Software Design, Modelling and Analysis in UML

Lecture 8: Class Diagrams III

2015-11-26

Prof. Dr. Andreas Podelski, **Dr. Bernd Westphal**

Albert-Ludwigs-Universität Freiburg, Germany

Contents & Goals

Last Lectures:

- completed class diagrams. . . except for associations.

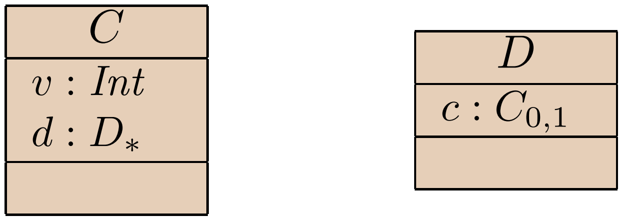
This Lecture:

- **Educational Objectives:** Capabilities for following tasks/questions.
  - Please explain this class diagram with associations.
  - Which annotations of an association arrow are semantically relevant?
  - What's a role name? What's it good for?
  - What is "multiplicity"? How did we treat them semantically?
  - What is "reading direction", "navigability", "ownership", . . . ?
  - What's the difference between "aggregation" and "composition"?
- **Content:**
  - Study concrete syntax for "associations".
  - (**Temporarily**) extend signature, define mapping from diagram to signature.
  - Study effect on OCL.
  - Btw.: where do we put OCL constraints?

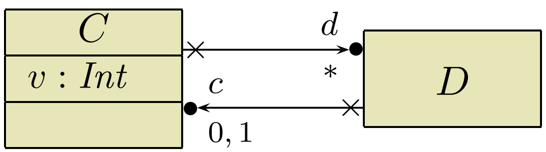
Overview

context Z inv : a->size > 3

- **Class diagram**:



Alternative presentation:



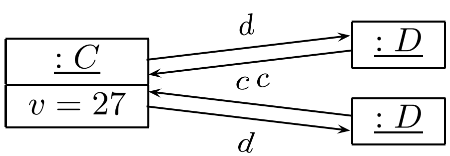
- **Signature**:

$$\mathscr{S} = (\{Int\}, \{C, D\}, \{v : Int, d : D_*, c : C_{0,1}\}, \{C \mapsto \{v, d\}, D \mapsto \{c\}\})$$

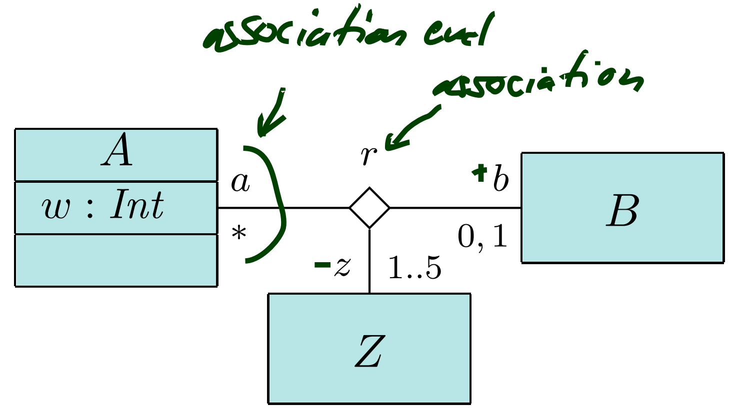
- **Example system state**:

$$\sigma = \{1_C \mapsto \{v \mapsto 27, d \mapsto \{5_D, 7_D\}\}, 5_D \mapsto \{c \mapsto \{1_C\}\}, 7_D \mapsto \{c \mapsto \{1_C\}\}\}$$

- **Object diagram**:



- **Class diagram** (with ternary association):



- **Signature**: extend again;

  represent association r
  with association ends a, b, and z
  (each with multiplicity, visibility, etc.)

- **Example system state**: (σ, λ) $3_Z \mapsto \emptyset$ $2_Z \mapsto \emptyset$

$$\sigma = \{1_A \mapsto \{w \mapsto 13\}, 1_B \mapsto \emptyset, 1_Z \mapsto \emptyset\}$$

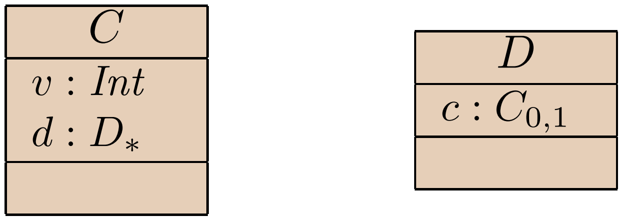
$$\lambda = \{\ r \mapsto \{(1_A, 1_B, 1_Z)\}\ \},\ (1_A, 1_B, 2_Z)$$

r:

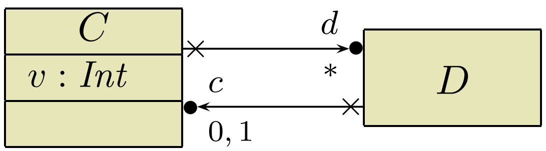
a	b	z
1_A	1_B	1_Z
1_A	1_B	2_Z

Overview

- **Class diagram**:



Alternative presentation:



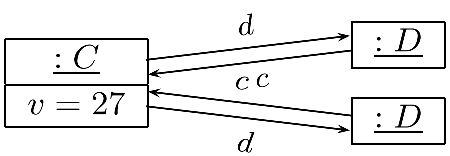
- **Signature**:

$$\mathscr{S} = (\{Int\}, \{C, D\}, \{v : Int, d : D_*, c : C_{0,1}\},$$
$$\{C \mapsto \{v, d\}, D \mapsto \{c\}\})$$

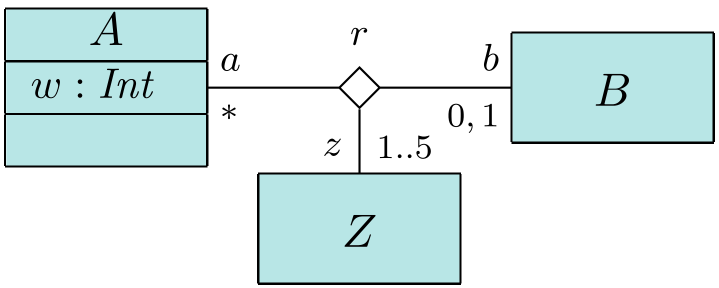
- **Example system state**:

$$\sigma = \{1_C \mapsto \{v \mapsto 27, c \mapsto \{5_D, 7_D\}\},$$
$$5_D \mapsto \{d \mapsto \{1_C\}\}, 7_D \mapsto \{d \mapsto \{1_C\}\}\}$$

- **Object diagram**:



- **Class diagram** (with ternary association):



- **Signature**: extend again;

represent **association** r
with **association ends** a, b, and z
(each with multiplicity, visibility, etc.)

- **Example system state**:

$$\sigma = \{1_A \mapsto \{w \mapsto 13\}, 1_B \mapsto \emptyset, 1_Z \mapsto \emptyset\}$$

$$\lambda = \{\ r \mapsto \{(1_A, 1_B, 1_Z)\}\ \}$$

- **Object diagram**: No...

Plan

(i) Study association **syntax**.

(ii) Extend **signature** accordingly.

(iii) Define (σ, λ) **system states** with

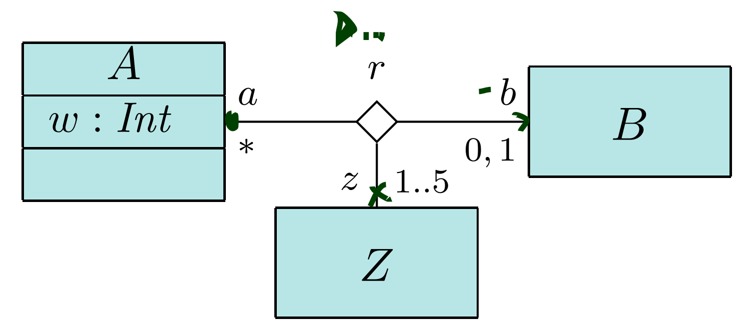
- **objects** in σ
  (instances of classes),
- **links** in λ
  (instances of associations).

(iv) Change **syntax** of OCL to refer to **association ends**.

(v) Adjust **interpretation** I accordingly.

(vi) ... go back to the special case of $C_{0,1}$ and C_* attributes.

- **Class diagram** (with ternary association):



- **Signature**: extend again;
  represent **association** r
  with **association ends** a, b, and z
  (each with multiplicity, visibility, etc.)

- **Example system state**:

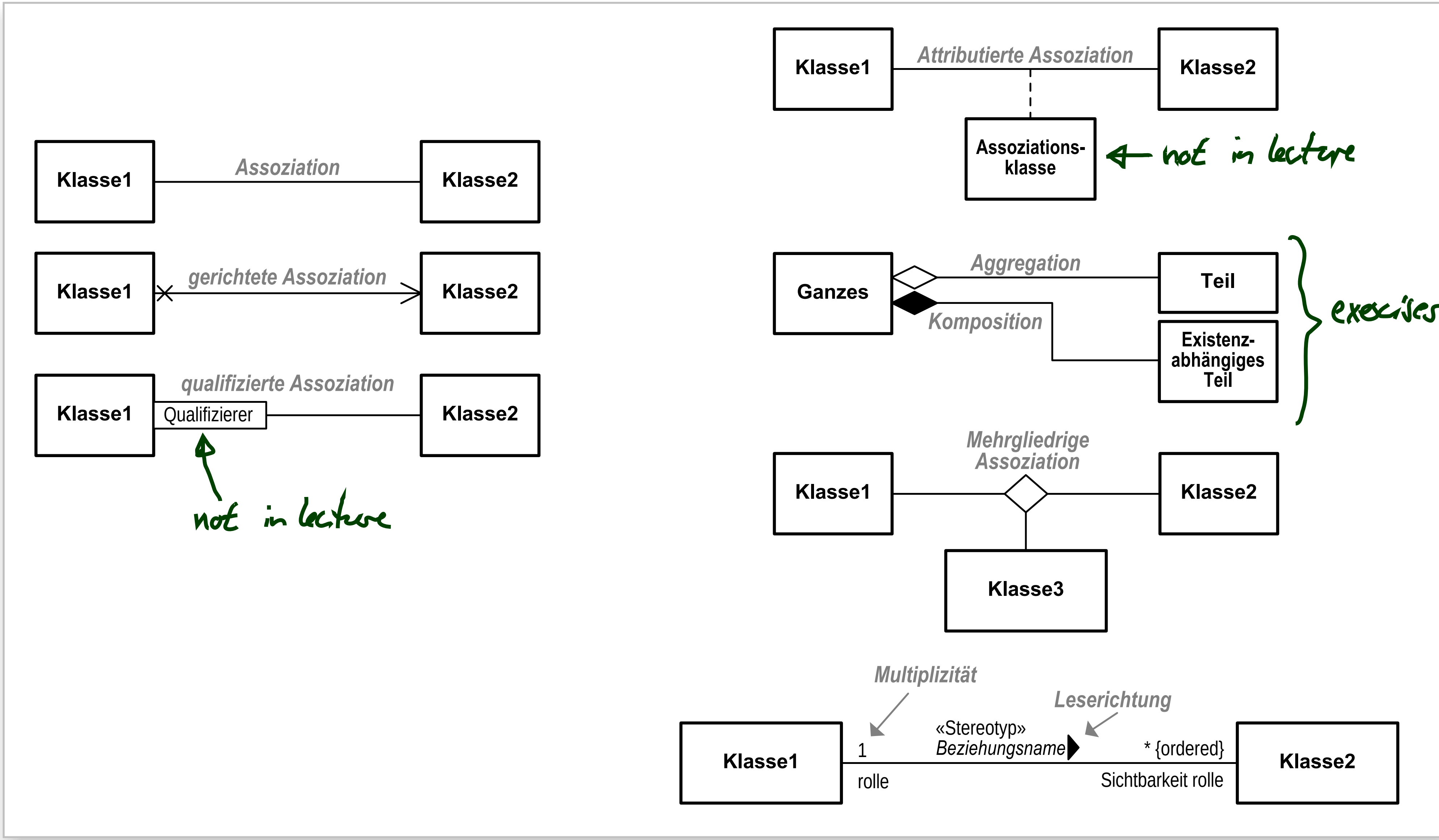
$$\sigma = \{1_A \mapsto \{w \mapsto 13\}, 1_B \mapsto \emptyset, 1_Z \mapsto \emptyset\}$$

$$\lambda = \{\ r \mapsto \{(1_A, 1_B, 1_Z)\}\ \}$$

- **Object diagram**: No...

Associations: Syntax

UML Association Syntax Oestereich (2006)



More Association Syntax *(OMG, 2011b, 61;43)*

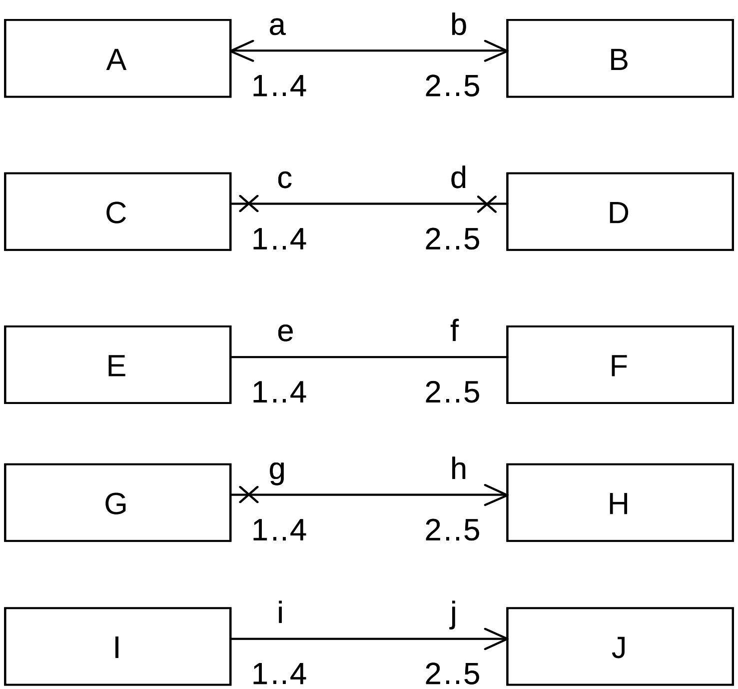


Figure 7.23 - Examples of navigable ends

Figure 7.19 - Graphic notation indicating exactly one association end owned by the association

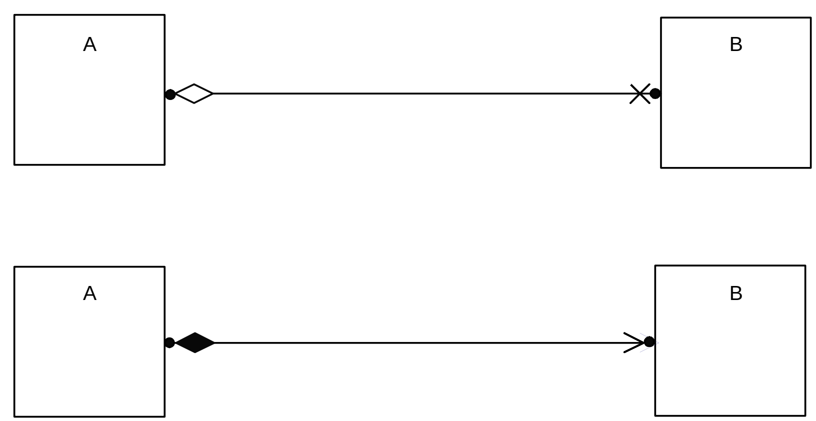


Figure 7.20 - Combining line path graphics

So, What Do We (Have to) Cover?

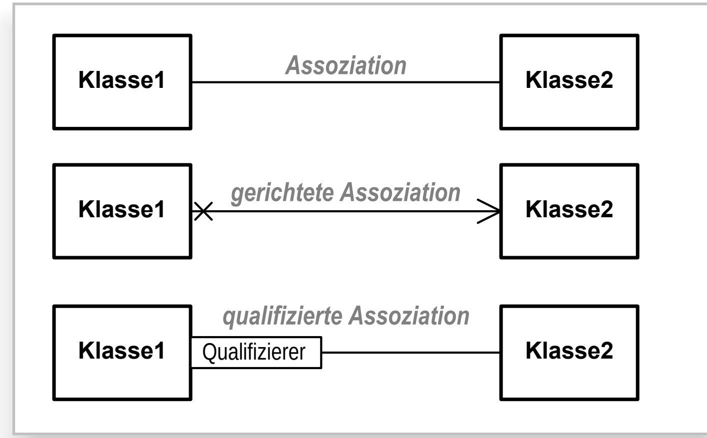
An **association** has

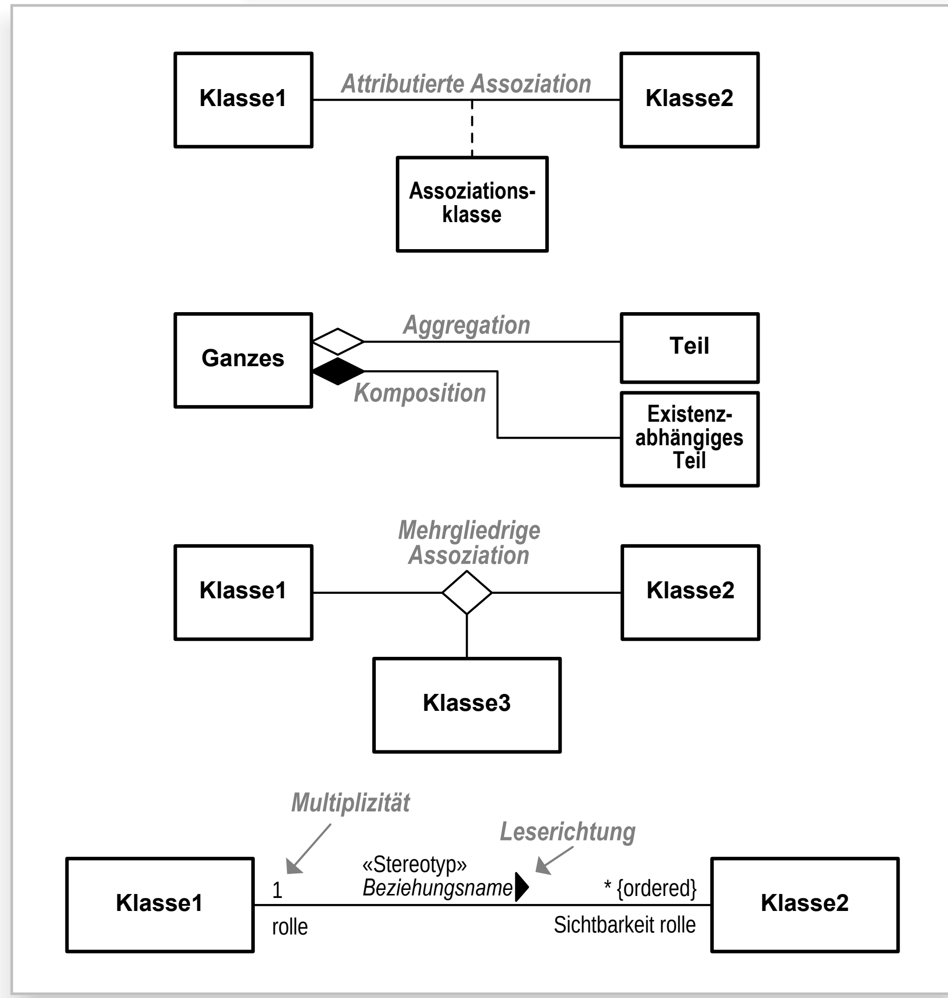
- a **name**,
- a **reading direction**, and
- at least two **ends**.

Each **end** has

- a **role name**,
- a **multiplicity**,
- a set of **properties**, such as **unique**, **ordered**, etc.
- a **qualifier**, (not in lecture)
- a **visibility**,
- a **navigability**,
- an **ownership**,
- and possibly a **diamond**. (exercises)

Wanted: places in the signature to represent the information from the picture.





(Temporarily) Extend Signature: Associations

Only for the course of Lectures 08/09 we assume that each element in V is

- **either** a **basic type attribute** $\langle v : T, \xi, expr_0, P_v \rangle$ with $T \in \mathscr{T}$ **(as before)**,
- **or** an **association** of the form

association end

$$\langle r : \quad \langle role_1 : C_1, \mu_1, P_1, \xi_1, \nu_1, o_1 \rangle, \\ \vdots \\ \langle role_n : C_n, \mu_n, P_n, \xi_n, \nu_n, o_n \rangle \rangle$$

association name

the class where this association end is located

- $n \geq 2$ (at least two ends),
- r, $role_i$ are just **names**, $\quad C_i \in \mathscr{C}$, $1 \leq i \leq n$,
- the **multiplicity** μ_i is an expression of the form

$$\mu ::= N..M \mid N..* \mid \mu, \mu \qquad (N, M \in \mathbb{N})$$

- P_i is a set of **properties (as before)**,
- $\xi \in \{+, -, \#, \sim\}$ **(as before)**,
- $\nu_i \in \{\times, -, >\}$ is the **navigability**,
- $o_i \in \mathbb{B}$ is the **ownership**.

- N for $N..N$,
- $*$ for $0..*$ (use with care!)

(Temporarily) Extend Signature: Associations

Only for the course of Lectures 08/09 we assume that each element in V is

- **either** a **basic type attribute** $\langle v : T, \xi, expr_0, P_v \rangle$ with $T \in \mathscr{T}$ **(as before)**,
- **or** an **association** of the form

$$\langle r : \quad \langle role_1 : C_1, \mu_1, P_1, \xi_1, \nu_1, o_1 \rangle, \\ \vdots \\ \langle role_n : C_n, \mu_n, P_n, \xi_n, \nu_n, o_n \rangle \rangle$$

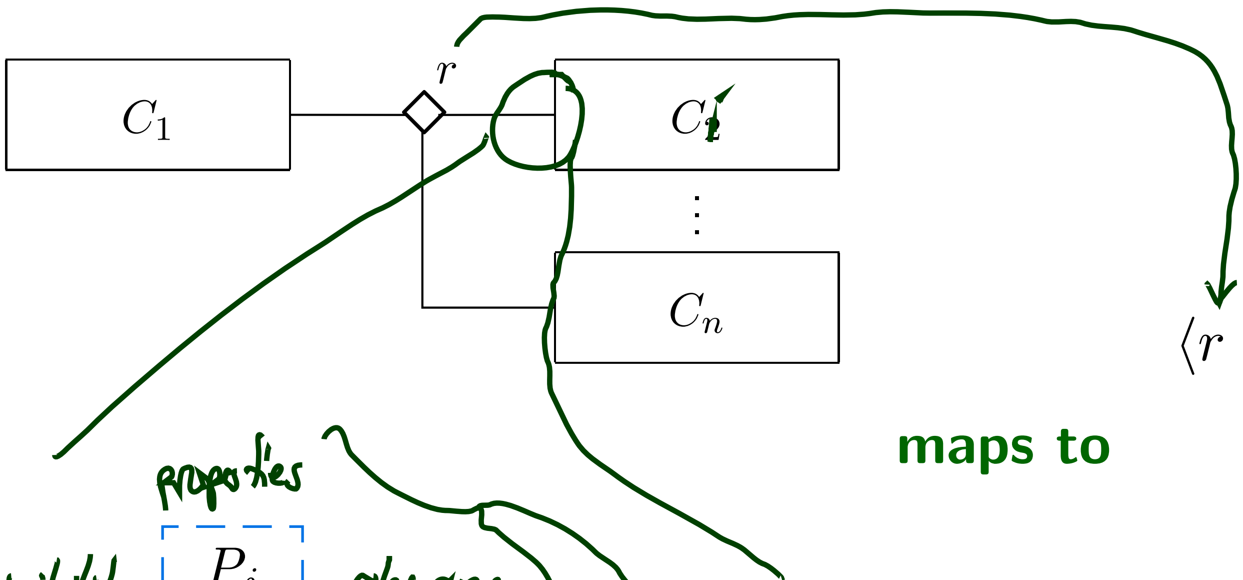
  - $n \geq 2$ (at least two ends),
  - r, $role_i$ are just **names**, $C_i \in \mathscr{C}$, $1 \leq i \leq n$,
  - the **multiplicity** μ_i is an expression of the form

$$\mu ::= N..M \mid N..* \mid \mu, \mu \qquad (N, M \in \mathbb{N})$$

  - P_i is a set of **properties (as before)**,
  - $\xi \in \{+, -, \#, \sim\}$ **(as before)**,
  - $\nu_i \in \{\times, -, >\}$ is the **navigability**,
  - $o_i \in \mathbb{B}$ is the **ownership**.

Multiplicity abbreviations:

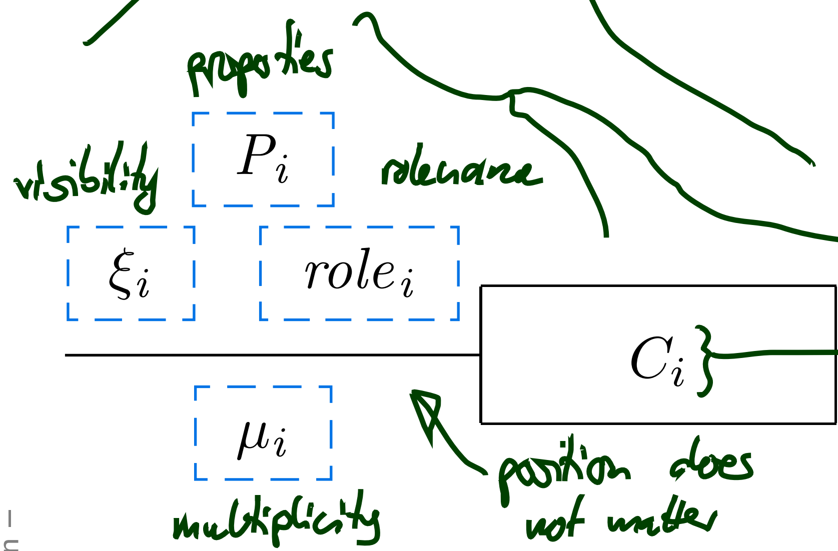
- N for $N..N$,
- $*$ for $0..*$ (use with care!)



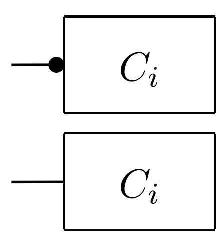
$$\langle r : \langle role_1 : C_1, \mu_1, P_1, \xi_1, \nu_1, o_1 \rangle$$

maps to $\vdots$

$$\langle role_n : C_n, \mu_n, P_n, \xi_n, \nu_n, o_n \rangle \rangle$$

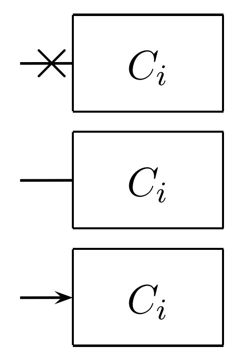


$o_i = \begin{cases} 1 & \text{, if} \\ 0 & \text{, if} \end{cases}$

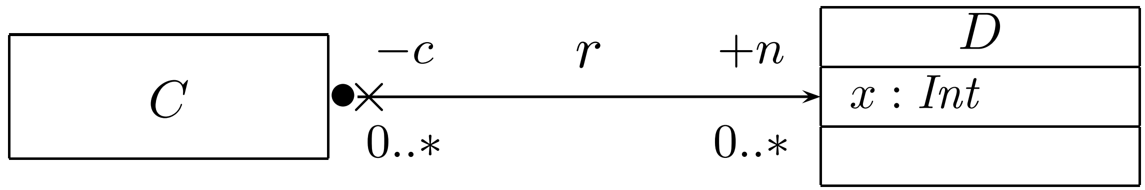


$\nu_i = \begin{cases} \times & \text{, if} \\ - & \text{, if} \\ > & \text{, if} \end{cases}$

navigability



Association Example



Signature:

$$\mathscr{S} = (\{Int\}, \{C, D\}, \{x : Int, \langle r : \langle c : C, 0..*, \{unique\}, -, \times, 1\rangle, \langle n : D, 0..*, \{unique\}, +, >, 0\rangle\rangle\}, \{C \mapsto \emptyset, D \mapsto \{x\}\})$$

assumption

What If Things Are Missing?

Most components of associations or association end may be omitted.
For instance (OMG, 2011b, 17), Section 6.4.2, proposes the following rules:

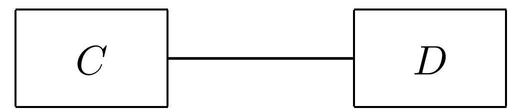
- **Name**: Use

$$A\_\langle C_1\rangle\_\cdots\_\langle C_n\rangle$$

  if the name is missing.

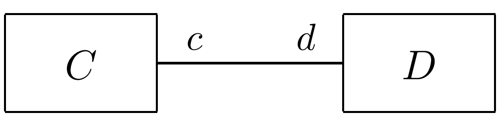
  Example:

for

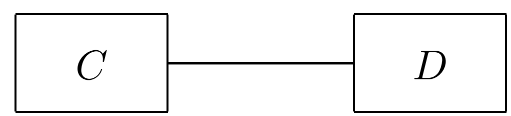


- **Reading Direction**: no default.
- **Role Name**: use the class name at that end in lower-case letters

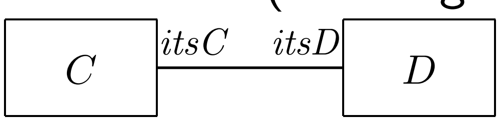
  Example:



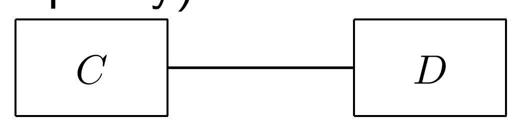
for



Other convention: (used e.g. by modelling tool Rhapsody)



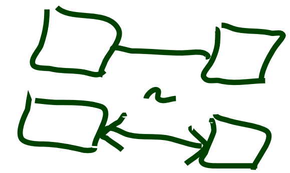
for



What If Things Are Missing?

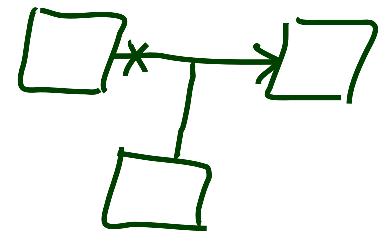
- **Multiplicity**: 1

  In my opinion, it's safer to assume $0..1$ or $*$ (for $0..*$)
  if there are no fixed, written, agreed conventions ("expect the worst").



- **Properties**: $\emptyset$

- **Visibility**: public



- **Navigability and Ownership**: not so easy. (OMG, 2011b, 43)

"Various options may be chosen for showing navigation arrows on a diagram.

In practice, it is often convenient to suppress some of the arrows and crosses and just show exceptional situations:

- *Show all arrows and x's. Navigation and its absence are made completely explicit.*
- *Suppress all arrows and x's. No inference can be drawn about navigation.*

  This is similar to any situation in which information is suppressed from a view.

- *Suppress arrows for associations with navigability in both directions, and show arrows only for associations with one- way navigability.*

  In this case, the two-way navigability cannot be distinguished from situations where there is no navigation at all; however, the latter case occurs rarely in practice."

Wait, If Omitting Things...

- ...**is causing so much trouble** (e.g. leading to misunderstanding), why does the standard say "**In practice, it is often convenient**..."?

  Is it a good idea to trade **convenience** for **precision/unambiguity**?

  It depends.

  - Convenience as such is a **legitimate goal**.
  - In UML-As-Sketch mode, precision "**doesn't matter**", so convenience (for writer) can even be a primary goal.
  - In UML-As-Blueprint mode, **precision** is the **primary goal**. And misunderstandings are in most cases annoying.

    But: (even in UML-As-Blueprint mode)

    If all associations in your model have multiplicity $*$, then it's probably a good idea not to write all these $*$'s.

    So: tell the reader about your convention and leave out the $*$'s.

Temporarily (Lecture 8/9) Extended Signature

Definition. An (Extended) Object System Signature (with Associations) is a quadruple $\mathscr{S} = (\mathscr{T}, \mathscr{C}, V, atr)$ where

- ...
- each element of V is
  - **either** a **basic type attribute** $\langle v : T, \xi, expr_0, P_v \rangle$ with $T \in \mathscr{T}$
  - **or** an **association** of the form

$$\begin{aligned} \langle r : \quad & \langle role_1 : C_1, \mu_1, P_1, \xi_1, \nu_1, o_1 \rangle, \\ & \vdots \\ & \langle role_n : C_n, \mu_n, P_n, \xi_n, \nu_n, o_n \rangle \rangle \end{aligned}$$

- ...
- $atr : \mathscr{C} \to 2^{\{v \in V \mid v:T,\ T \in \mathscr{T}\}}$
  maps each class to its set of **basic type** (!) attributes.

In other words:

- only **basic type attributes** **"belong"** to a class (may appear in $atr(C)$),
- **associations** are not "owned" by a particular class (do not appear in any $atr(C)$), but **"live on their own"**.

Associations: Semantics

Associations in General

Recall: We consider associations of the following form:

$$\langle r : \langle role_1 : C_1, \mu_1, P_1, \xi_1, \nu_1, o_1 \rangle, \dots, \langle role_n : C_n, \mu_n, P_n, \xi_n, \nu_n, o_n \rangle \rangle$$

Only these parts are relevant for extended system states:

$$\langle r : \langle role_1 : C_1, \_, P_1, \_, \_, \_ \rangle, \dots, \langle role_n : C_n, \_, P_n, \_, \_, \_ \rangle$$

(recall: we assume $P_1 = P_n = \{\texttt{unique}\}$).

The UML standard thinks of associations as **n-ary relations** which "**live on their own**" in a system state.

That is, **links** (= association instances)

- **do not** belong (in general) to certain objects (in contrast to pointers, e.g.)
- are "first-class citizens" **next to objects**,
- are (in general) **not** directed (in contrast to pointers).

Links in System States

$$\langle r : \langle role_1 : C_1, \_, P_1, \_, \_, \_\rangle, \ldots, \langle role_n : C_n, \_, P_n, \_, \_, \_\rangle$$

Only for the course of lectures 8 / 9 we change the definition of system states:

Definition. Let $\mathscr{D}$ be a structure of the (extended) signature with associations $\mathscr{S} = (\mathscr{T}, \mathscr{C}, V, atr)$.

A system state of $\mathscr{S}$ wrt. $\mathscr{D}$ is a pair (σ, λ) consisting of

- a type-consistent mapping (as before) only basic type attributes in here

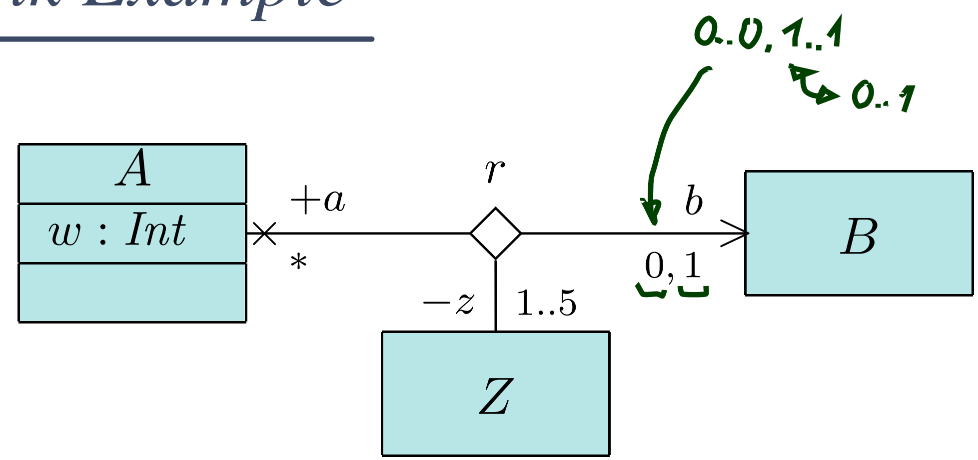
$$\sigma : \mathscr{D}(\mathscr{C}) \nrightarrow (atr(\mathscr{C}) \nrightarrow \mathscr{D}(\mathscr{T})),$$

- a mapping λ which maps each association $\langle r : \langle role_1 : C_1\rangle, \ldots, \langle role_n : C_n\rangle\rangle \in V$ to a **relation**

$$\lambda(r) \subseteq \mathscr{D}(C_1) \times \cdots \times \mathscr{D}(C_n)$$

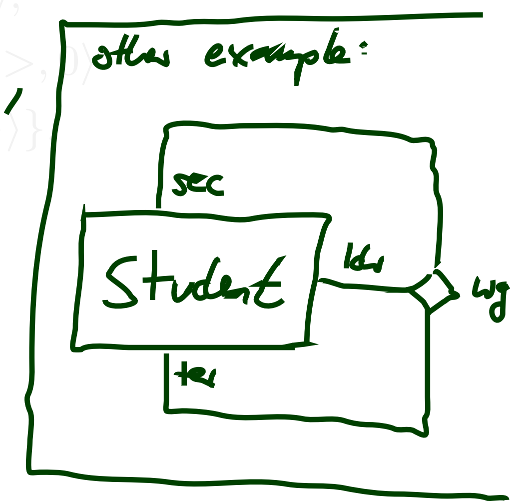
(i.e. a set of type-consistent n-tuples of identities).

Association / Link Example



Signature:

$$\mathscr{S} = (\{Int\}, \{A, B, Z\}, \{w : Int,$$
$$\langle r : \langle a : A, 0..*, +, \{unique\}, \times, 0\rangle,$$
$$\langle b : B, 0..1, +, \{unique\}, >, 0\rangle,$$
$$\langle z : Z, 1..5, -, \{unique\}, -, 0\rangle\rangle\},$$
$$\{A \mapsto \{w\}, B \mapsto \emptyset, C \mapsto \emptyset\})$$



System state:

$$\sigma : \{1_A \mapsto \{w \mapsto 0\}, 2_A \mapsto \{w \mapsto 1\}, 3_A \mapsto \{w \mapsto 2\}, 10_B \mapsto \emptyset, 11_B \mapsto \emptyset, 27_Z \mapsto \emptyset, 28_Z \mapsto \emptyset\}$$

$$\lambda : \{r \mapsto \{(1_A, 10_B, 27_Z), (2_A, 10_B, 27_Z), (1_A, 11_B, 27_Z), (5_A, 13_B, 29_Z)\}$$

Associations and OCL

OCL and Associations: Syntax

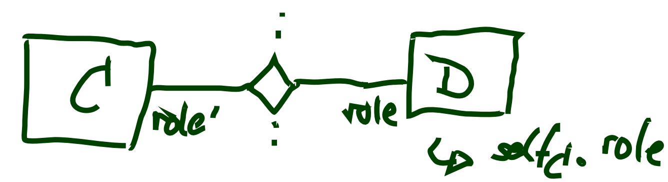
Recall: OCL syntax as introduced in Lecture 3, interesting part:

$$
\begin{array}{llll}
expr ::= \dots & \mid r_1(expr_1) & : \tau_C \to \tau_D & r_1 : D_{0,1} \in atr(C) \\
 & \mid r_2(expr_1) & : \tau_C \to Set(\tau_D) & r_2 : D_* \in atr(C)
\end{array}
$$

OCL and Associations: Syntax

Recall: OCL syntax as introduced in Lecture 3, interesting part:

$$
\begin{array}{llll}
expr ::= \dots & \mid r_1(expr_1) & : \tau_C \to \tau_D & r_1 : D_{0,1} \in atr(C) \\
 & \mid r_2(expr_1) & : \tau_C \to Set(\tau_D) & r_2 : D_* \in atr(C)
\end{array}
$$



Now becomes

$$
\begin{array}{llll}
expr ::= \dots & \mid role(expr_1) & : \tau_C \to \tau_D & \mu = 0..1 \text{ or } \mu = 1..1 \\
 & \mid role(expr_1) & : \tau_C \to Set(\tau_D) & \text{otherwise}
\end{array}
$$

if there is

$$
\langle r : \dots, \langle role : D, \mu, \_, \_, \_, \_\rangle, \dots, \langle role' : C, \_, \_, \_, \_, \_\rangle, \dots\rangle \in V \text{ or}
$$

$$
\langle r : \dots, \langle role' : C, \_, \_, \_, \_, \_\rangle, \dots, \langle role : D, \mu, \_, \_, \_, \_\rangle, \dots\rangle \in V, \quad role \neq role'.
$$

Note:

- Association name as such **does not occur** in OCL syntax, role names do.
- $expr_1$ has to denote an object of a class which "participates" in the association.

OCL and Associations Syntax: Example

$$expr ::= \ldots \quad | \; role(expr_1) \quad : \tau_C \to \tau_D \qquad \mu = 0..1 \text{ or } \mu = 1..1$$

$$| \; role(expr_1) \quad : \tau_C \to Set(\tau_D) \qquad \text{otherwise}$$

if there is

$$\langle r : \ldots, \langle role : D, \mu, \_, \_, \_, \_\rangle, \ldots, \langle role' : C, \_, \_, \_, \_, \_\rangle, \ldots\rangle \in V \text{ or}$$

$$\langle r : \ldots, \langle role' : C, \_, \_, \_, \_, \_\rangle, \ldots, \langle role : D, \mu, \_, \_, \_, \_\rangle, \ldots\rangle \in V, role \neq role'.$$

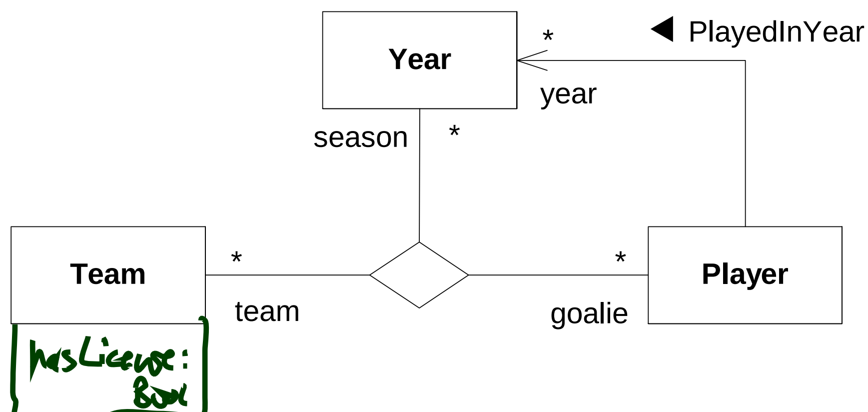


Figure 7.21 - Binary and ternary associations (OMG, 2011b, 44).

context Player inv : team -> forAll(t | t.hasLicense)

OCL and Associations: Semantics

Recall:

Assume $expr_1 : \tau_C$ for some $C \in \mathscr{C}$. Set $u_1 := I[\![expr_1]\!](\sigma, \beta) \in \mathscr{D}(T_C)$.

- $I[\![r_1(expr_1)]\!](\sigma, \beta) := \begin{cases} u & \text{, if } u_1 \in \mathrm{dom}(\sigma) \text{ and } \sigma(u_1)(r_1) = \{u\} \\ \bot & \text{, otherwise} \end{cases}$
- $I[\![r_2(expr_1)]\!](\sigma, \beta) := \begin{cases} \sigma(u_1)(r_2) & \text{, if } u_1 \in \mathrm{dom}(\sigma) \\ \bot & \text{, otherwise} \end{cases}$

Now needed:

$$I[\![role(expr_1)]\!]((\sigma, \lambda), \beta)$$

- We cannot simply write $\sigma(u)(role)$.

  Recall: $role$ is (**for the moment**) not an attribute of object u (not in $atr(C)$).
- What we have is $\lambda(r)$ (with association name r, not with role name $role$!).

  $$\langle r : \ldots, \langle role : D, \mu, \_, \_, \_, \_\rangle, \ldots, \langle role' : C, \_, \_, \_, \_, \_\rangle, \ldots\rangle$$

  But it yields a set of n-tuples, of which **some** relate u and some instances of D.
- $role$ denotes the position of the D's in the tuples constituting the value of r.

References

References

Oestereich, B. (2006). *Analyse und Design mit UML 2.1, 8. Auflage*. Oldenbourg, 8. edition.

OMG (2011a). Unified modeling language: Infrastructure, version 2.4.1. Technical Report formal/2011-08-05.

OMG (2011b). Unified modeling language: Superstructure, version 2.4.1. Technical Report formal/2011-08-06.