### Contents & Goals

**Last Lectures:**
- class diagram — except for associations; visibility within OCL typesystem

**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 rolename? What’s it good for?
  - What’s “multiplicity”? How did we treat them semantically?
  - What’s the difference between “aggregation” and “composition”?
  - What’s “reading direction”, “navigability”, “ownership”, . . .?

**Content:**
- Complete visibility
- Study concrete syntax for “associations”.
  - (Temporarily) extends signature, define mapping from diagram to signature.
- Study effect on OCL.
- Wheredo we put OCL constraints?

---

### Casting in the Type System

**One Possible Extension: Implicit Casts**

- We may wish to have
  $\exists$ $\lambda:\text{false}:\text{Bool}$
  In other words: We may wish that the typesystem allowstouse 0, 1: Int instead of true and false without breaking well-typedness.

- Then just have a rule:
  - ($\text{Cast}$)
  - $A \vdash \text{expr}:\text{Int}$
  - $A \vdash \text{expr}:\text{Bool}$

- With ($\text{Cast}$) (and (Int), and (Bool), and (Fun 0)), we can derivethesentence ($\ast$), thus conclude well-typedness.

- But: that’s only half of the story— the definition of the interpretation function $I$ that we have is not prepared, it doesn’t tell us what ($\ast$) means...

### Implicit Casts Cont’d

- So, why isn’t there an interpretation for ($1 \land \text{false}$)?

- First of all, we have (syntax)
  $\text{expr}_1 \land \text{expr}_2: \text{Bool} \times \text{Bool} \rightarrow \text{Bool}$

- Thus, \[ I(\land): I(\text{Bool}) \times I(\text{Bool}) \rightarrow I(\text{Bool}) \] where \[ I(\text{Bool}) = \{ \text{true}, \text{false} \} \cup \{ \bot_{\text{Bool}} \} \].

- By definition, \[ I(\land)_1(\text{true}, \text{false}) = I(\land)_2(\text{true}) = I(\text{true}) \]
  \[ I(\land)_2(\text{false}, \text{false}) = I(\text{false}) \]
  and therewe’re stuck.

### Implicit Casts: Quickfix

- Explicitly define $I(\land)_1$ and ($\text{expr}_1, \text{expr}_2$):
  - \[ \{ \text{b}_1 \land \text{b}_2, \text{if } \text{b}_1 \neq \bot_{\text{Bool}} \neq \text{b}_2, \text{otherwise} \} \]

- where
  - $\text{b}_1 := \text{toBool}(I(\land)_1(\text{expr}_1))$,
  - $\text{b}_2 := \text{toBool}(I(\land)_2(\text{expr}_2))$,

- and where $\text{toBool}: I(\text{Int}) \cup I(\text{Bool}) \rightarrow I(\text{Bool})$:
  - $x \mapsto \begin{cases} \text{true}, & \text{if } x \in \{ \text{true} \} \cup I(\text{Int}) \setminus \{ 0, \bot_{\text{Int}} \} \\ \text{false}, & \text{if } x \in \{ \text{false}, 0 \} \\ \bot_{\text{Bool}}, & \text{otherwise} \end{cases}$
In particular: prepare to treat "protected" later (when doing inheritance).

• $w \in w \oplus A = \text{where}$

\begin{align*}
\text{Accessing attribute} & \quad \text{of a} \\
\text{is object of via expression} & \quad \text{C}_{\tau}:
\end{align*}

\begin{align*}
\text{is well-typed if} & \quad \text{w or} \\
\text{whether an expression involving attributes with visibility is} & \quad \text{is type} \\
\text{is a problem?} & \quad \text{or}
\end{align*}

\begin{align*}
\text{Context Intro} & \quad \text{in a type environment, then it is in} \\
\text{operator application:} & \quad \tau:
\end{align*}

\begin{align*}
\text{and} & \quad \text{in context}
\end{align*}

\begin{align*}
\text{Altthesis Alikes in Context?} & \quad \text{expr}, \ldots, \text{expr}
\end{align*}

\begin{align*}
\text{w} \in (\omega_{1}/\nu_{1}) & \quad \text{A,B}_{\tau}:
\end{align*}
Attribute Access in Context Example

\[ A \vdash expr : \tau \]

\[ A,B \vdash expr : \tau \]

\[ Attr_1 \]

\[ A,B \vdash expr_1 : \tau \]

\[ C \]

\[ A,B \vdash v(\text{expr}_1) : \tau, \langle v : \tau, \xi, \text{expr}_0, P/PBV \rangle \in \text{atr}(C), \xi = + \text{or } \xi = - \]

\[ x = 3 \]

Example:

\[ \text{self} : \tau \]

\[ C \vdash \text{self} . r . v > 0 \]

The Semantics of Visibility

- **Observation:**
  - Whether an expression does or does not respect visibility is a matter of well-typedness only.
  - We only evaluate \( I \) to well-typed expressions.
  - \( \Rightarrow \) We need not adjust the interpretation function \( I \) to support visibility.

What is Visibility Good For?

- Visibility is a property of attributes—
  - Is it useful to consider it in OCL?
  - In other words: given the picture above, is it useful to state the following invariant (even though \( x \) is private in \( D \))
    \[ \text{inv}: n.x > 0? \]
  - It depends.
    \( \text{cf.} \ [?, \text{Sect.12 and 9.2.2}] \)

- Constraints and pre/postconditions:
  - Visibility is sometimes not taken into account. To state "global" requirements, it may be adequate to have a "global view", be able to look into all objects.
  - But: visibility supports "narrow interfaces", "information hiding", and similar good design practices. To be more robust against changes, try to state requirements only in the terms which are visible to a class.

  **Rule of thumb:**
  - If attributes are important to state requirements on design models, leave them public or provide get-methods (later).

- Guards and operation bodies:
  - If in doubt, yes (\( \Rightarrow \) take visibility into account).
  - Any so-called action language typically takes visibility into account.

Recapitulation

- We extended the typesystem for:
  - casts (requires change of \( I \)) and
  - visibility (no change of \( I \)).

  \( \Rightarrow \) Later: navigability of associations.

Good:
- Well-typedness is decidable for these type-systems. That is, we can have automatic tools that check whether OCL expressions in a model are well-typed.
What If Things Are Missing?

In practice, it is often convenient to suppress some of the arrows and crosses and just show what is needed. For example, if the multiplicity is 1, it is often sufficient to suppress the cross and show the arrow only.

Visibility

Convenience as such is a legitimate goal. It depends, however, why does the standards say "expect the worst." It depends, why do we assume versatility? One of the reasons is that we want to avoid misunderstandings. Convenience as such is a legitimate goal. So, it is a good idea not to write all these things on the diagram.

Roles

When in doubt about the role name, it is often best to use the class name at that end in lower-case letters.

For instance, if an association name is "Customer," we only write it as \textit{customer}. But we must be careful about the role names. If there are no role names, it is best to use the class name at that end in lower-case letters.


d_{1} \rightarrow \tau_{V}, \tau_{V} = 1 \rightarrow \emptyset_{C}\{\textrole{RoleName}\} \mapsto \emptyset_{C}, \emptyset_{D} \mapsto \emptyset_{C}\{\textrole{RoleName}\} 

\text{Associative Example

Formally, we only call (C,D) \textrole{RoleName} if OmittingThings... Wait, If OmittingThings... In practice, it is often convenient to suppress some of the arrows and crosses and just show what is needed.

Properties

In my opinion, it's safe to assume that the name is missing. Is it a good idea to trade precision/unambiguity for convenience? This is similar to any situation in which information is suppressed from a view.

Properties

For instance, if an association name is "Customer," we only write it as \textit{customer}. But we must be careful about the role names. If there are no role names, it is best to use the class name at that end in lower-case letters.

Microstructure

In practice, it is often convenient to suppress some of the arrows and crosses and just show what is needed. For example, if the multiplicity is 1, it is often sufficient to suppress the cross and show the arrow only.

Microstructure

Convenience as such is a legitimate goal. It depends, however, why does the standards say "expect the worst." It depends, why do we assume versatility? One of the reasons is that we want to avoid misunderstandings. Convenience as such is a legitimate goal. So, it is a good idea not to write all these things on the diagram.

Roles

When in doubt about the role name, it is often best to use the class name at that end in lower-case letters.

For instance, if an association name is "Customer," we only write it as \textit{customer}. But we must be careful about the role names. If there are no role names, it is best to use the class name at that end in lower-case letters.