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?
Class diagram:

\[ C \rightarrow \text{Int} \]
\[ d : D, c : C \rightarrow 0,1 \]

Alternative presentation:

\[ C \rightarrow \text{Int} \]
\[ d : D, c : C \rightarrow 0,1 \]

Signature:

\[ \mathcal{S} = (\{ \text{Int} \}, \{ C, D \}, \{ v : \text{Int}, d : D, c : C \rightarrow 0,1 \}) \]

Example system state:

\[ \sigma = \{ 1_C \rightarrow \{ v \rightarrow 27, d \rightarrow 5_D, 7_D \}, 5_D \rightarrow \{ d \rightarrow \{ 1_C \}, 7_D \rightarrow \{ d \rightarrow \{ 1_C \} \} \} \]

Object diagram:

\[ \text{Class diagram (with ternary association):} \]

\[ A \rightarrow \text{Int} \]
\[ w : C \rightarrow \text{Int} \]
\[ r : B \rightarrow Z \]
\[ \lambda = \{ r \rightarrow \{ (1_A, 1_B, 1_Z) \} \} \]

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

Example system state:

\[ \sigma = \{ 1_A \rightarrow \{ w \rightarrow 13 \}, 1_B \rightarrow 0, 1_Z \rightarrow \emptyset \} \]

Example system state:

\[ \lambda = \{ r \rightarrow \{ (1_A, 1_B, 1_Z) \} \} \]

Object diagram: No...
Plan

(i) Study association syntax.

(ii) Extend signature accordingly.

(iii) Define \((\sigma, \lambda)\) system states with
\begin{itemize}
  \item objects in \(\sigma\) (instances of classes),
  \item links in \(\lambda\) (instances of associations).
\end{itemize}

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

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

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

Class diagram (with ternary association):

```
  A
 / \ r
0 \ 1
W - r - z
\ ---- \------\------
\    0 \   0.5 \ 1
A - r - B
   1
   ↓
Z
```

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
More Association Syntax \((\text{OMG, 2011b, 61-43})\)

More Association Syntax \((\text{OMG, 2011b, 61-43})\)
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,
- an ownership,
- and possibly a diamond. (exembe)

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 $\langle v : T, \xi, \text{expr}_v, P_v \rangle$ with $T \in \mathcal{T}$ (as before),
- or an association of the form

\[
\langle r : \langle \text{role}_1 : C_1, \mu_1, P_1, \xi_1, \nu_1, o_1 \rangle, \ldots, \langle \text{role}_n : C_n, \mu_n, P_n, \xi_n, \nu_n, o_n \rangle \rangle
\]

- $n \geq 2$ (at least two ends),
- $r, \text{role}_i$ are just names, $C_i \in \mathcal{C}, 1 \leq i \leq n$,
- the multiplicity $\mu_i$ is an expression of the form

\[
\mu ::= N..M | N..* | \mu, \mu \quad (N, M \in \mathbb{N})
\]

- $P_i$ is a set of properties (as before),
- $\xi \in \{+, -, #, \sim\}$ (as before),
- $\nu_i \in \{\times, \sim, >\}$ 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 $T \in \mathcal{T}$ (as before),
- or an association of the form

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

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

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

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

From Association Lines to Extended Signatures

<table>
<thead>
<tr>
<th>Multiplicity abbreviations:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N$ for $N..N$,</td>
</tr>
<tr>
<td>$<em>$ for $0..</em>$ (use with care!)</td>
</tr>
</tbody>
</table>

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

$$
\nu_i = \begin{cases} 
\times & \text{if } C_i \\
- & \text{if } C_i \\
> & \text{if } C_i
\end{cases}
$$
Association Example

![Diagram of association example]

**Signature:**

\[ \mathcal{A} = (\{\text{Int}\}, \{\text{C}, \text{D}\}, \{x: \text{Int}, \langle r: \langle c: \text{D}, 0..\ast, \{\text{unique}\}, -, x, 1 \rangle, \langle n: \text{C}, 0..\ast, \{\text{unique}\}, +, >, 0 \rangle \rangle, \{ \text{C} \mapsto \emptyset, \text{D} \mapsto \{x\} \}) \]

**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 \( A_{(C_1), \ldots, (C_n)} \) if the name is missing.
  
  **Example:**
  
  \[ \begin{array}{c}
  C \quad \leftarrow \text{A}_{\ldots D} \\
  \end{array} \quad \begin{array}{c}
  D \\
  \end{array} \quad \text{for} \quad \begin{array}{c}
  C \\
  \end{array} \quad \begin{array}{c}
  D \\
  \end{array} \]

- **Reading Direction:** no default.

- **Role Name:** use the class name at that end in lower-case letters.
  
  **Example:**
  
  \[ \begin{array}{c}
  C \quad e \quad d \\
  \end{array} \quad \begin{array}{c}
  D \\
  \end{array} \quad \text{for} \quad \begin{array}{c}
  C \\
  \end{array} \quad \begin{array}{c}
  D \\
  \end{array} \]

  **Other convention:** (used e.g. by modelling tool Rhapsody)
  
  \[ \begin{array}{c}
  C \quad \text{has} \quad \text{has} \\
  \end{array} \quad \begin{array}{c}
  D \\
  \end{array} \quad \text{for} \quad \begin{array}{c}
  C \\
  \end{array} \quad \begin{array}{c}
  D \\
  \end{array} \]
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**: ∅

- **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.
Definition. An (Extended) Object System Signature (with Associations) is a quadruple \( \mathcal{S} = (\mathcal{F}, \mathcal{C}, V, \text{atr}) \) where

- ... 
- each element of \( V \) is
  - either a basic type attribute \( \langle v : T, \xi, \text{expr}_0, P_v \rangle \) with \( T \in \mathcal{F} \)
  - or an association of the form
    \[
    \langle r : \langle \text{role}_1 : C_1, \mu_1, P_1, \xi_1, \nu_1, o_1 \rangle, \\
    \quad \ldots \\
    \quad \langle \text{role}_n : C_n, \mu_n, P_n, \xi_n, \nu_n, o_n \rangle \rangle
    \]
- ... 
- \( \text{atr} : \mathcal{C} \to 2^{\{v \in V \mid v : T, T \in \mathcal{F} \}} \)
  - maps each class to its set of basic type (!) attributes.

In other words:
- only basic type attributes “belong” to a class (may appear in \( \text{atr}(C) \)),
- associations are not “owned” by a particular class (do not appear in any \( \text{atr}(C) \)),
  - but “live on their own”.

**Associations: Semantics**
**Associations in General**

**Recall:** We consider associations of the following form:

\[
\langle r : \langle \text{role}_1 : C_1, \mu_1, P_1, \xi_1, \nu_1, o_1 \rangle, \ldots, \langle \text{role}_n : C_n, \mu_n, P_n, \xi_n, \nu_n, o_n \rangle \rangle
\]

Only these parts are relevant for extended system states:

\[
\langle r : \langle \text{role}_1 : C_1, P_1, \ldots \rangle, \ldots, \langle \text{role}_n : C_n, P_n, \ldots \rangle \rangle
\]

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

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**

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

**Definition.** Let \( \mathcal{D} \) be a structure of the (extended) signature with associations \( \mathcal{S} = (\mathcal{F}, \mathcal{C}, V, \text{atr}) \).

A **system state** of \( \mathcal{S} \) wrt. \( \mathcal{D} \) is a pair \((\sigma, \lambda)\) consisting of

- a type-consistent mapping (as before)
  \[
  \sigma : \mathcal{D}(\mathcal{C}) \mapsto (\text{atr}(\mathcal{C}) \mapsto \mathcal{D}(\mathcal{F}))
  \]

- a mapping \( \lambda \) which maps each association \( \langle r : \langle \text{role}_1 : C_1 \rangle, \ldots, \langle \text{role}_n : C_n \rangle \rangle \in V \) to a **relation**
  \[
  \lambda(r) \subseteq \mathcal{D}(C_1) \times \cdots \times \mathcal{D}(C_n)
  \]
  (i.e. a set of type-consistent \( n \)-tuples of identities).
**Association / Link Example**

\[ A \xrightarrow{-z} 1.5 Z \xrightarrow{b} B \]

**Signature:**

\[ \mathcal{Y} = \{ \{ h:1 \}, \{ a, b, 2 \}, \{ w:1, 6 \}, \{ r:1, 0 \} \} \]

\[ \mathcal{Z} = \{ \{ a:1 \}, \{ b:1 \}, \{ z:1 \} \} \]

**System state:**

\[ \sigma: \{ a \mapsto \{ w:10 \} \}
\{ 2a \mapsto \{ w:4 \} \}
\{ 3a \mapsto \{ w:2 \} \}
\{ 16 \mapsto \{ b:1 \} \}
\{ 21 \mapsto \{ b:2, 6 \} \} \]

**\( \lambda \):**

\[ \{ r \mapsto \{ (a, 10, 23) \}
\{ (2a, 10, 22) \}
\{ (2b, 11, 23) \}
\{ (3a, 23, 2g) \} \} \]

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

\[ \text{expr ::= ... | } r_1(\text{expr}_1) : \tau_C \rightarrow \tau_D \]  
\[ r_1 : D_0,1 \in \text{atr}(C) \]  
\[ | r_2(\text{expr}_1) : \tau_C \rightarrow \text{Set}(\tau_D) \]  
\[ r_2 : D_*,1 \in \text{atr}(C) \]

Now becomes

\[ \text{expr ::= ... | role(\text{expr}_1) : \tau_C \rightarrow \tau_D } \]  
\[ \mu = 0..1 \text{ or } 1..1 \]  
\[ | \text{role(\text{expr}_1) : } \tau_C \rightarrow \text{Set}(\tau_D) \]  
\[ \text{otherwise} \]

if there is

\[ \langle r : \ldots, \langle \text{role} : D, \mu \ldots \rangle, \ldots, \langle \text{role} : C, \ldots \rangle, \ldots \rangle \in V \text{ or } \]
\[ \langle r : \ldots, \langle \text{role} : C, \ldots \rangle, \ldots, \langle \text{role} : D, \mu \ldots \rangle, \ldots \rangle \in V, \]  
\[ \text{role } \neq \text{ role'}. \]

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

\[
\begin{align*}
\text{expr} &::= \ldots | \text{role} (\text{expr}_1) : \tau_C \rightarrow \tau_D & \mu = 0..1 \\
& & \text{otherwise}
\end{align*}
\]

if there is
\[
\langle r : \ldots , \langle \text{role} : D, \mu, \ldots \rangle, \ldots \rangle \in V \text{ or } \\
\langle r : \ldots , \langle \text{role} : C, \ldots \rangle, \ldots \rangle \in V, \text{role} \neq \text{role}'.
\]

\[\text{context Players inv: team} \rightarrow \text{nat}(\{1, \ldots, \text{holice}\})\]

OCL and Associations: Semantics

Recall:

Assume \(\text{expr}_1 : \tau_C\) for some \(C \in \mathcal{E}\). Set \(u_1 := I[\text{expr}_1](\sigma, \beta) \in \mathcal{D}(T_C)\).

- \(I[r_1(\text{expr}_1)](\sigma, \beta) := \begin{cases} u, & \text{if } u_1 \in \text{dom}(\sigma) \text{ and } \sigma(u_1)(r_1) = \{u\} \\ \perp, & \text{otherwise} \end{cases}\)
- \(I[r_2(\text{expr}_1)](\sigma, \beta) := \begin{cases} \sigma(u_1)(r_2), & \text{if } u_1 \in \text{dom}(\sigma) \\ \perp, & \text{otherwise} \end{cases}\)

Now needed:

\[I[\text{role} (\text{expr}_1)]((\sigma, \lambda), \beta)\]

- We cannot simply write \(\sigma(u)(\text{role})\).
  
Recall: role is (for the moment) not an attribute of object \(u\) (not in \(\text{atr}(C)\)).

- What we have is \(\lambda(r)\) (with association name \(r\), not with role name role!).

\(\langle r : \ldots , \langle \text{role} : D, \mu, \ldots \rangle, \ldots \rangle\)

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

