If then new aspects are irrelevant (for a given context), we write
\(\langle\cdot\rangle\in C,S\{\langle\cdot\rangle\rangle\)

\(\tau\in\mathcal{C}\) or \(\tau\in\mathcal{A}\), if details are irrelevant.

\(\{\langle\cdot\rangle\rangle\rangle\)

\(\mathcal{C}\) or \(\mathcal{A}\), if details are irrelevant.

\(\mathcal{C}\) or \(\mathcal{A}\), if details are irrelevant.

\(\tau\in\mathcal{C}\) or \(\tau\in\mathcal{A}\), if details are irrelevant.

\(\{\langle\cdot\rangle\rangle\rangle\)

\(\mathcal{C}\) or \(\mathcal{A}\), if details are irrelevant.

\(\{\langle\cdot\rangle\rangle\rangle\rangle\)

\(\mathcal{C}\) or \(\mathcal{A}\), if details are irrelevant.

\(\{\langle\cdot\rangle\rangle\rangle\rangle\rangle\)

\(\mathcal{C}\) or \(\mathcal{A}\), if details are irrelevant.

\(\{\langle\cdot\rangle\rangle\rangle\rangle\rangle\rangle\)

\(\mathcal{C}\) or \(\mathcal{A}\), if details are irrelevant.
(ii) Well-typedness in a type environment \( \tau \):
\[
\text{If } \tau \vdash n : \tau \text{ then } \tau \text{ which are of type } \tau \text{ to express operations is the application of } expr \cdot \bullet
\]

(i) Universal well-typedness:
\[
\text{if } \tau \vdash \text{true} : \tau \text{ then } \tau \text{ boolean constant is of type } \tau \text{ logical variable if and only if we can derive } expr \rightarrow \bullet)
\]

Excursus: Type Theory (cf. Thiemann, 2008)
function \( I \in \mathbb{C}_2 \to \mathbb{C}_3 \), \( I \) and \( (I) = \mathit{false} \) and \( 1\mathbb{C}_2\mathbb{0} \), \( \mathbb{I} \in \{x, \mathit{false}\} \), \( \mathit{false} \times \mathbb{I} \rightarrow \mathbb{Int} \), \( \mathit{true} \times \mathbb{I} \rightarrow \mathbb{Int} \). Thus, \( \mathit{false} \) and \( \mathit{true} \)

\( \mathbb{I} \) to \( \mathit{false} \) without breaking well-typedness.

In other words, \( \mathbb{I} \) to have \( \mathit{false} \) and \( \mathit{true} \)

\( \mathbb{I} \) to \( \mathit{false} \) and \( \mathit{true} \)

Thus, \( \mathbb{I} \) to \( \mathit{false} \) and \( \mathit{true} \)

So, why isn't there an interpretation for

Example

Casting in the Type System

First Recapitulation

Iterate Example

False Example.
Visibility—The Intuition

Example
Attribute Access in Context Example

\[ A \vdash \text{expr} : \tau \]
\[ (\text{Attr 1}) \quad A \vdash \text{expr} : \tau \]
\[ A \vdash \text{v} (\text{expr 1}) : \tau, \langle \text{v} : \tau, \xi, \text{expr 0}, P / \text{BV} \rangle \in \text{atr} (C) \]
\[ \xi = +, \text{or} \xi = - \]
\[ C = B - v : \text{Int} \]

Example:

\[ \text{self} : \tau \quad A \vdash \text{self}.r.v > 0 \]

The Semantics of Visibility

• Observation:
  • Whether an expression does or does not respect visibility is a matter of well-typedness only.
  • We only evaluate (\( = \) 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—
  • 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 \( D \))
    \[ \text{context } C \quad \text{inv} : n.x > 0? \]
  • It depends.
  • Constraints and pre/postconditions:
    • Visibility is sometimes not taken into account. To state "global" requirements, it might 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 (\( = \) take visibility into account).
    • Any so-called action language typically takes visibility into account.

Recapitulation

Class Diagrams

• We extended the typesystem for:
  • casts (requires change of \( I \)) and
  • visibility (no change of \( I \)).
• Later: navigability of associations.

Good:
• Well-typedness is decidable for these typesystems. That is, we can have automatic tools that check, whether OCL expressions in a model are well-typed.

References
References

