Software Design, Modelling and Analysis in UML

Lecture 04: OCL Cont'd, Object Diagrams

2013-11-04

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

Albert-Ludwigs-Universität Freiburg, Germany

Contents & Goals

Last Lecture:

- OCL Syntax

This Lecture:

- **Educational Objectives:** Capabilities for following tasks/questions.
  - What is an object diagram? What are object diagrams good for?
  - When is an object diagram called partial? What are partial ones good for?
  - When is an object diagram an object diagram (wrt. what)?
  - Is this an object diagram wrt. to that other thing?
  - How are system states and object diagrams related?
  - What does it mean that an OCL expression is satisfiable?
  - When is a set of OCL constraints said to be consistent?
  - Can you think of an object diagram which violates this OCL constraint?
- **Content:**
  - OCL Semantics
  - Object Diagrams
  - Example: Object Diagrams for Documentation
  - OCL: consistency, satisfiability

OCL Semantics [OMG, 2006]

The Task

OCL Syntax 1/4: Expressions

$$
\begin{array}{lll}
\mathit{expr} ::= & & \\
\quad w & : \tau(w) & \\
\quad \mid \mathit{expr}_1 =_\tau \mathit{expr}_2 & : \tau \times \tau \to \mathit{Bool} & \\
\quad \mid \mathsf{oclIsUndefined}_\tau(\mathit{expr}_1) & : \tau \to \mathit{Bool} & \\
\quad \mid \{\mathit{expr}_1, \dots, \mathit{expr}_n\} & : \tau \times \dots \times \tau \to \mathit{Set}(\tau) & \\
\quad \mid \mathsf{isEmpty}(\mathit{expr}_1) & : \mathit{Set}(\tau) \to \mathit{Bool} & \\
\quad \mid \mathsf{size}(\mathit{expr}_1) & : \mathit{Set}(\tau) \to \mathit{Int} & \\
\quad \mid \mathsf{allInstances}_C & : \mathit{Set}(\tau_C) & \\
\quad \mid v(\mathit{expr}_1) & : \tau_C \to \tau(v) & \\
\quad \mid r_1(\mathit{expr}_1) & : \tau_C \to \tau_D & \\
\quad \mid r_2(\mathit{expr}_1) & : \tau_C \to \mathit{Set}(\tau_D) &
\end{array}
$$

Where, given $\mathscr{S} = (\mathscr{T}, \mathscr{C}, V, atr)$,

- $W \supseteq \{self\}$ is a set of typed logical variables, w has type $\tau(w)$
- τ is any type from $\mathscr{T} \cup T_B \cup T_\mathscr{C}$ $\cup \{Set(\tau_0) \mid \tau_0 \in T_B \cup T_\mathscr{C}\}$
  - T_B is a set of basic types, in the following we use $T_B = \{Bool, Int, String\}$
  - $T_\mathscr{C} = \{\tau_C \mid C \in \mathscr{C}\}$ is the set of object types,
  - $Set(\tau_0)$ denotes the set-of-τ_0 type for $\tau_0 \in T_B \cup T_\mathscr{C}$ (sufficient because of "flattening" (cf. standard))
- $v : \tau(v) \in atr(C)$, $\tau(v) \in \mathscr{T}$,
- $r_1 : D_{0,1} \in atr(C)$,
- $r_2 : D_* \in atr(C)$,
- $C, D \in \mathscr{C}$.

– 03 – 2010-10-27 – Soclsyn –

7/30

- Given an OCL expression $expr$, a system state $\sigma \in \Sigma_\mathscr{S}^\mathscr{D}$, and a valuation of logical variables β, define

$$I[\![\cdot]\!](\cdot\,,\cdot) : OCLExpressions(\mathscr{S}) \times \Sigma_\mathscr{S}^\mathscr{D} \times (W \to I(\mathscr{T} \cup T_B \cup T_\mathscr{C})) \to I(Bool)$$

such that

$$I[\![expr]\!](\sigma, \beta) \in \{\mathit{true}, \mathit{false}, \bot_{Bool}\}. = I(Bool)$$

Basically business as usual...

(i) Equip each OCL (!) **basic type** with a reasonable **domain**, i.e. define function

$$I_{(i)} \text{ with } \mathrm{dom}(I) = T_B$$

(ii) Equip each **object type** τ_C with a reasonable **domain**, i.e. define function

$$I_{(ii)} \text{ with } \mathrm{dom}(I) = \tau_C$$

(most reasonable: $\mathscr{D}(C)$ determined by structure $\mathscr{D}$ of $\mathscr{S}$).

(iii) Equip each **set type** $Set(\tau_0)$ with reasonable **domain**, i.e. define function

$$I_{(iii)} \text{ with } \mathrm{dom}(I) = \{Set(\tau_0) \mid \tau_0 \in T_B \cup T_{\mathscr{C}}\}$$

(iv) Equip each **arithmetical operation** with a reasonable **interpretation** (that is, with a **function** operating on the corresponding **domains**).

$$I_{(iv)} \text{ with } \mathrm{dom}(I) = \{+, -, \leq, \ldots\}, \text{ e.g., } I(+) \in I(Int) \times I(Int) \to I(Int)$$

(v) **Set operations** similar: $I_{(v)}$ with $\mathrm{dom}(I) = \{\mathsf{isEmpty}, \ldots\}$

(vi) Equip each **expression** with a reasonable **interpretation**, i.e. define function

$$I_{(vi)} : Expr \times \Sigma_{\mathscr{S}}^{\mathscr{D}} \times (W \to I(\mathscr{T} \cup T_B \cup T_{\mathscr{C}})) \to I(Bool)$$

...except for OCL being a **three-valued logic**, and the "iterate" expression.

(i) Domains of Basic Types (of OCL)

Recall:

- $T_B = \{Bool, Int, String\}$

assume both sets disjoint

"undefined"

We set:

- $I_{(i)}(Bool) := \{true, false\} \dot{\cup} \{\bot_{Bool}\}$
- $I(Int) := \mathbb{Z} \cup \{\bot_{Int}\}$
- $I(String) := \ldots \cup \{\bot_{String}\}$

finite sequences of characters

We may omit index τ of $\bot_\tau$ if it is clear from context.

(ii) Domains of Object and (iii) Set Types

- Now we need a structure $\mathscr{D}$ of our signature $\mathscr{S} = (\mathscr{T}, \mathscr{C}, V, atr)$.
- **Recall:** $\mathscr{D}$ assigns an (infinite) domain $\mathscr{D}(C)$ to each class $C \in \mathscr{C}$.

- Let τ_C be an (OCL) **object type** for a class $C \in \mathscr{C}$.
- We set

$$I_{(ii)}(\tau_C) := \mathscr{D}(C) \,\dot{\cup}\, \{\bot_{\tau_C}\}$$

2^A is powerset of A

- Let τ be a type from $T_B \cup T_{\mathscr{C}}$.
- We set

$$I_{(iii)}(Set(\tau)) := 2^{I(\tau)} \,\dot{\cup}\, \{\bot_{Set(\tau)}\}$$

Note: in the OCL standard, only **finite** subsets of $I(\tau)$.
But infinity doesn't scare **us**, so we simply allow it.

(iv) Interpretation of Arithmetic Operations

- **Literals** map to fixed values: $\mathcal{I}_{(i)}(Bool)$

$$I_{(iv)}(\mathtt{true}) := \mathit{true}, \quad I(\mathtt{false}) := \mathit{false}, \qquad I(\mathtt{0}) := 0, \quad I(\mathtt{1}) := 1, \ldots \quad \in \mathcal{I}(Int) = \mathbb{Z} \cup \{\bot_{Int}\}$$

$$I(\mathtt{OclUndefined}_\tau) := \bot_\tau \in \mathcal{I}(\tau) \leftarrow \mathcal{I}_{(i)} \cup \mathcal{I}_{(ii)} \cup \mathcal{I}_{(iii)}$$

- **Boolean operations** (defined point-wise for $x_1, x_2 \in I(\tau)$):

$$I(=_\tau)(x_1, x_2) := \begin{cases} \mathit{true} & \text{, if } x_1 \neq \bot_\tau \neq x_2 \text{ and } x_1 = x_2 \\ \mathit{false} & \text{, if } x_1 \neq \bot_\tau \neq x_2 \text{ and } x_1 \neq x_2 \\ \bot_{Bool} & \text{, otherwise} \end{cases}$$

- **Integer operations** (defined point-wise for $x_1, x_2 \in I(Int)$):

$$I(+)(x_1, x_2) := \begin{cases} x_1 + x_2 & \text{, if } x_1 \neq \bot \neq x_2 \\ \bot & \text{, otherwise} \end{cases}$$

Note: There is a **common principle**.
Namely, the **interpretation** of an operation $\omega : \tau_1 \times \ldots \tau_n \to \tau$
is a function $I_{(iv)}(\omega) : I(\tau_1) \times \cdots \times I(\tau_n) \to I(\tau)$ on corresponding semantical domain(s).

(iv) Interpretation of OclIsUndefined

- The **is-undefined** predicate (defined point-wise for $x \in I(\tau)$):

$$I(\mathsf{oclIsUndefined}_\tau)(x) := \begin{cases} \mathit{true} & \text{, if } x = \bot_\tau \\ \mathit{false} & \text{, otherwise} \end{cases}$$

(v) Interpretation of Set Operations

Basically the same principle as with arithmetic operations...

$\{x_1, \ldots, x_n\}$ $\tau \times \cdots \times \tau \to Set(\tau)$
$= \{\}^\tau_n(x_1, \ldots, x_n)$

Let $\tau \in T_B \cup T_{\mathscr{C}}$.

- **Set comprehension** ($x_1, \ldots, x_n \in I(\tau)$):

$$(I(\{\}^\tau_n))(x_1, \ldots, x_n) := \{x_1, \ldots, x_n\}$$

  for all $n \in \mathbb{N}_0$

- **Empty-ness check** ($x \in I(Set(\tau))$):

$$I(\mathit{isEmpty}^\tau)(x) := \begin{cases} \mathit{true} & \text{, if } x = \emptyset \\ \bot_{Bool} & \text{, if } x = \bot_{Set(\tau)} \\ \mathit{false} & \text{, otherwise} \end{cases}$$

- **Counting** ($x \in I(Set(\tau))$): cardinality

$$(I(\mathit{size}^\tau))(x) := |x| \text{ if } x \neq \bot_{Set(\tau)} \text{ and } \bot_{Int} \text{ otherwise}$$

(vi) Putting It All Together

OCL Syntax 1/4: Expressions

$$
\begin{array}{ll}
expr ::= & \\
\quad w & : \tau(w) \\
\quad \mid expr_1 =_\tau expr_2 & : \tau \times \tau \to Bool \\
\quad \mid \mathsf{oclIsUndefined}_\tau(expr_1) & : \tau \to Bool \\
\quad \mid \{expr_1, \dots, expr_n\} & : \tau \times \dots \times \tau \to Set(\tau) \\
\quad \mid \mathsf{isEmpty}(expr_1) & : Set(\tau) \to Bool \\
\quad \mid \mathsf{size}(expr_1) & : Set(\tau) \to Int \\
\quad \mid \mathsf{allInstances}_C & : Set(\tau_C) \\
\quad \mid v(expr_1) & : \tau_C \to \tau(v) \\
\quad \mid r_1(expr_1) & : \tau_C \to \tau_D \\
\quad \mid r_2(expr_1) & : \tau_C \to Set(\tau_D)
\end{array}
$$

Where, given $\mathscr{S} = (\mathscr{T}, \mathscr{C},$

- $W \supseteq \{self\}$ is a set of t
  logical variables, w has
- τ is any type from $\mathscr{T} \cup$
  $\cup \{Set(\tau_0) \mid \tau_0 \in T_B \cup$
  - T_B is a set of basic
    the following we use
    $T_B = \{Bool, Int, St$
  - $T_\mathscr{C} = \{\tau_C \mid C \in \mathscr{C}\}$
    set of object types,
  - $Set(\tau_0)$ denotes the
    set-of-τ_0 type for
    $\tau_0 \in T_B \cup T_\mathscr{C}$
    (sufficient because o
    "flattening" (cf. star
- $v : \tau(v) \in atr(C)$, $\tau(v)$
- $r_1 : D_{0,1} \in atr(C)$,
- $r_2 : D_* \in atr(C)$,
- $C, D \in \mathscr{C}$.

– 03 – 2010-10-27 – Soclsyn –

OCL Syntax 2/4: Constants, Arithmetical Operat

For example:

$$
\begin{array}{lll}
expr ::= & \dots & \\
& \mid \mathsf{true}, \mathsf{false} & : Bool \\
& \mid expr_1 \ \{\mathsf{and}, \mathsf{or}, \mathsf{implies}\} \ expr_2 & : Bool \times Bool \to \\
& \mid \mathsf{not} \ expr_1 & : Bool \to Bool \\
& \mid 0, -1, 1, -2, 2, \dots & : Int \\
& \mid \mathsf{OclUndefined} & : \tau \\
& \mid expr_1 \ \{+, -, \dots\} \ expr_2 & : Int \times Int \to I \\
& \mid expr_1 \ \{<, \leq, \dots\} \ expr_2 & : Int \times Int \to B
\end{array}
$$

Generalised notation:

$$expr ::= \omega(expr_1, \dots, expr_n) \qquad : \tau_1 \times \dots \times \tau_n \to$$

with $\omega \in \{+, -, \dots\}$

– 03 – 2010-10-27 – Soclsyn –

OCL Syntax 3/4: Iterate

$$expr ::= \dots \mid expr_1\text{->}\mathsf{iterate}(w_1 : \tau_1 \ ; \ w_2 : \tau_2 = expr_2 \mid expr_3)$$

or, with a little renaming,

$$expr ::= \dots \mid expr_1\text{->}\mathsf{iterate}(iter : \tau_1; \ result : \tau_2 = expr_2 \mid expr_3)$$

OCL Syntax 4/4: Context

$$context ::= \mathsf{context} \ w_1 : \tau_1, \dots, w_n : \tau_n \ \mathsf{inv} : expr$$

where $w \in W$ and $\tau_i \in T_\mathscr{C}$, $1 \leq i \leq n$, $n \geq 0$.

– 04 – 2013-11-04 – Soclsem –

Valuations of Logical Variables

- **Recall**: we have typed logical variables $(w \in)\ W$, $\tau(w)$ is the type of w.

$\supseteq \{ self_{C} \mid C \in \mathscr{C} \}$

- By β, we denote a valuation of the logical variables, i.e. for each $w \in W$,

$$\beta(w) \in I(\tau(w)).$$

$$\beta : W \longrightarrow \bigcup_{w \in W} I(\tau(w))$$

$$W = \{ x : Int,\ self_C : \tau_C \}$$

$$\beta : W \to I(Int) \cup I(\tau_C)$$

Example:

- $\beta(x) = 27 \in I(Int)$
- $\beta(self_C) = 1_C \in I(\tau_C) = \mathscr{D}(C) \cup \{\bot\}$

- $\beta_2(x) = \bot_{Int}$
- $\beta_2(self) = \bot_{\tau_C}$

(vi) Putting It All Together...

$I : OCLExpr \times \Sigma_\mathscr{S}^\mathscr{D} \times (W \to \bigcup_{\tau \in T} I(\tau(w))) \to \{true, false, \bot_{Bool}\}$

$$expr ::= w \mid \omega(expr_1, \dots, expr_n) \mid \mathsf{allInstances}_C \mid v(expr_1) \mid r_1(expr_1)$$
$$\mid r_2(expr_1) \mid expr_1\text{->}\mathsf{iterate}(v_1 : \tau_1 \,;\, v_2 : \tau_2 = expr_2 \mid expr_3)$$

- $I[\![w]\!](\sigma, \beta) := \beta(w)$
- $I[\![\omega(expr_1, \dots, expr_n)]\!](\sigma, \beta) := I(\omega)(I[\![expr_1]\!](\sigma,\beta), \dots, I[\![expr_n]\!](\sigma,\beta))$
- $I[\![\mathsf{allInstances}_C]\!](\sigma, \beta) := dom(\sigma) \cap \mathscr{D}(C)$

Note: in the OCL standard, $\mathrm{dom}(\sigma)$ is assumed to be **finite**.
Again: doesn't scare us.

$\mathcal{S} = (\emptyset, \{C, D\}, \emptyset, \emptyset)$

- $\sigma_1 = \{1_C \mapsto \emptyset,\ 3_C \mapsto \emptyset,\ 27_C \mapsto \emptyset,\ 5_D \mapsto \emptyset\}$
- $W = \{x : Int,\ c : \tau_C\}$
- $\beta_1 = \{x \mapsto 13,\ c \mapsto 3_C\}$ $(**)$
- $I[\![allInstances_D]\!](\sigma_1, \beta_1) = dom(\sigma_1) \cap \mathscr{D}(D) = \{5_D\}$ $(*)$
- $I[\![size(\underbrace{allInstances_D}_{expr_1})]\!](\sigma_1, \beta_1) = (I(size))(I[\![allInstances_D]\!](\sigma_1, \beta_1))$
  $= (I(size))(\{5_D\}) = |\{5_D\}| = 1$ $(***)$
  by $(*)$ by def. of $I(size)$
- $I[\![x > size(allInstances_D)]\!](\sigma_1, \beta_1) = (I(>))(I[\![x]\!](\sigma_1, \beta_1), I[\![size(allInstances_D)]\!](\sigma_1, \beta_1))$
  $>(x, size(allInstances_D))$
  $= I(>)(\beta_1(x), 1) = I(>)(13, 1)$ $(**)$
  $(***)$ by def. of $I[\![\cdot]\!]$
  $= true$
  by def. $I(>)$
- $I[\![1/(\underbrace{\underbrace{size(allInstances_D)}_{1} - 1}_{0})]\!](\sigma_1, \beta_1) = \bot_{Int}$
  $I[\![\underset{\omega}{1}]\!](\sigma_1, \beta_1) = I(1) = 1 \in I(Int)$

assuming
$I(/)(x_1, x_2) = \begin{cases} x_1/x_2 & \text{if } x_1 \neq \bot \text{ and } x_2 \neq \bot \text{ and } x_2 \neq 0 \\ \bot_{Int} & \text{otherwise} \end{cases}$

(vi) Putting It All Together...

$$expr ::= w \mid \omega(expr_1, \ldots, expr_n) \mid \mathsf{allInstances}_C \mid v(expr_1) \mid r_1(expr_1)$$
$$\mid r_2(expr_1) \mid expr_1\text{->iterate}(v_1 : \tau_1 \; ; \; v_2 : \tau_2 = expr_2 \mid expr_3)$$

Assume $expr_1 : \tau_C$ for some $C \in \mathscr{C}$. Set $\underline{u_1 := I[\![expr_1]\!](\sigma, \beta)} \in \mathscr{D}(\tau_C)$.

- $I[\![v(expr_1)]\!](\sigma, \beta) := \begin{cases} \sigma(u_1)(v) & \text{, if } u_1 \in dom(\sigma) \\ \bot_\tau & \text{, otherwise} \end{cases}$ assuming $v : \tau$

- $I[\![r_1(expr_1)]\!](\sigma, \beta) := \begin{cases} u & \text{, if } u_1 \in dom(\sigma) \text{ and } \sigma(u_1)(r_1) = \{u\} \\ \bot & \text{, otherwise} \end{cases}$

  $r_1 : C_{0,1}$

- $I[\![r_2(expr_1)]\!](\sigma, \beta) := \begin{cases} \sigma(u_1)(r_2) & \text{, if } u_1 \in dom(\sigma) \\ \bot_{Set(\tau_C)} & \text{, otherwise} \end{cases}$

  (Recall: σ evaluates r_2 of type C_* to a set)

$\mathcal{S} = (\{Int, Colour\}, \{C, D\}, \{ms : C_{0,1},\ sl : C_*,\ r : Int,\ c : Colour\}, \{C \mapsto \{ms, sl, r\},$
$D \mapsto \{c\}\})$

$\mathcal{D}(Int) = \mathbb{Z}$, $\mathcal{D}(Colour) = \{red, green, blue\}$

$\sigma_2 = \{1_C \mapsto \{ms \mapsto \emptyset, sl \mapsto \{2_C, 3_C\}, r \mapsto 9\},$
$\quad 2_C \mapsto \{ms \mapsto \{1_C\}, sl \mapsto \emptyset, r \mapsto 5\},$
$\quad 3_C \mapsto \{ms \mapsto \{4_C\}, sl \mapsto \emptyset, r \mapsto 3\},$
$\quad 5_D \mapsto \{c \mapsto blue\}\}$

$\beta_2 = \{p : \tau_C \mapsto 1_C,\ q : \tau_D \mapsto 5_D,\ x : Int \mapsto 4,\ d : Colour \mapsto green,\ m : \tau_C \mapsto 2_C\}$

- $I[\![c(q)]\!](\sigma_2, \beta_2) = \sigma_2(5_D)(c) = blue$, $I[\![q]\!](\sigma_2, \beta_2) = 5_D$
- $I[\![c(q) = d]\!](\sigma_2, \beta_2) = false$
- $I[\![sl(p)]\!](\sigma_2, \beta_2) = \{2_C, 3_C\}$
- $I[\![r(ms(m))]\!](\sigma_2, \beta_2) = 9$, $I[\![ms(m)]\!](\sigma_2, \beta_2) = 1_C$
- $I[\![r(ms(p))]\!](\sigma_2, \beta_2) = \bot_{Int}$, $I[\![ms(p)]\!](\sigma_2, \beta_2) = \bot_{\tau_C}$

(vi) Putting It All Together...

assign to hlp the set denoted by $expr_1$ (in σ, under β)

$$expr ::= w \mid \omega(expr_1, \dots, expr_n) \mid \mathsf{allInstances}_C \mid v(expr_1) \mid r_1(expr_1)$$
$$\mid r_2(expr_1) \mid expr_1\text{->}\mathsf{iterate}(v_1 : \tau_1 \, ; v_2 : \tau_2 = expr_2 \mid expr_3)$$

iterate result init. value expr. assign to v_2 the initial value as given by $expr_2$

- $I[\![expr_1\text{->}\mathsf{iterate}(v_1 : \tau_1 \, ; v_2 : \tau_2 = expr_2 \mid expr_3)]\!](\sigma, \beta)$

modification of β at hlp and v_2

$$:= \begin{cases} I[\![expr_2]\!](\sigma, \beta) & \text{, if } I[\![expr_1]\!](\sigma, \beta) = \emptyset \\ iterate(hlp, v_1, v_2, expr_3, \sigma, \beta') & \text{, otherwise} \end{cases}$$

where $\beta' = \beta[hlp \mapsto I[\![expr_1]\!](\sigma, \beta), v_2 \mapsto I[\![expr_2]\!](\sigma, \beta)]$ and

- $iterate(hlp, v_1, v_2, expr_3, \sigma, \beta')$

hlp has more than one element left

$$:= \begin{cases} I[\![expr_3]\!](\sigma, \beta'[v_1 \mapsto x]) & \text{, if } \beta'(hlp) = \{x\} \\ I[\![expr_3]\!](\sigma, \beta'') & \text{, if } \beta'(hlp) = X \,\dot\cup\, \{x\} \text{ and } X \neq \emptyset \end{cases}$$

where $\beta'' = \beta'[v_1 \mapsto x, v_2 \mapsto iterate(hlp, v_1, v_2, expr_3, \sigma, \beta'[\boxed{hlp \mapsto X}])]$

new hlp is earlier hlp without x

(vi) Putting It All Together...

$$expr ::= w \mid \omega(expr_1, \dots, expr_n) \mid \mathsf{allInstances}_C \mid v(expr_1) \mid r_1(expr_1)$$
$$\mid r_2(expr_1) \mid expr_1\texttt{->iterate}(v_1 : \tau_1 \,;\, v_2 : \tau_2 = expr_2 \mid expr_3)$$

- $I[\![expr_1\texttt{->iterate}(v_1 : \tau_1 \,;\, v_2 : \tau_2 = expr_2 \mid expr_3)]\!](\sigma, \beta)$

$$:= \begin{cases} I[\![expr_2]\!](\sigma,\beta) & \text{, if } I[\![expr_1]\!](\sigma,\beta) = \emptyset \\ iterate(hlp, v_1, v_2, expr_3, \sigma, \beta') & \text{, otherwise} \end{cases}$$

where $\beta' = \beta[hlp \mapsto I[\![expr_1]\!](\sigma,\beta), v_2 \mapsto I[\![expr_2]\!](\sigma,\beta)]$ and

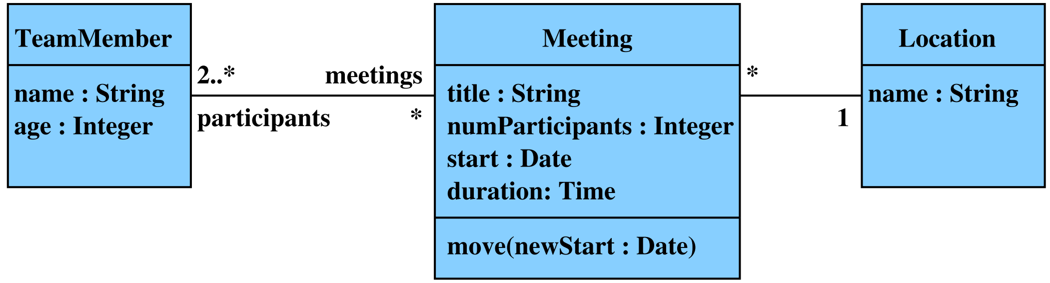
- $iterate(hlp, v_1, v_2, expr_3, \sigma, \beta')$

$$:= \begin{cases} I[\![expr_3]\!](\sigma, \beta'[v_1 \mapsto x]) & \text{, if } \beta'(hlp) = \{x\} \\ I[\![expr_3]\!](\sigma, \beta'') & \text{, if } \beta'(hlp) = X \,\dot\cup\, \{x\} \text{ and } X \neq \emptyset \end{cases}$$

where $\beta'' = \beta'[v_1 \mapsto x, v_2 \mapsto iterate(hlp, v_1, v_2, expr_3, \sigma, \beta'[hlp \mapsto X])]$

Quiz: Is (our) I a function?

Example



* **context** TeamMember **inv:** age => 18
* **context** Meeting **inv:** duration > 0

References

References

[Cabot and Clarisó, 2008] Cabot, J. and Clarisó, R. (2008). UML-OCL verification in practice. In Chaudron, M. R. V., editor, *MoDELS Workshops*, volume 5421 of *Lecture Notes in Computer Science*. Springer.

[Cengarle and Knapp, 2001] Cengarle, M. V. and Knapp, A. (2001). On the expressive power of pure OCL. Technical Report 0101, Institut für Informatik, Ludwig-Maximilians-Universität München.

[Cengarle and Knapp, 2002] Cengarle, M. V. and Knapp, A. (2002). Towards OCL/RT. In Eriksson, L.-H. and Lindsay, P. A., editors, *FME*, volume 2391 of *Lecture Notes in Computer Science*, pages 390–409. Springer-Verlag.

[Flake and Müller, 2003] Flake, S. and Müller, W. (2003). Formal semantics of static and temporal state-oriented OCL constraints. *Software and Systems Modeling*, 2(3):164–186.

[Jackson, 2002] Jackson, D. (2002). Alloy: A lightweight object modelling notation. *ACM Transactions on Software Engineering and Methodology*, 11(2):256–290.

[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.

[Schumann et al., 2008] Schumann, M., Steinke, J., Deck, A., and Westphal, B. (2008)