Software Design, Modelling and Analysis in UML

Lecture 08: Class Diagrams II

2014-11-20

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

Albert-Ludwigs-Universität Freiburg, Germany

Contents & Goals

Last Lectures:

- completed class diagrams... except for visibility and associations

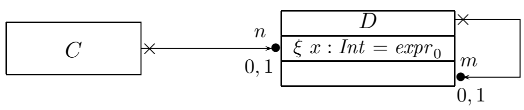
This Lecture:

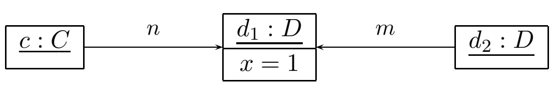
- **Educational Objectives:** Capabilities for following tasks/questions.
  - Please explain this class diagram with associations.
  - Which annotations of an association arrow are semantically relevant?
  - What's a role name? What's it good for?
  - What is "multiplicity"? How did we treat them semantically?
  - 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?

Visibility Cont'd

The Intuition by Example

$$\mathscr{S} = (\{Int\}, \{C, D\}, \{n : D_{0,1}, m : D_{0,1}, \langle x : Int, \xi, expr_0, \emptyset\rangle\}, \{C \mapsto \{n\}, D \mapsto \{x, m\}\}$$





Assume $\boxed{w_1 : \tau_C}$ and $\boxed{w_2 : \tau_D}$ are logical variables. **Which** of the following **syntactically correct** (?) OCL expressions **shall** we consider to be **well-typed**?

by class ~ OCL, C++, ... !

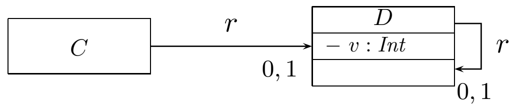
ξ of x:	public	private	protected	package
$w_1 . n . x = 0$ $x(n(w_1))$	✔ 卌 | ✘ ? ||	✔ 0 ✘ 卌 ||| ?	later	not
$w_2 . m . x = 0$ $x(m(w_2))$	✔ 卌 |||| ✘ 0 ?	✔ ||| ✘ ||| ?	later	not

by object

Context

$$\mathscr{S} = (\{Int\}, \{C, D\}, \{r : D_{0,1}, \langle v : Int, \xi, \star, \emptyset\rangle\}, \{C \mapsto \{r\}, D \mapsto \{v, r\}\}$$

- **Example**:



$$self_D \,.\, v > 0 \quad ✓$$

$$self_D \,.\, r \,.\, v > 0 \quad ✓$$

$$self_C \,.\, r \,.\, v > 0 \quad ✗$$

- That is, whether an expression involving attributes with visibility is well-typed **depends** on the class of objects "for which it is evaluated".

Attribute Access in Context

Recall: attribute access in OCL Expressions, $C, D \in \mathscr{C}$.

$v(expr_1)$	$: \tau_C \to \tau(v)$	• $v : \tau(v) \in atr(C)$, $\tau(v) \in \mathscr{T}$,
$r_1(expr_1)$	$: \tau_C \to \tau_D$	• $r_1 : D_{0,1} \in atr(C)$,
$r_2(expr_1)$	$: \tau_C \to Set(\tau_D)$	• $r_2 : D_* \in atr(C)$,

New rules:

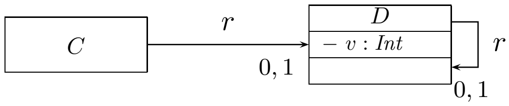
$v(w)$	$: \tau_C \to \tau(v)$	$\langle v : \tau, \xi, expr_0, P_{\mathscr{C}}\rangle \in atr(C)$
$r_1(w)$	$: \tau_C \to \tau_D$	$\langle r_1 : D_{0,1}, \xi, expr_0, P_{\mathscr{C}}\rangle \in atr(C)$
$r_2(w)$	$: \tau_C \to Set(\tau_D)$	$\langle r_1 : D_*, \xi, expr_0, P_{\mathscr{C}}\rangle \in atr(C)$

$v(expr_1(w)) \quad : \tau_{C_2} \to \tau(v) \qquad \langle v : \tau, \xi, expr_0, P_{\mathscr{C}}\rangle \in atr(C)$,
$expr_1(w) : \tau_{C_2}$, $w : \tau_{C_1}$, and $C_1 = C_2$ or $\xi = +$

$r_1(expr_1(w)) \quad : \tau_{C_2} \to \tau_D \qquad \langle v : D_{0,1}, \xi, expr_0, P_{\mathscr{C}}\rangle \in atr(C)$,
$expr_1(w) : \tau_{C_2}$, $w : \tau_{C_1}$, and $C_1 = C_2$ or $\xi = +$

Example

1	$v(w)$	$: \tau_C \to \tau(v)$	$\langle v : \tau, \xi, expr_0, P_{\mathscr{C}} \rangle \in atr(C)$
2	$r_1(w)$	$: \tau_C \to \tau_D$	$\langle r_1 : D_{0,1}, \xi, expr_0, P_{\mathscr{C}} \rangle \in atr(C)$
3	$v(expr_1(w))$	$: \tau_{C_2} \to \tau(v)$	$\langle v : \tau, \xi, expr_0, P_{\mathscr{C}} \rangle \in atr(C)$, $expr_1(w) : \tau_{C_2}$, $w : \tau_{C_1}$, and $C_1 = C_2$ or $\xi = +$
4	$r_1(expr_1(w))$	$: \tau_{C_2} \to \tau_D$	$\langle v : D_{0,1}, \xi, expr_0, P_{\mathscr{C}} \rangle \in atr(C)$, $expr_1(w) : \tau_{C_2}$, $w : \tau_{C_1}$, and $C_1 = C_2$ or $\xi = +$



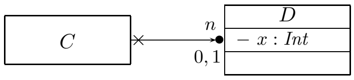
- $self_D \, . \, v > 0$ $\rightsquigarrow$ $v(self_D) > 0$ ok by 1
- $self_D \, . \, r \, . \, v > 0$ $\rightsquigarrow$ $v(r(self_D)) > 0$
  $\underbrace{r(self_D)}^{: \tau_{C_2} (= \tau_D)}$ ok by 2
  $v(r(\underbrace{self_D}_{: \tau_D}))$ ok by 3 because $C_2 = D$
- $self_C \, . \, r \, . \, v > 0$ $\rightsquigarrow$ $v(r(self_C)) > 0$
  $\underbrace{r(self_C)}_{: \tau_{C_2} (= \tau_D)}$ ok by 2 $\quad \xi = -$
  $v(r(self_C))$ not ok and $C_2 \neq C$

The Semantics of Visibility

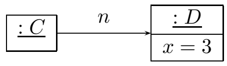
- **Observation**:
  - Whether an expression **does** or **does not** respect visibility is a matter of well-typedness **only**.
  - We only evaluate (= apply I to) **well-typed** expressions.

$\rightarrow$ 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 diagram above, **is it useful** to state the following invariant (even though x is private in D)

$$\text{context } C \text{ inv} : n.x > 0 \ ?$$

It depends. (cf. [OMG, 2006], Sect. 12 and 9.2.2)

- **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

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

[OMG, 2006] OMG (2006). Object Constraint Language, version 2.0. Technical Report formal/06-05-01.

[OMG, 2007a] OMG (2007a). Unified modeling language: Infrastructure, version 2.1.2. Technical Report formal/07-11-04.

[OMG, 2007b] OMG (2007b). Unified modeling language: Superstructure, version 2.1.2. Technical Report formal/07-11-02.