well-typed on the class of objects for which it is evaluated.

Contents & Goals

- Study effect on OCL
- What's a role name? What's it good for?
- completed class diagrams... except for visibility and associations
- Last Lectures:
The Semantics of Visibility

Observation:

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

What is Visibility Good For?

Visibility is a property of attributes —

\[ C \xrightarrow{r_0, 1} D \]

\[ \langle v : D_0, 1, \xi, \text{expr}_0, P_C \rangle \in \text{atr}(C) \]

\[ \text{expr}_1(w) : \tau \]

\[ \tau \]

\[ \text{expr}_1(w) : \tau \]

\[ \text{expr}_1(w) : \tau \]

\[ \text{expr}_1(w) : \tau \]

\[ \text{expr}_1(w) : \tau \]

\[ C_1 = C_2 \] or \( \xi = + \)

Constraints and pre/post conditions:

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 (\( = \) do take visibility into account). Any so-called action language typically takes visibility into account.

References


