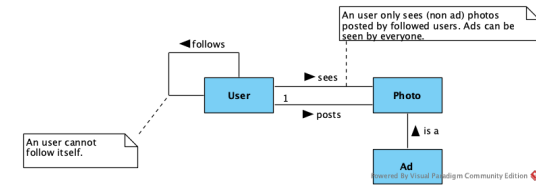# Formalizing Domain Models in FOL

Maria João Frade

HASLab - INESC TEC
Departamento de Informática, Universidade do Minho

2022/2023

Domain models

Recall that a domain model is a diagram where

- entities are represented by boxes and

- relationships between entities by arcs annotated with
  - the name of the relationship/association,
  - the reading direction (indicated by an arrow) and
  - its multiplicity (annotated at the tip of the arcs).

- annotations with restrictions that are informally indicated in boxes may also appear in the diagram.


Domain models

- There is no standard notation for domain models. We assume that:

  - all entities present in a diagram are disjoint, unless between two entities there is a relation/association *"is a"* which denotes a specialization/extension;

  - if there are multiple specializations for the same entity, those specializations are disjoint;

  - the relation/association *"is a"* may be a subset or a membership relation. If it is a membership relation, the entity that "belongs to" another will be modeled as a constant.


Formalizing a domain model

- First establish the logical language (i.e., the vocabulary).
  - a unary predicate for each entity (these predicates will act as the "type" of the entity, in a domain that is untyped);
  - a predicate for each association, except specializations (which will be codified by formulas that establish the kind of specialization);
  - a constant for each entity belonging to an "enumerated type".

- Next define the set of formulas that describe the system. They are of different nature:
  - codification of specialization relationships (which may be subset or membership relations, depending on the case);
  - partitioning the universe of discourse with the types of the entities;
  - disjunction of specialization entities;
  - typing associations;
  - multiplicity restrictions on associations;
  - restrictions annotated informally.

Slide 5/34

Example: Train station

In simplistic terms, the layout of a train station is composed of tracks that are connected to each other. A track can have more than one successor, namely if it ends in a railway switch. A switch can also be used to form a junction between two or more tracks. Trains arrive at the station through entry tracks and leave the station through exit tracks. The interlocking system is responsible for ensuring the safe flow of the trains through a station. A key part of the interlocking are signals. Possible signals are semaphores and speed limits.
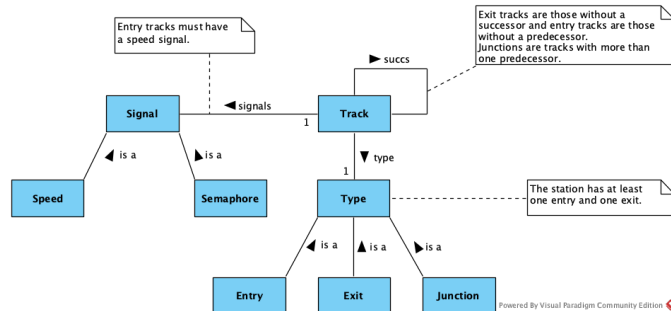
Slide 6/34
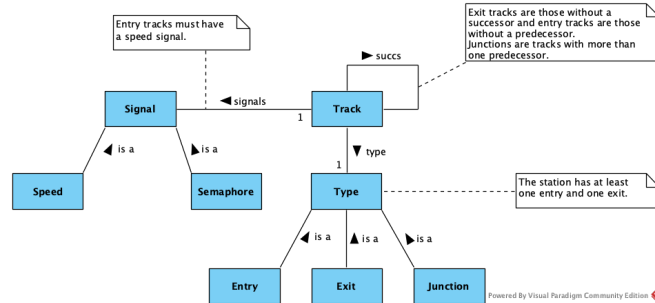
Example: Train station
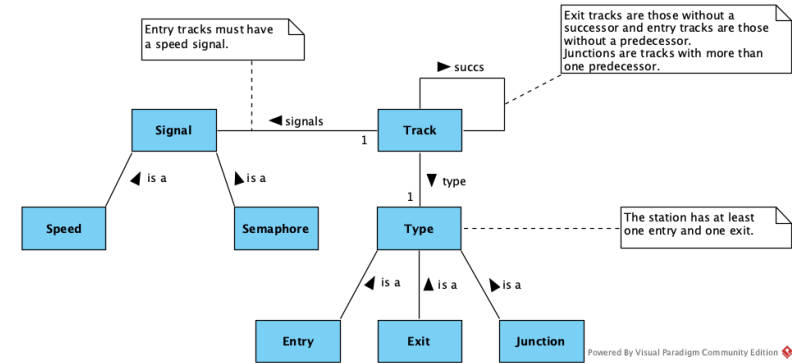
Predicates: Signal(-)   Speed(-)   Semaphore(-)   Track(-)   Type(-)
signals(-,-)   type(-,-)   succs(-,-)

Constants: Entry   Exit   Junction

Slide 7/34

Example: Train station

- Codification of specialization relationships.
  - The Signal entity has two subsets.
    $$\forall x.\, \text{Speed}(x) \rightarrow \text{Signal}(x)$$
    $$\forall x.\, \text{Semaphore}(x) \rightarrow \text{Signal}(x)$$
  - The Type entity is an "enumerated type" with at least 3 elements.
    $$\text{Type(Entry)}, \quad \text{Type(Exit)}, \quad \text{Type(Junction)}$$

Slide 8/34

Example: Train station

- Partitioning the universe of discourse with the types of the entities.
$$\forall x.\, \text{Signal}(x) \leftrightarrow \neg\text{Track}(x) \wedge \neg\text{Type}(x)$$
$$\forall x.\, \text{Track}(x) \leftrightarrow \neg\text{Signal}(x) \wedge \neg\text{Type}(x)$$
$$\forall x.\, \text{Type}(x) \leftrightarrow \neg\text{Track}(x) \wedge \neg\text{Signal}(x)$$

Example: Train station

Entry tracks must have a speed signal.

Exit tracks are those without a successor and entry tracks are those without a predecessor. Junctions are tracks with more than one predecessor.

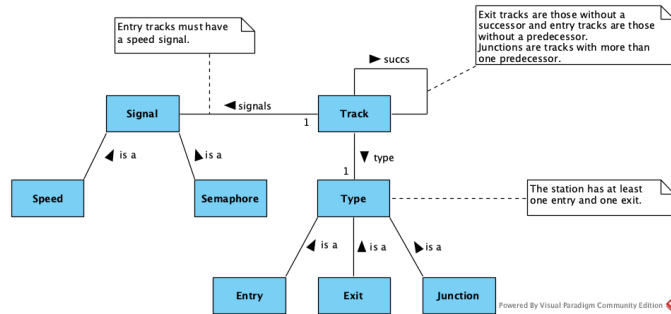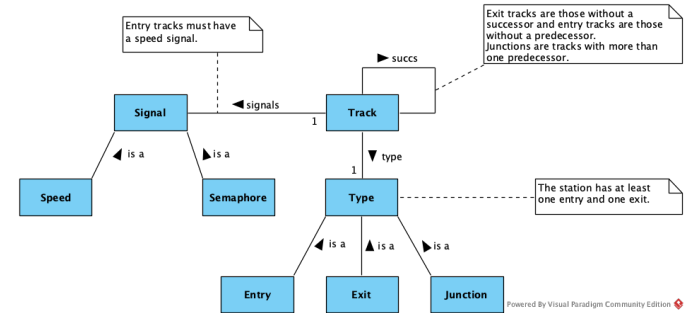▶ succs

Signal ◀ signals Track 1

◀ is a ▲ is a

Speed Semaphore Type ▼ type 1

The station has at least one entry and one exit.

◀ is a ▲ is a ▶ is a

Entry Exit Junction

Powered By Visual Paradigm Community Edition

- Disjunction of specialization entities.

$$\forall x.\, \mathsf{Speed}(x) \rightarrow \neg \mathsf{Semaphore}(x)$$
$$\mathsf{Entry} \neq \mathsf{Exit} \wedge \mathsf{Entry} \neq \mathsf{Junction} \wedge \mathsf{Exit} \neq \mathsf{Junction}$$

Example: Train station

Entry tracks must have a speed signal.

Exit tracks are those without a successor and entry tracks are those without a predecessor. Junctions are tracks with more than one predecessor.

▶ succs

Signal ◀ signals Track 1

◀ is a ▲ is a

Speed Semaphore Type ▼ type 1

The station has at least one entry and one exit.

◀ is a ▲ is a ▶ is a

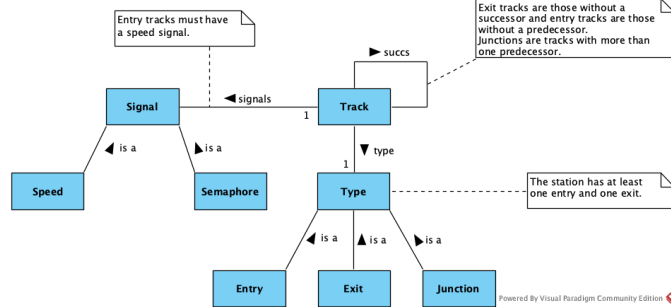Entry Exit Junction

Powered By Visual Paradigm Community Edition

- Typing associations.

$$\forall x, y.\, \mathsf{signals}(x, y) \rightarrow \mathsf{Track}(x) \wedge \mathsf{Signal}(y)$$
$$\forall x, y.\, \mathsf{succs}(x, y) \rightarrow \mathsf{Track}(x) \wedge \mathsf{Track}(y)$$
$$\forall x, y.\, \mathsf{type}(x, y) \rightarrow \mathsf{Track}(x) \wedge \mathsf{Type}(y)$$

Example: Train station

Entry tracks must have a speed signal.

Exit tracks are those without a successor and entry tracks are those without a predecessor. Junctions are tracks with more than one predecessor.

▶ succs

Signal ◀ signals Track 1

◀ is a ▲ is a

Speed Semaphore Type ▼ type 1

The station has at least one entry and one exit.

◀ is a ▲ is a ▶ is a

Entry Exit Junction

Powered By Visual Paradigm Community Edition

- Multiplicity restrictions on associations.

$$(\forall x.\, \mathsf{Signal}(x) \rightarrow \exists y.\, \mathsf{signals}(y, x)) \; \wedge$$
$$(\forall x, y, z.\, \mathsf{signals}(x, z) \wedge \mathsf{signals}(y, z) \rightarrow x = y)$$

$$(\forall x.\, \mathsf{Track}(x) \rightarrow \exists y.\, \mathsf{type}(x, y)) \; \wedge$$
$$(\forall x, y, z.\, \mathsf{type}(x, z) \wedge \mathsf{type}(x, y) \rightarrow z = y)$$

Example: Train station

Entry tracks must have a speed signal.

Exit tracks are those without a successor and entry tracks are those without a predecessor. Junctions are tracks with more than one predecessor.

▶ succs

Signal ◀ signals Track 1

◀ is a ▲ is a

Speed Semaphore Type ▼ type 1

The station has at least one entry and one exit.

◀ is a ▲ is a ▶ is a

Entry Exit Junction

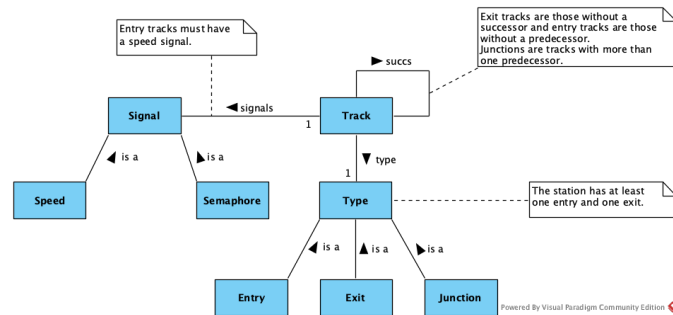Powered By Visual Paradigm Community Edition
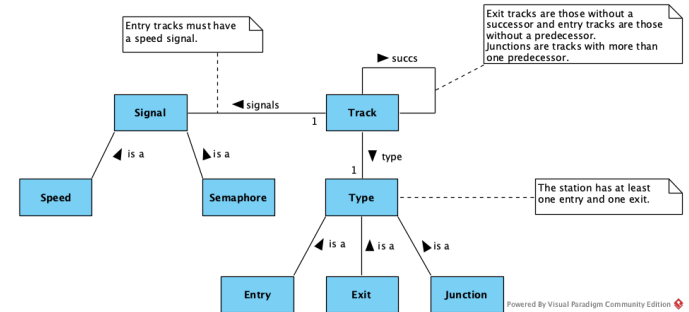
- Restrictions annotated informally.
  - ▶ *Entry tracks must have a speed signal.*

$$\forall x.\, \mathsf{type}(x, \mathsf{Entry}) \rightarrow \exists y.\, \mathsf{signals}(x, y) \wedge \mathsf{Speed}(y)$$

Example: Train station

Entry tracks must have a speed signal.

Exit tracks are those without a successor and entry tracks are those without a predecessor. Junctions are tracks with more than one predecessor.
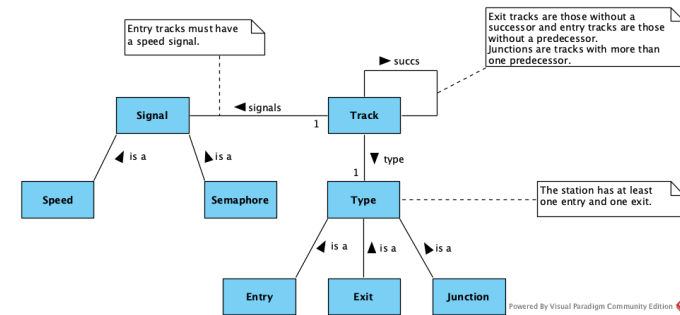
The station has at least one entry and one exit.

► Exit tracks are those without a successor and entry tracks are those without a predecessor. Junctions are tracks with more than one predecessor.

$$\forall x.\, \mathsf{type}(x, \mathsf{Exit}) \leftrightarrow \forall y.\, \neg\mathsf{succs}(x, y)$$
$$\forall x.\, \mathsf{type}(x, \mathsf{Entry}) \leftrightarrow \neg\exists y.\, \mathsf{succs}(y, x)$$
$$\forall x.\, \mathsf{type}(x, \mathsf{Junction}) \leftrightarrow \exists y, z.\, \mathsf{succs}(y, x) \wedge \mathsf{succs}(z, x) \wedge y \neq z$$
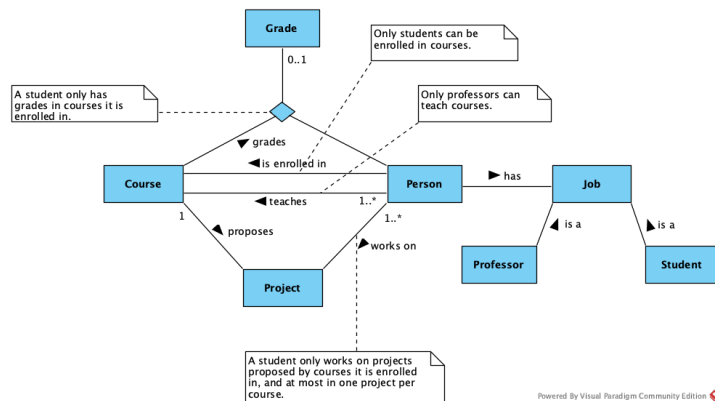

Example: Train station

Entry tracks must have a speed signal.

Exit tracks are those without a successor and entry tracks are those without a predecessor. Junctions are tracks with more than one predecessor.

The station has at least one entry and one exit.

► The station has at least one entry and one exit.

$$\exists x.\, \mathsf{type}(x, \mathsf{Entry})$$
$$\exists x.\, \mathsf{type}(x, \mathsf{Exit})$$

Example: Courses

Only students can be enrolled in courses.

A student only has grades in courses it is enrolled in.

Only professors can teach courses.

A student only works on projects proposed by courses it is enrolled in, and at most in one project per course.

In this diagram, the diamond symbol is used to represent an n-ary association. In this case *grades* will be a ternary predicate that associates *courses* with *persons* and *grades*.

Example: Courses

Only students can be enrolled in courses.

A student only has grades in courses it is enrolled in.

Only professors can teach courses.

A student only works on projects proposed by courses it is enrolled in, and at most in one project per course.
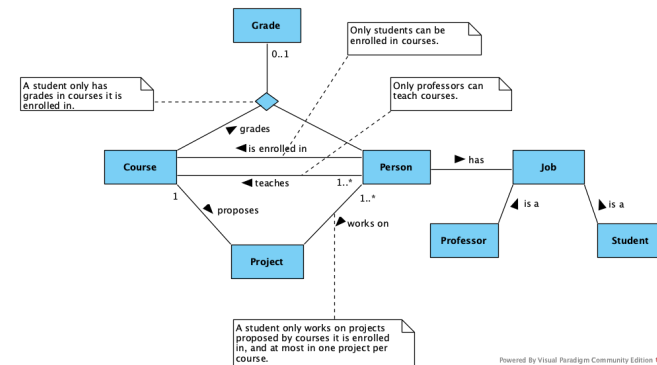
Predicates: Course(-)　Grade(-)　Person(-)　Project(-)
Job(-)　Professor(-)　Student(-)
enrolledin(-,-)　teaches(-,-)　has(-,-)
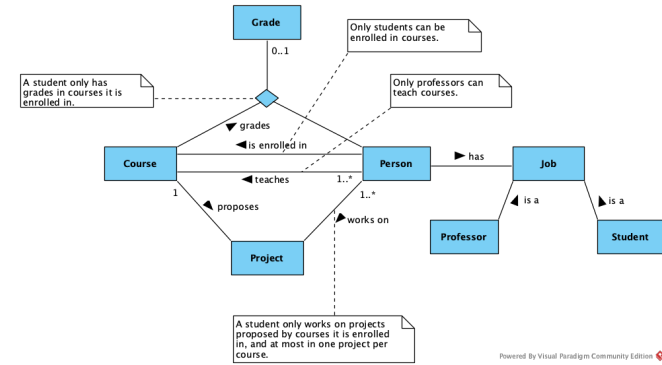porposes(-,-)　workson(-,-)　grades(-,-,-)

Example: Courses
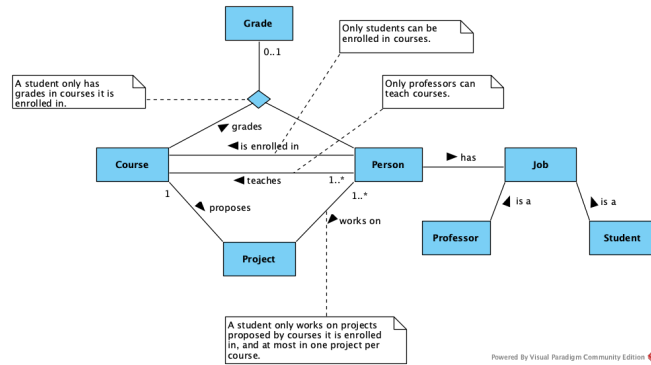
- Codification of specialization relationships.
  - The Job entity has two subsets.

$$\forall x.\, \mathsf{Professor}(x) \to \mathsf{Job}(x)$$
$$\forall x.\, \mathsf{Student}(x) \to \mathsf{Job}(x)$$

Example: Courses

- Partitioning the universe of discourse with the types of the entities.
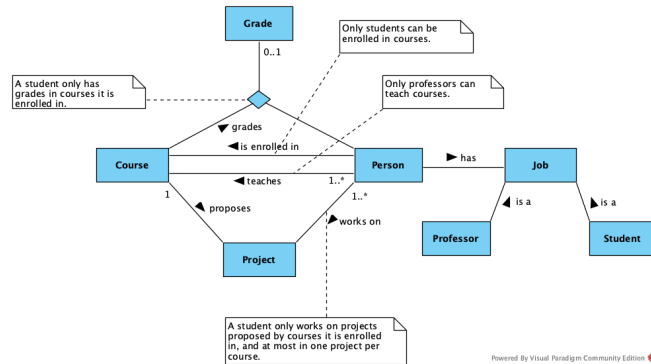
$$\forall x.\, \mathsf{Course}(x) \leftrightarrow \neg\mathsf{Project}(x) \wedge \neg\mathsf{Grade}(x) \wedge \neg\mathsf{Person}(x) \wedge \neg\mathsf{Job}(x)$$
$$\forall x.\, \mathsf{Project}(x) \leftrightarrow \neg\mathsf{Course}(x) \wedge \neg\mathsf{Grade}(x) \wedge \neg\mathsf{Person}(x) \wedge \neg\mathsf{Job}(x)$$
$$\forall x.\, \mathsf{Grade}(x) \leftrightarrow \neg\mathsf{Project}(x) \wedge \neg\mathsf{Course}(x) \wedge \neg\mathsf{Person}(x) \wedge \neg\mathsf{Job}(x)$$
$$\forall x.\, \mathsf{Job}(x) \leftrightarrow \neg\mathsf{Project}(x) \wedge \neg\mathsf{Grade}(x) \wedge \neg\mathsf{Person}(x) \wedge \neg\mathsf{Course}(x)$$
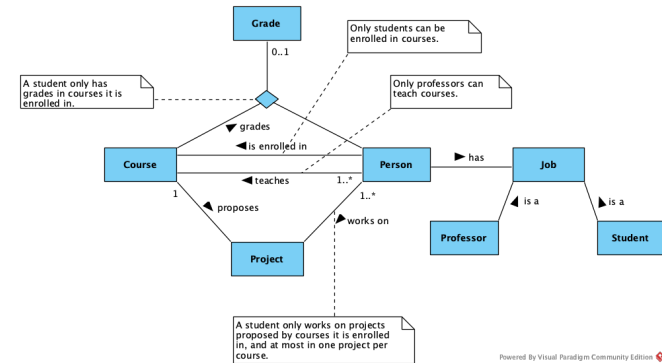

Example: Courses

- Disjunction of specialization entities.

$$\forall x.\, \mathsf{Professor}(x) \to \neg\mathsf{Student}(x)$$

Example: Courses

- Typing associations.

$$\forall x, y, z.\, \mathsf{grades}(x, y, z) \to \mathsf{Person}(x) \wedge \mathsf{Course}(y) \wedge \mathsf{Grade}(z)$$
$$\forall x, y.\, \mathsf{enrolledin}(x, y) \to \mathsf{Person}(x) \wedge \mathsf{Course}(y)$$
$$\forall x, y.\, \mathsf{porposes}(x, y) \to \mathsf{Course}(x) \wedge \mathsf{Project}(y)$$
$$\forall x, y.\, \mathsf{workson}(x, y) \to \mathsf{Person}(x) \wedge \mathsf{Project}(y)$$
$$\forall x, y.\, \mathsf{teaches}(x, y) \to \mathsf{Person}(x) \wedge \mathsf{Course}(y)$$
$$\forall x, y.\, \mathsf{has}(x, y) \to \mathsf{Person}(x) \wedge \mathsf{Job}(y)$$

Example: Courses
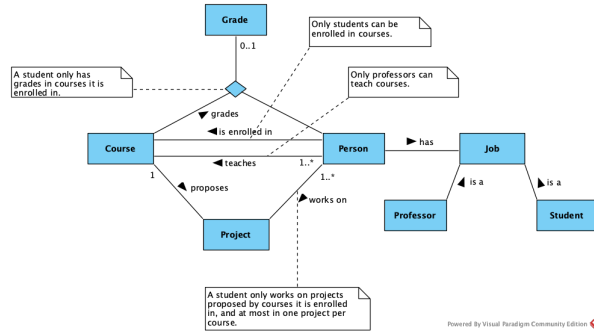
- Multiplicity restrictions on associations.

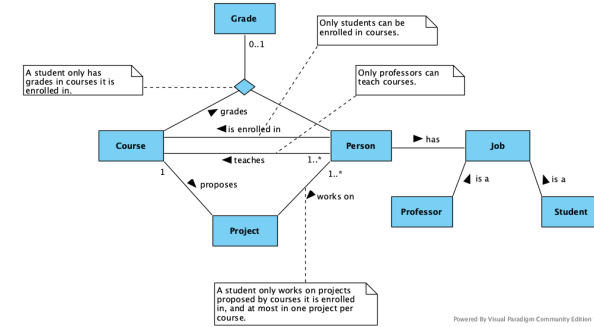 $(\forall x.\, \mathsf{Project}(x) \rightarrow \exists y.\, \mathsf{porposes}(y,x)) \;\wedge$
  $(\forall x,y,z.\, \mathsf{porposes}(x,z) \wedge \mathsf{porposes}(y,z) \rightarrow x=y)$

  $\forall x.\, \mathsf{Course}(x) \rightarrow \exists y.\, \mathsf{teaches}(y,x)$

  $\forall x.\, \mathsf{Project}(x) \rightarrow \exists y.\, \mathsf{workson}(y,x))$

  $\forall x,y,a,b.\, \mathsf{grades}(x,y,a) \wedge \mathsf{grades}(x,y,b) \rightarrow a=b$

Example: Courses

- Restrictions annotated informally.

  ▸ *A student only has grades in courses it is enrolled in.*

  $\forall x,y,z,j.\, \mathsf{has}(x,j) \wedge \mathsf{Student}(j) \wedge \mathsf{grades}(x,y,z) \rightarrow \mathsf{enrolledin}(x,y)$

Example: Courses
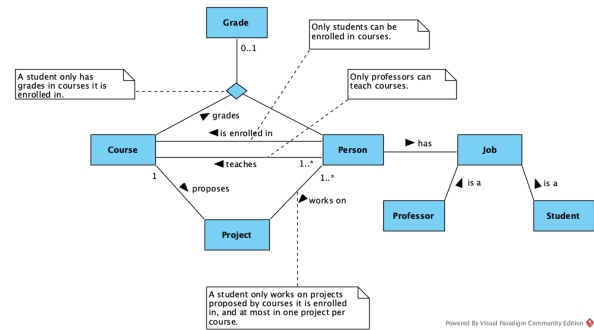
  ▸ *Only students can be enrolled in courses.*

  $\forall x,y.\, \mathsf{enrolledin}(x,y) \rightarrow \exists j.\, \mathsf{has}(x,j) \wedge \mathsf{Student}(j)$

Example: Courses

  ▸ *Only professors can teach courses.*

  $\forall x,y.\, \mathsf{teaches}(x,y) \rightarrow \exists j.\, \mathsf{has}(x,j) \wedge \mathsf{Professor}(j)$

Example: Courses

Grade
0..1

Only students can be enrolled in courses.

A student only has grades in courses it is enrolled in.

Only professors can teach courses.

grades
is enrolled in
Course ◄ teaches — Person ► has — Job
1 proposes
works on
Project

A student only works on projects proposed by courses it is enrolled in, and at most in one project per course.

Professor   Student

Powered By Visual Paradigm Community Edition

▶ *A student only works on projects proposed by courses it is enrolled in, and at most in one project per course.*
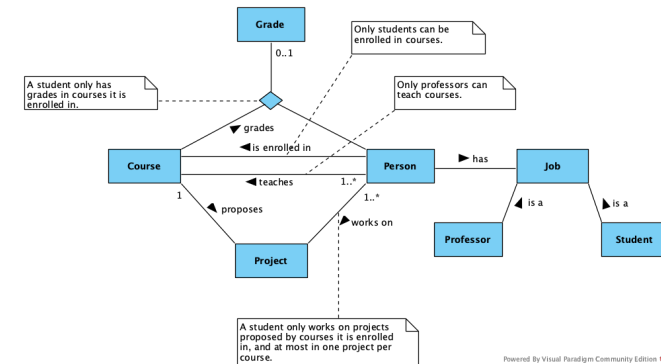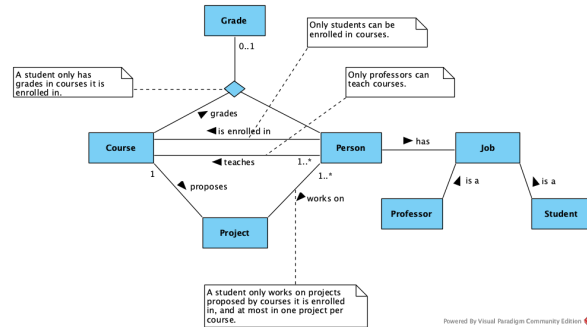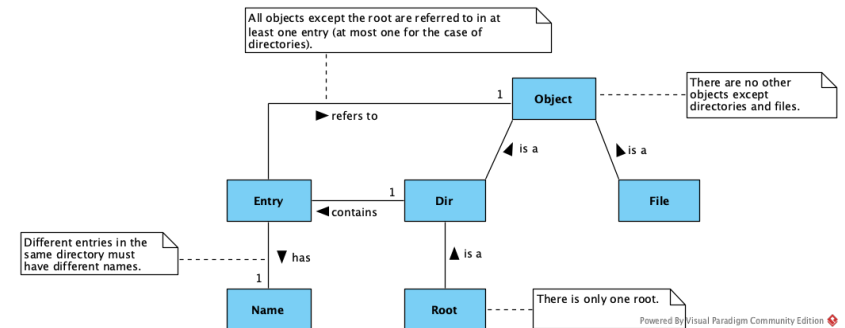
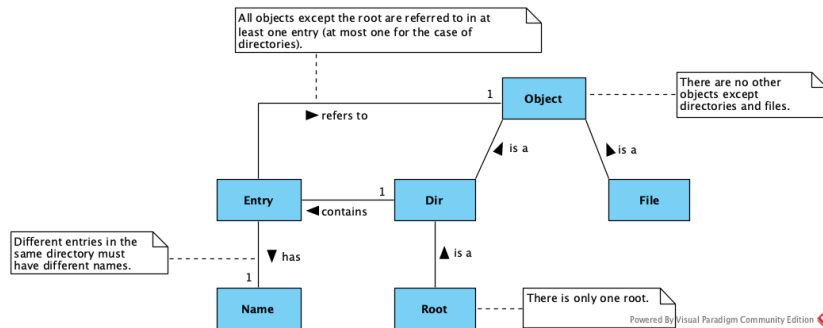$$(\forall x, y.\, \mathsf{workson}(x, y) \rightarrow \exists z.\, \mathsf{porposes}(z, y) \wedge \mathsf{enrolledin}(x, z))$$
$$\wedge \quad (\forall x, y, z, w.\, \mathsf{workson}(x, y) \wedge \mathsf{workson}(x, z)$$
$$\wedge\, \mathsf{porposes}(w, y) \wedge \mathsf{porposes}(w, z) \rightarrow y = z)$$

Example: File system

All objects except the root are referred to in at least one entry (at most one for the case of directories).

There are no other objects except directories and files.

1   Object
► refers to
is a   is a
Entry — 1 — Dir   File
◄ contains

Different entries in the same directory must have different names.

▼ has   ▲ is a
1
Name   Root

There is only one root.

Powered By Visual Paradigm Community Edition

Predicates: Entry(-)   Object(-)   Dir(-)
File(-)   Name(-)   Root(-)
refersto(-,-)   has(-,-)   contains(-,-)

Example: File system

All objects except the root are referred to in at least one entry (at most one for the case of directories).

There are no other objects except directories and files.

1   Object
► refers to
is a   is a
Entry — 1 — Dir   File
◄ contains

Different entries in the same directory must have different names.

▼ has   ▲ is a
1
Name   Root

There is only one root.

Powered By Visual Paradigm Community Edition

● Codification of specialization relationships.

$$\forall x.\, \mathsf{Dir}(x) \rightarrow \mathsf{Object}(x)$$
$$\forall x.\, \mathsf{File}(x) \rightarrow \mathsf{Object}(x)$$
$$\forall x.\, \mathsf{Root}(x) \rightarrow \mathsf{Dir}(x)$$
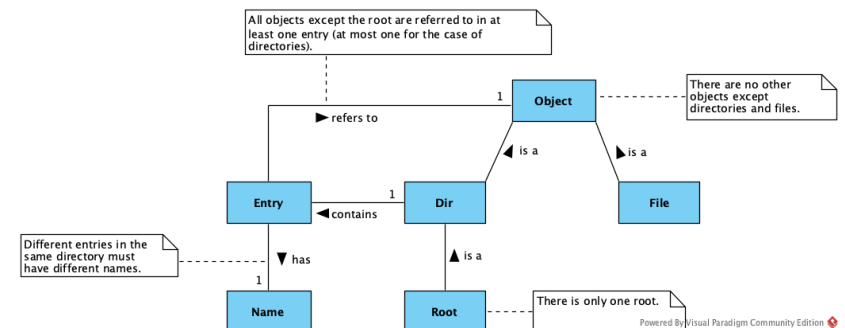
Example: File system

All objects except the root are referred to in at least one entry (at most one for the case of directories).

There are no other objects except directories and files.

1   Object
► refers to
is a   is a
Entry — 1 — Dir   File
◄ contains

Different entries in the same directory must have different names.

▼ has   ▲ is a
1
Name   Root

There is only one root.

Powered By Visual Paradigm Community Edition

● Partitioning the universe of discourse with the types of the entities.

$$\forall x.\, \mathsf{Entry}(x) \leftrightarrow \neg\mathsf{Name}(x) \wedge \neg\mathsf{Object}(x)$$
$$\forall x.\, \mathsf{Name}(x) \leftrightarrow \neg\mathsf{Entry}(x) \wedge \neg\mathsf{Object}(x)$$
$$\forall x.\, \mathsf{Object}(x) \leftrightarrow \neg\mathsf{Entry}(x) \wedge \neg\mathsf{Name}(x)$$

Example: File system

All objects except the root are referred to in at least one entry (at most one for the case of directories).

There are no other objects except directories and files.

Different entries in the same directory must have different names.
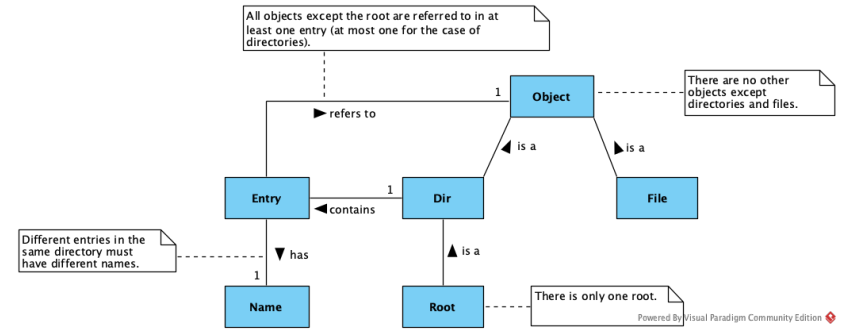
There is only one root.

Powered By Visual Paradigm Community Edition
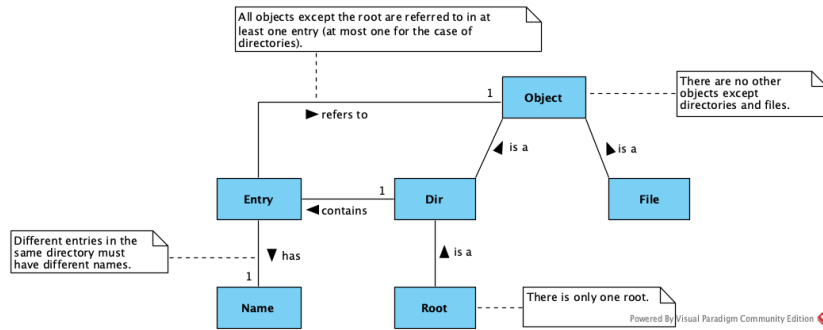
- Disjunction of specialization entities.

$$\forall x.\, \mathsf{Dir}(x) \rightarrow \neg\mathsf{File}(x)$$
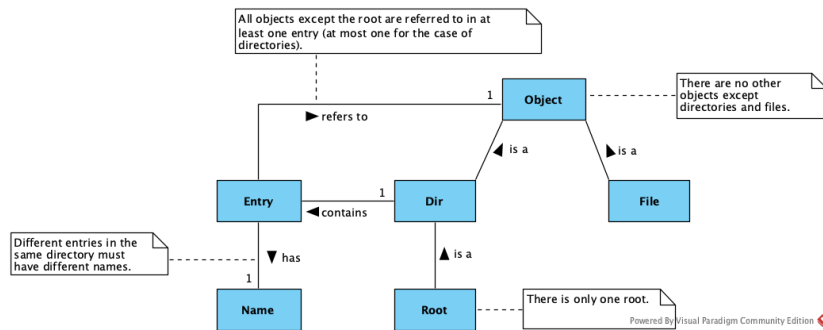
Example: File system

All objects except the root are referred to in at least one entry (at most one for the case of directories).

There are no other objects except directories and files.

Different entries in the same directory must have different names.

There is only one root.

Powered By Visual Paradigm Community Edition

- Typing associations.

$$\forall x, y.\, \mathsf{has}(x, y) \rightarrow \mathsf{Entry}(x) \wedge \mathsf{Name}(y)$$
$$\forall x, y.\, \mathsf{refersto}(x, y) \rightarrow \mathsf{Entry}(x) \wedge \mathsf{Object}(y)$$
$$\forall x, y.\, \mathsf{contains}(x, y) \rightarrow \mathsf{Dir}(x) \wedge \mathsf{Entry}(y)$$

Example: File system

All objects except the root are referred to in at least one entry (at most one for the case of directories).

There are no other objects except directories and files.

Different entries in the same directory must have different names.

There is only one root.
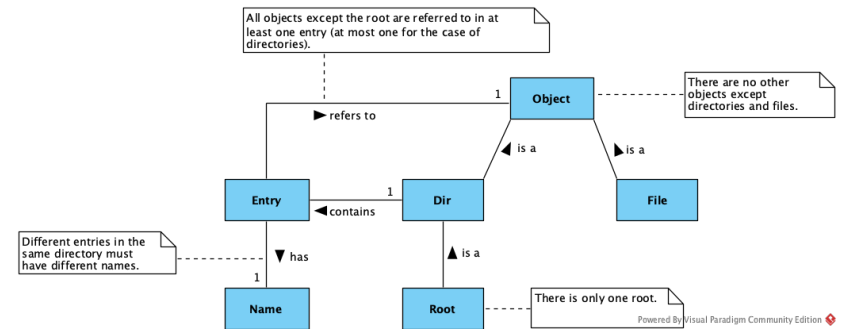
Powered By Visual Paradigm Community Edition

- Multiplicity restrictions on associations.

$$(\forall x.\, \mathsf{Entry}(x) \rightarrow \exists y.\, \mathsf{has}(x, y)) \wedge (\forall x, y, z.\, \mathsf{has}(x, y) \wedge \mathsf{has}(x, z) \rightarrow y = z)$$

$$(\forall x.\, \mathsf{Entry}(x) \rightarrow \exists y.\, \mathsf{refersto}(x, y)) \wedge$$
$$(\forall x, y, z.\, \mathsf{refersto}(x, y) \wedge \mathsf{refersto}(x, z) \rightarrow y = z)$$

$$(\forall x.\, \mathsf{Entry}(x) \rightarrow \exists y.\, \mathsf{contains}(y, x)) \wedge$$
$$(\forall x, y, z.\, \mathsf{contains}(x, y) \wedge \mathsf{contains}(x, z) \rightarrow y = z)$$

Example: File system

All objects except the root are referred to in at least one entry (at most one for the case of directories).

There are no other objects except directories and files.

Different entries in the same directory must have different names.

There is only one root.

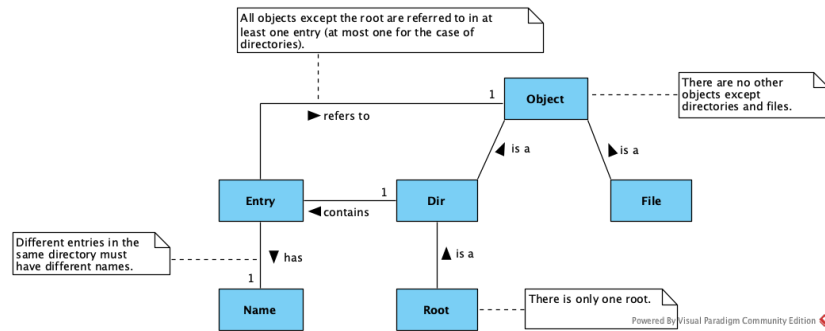Powered By Visual Paradigm Community Edition

- Restrictions annotated informally.

  ▶ *Different entries in the same directory must have different names.*

$$\forall x, y, z, n.\, \mathsf{contains}(x, y) \wedge \mathsf{contains}(x, z) \wedge y \neq z \wedge \mathsf{has}(y, n) \rightarrow \neg\mathsf{has}(z, n)$$
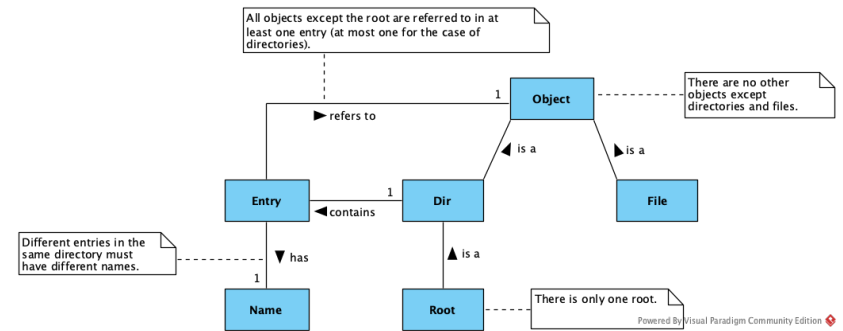
Example: File system

All objects except the root are referred to in at least one entry (at most one for the case of directories).

There are no other objects except directories and files.

Different entries in the same directory must have different names.

There is only one root.

- *There is only one root.*

$$\forall x, y. \, \mathsf{Root}(x) \wedge \mathsf{Root}(y) \rightarrow x = y$$

Example: File system

All objects except the root are referred to in at least one entry (at most one for the case of directories).

There are no other objects except directories and files.

Different entries in the same directory must have different names.

There is only one root.

- *There are no other objects except directories and files.*

$$\forall x. \, \mathsf{Object}(x) \rightarrow \mathsf{Dir}(x) \vee \mathsf{File}(x)$$