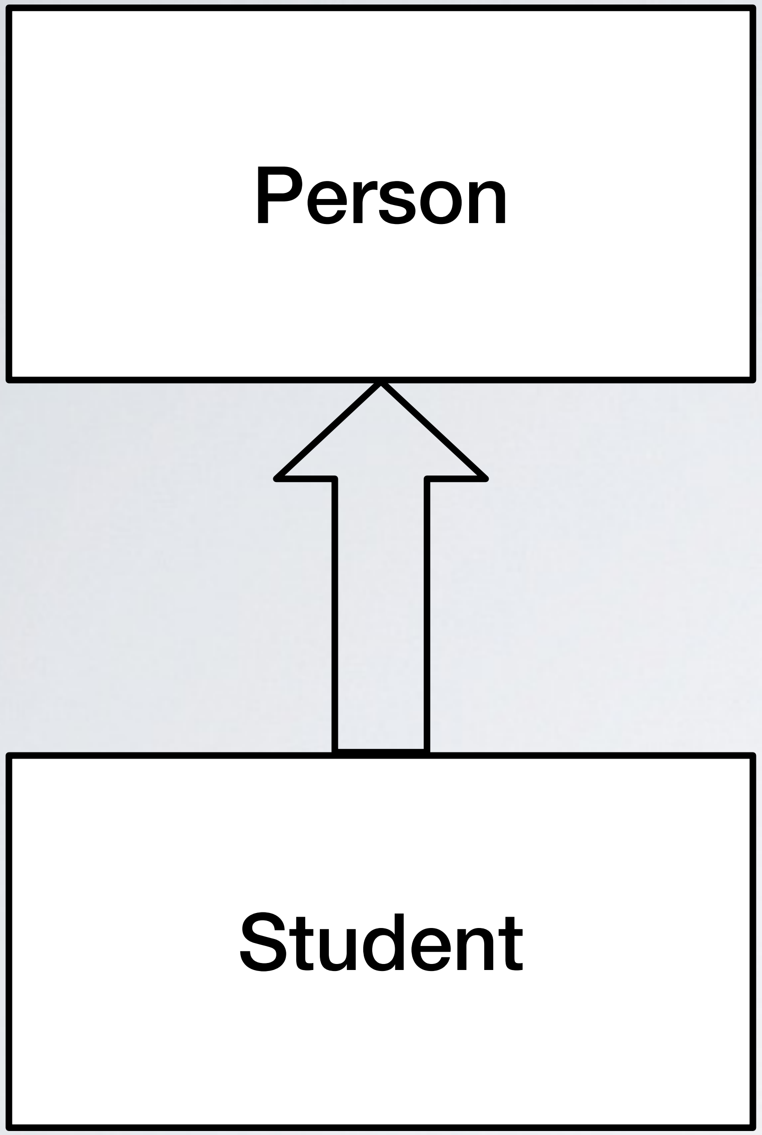


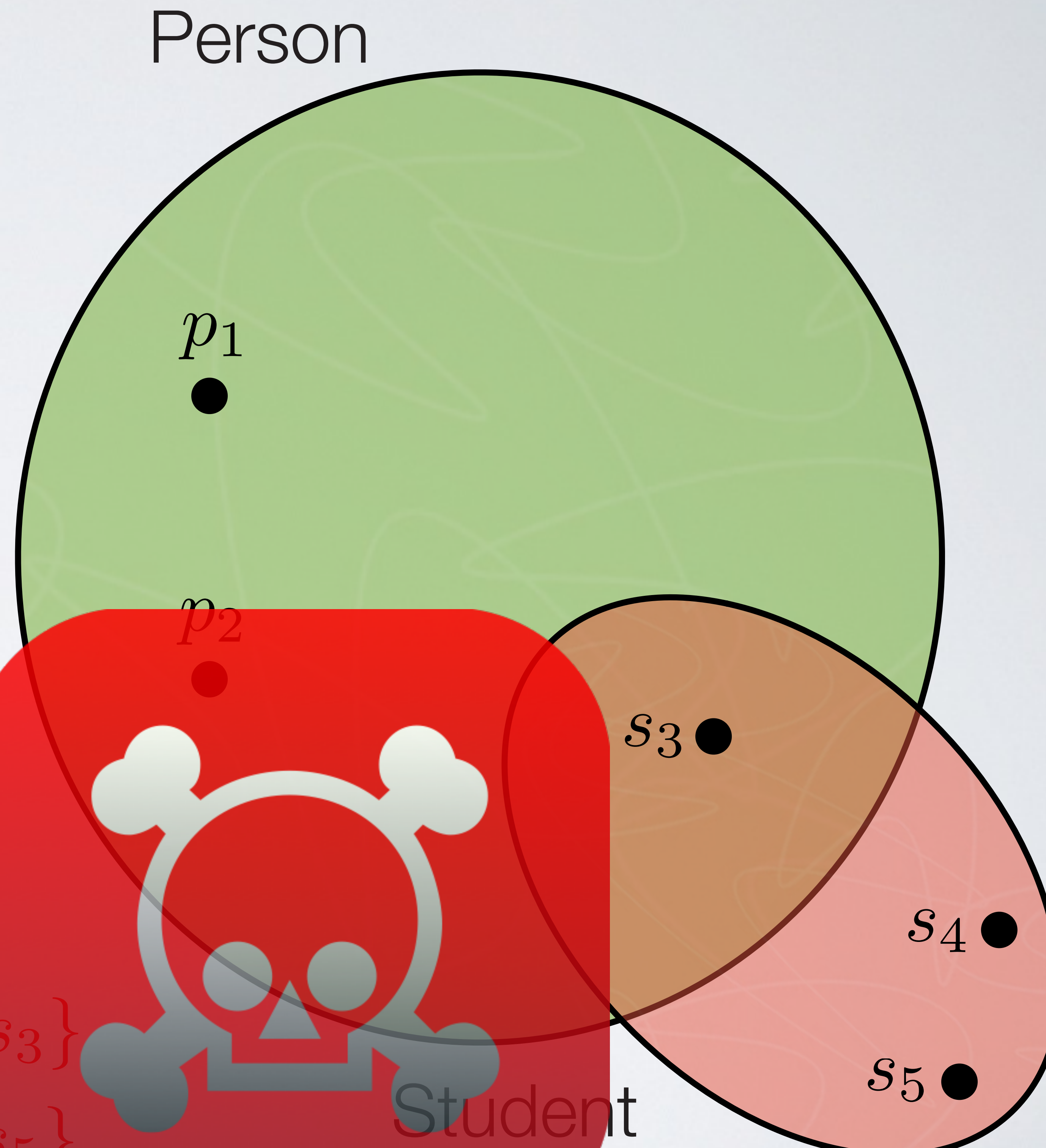


$$\text{instance}(\mathsf{i}, \mathsf{Person}) = \{p_1, p_2, s_3\}$$

$$\text{instance}(\mathsf{i}, \mathsf{Student}) = \{s_3, s_4, s_5\}$$

Example of Extensional Semantics of the IS-A Relation





$$\text{instance}(\text{i}, \text{Person}) = \{p_1, p_2, s_3\}$$

$$\text{instance}(\text{i}, \text{Student}) = \{s_3, s_4, s_5\}$$

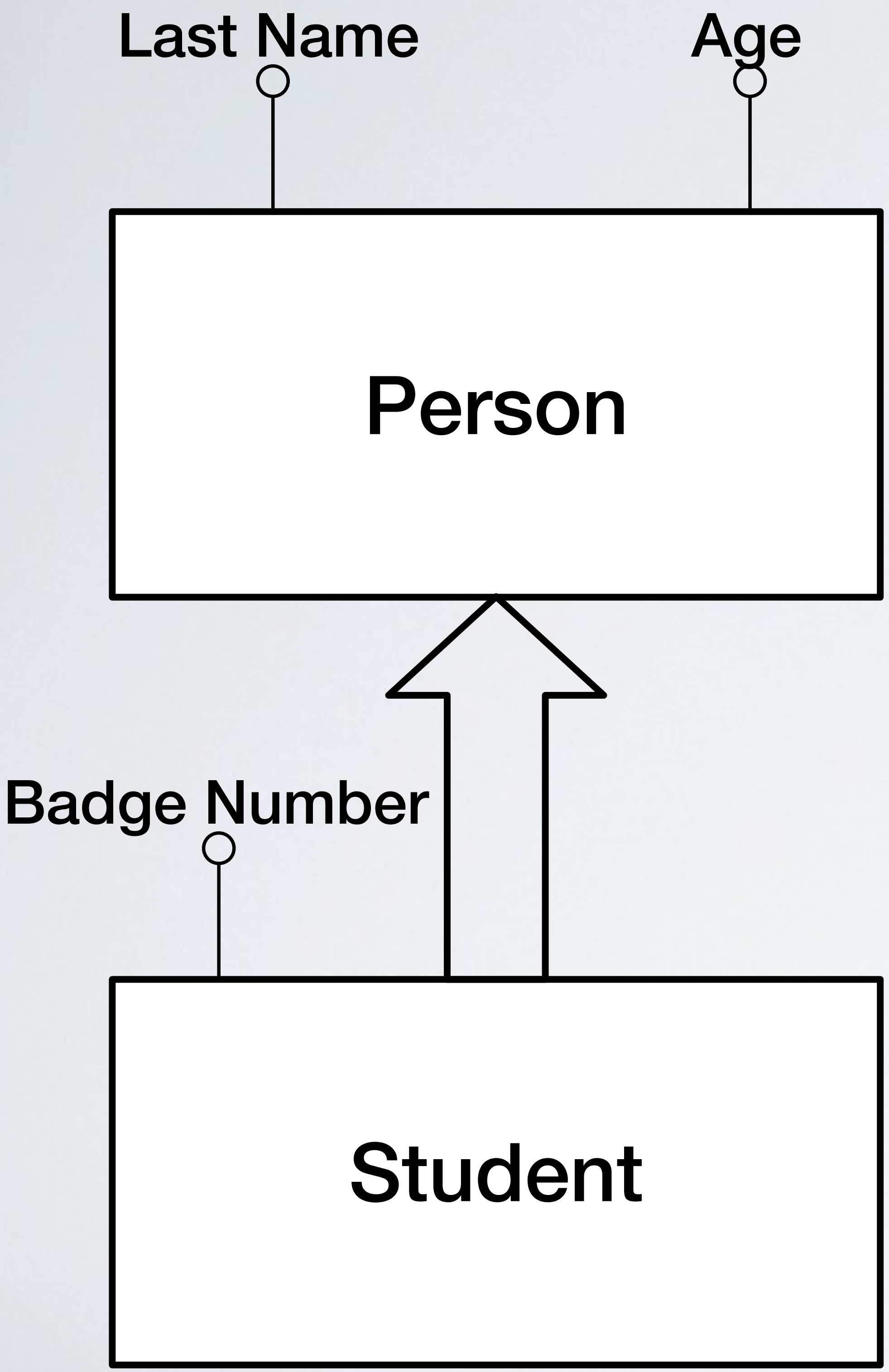
Inheritance Principle

Each property of the superclass is also a property of the subclass and it is not represented (twice) in the ER diagram. The subclass may have additional properties.

By "property" we mean

- Attributes
- Relationships
- Integrity constraints

Example of Attribute Inheritance

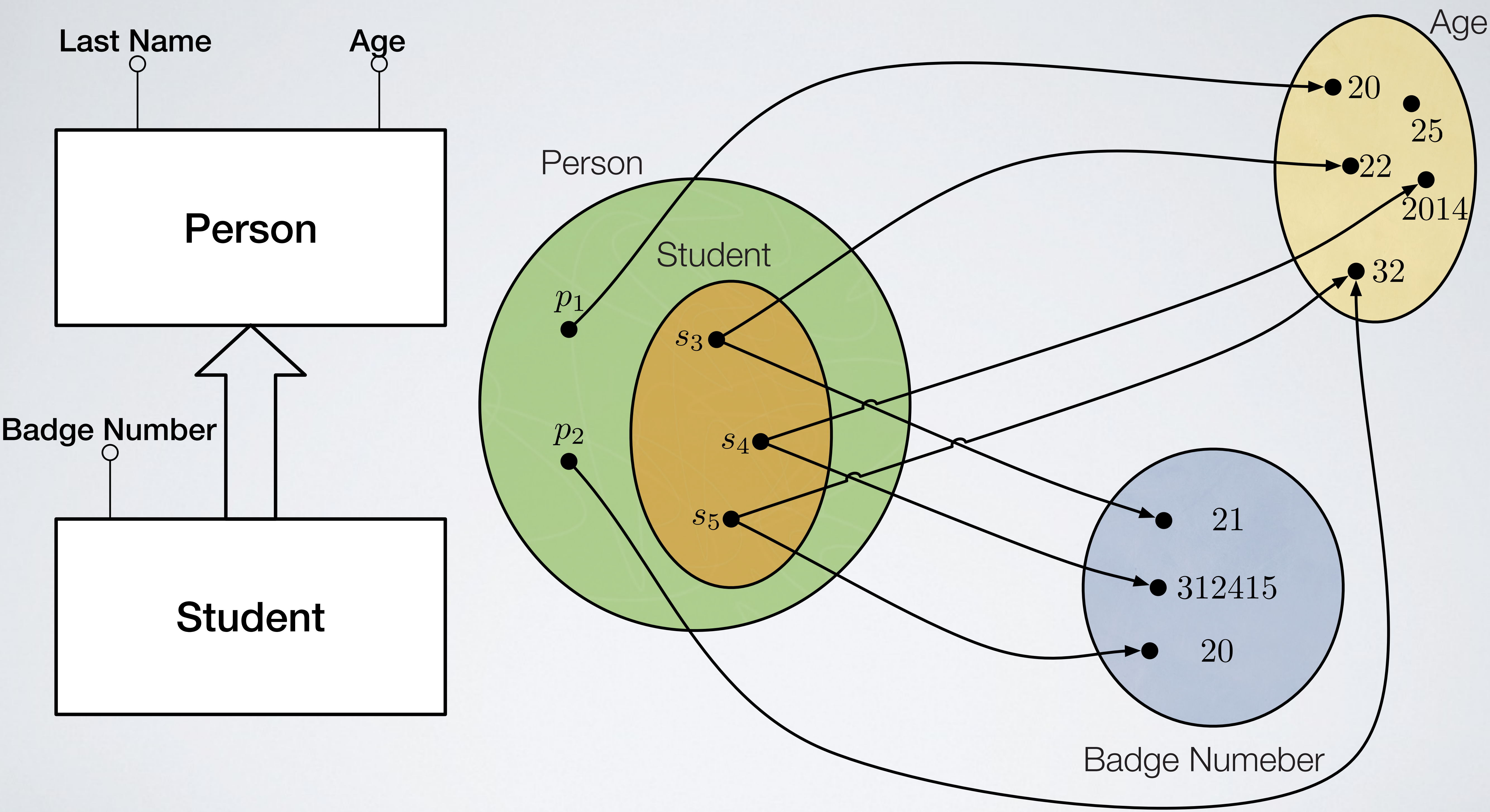


- Each instance of **Person** has an **Age**
- Each instance of **Student** is an instance of **Person**

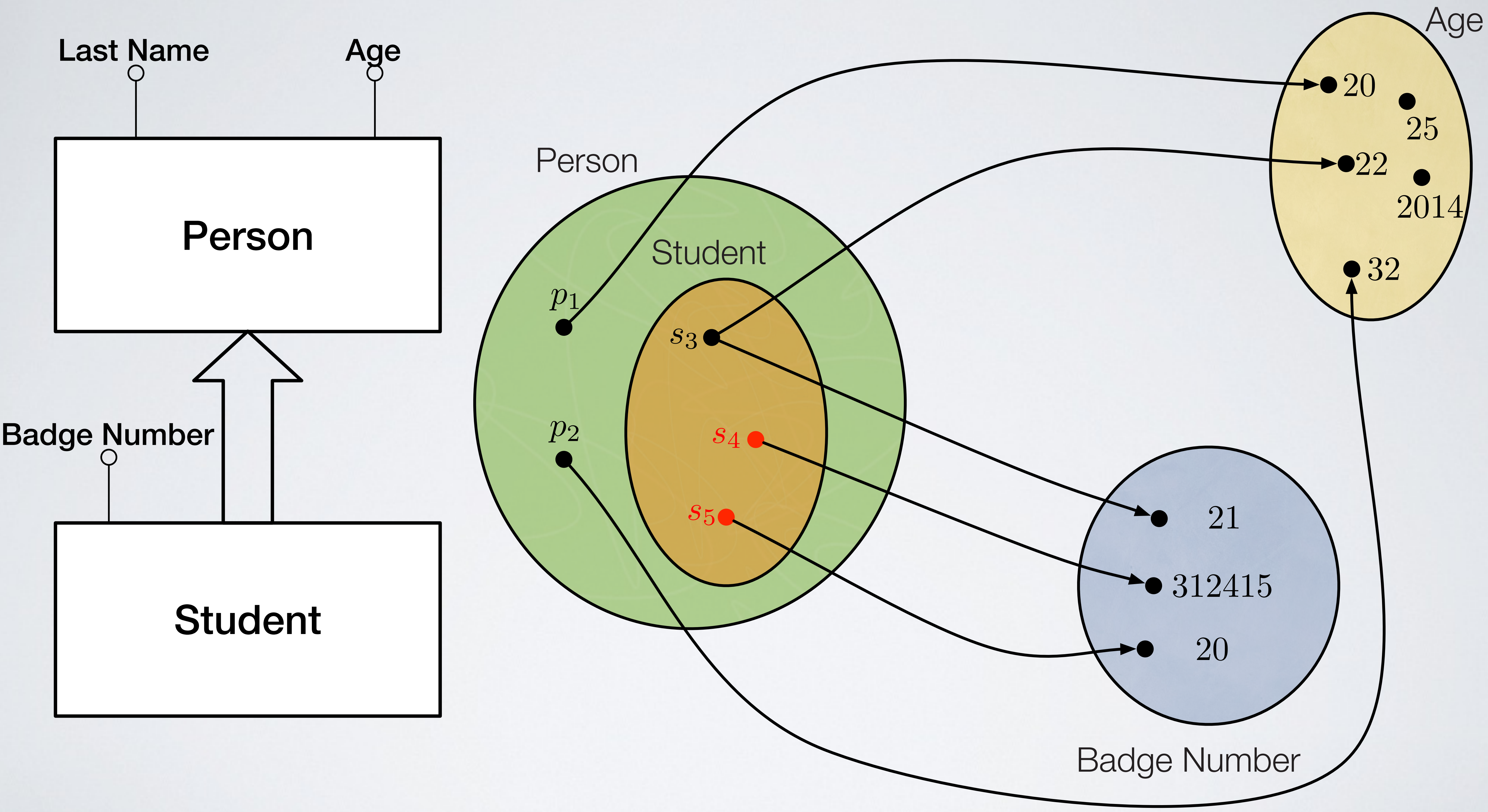
therefore

- Each instance of **Student** has an **Age**

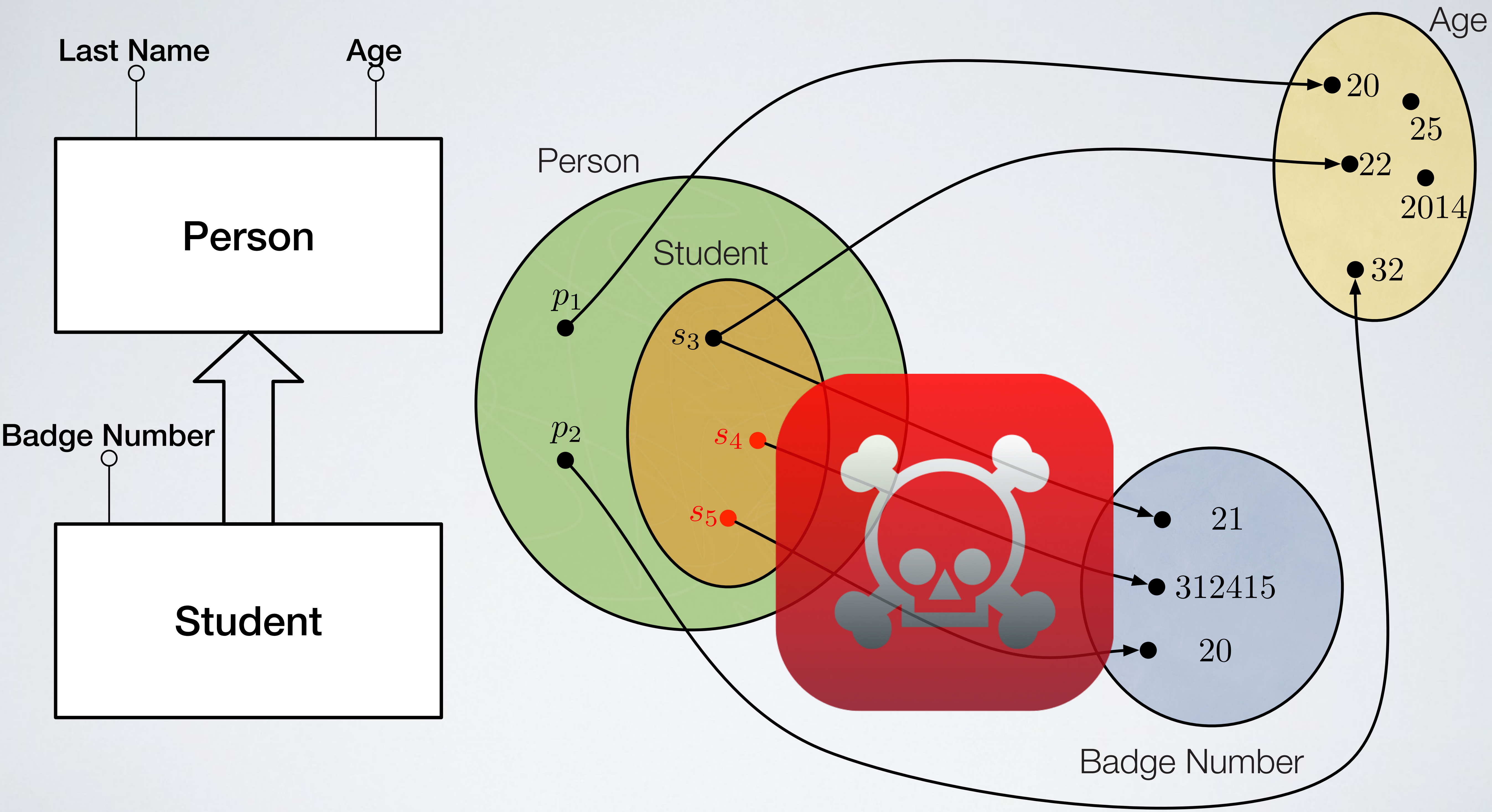
Example of Attribute Inheritance



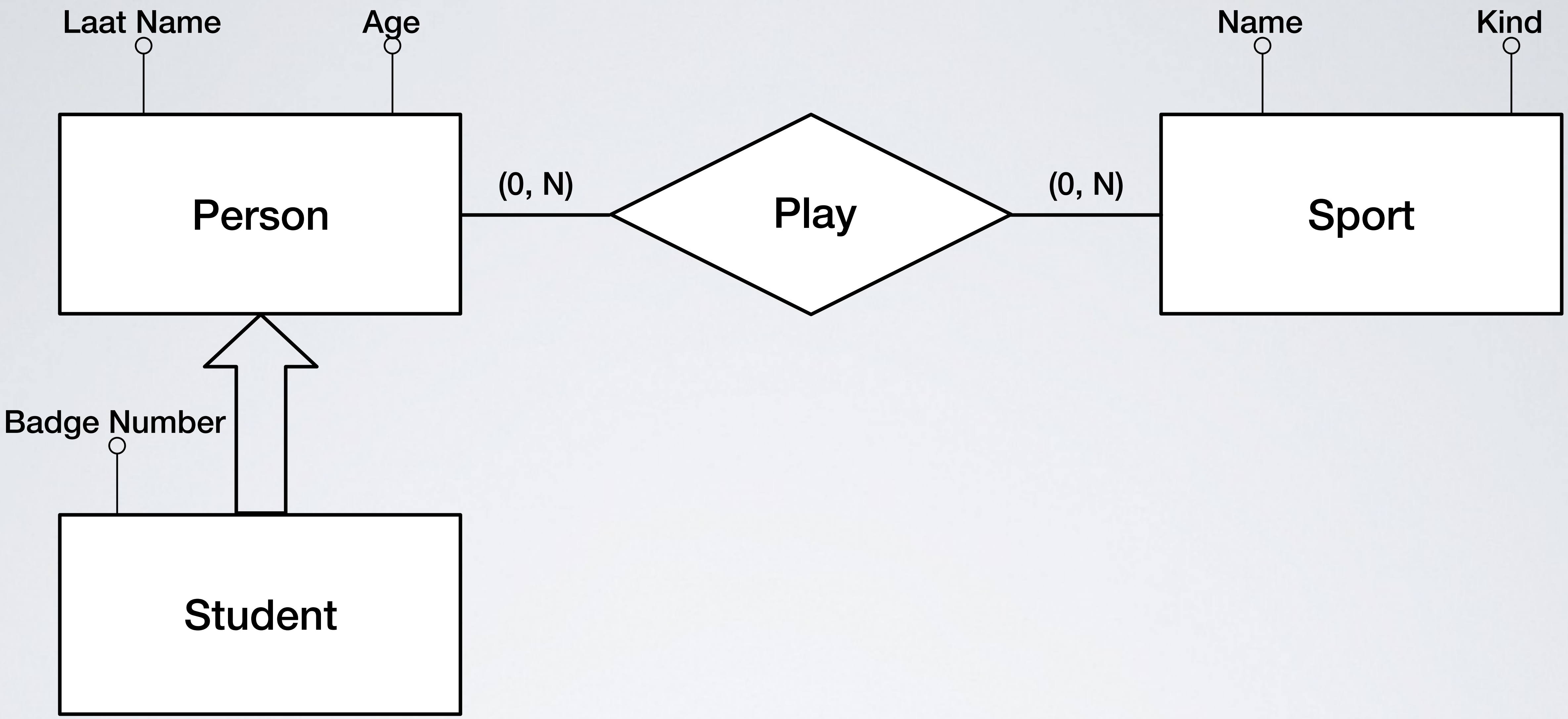
Example of Attribute Inheritance



Example of Attribute Inheritance



Example of Relationship Inheritance

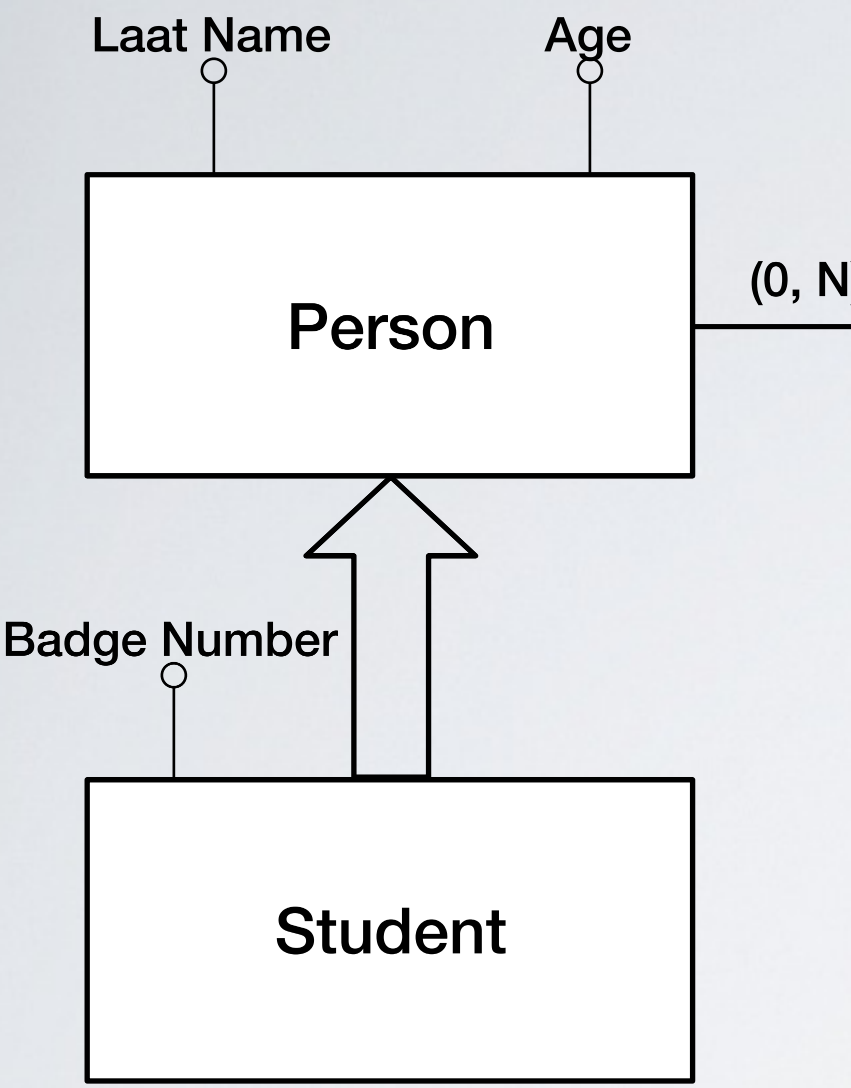


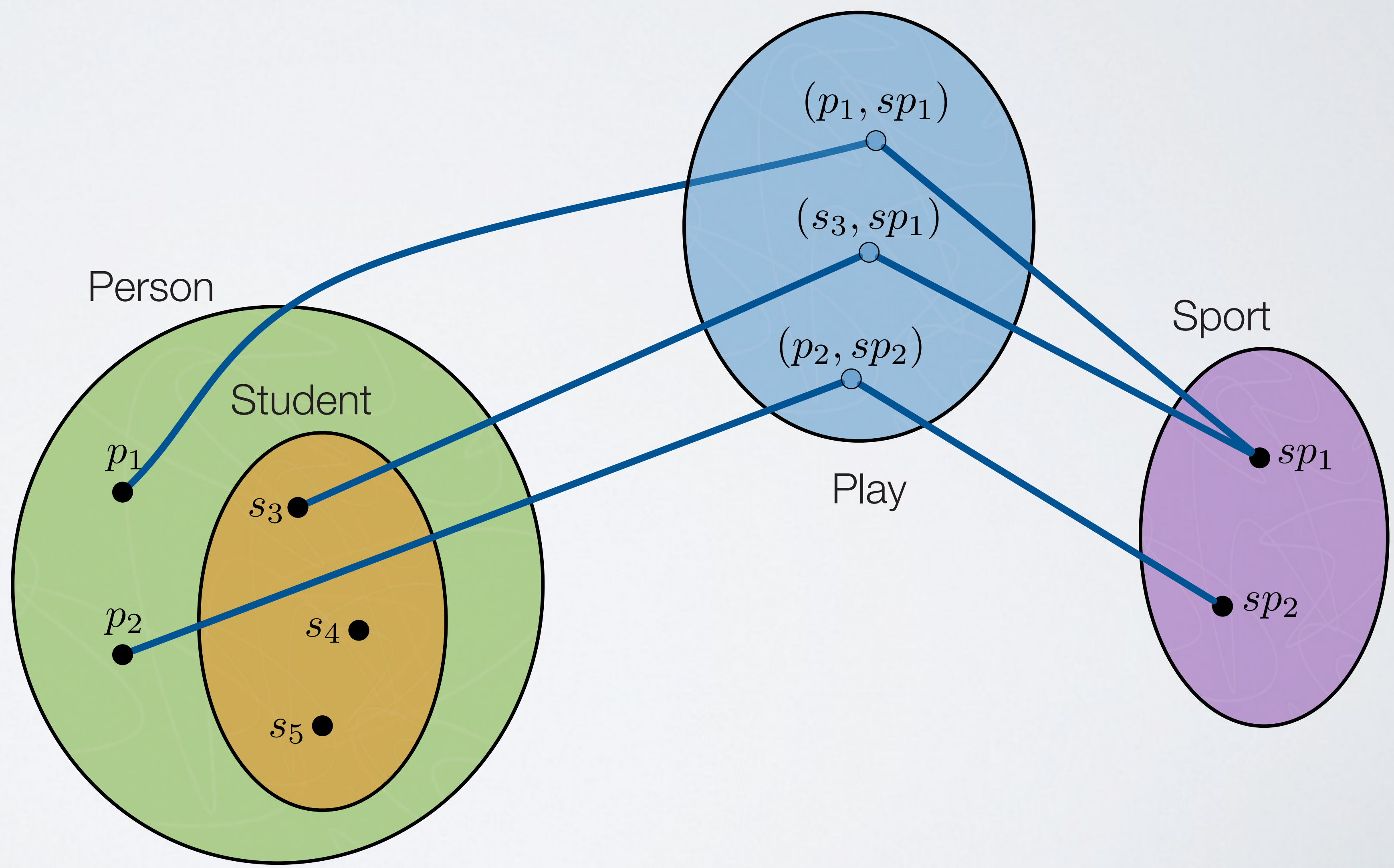
- Each instance of **Person** can take part to any number of instances of **Play**
- Each instance of **Student** is an instance of **Person**

therefore

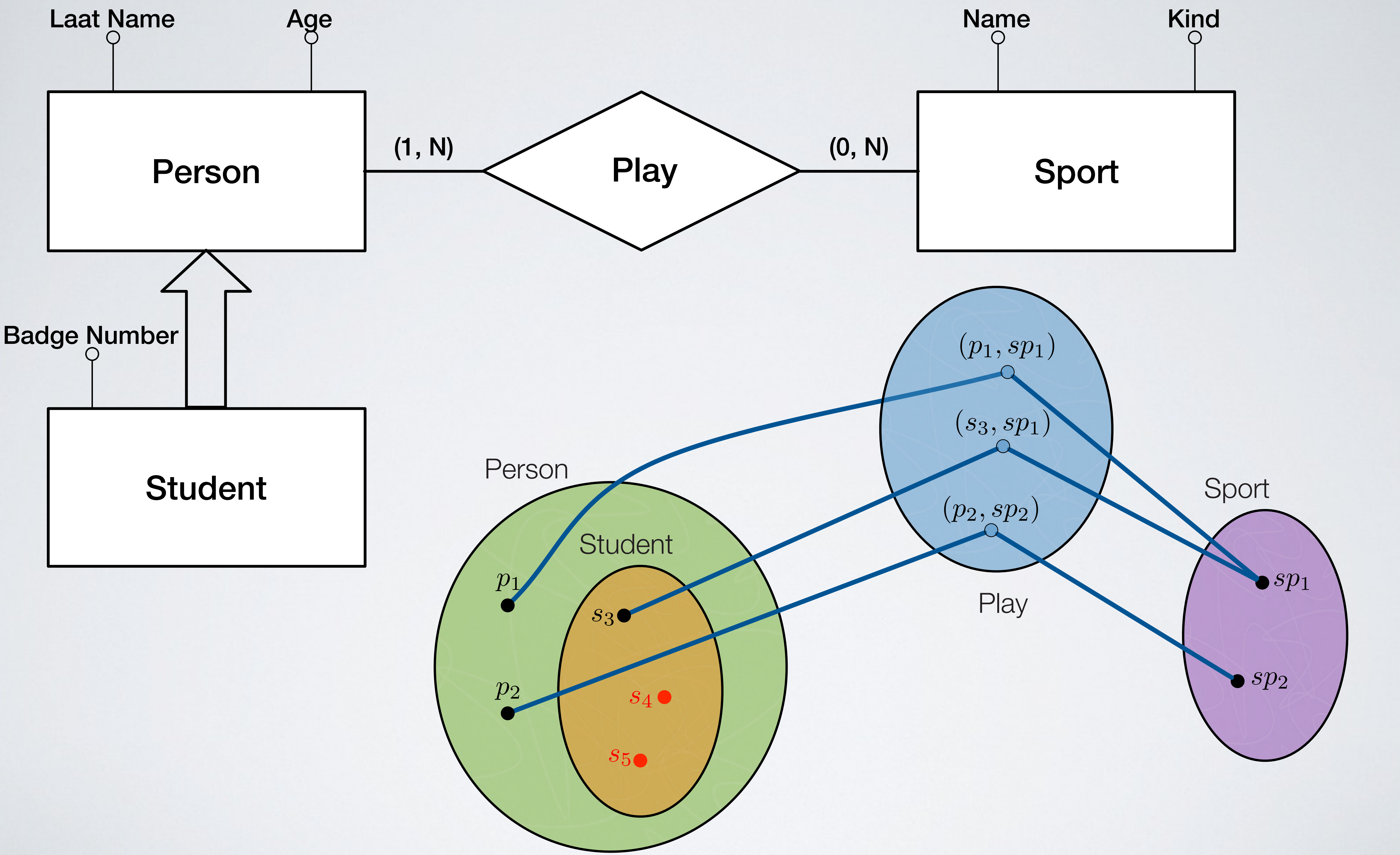
- Each instance of **Student** can take part to any number of instances of **Play**

Example of Relationship Inheritance

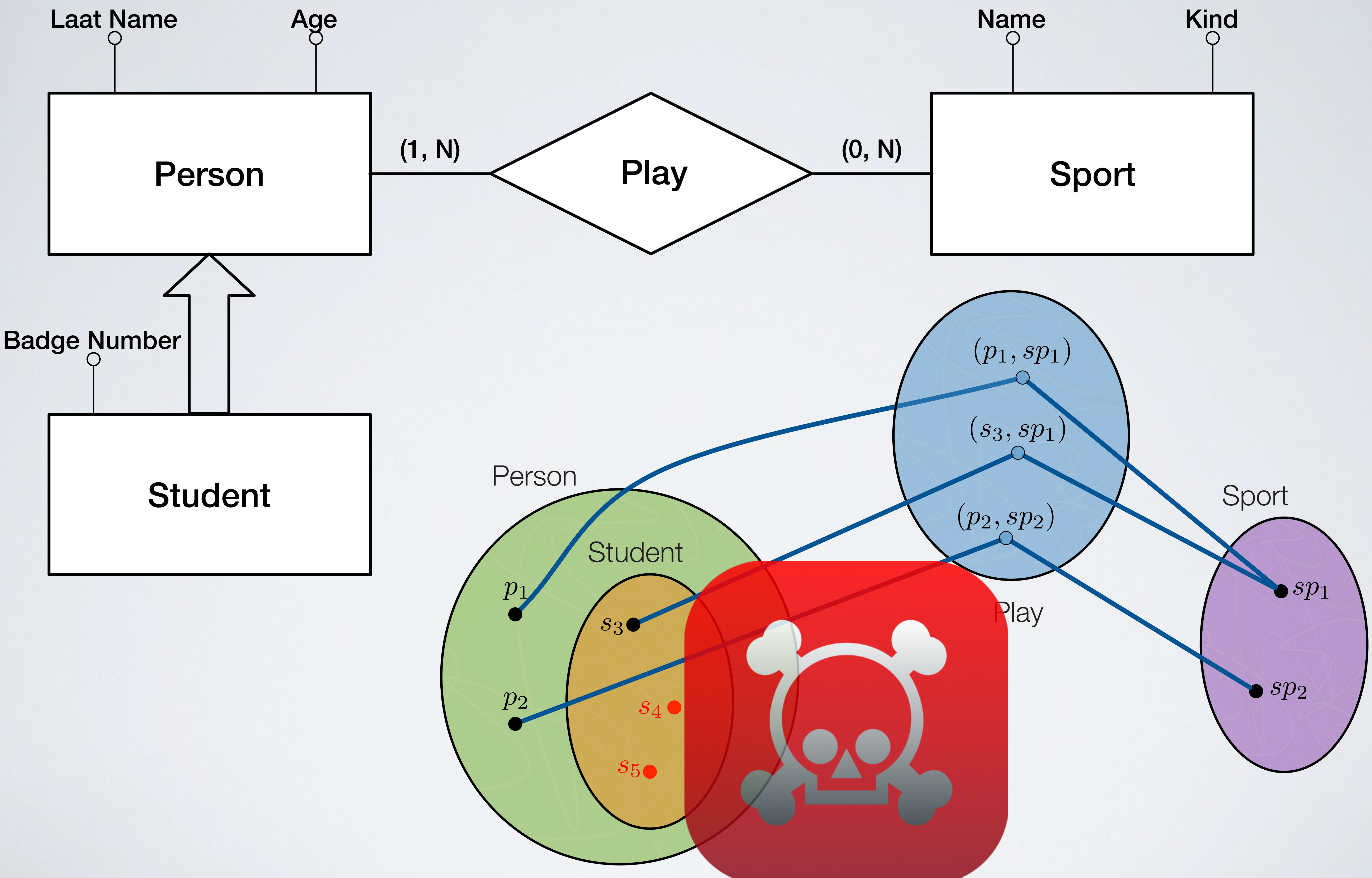




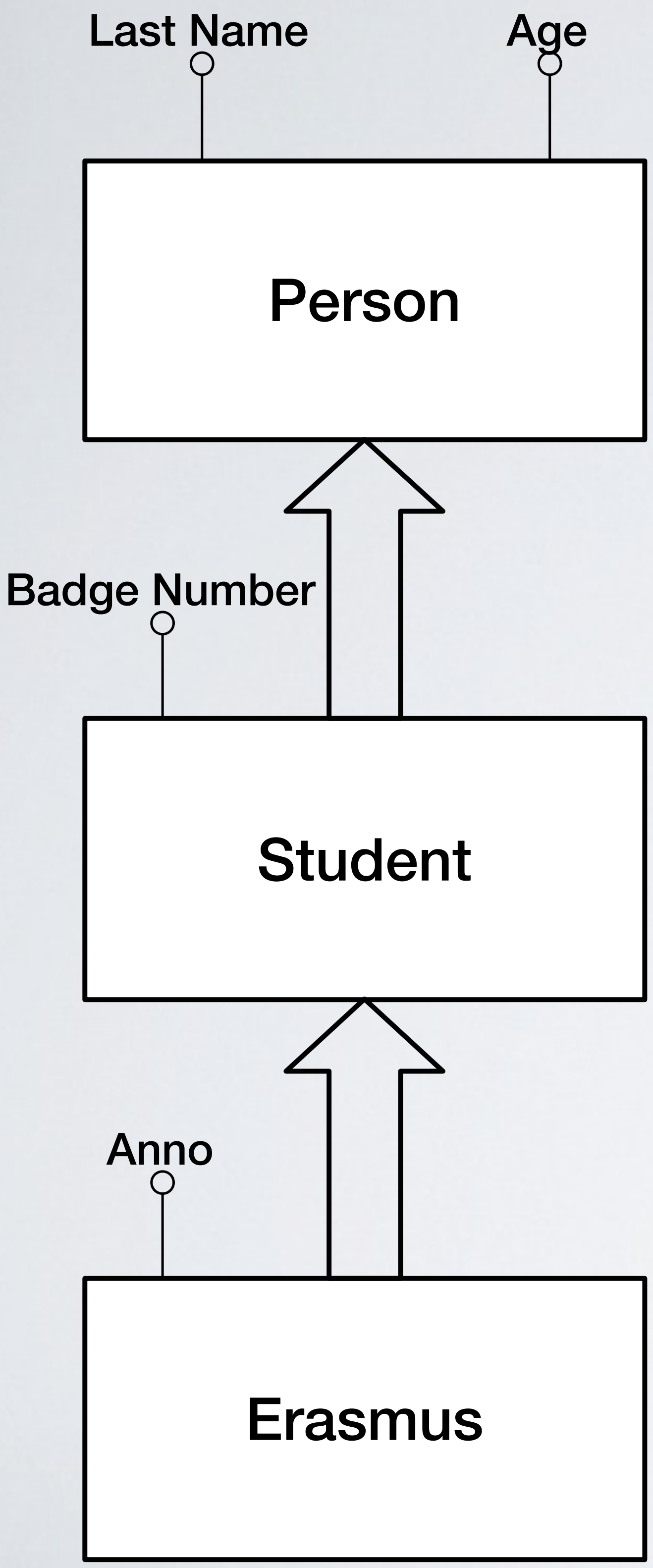
Example of Relationship Inheritance



Example of Relationship Inheritance



Inheritance of IS-A Relation: Transitivity



- The IS-A relation is inherited too. This means that the IS-A relation is transitive (in addition to being reflexive)

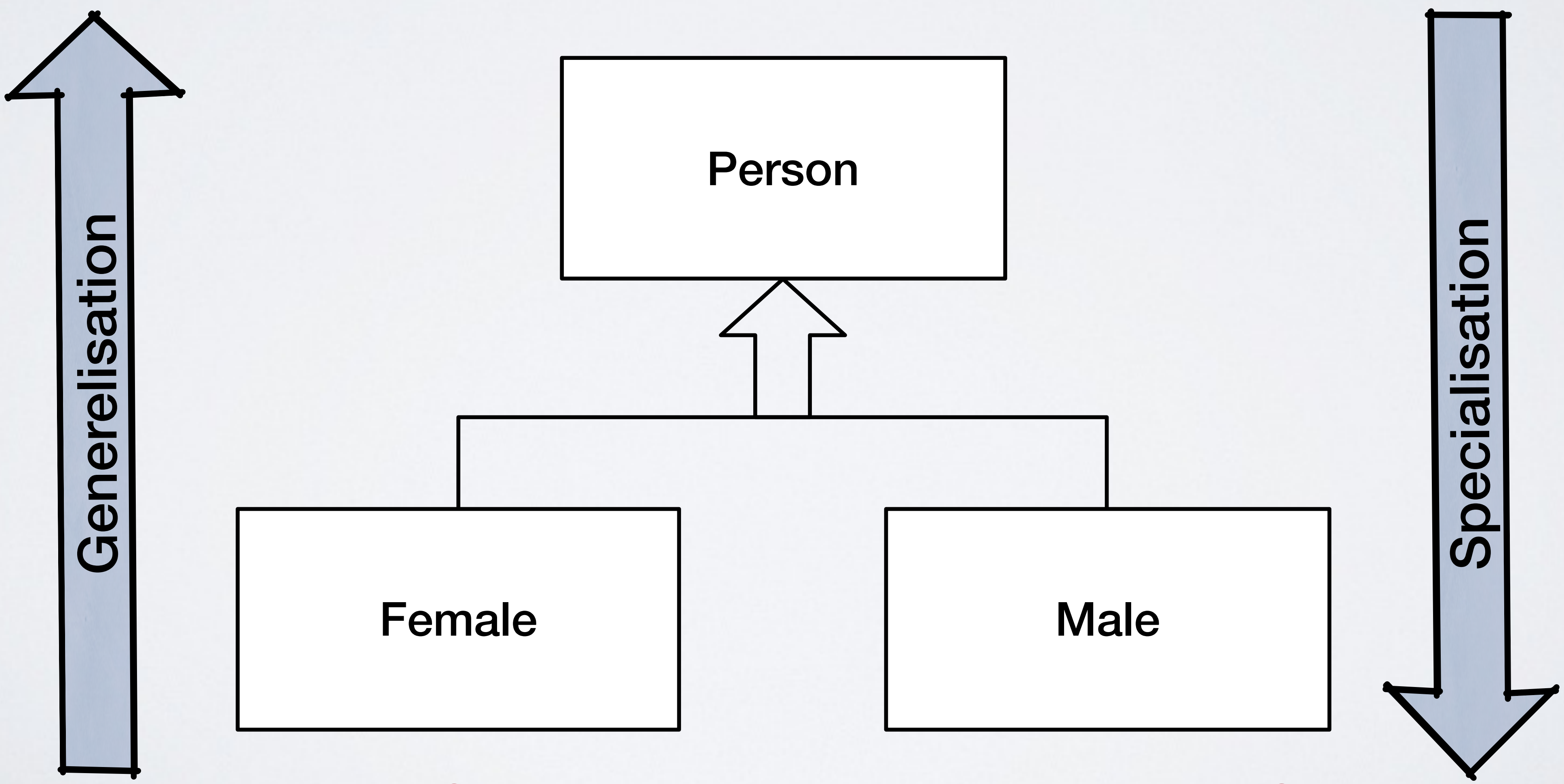
- Each instance of **Student** is an instance of **Person**
- Each instance of **Erasmus** is an instance of **Student**

therefore

- Each instance of **Erasmus** is an instance of **Person**

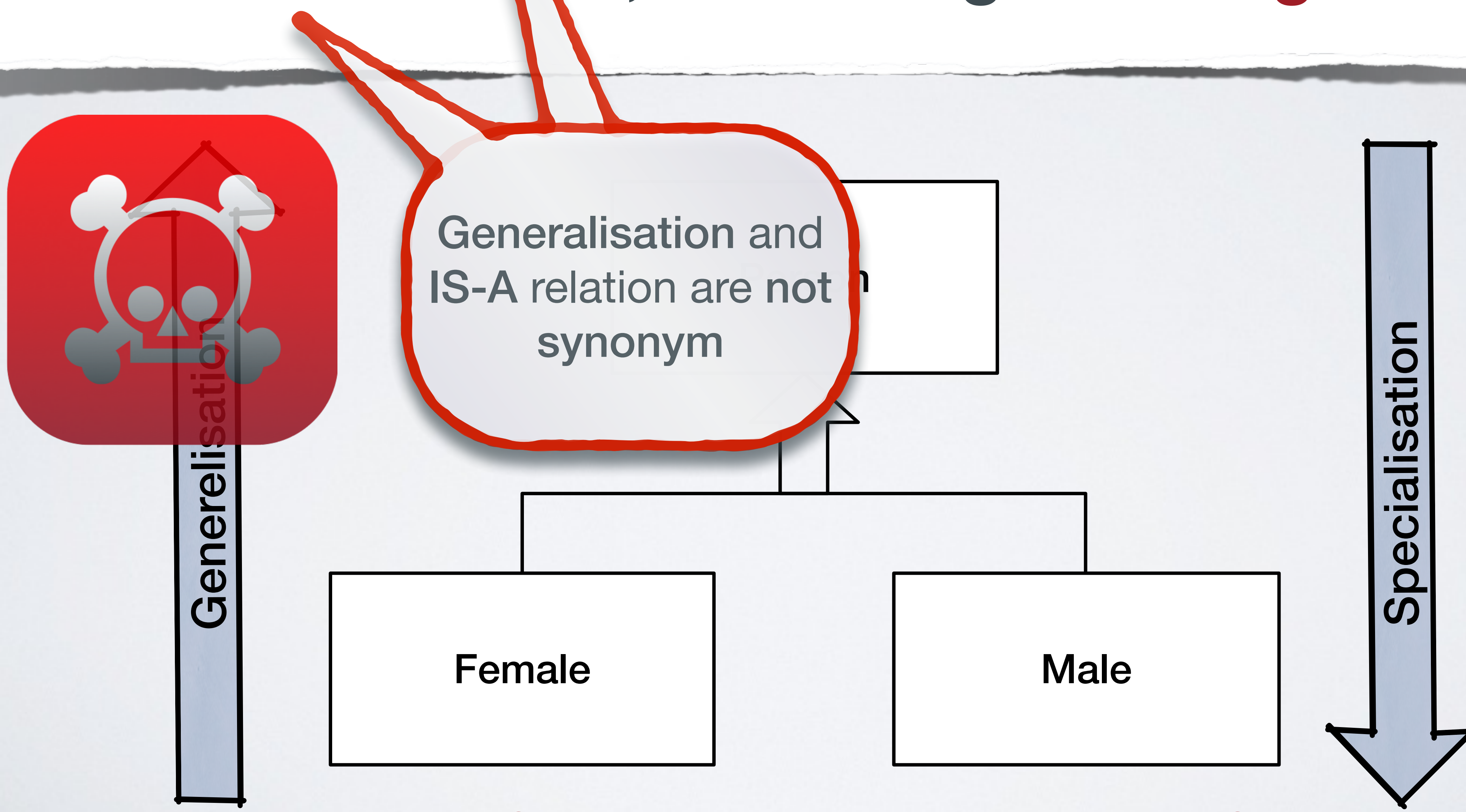
Generalisation

The generalisation consists of a super class extended by several subclasses, that is we have several IS-A relations, according to a single criterion.



Generalisation

The **generalisation** consists of a **super class** extended by **several subclasses**, that is we have several **IS-A** relations, according to a **single criterion.**



Extensional Semantics of Generalisation

- If in a schema **S** a generalisation between superclass **p** and subclasses **f_1**, **f_2**, …, **f_n** is defined, then in each instance **i** of the schema **S** it must hold:

$$\mathrm{instance}(\mathrm{i}, \mathrm{f}_1) \subseteq \mathrm{instance}(\mathrm{i}, \mathrm{p})$$
$$\mathrm{instance}(\mathrm{i}, \mathrm{f}_2) \subseteq \mathrm{instance}(\mathrm{i}, \mathrm{p})$$
$$\dots$$
$$\mathrm{instance}(\mathrm{i}, \mathrm{f}_n) \subseteq \mathrm{instance}(\mathrm{i}, \mathrm{p})$$

Generalisation: inheritance and constraints

- The **inheritance principle** holds also in the case of generalisations
  - each property (attributes, relationships, integrity constraints) of the superclass is also a property of all the subclasses and it is not drawn in the ER diagram. Subclasses may have additional properties.

- Two constraints may hold for generalisations
  - **completeness**: each instance of the superclass must be an instance of at least one subclass

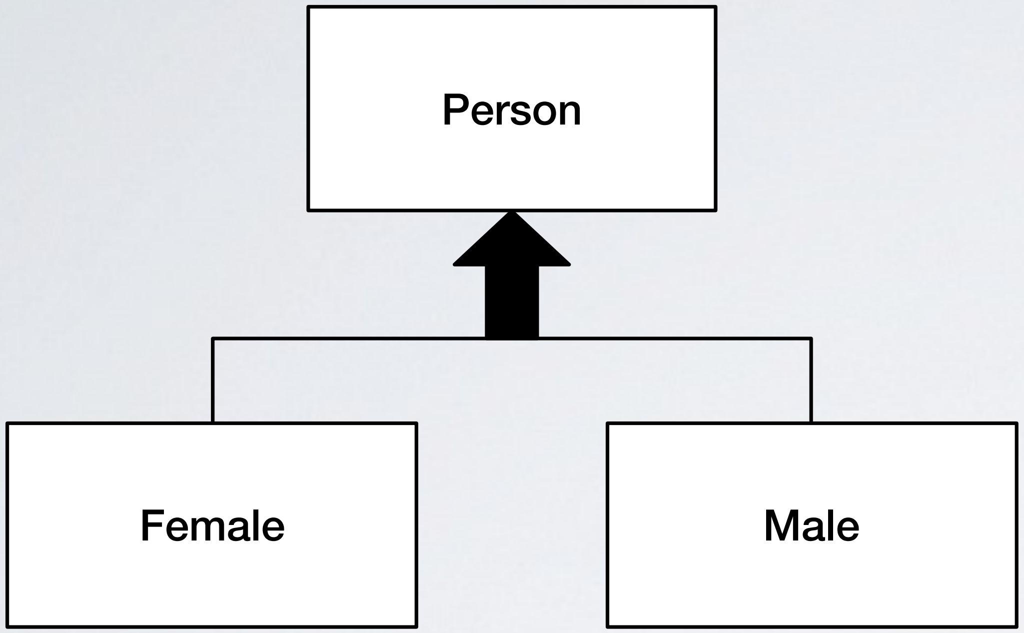
$$\text{instance}(\mathsf{i}, \mathsf{f}_1) \cup \text{istanze}(\mathsf{i}, \mathsf{f}_2) \cup \ldots \cup \text{instance}(\mathsf{i}, \mathsf{f}_n) = \text{instance}(\mathsf{i}, \mathsf{p})$$

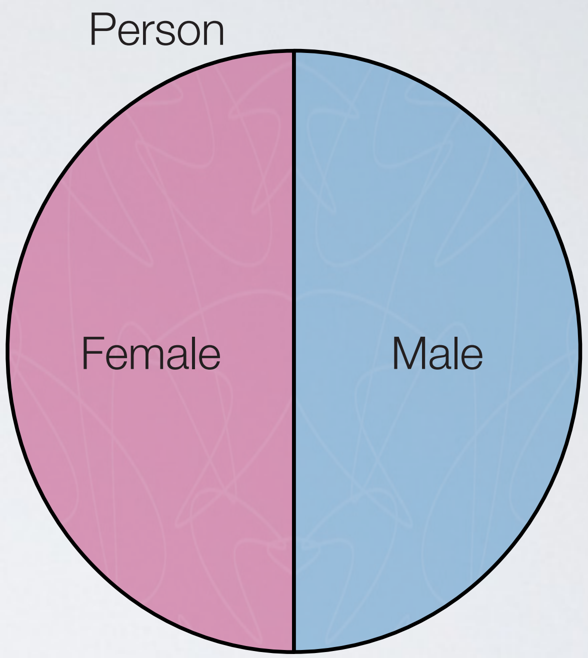
  - **disjointness**: each instance of the superclass can be a member of at most one of the subclasses

$$\text{instance}(\mathsf{i}, \mathsf{f}_i) \cap \text{instance}(\mathsf{i}, \mathsf{f}_j) = \emptyset, \quad 1 \leq i, j \leq n, \; i \neq j$$

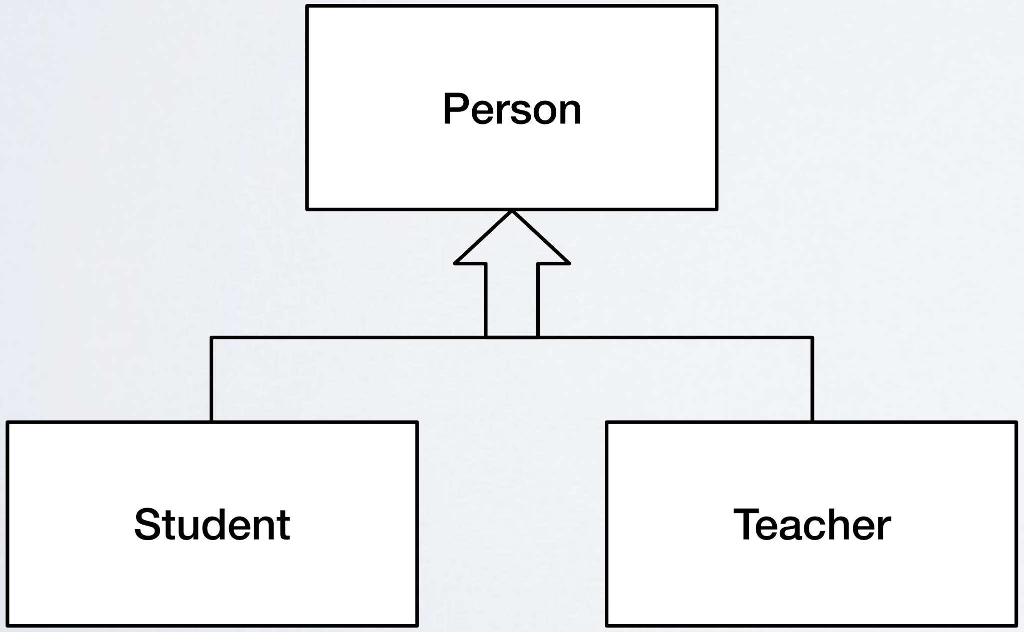
Completness and Disjointness

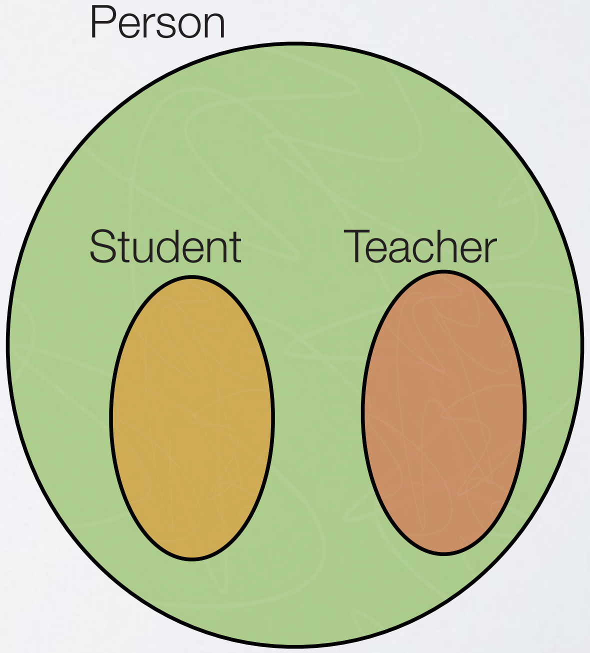
- Complete and disjoint generalisation, a.k.a. partition





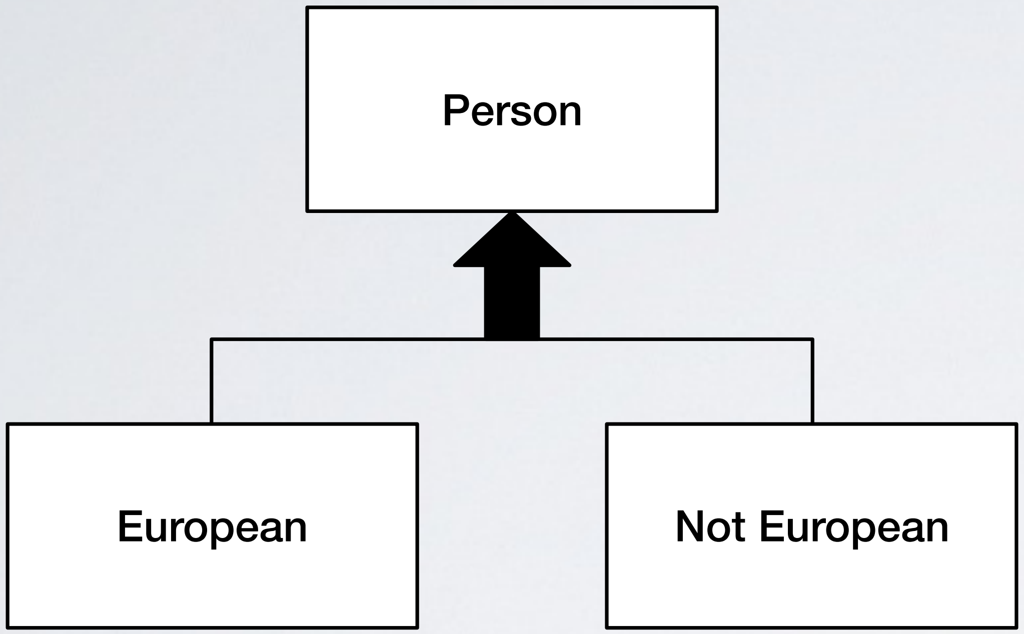
- Not complete but disjoint generalisation

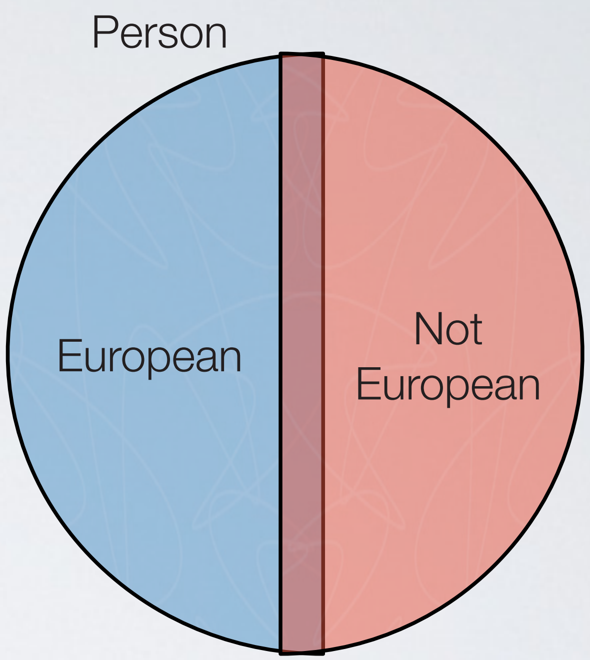




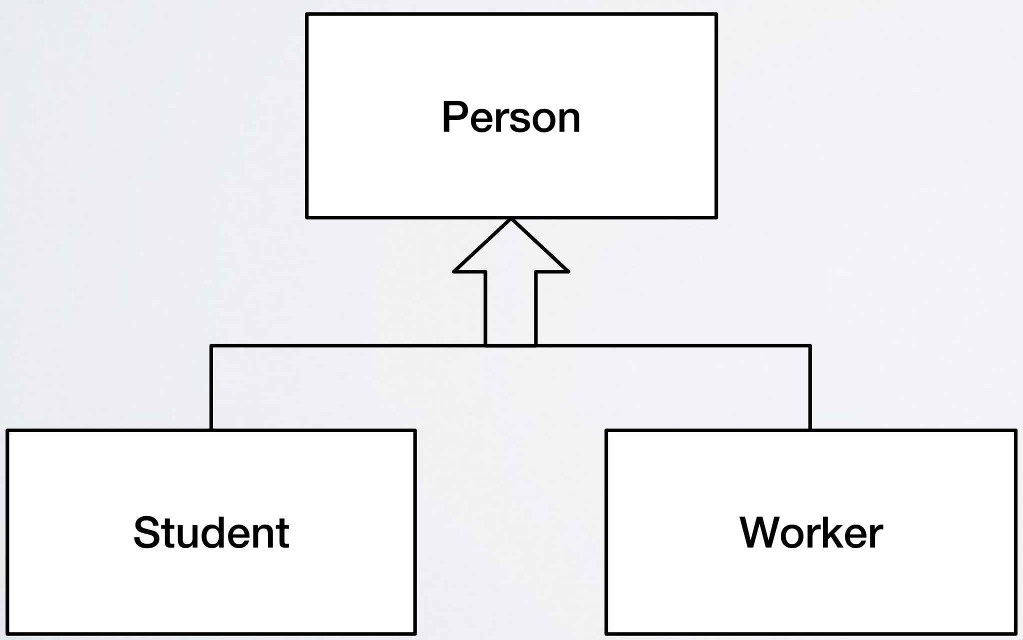
Completness and Disjointness

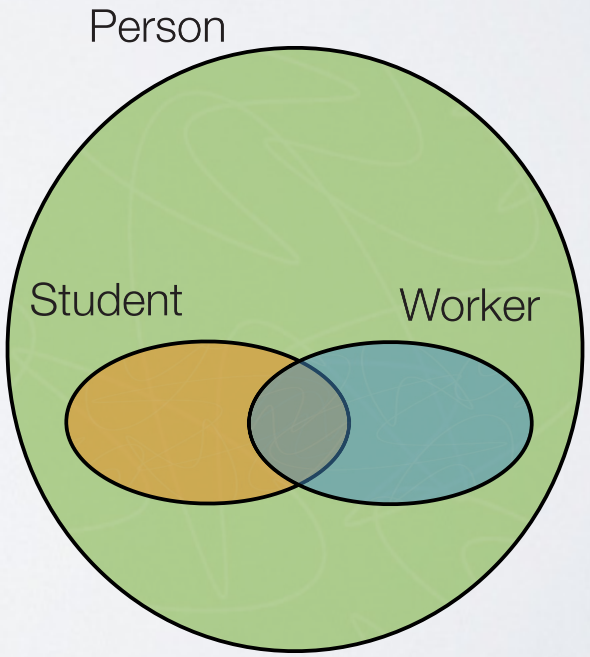
- Complete and not disjoint generalisation



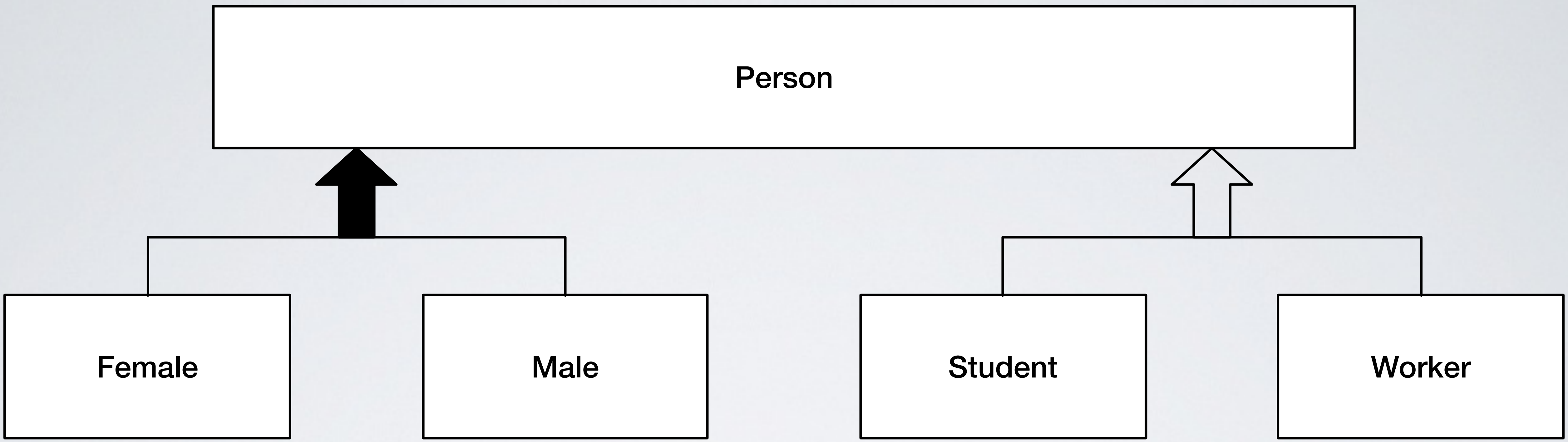


- Not complete and not disjoint generalisation



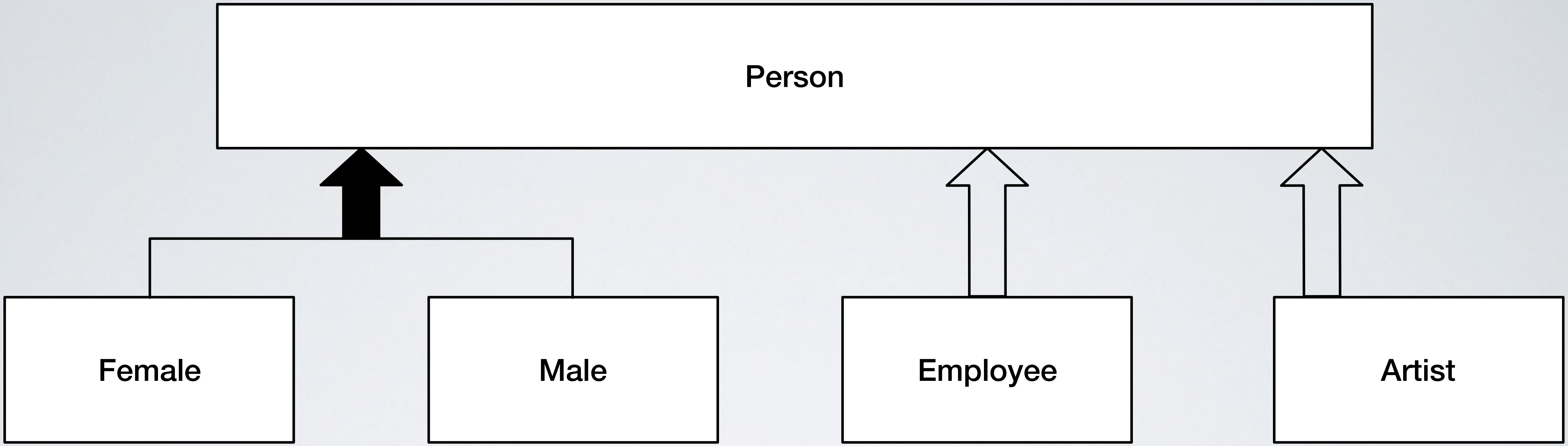


Different Generalisations on the Same Entity



- The same entity can be a superclass in different generalisations created according to different criteria
- There is no link between two different generalisations because they correspond to different criteria

Generalisation and IS-A Relation



The two subclasses derive from the same classification criterion of the instances of the superclass

The two subclasses are “independent”, meaning that they derive from two different classification criteria for the instances of the superclass

Further Readings (1/2)

- Batini, C., Ceri, S., and Navathe, S. B. (1992). *Conceptual Database Design. An Entity-Relationship Approach*. The Benjamin/Cummings Publishing Company, Inc., Redwood City (CA), USA.
- Batini, C., De Petra, G., Lenzerini, M., and Santucci, G. (2002). *La progettazione concettuale dei dati*. Franco Angeli, Milano.
- Chen, P. P. (1976). The Entity–Relationship Model – Towards a Unified View of Data. *ACM Transactions on Database Systems (TODS)*, 1(1):9– 36.
- Chen, P. P. (2002). Entity-Relationship Modeling: Historical Events, Future Trends, and Lessons Learned. In Broy, M. and Denert, E., editors, *Software Pioneers: Contributions to Software Engineering*, pages 296–310. Springer-Verlag, New York, USA.
- dos Santos, C. S., Neuhold, E. J., and Furtado, A. L. (1980). A Data Type Approach to the Entity-Relationship Approach. In Chen, P. P., editor (1980). *Proc. 1st International Conference on the Entity-Relationship Approach to Systems Analysis and Design (ER 1980)*. North-Holland Publishing Co. Amsterdam, The Netherlands, pages 103–120.
- Elmasri, E., Weeldreyer, J., and Hevner, A. (1985). The category concept: An extension to the entity–relationship model. *Data & Knowledge Engineering*, 1(1):75–116.

Further Readings (2/2)

- Scheuermann, P., Schiffner, G., and Weber, H. (1980). Abstraction Capabilities and Invariant Properties Modelling within the Entity-Relationship Approach. In Chen, P. P., editor (1980). *Proc. 1st International Conference on the Entity-Relationship Approach to Systems Analysis and Design (ER 1980)*. North-Holland Publishing Co. Amsterdam, The Netherlands, pages 121–140.
- Smith, J. M. and Smith, D. C. P. (1977). Database Abstractions: Aggregation and Generalization. *ACM Transactions on Database Systems (TODS)*, 2(2):105–133.
- Teorey, T. J. (1990). *Database Modeling and Design. The Entity–Relationship Approach*. Morgan Kaufmann Publishers, San Francisco (CA), USA.
- Teorey, T. J., Yang, D., and Fry, J. (1986). A Logical Design Methodology for Relational Databases Using the Extended Entity-Relationship Model. *ACM Computing Surveys (CSUR)*, 18(2):197–222.


Questions?


WE'RE GOING TO TRY SOMETHING CALLED AGILE PROGRAMMING.
THAT MEANS NO MORE PLANNING AND NO MORE DOCUMENTATION. JUST START WRITING CODE AND COMPLAINING.
I'M GLAD IT HAS A NAME.
THAT WAS YOUR TRAINING.
www.dilbert.com
scottadams@aol.com
©2007 Scott Adams, Inc./Dist. by UFS, Inc.