# *Software Design, Modelling and Analysis in UML*

# *Lecture 4: OCL Semantics*

*2016-11-03*

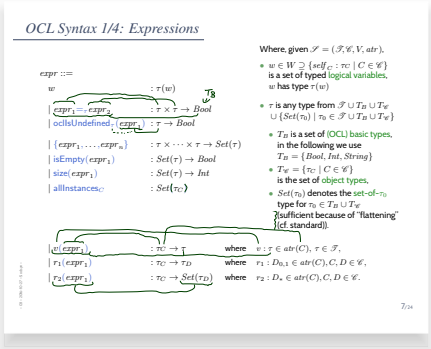
Prof. Dr. Andreas Podelski, **Dr. Bernd Westphal**

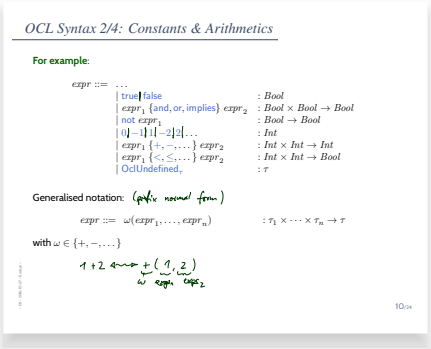
Albert-Ludwigs-Universität Freiburg, Germany

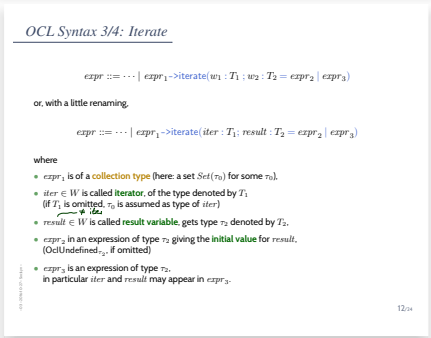


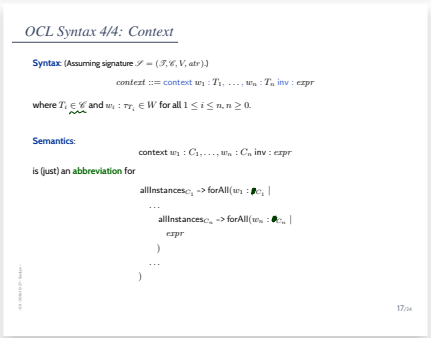
## *Content*

# Recall






# OCL Semantics: The Task

- **Given**
  - an OCL expression (over signature $\mathscr{S}$), e.g.

$$expr_1 = \text{context } CP \text{ inv : } wen \text{ implies } dd\,.\,wis > 0$$

  - and a system state

$$\sigma_1 = \{7_{VM} \mapsto \{dd \mapsto \{1_{DD}\}, cp \mapsto \{3_{DD}, 5_{DD}\}\}, \quad 1_{DD} \mapsto \{wis \mapsto 13\},$$
$$3_{CP} \mapsto \{dd \mapsto \{1_{DD}\}, wen \mapsto \textit{true}\}, \quad 5_{CP} \mapsto \{dd \mapsto \{1_{DD}\}, wen \mapsto \textit{false}\}\} \in \Sigma^{\mathscr{D}}_{\mathscr{S}}$$

  - and a valuation of the logical variables $\beta_1 : W \to I(\mathscr{T} \cup T_B \cup T_{\mathscr{C}})$,

- **compute** the value $I[\![expr_1]\!](\sigma_1, \beta_1) \in \{\textit{true}, \textit{false}, \bot_{Bool}\}$ of $expr_1$ in $\sigma_1$ under $\beta_1$.

- More general: **Define** the **interpretation** $I[\![expr]\!](\sigma, \beta)$ of $expr$ **in** $\sigma$ **under** $\beta$:

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

# *OCL Semantics OMG (2006)*



## *Basically business as usual...*

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

$$I_{(i)} \text{ with } \mathrm{dom}(I_{(i)}) = \mathscr{T} \cup T_B \cup T_{\mathscr{C}}$$

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

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

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

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

(iv) Same game for **set operations**: **define function** $I_{(iv)}$ with $\mathrm{dom}(I) = \{\mathrm{isEmpty}, \dots\}$

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

$$I_{(v)} : Expr \times \Sigma^{\mathscr{D}}_{\mathscr{S}} \times (W \to I_{(iv)}(\mathscr{T} \cup T_B \cup T_{\mathscr{C}})) \to I_{(i)}(Bool)$$

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

$I := I_{(i)} \cup I_{(ii)} \cup I_{(iii)} \cup I_{(iv)} \cup I_{(v)}$

## (i) Domains of OCL and (!) Model Basic Types

**Recall: OCL basic types**

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

**We set:** (three-valued)

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

($\dot\cup$: disjoint union)

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

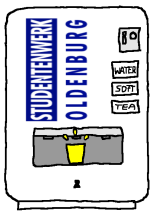
Given signature $\mathscr{S}$ with **model basic types** $\mathscr{T}$ and domain $\mathscr{D}$, set

$$I(T) := \mathscr{D}(T) \dot\cup \{\bot_T\}$$

for each model basic type $T \in \mathscr{T}$.



## OCL and Model Types?! An Example.




$$\mathscr{S} = (\{Bool_M, Nat\}, \{VM, CP, DD\},$$
$$\{cp : CP_*, dd : DD_{0,1}, wen : Bool, wis : Nat\},$$
$$\{VM \mapsto \{cp, dd\}, CP \mapsto \{wen, dd\}, DD \mapsto \{wis\}\})$$

Model Types:

$\mathscr{D}(Bool_M) = \{0, 1\}$

$\mathscr{D}(Nat) = \{0, \ldots, 255\}$

$\mathscr{D}(VM) = \mathbb{N} \times \{VM\}$
$= \{1_{VM}, 2_{VM}, \ldots\}$

OCL Types:

$I(Bool) = \{true, false, \bot\}$ } fixed for OCL $T_B$
$I(Int) = \mathbb{Z} \dot\cup \{\bot_{Int}\}$

$I(Bool_M) = \mathscr{D}(Bool_M) \dot\cup \{\bot_{Bool_M}\}$
$= \{0, 1, \bot_{Bool_M}\}$

$I(Nat) = \mathscr{D}(Nat) \dot\cup \{\bot_{Nat}\}$
$= \{0, \ldots, 255\} \dot\cup \{\bot_{Nat}\}$

$I(\tau_{VM}) = \mathscr{D}(VM) \dot\cup \{\bot_{VM}\}$

## *(i) Domains of Object and (ii) Set Types*

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

$$I(\tau_C) := \mathscr{D}(C) \dot\cup \{\bot_{\tau_C}\}$$

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

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

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



## *(iii) Interpretation of Arithmetic Operations*

- **Literals** map to fixed values:

$$I(\mathsf{true}) := \mathit{true}, \quad I(\mathsf{false}) := \mathit{false}, \quad I(0) := 0, \quad I(1) := 1, \ldots$$
$$I(\mathsf{OclUndefined}_\tau) := \bot_\tau$$

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

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

- **Logical connectives**, e.g. $I(\mathsf{and})(\cdot,\cdot) : \{\mathit{true}, \mathit{false}, \bot\} \times \{\mathit{true}, \mathit{false}, \bot\} \to \{\mathit{true}, \mathit{false}, \bot\}$ is defined by the following truth table:

| $x_1$ | *true* | *true* | *true* | *false* | *false* | *false* | $\bot$ | $\bot$ | $\bot$ |
|---|---|---|---|---|---|---|---|---|---|
| $x_2$ | *true* | *false* | $\bot$ | *true* | *false* | $\bot$ | *true* | *false* | $\bot$ |
| $I(\mathsf{and})(x_1, x_2)$ | *true* | *false* | $\bot$ | *false* | *false* | *false* | $\bot$ | *false* | $\bot$ |

We assume common logical connectives not, or, ... with the canonical 3-valued interpretation.

## *(iii) Interpretation of OclIsUndefined*

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

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

  **Note**: $I(\text{oclIsUndefined}_\tau)$ is **definite**, i.e., it never yields $\bot$.

- **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**.
The **interpretation** of an operation (symbol)

$$\omega : \tau_1 \times \ldots \tau_n \to \tau, \quad n \geq 0$$

is a function

$$I(\omega) : I(\tau_1) \times \cdots \times I(\tau_n) \to I(\tau)$$

on corresponding semantical domain(s) of OCL (!) types.



## *(iv) Interpretation of Set Operations*

Basically the same principle as with arithmetic operations...

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

- **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 $n \in \mathbb{N}_0$

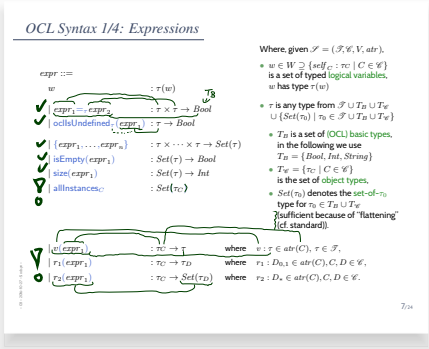
- **Empty-ness check** ($x \in I(Set(\tau))$):

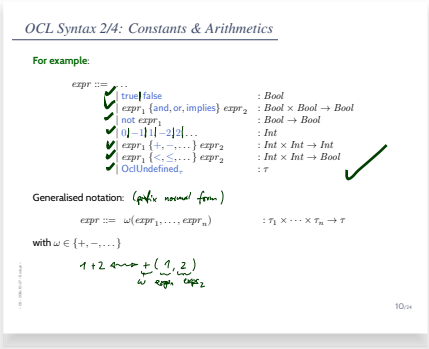
$$I(\text{isEmpty}^\tau)(x) := \begin{cases} true & \text{, if } x = \emptyset \\ \bot_{Bool} & \text{, if } x = \bot_{Set(\tau)} \\ false & \text{, otherwise} \end{cases}$$

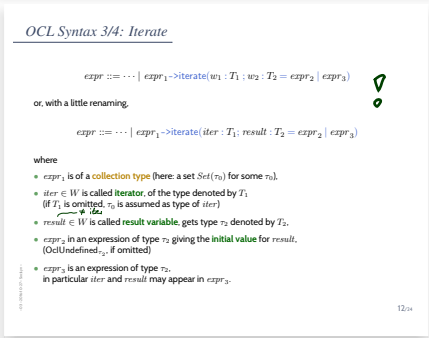
- **Counting** ($x \in I(Set(\tau))$):

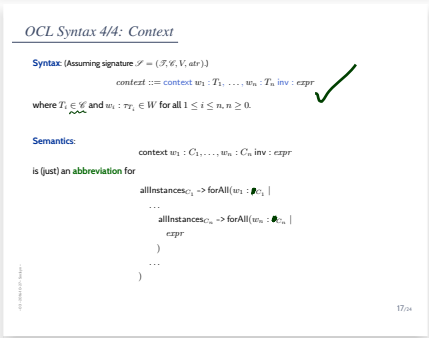
$$I(\text{size}^\tau)(x) := \begin{cases} |x| & \text{, if } x \neq \bot_{Set(\tau)} \\ \bot_{Int} & \text{, otherwise} \end{cases}$$

# (v) Interpretation of OCL Expressions






# Valuations of Logical Variables

- **Recall**: we have typed logical variables $(w \in) W$, $\tau(w)$ is the type of $w$.
- By $\beta$, we denote a valuation of the logical variables, i.e. for each $w \in W$,

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

- $self_{VM} \in W$
- $self_{VM} : \tau_{VM}$ is an OCL expression
- $I[\![self_{VM}]\!](\sigma, \beta) := \beta(self_{VM})$
- $\beta_1 = \{ self_{VM} \mapsto 1_{VM} \}$
  $\hookrightarrow I[\![self_{VM}]\!](\sigma, \beta_1) = \beta_1(self_{VM}) = 1_{VM}$
- $\beta : W \longrightarrow I(T_B \cup T_C \cup \mathcal{F})$

## (v) Interpretation of OCL Expressions

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

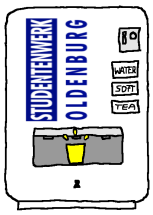
- $I[\![w]\!](\sigma, \beta) := \beta(w)$
- $I[\![\omega(expr_1, \ldots, expr_n)]\!](\sigma, \beta) := I(\omega)(I[\![expr_1]\!](\sigma, \beta), \ldots, I[\![expr_n]\!](\sigma, \beta))$
- $I[\![\mathsf{allInstances}_C]\!](\sigma, \beta) := \mathrm{dom}(\sigma) \cap \mathscr{D}(C)$

  (all alive objects in $\sigma$; objects of class $C$)

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



## Example




$$\mathscr{S} = (\{Bool, Nat\}, \{VM, CP, DD\}, \{cp : CP_*, dd : DD_{0,1}, wen : Bool, wis : Nat\}, \{VM \mapsto \{cp, dd\}, CP \mapsto \{wen, dd\}, DD \mapsto \{wis\}\})$$

$$\sigma_1 = \{7_{VM} \mapsto \{dd \mapsto \{1_{DD}\}, cp \mapsto \{3_{DD}, 5_{DD}\}\}, \quad 1_{DD} \mapsto \{wis \mapsto 13\}, \quad 3_{CP} \mapsto \{dd \mapsto \{1_{DD}\}, wen \mapsto \mathit{true}\}, \quad 5_{CP} \mapsto \{dd \mapsto \{1_{DD}\}, wen \mapsto \mathit{false}\}\}$$

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

- $I[\![\mathsf{allInstances}_{CP}]\!](\sigma_1, \beta) = \mathrm{dom}(\sigma_1) \cap \mathscr{D}(CP) = \{7_{VM}, 1_{DD}, 3_{CP}, 5_{CP}\} \cap \mathscr{D}(CP) = \{3_{CP}, 5_{CP}\}$
- $I[\![\mathsf{allInstances}_{CP}\text{->}\mathsf{size}]\!](\sigma_1, \beta) = I[\![\mathsf{size}(\mathsf{allInstances}_{CP})]\!](\sigma, \beta) = I(\mathsf{size})(I[\![\mathsf{allInstances}_{CP}]\!](\sigma_1, \beta)) = I(\mathsf{size})(\{3_{CP}, 5_{CP}\}) = 2$
- $\beta_1 := \{\overset{self \mapsto}{3_{CP}}\}, \quad I[\![self]\!](\sigma_1, \beta_1) = \beta_1(self) = 3_{CP}$

## (v) Interpretation of OCL Expressions

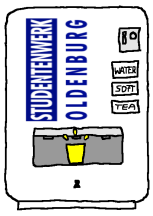
$$expr ::= w \mid \omega(expr_1, \ldots, expr_n) \mid \text{allInstances}_C \mid v(expr_1) \mid r_1(expr_1) \mid r_2(expr_1) \mid expr_1\text{->iterate}(v_1 : \tau_1 ; v_2 : \tau_2 = expr_2 \mid expr_3)$$

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

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



## Example




$$\mathscr{S} = (\{Bool, Nat\}, \{VM, CP, DD\}, \{cp : CP_*, dd : DD_{0,1}, wen : Bool, wis : Nat\}, \{VM \mapsto \{cp, dd\}, CP \mapsto \{wen, dd\}, DD \mapsto \{wis\}\})$$

$$\sigma_1 = \{7_{VM} \mapsto \{dd \mapsto \{1_{DD}\}, cp \mapsto \{3_{DD}, 5_{DD}\}\}, \quad 1_{DD} \mapsto \{wis \mapsto 13\}, \quad 3_{CP} \mapsto \{dd \mapsto \{1_{DD}\}, wen \mapsto \textit{true}\}, \quad 5_{CP} \mapsto \{dd \mapsto \{1_{DD}\}, wen \mapsto \textit{false}\}\}$$

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

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

- $\beta_1 := \{3_{CP}\}$ (self ↦), $I[\![wen(self)]\!](\sigma_1, \beta_1) = \sigma_1(u_1)(wen) = \sigma_1(3_{CP})(wen) = true$

$u_1 = I[\![self]\!](\sigma_1, \beta_1) = 3_{CP}$

## (v) Interpretation of OCL Expressions

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

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

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

  $r_1 : C_{0,1}$
- $I[\![r_2(expr_1)]\!](\sigma,\beta) := \begin{cases} \sigma(u_1)(r_2) & \text{, if } u_1 \in \mathrm{dom}(\sigma) \\ \bot & \text{, otherwise} \end{cases}$

  $r_2 : C_*$

Recall: $\sigma$ evaluates $r_2$ of type $C_*$ to a set.



## Iterate: Intuitive Semantics

$$expr ::= expr_1\text{->iterate}(iter : T_1; result : T_2 = expr_2 \mid expr_3)$$

$Set(\tau_0)\ hlp := expr_1;$
$\tau_1\ iter;$
$\tau_2\ result := expr_2;$
$while\ (!hlp.empty())\ do$
$\quad iter := hlp.pop();$
$\quad result := expr_3;$
$od;$
$return\ result;$

pick and remove one element

may comprise iter and result

context CP inv : expr

allInst$_{CP}$ → forAll( self | expr(self) )

## *Iterate: Intuitive Semantics*

$$expr ::= expr_1\text{->iterate}(iter : T_1;\ result : T_2 = expr_2 \mid expr_3)$$

$Set(\tau_0)\ hlp := expr_1;$
$\tau_1\ iter;$
$\tau_2\ result := expr_2;$
$while\ (!hlp.empty())\ do$
  $iter := hlp.pop();$
  $result := expr_3;$
$od;$
$return\ result;$

**Recall**: In our (simplified) setting, we always have $expr_1 : Set(\tau_0)$ and $\tau_1 = \tau_0$.
In the type hierarchy of full OCL with inheritance and `oclAny`, $\tau_0$ and $\tau_1$ may be different and still type consistent.



## *(v) Interpretation of OCL Expressions*

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

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

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

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

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

modify $\beta'$ at $v_1$. $\begin{cases} x & \text{, if } v_1 \text{ given} \\ \beta'(v_1) & \text{, otherwise} \end{cases}$

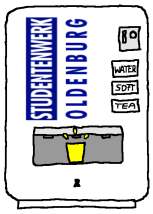
$$:= \begin{cases} I[\![expr_3]\!](\sigma, \beta'[v_1 \mapsto x]) & \text{, if } \beta'(hlp) = \{x\} \\ I[\![expr_3]\!](\sigma, \beta'') & \text{, if } \beta'(hlp) = X \dot\cup \{x\} \text{ and } X \neq \emptyset \end{cases}$$

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

**Quiz**: Is (our) $I$ a function?

## *Example*

$$\mathscr{S} = (\{Bool, Nat\}, \{VM, CP, DD\},$$
$$\{cp : CP_*, dd : DD_{0,1}, wen : Bool, wis : Nat\},$$
$$\{VM \mapsto \{cp, dd\}, CP \mapsto \{wen, dd\}, DD \mapsto \{wis\}\})$$




$$\sigma_1 = \{7_{VM} \mapsto \{dd \mapsto \{1_{DD}\}, cp \mapsto \{3_{DD}, 5_{DD}\}\}, \quad 1_{DD} \mapsto \{wis \mapsto 13\},$$
$$3_{CP} \mapsto \{dd \mapsto \{1_{DD}\}, wen \mapsto \textit{true}\}, \quad 5_{CP} \mapsto \{dd \mapsto \{1_{DD}\}, wen \mapsto \textit{false}\}\}$$

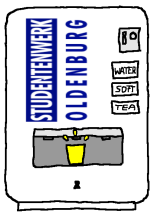
context $CP$ inv : $wen$ implies $dd\,.\,wis > 0$

allInstances$_{CP}$ -> forAll ( self | wen implies dd.wis > 0 )



## *Example*

$$\mathscr{S} = (\{Bool, Nat\}, \{VM, CP, DD\},$$
$$\{cp : CP_*, dd : DD_{0,1}, wen : Bool, wis : Nat\},$$
$$\{VM \mapsto \{cp, dd\}, CP \mapsto \{wen, dd\}, DD \mapsto \{wis\}\})$$




$$\sigma_1 = \{7_{VM} \mapsto \{dd \mapsto \{1_{DD}\}, cp \mapsto \{3_{DD}, 5_{DD}\}\}, \quad 1_{DD} \mapsto \{wis \mapsto 13\},$$
$$3_{CP} \mapsto \{dd \mapsto \{1_{DD}\}, wen \mapsto \textit{true}\}, \quad 5_{CP} \mapsto \{dd \mapsto \{1_{DD}\}, wen \mapsto \textit{false}\}\}$$

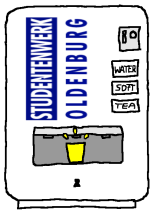
context $CP$ inv : $wen$ implies $dd\,.\,wis > 0$

allInstances$_{CP}$ -> forAll ($self$ | $self\,.\,wen$ implies $self\,.\,dd\,.\,wis > 0$)

allInstances$_{CP}$ -> iterate ( self ; r : Bool = true | r and ... )

## *Example*

$$\mathscr{S} = (\{Bool, Nat\}, \{VM, CP, DD\},$$
$$\{cp : CP_*, dd : DD_{0,1}, wen : Bool, wis : Nat\},$$
$$\{VM \mapsto \{cp, dd\}, CP \mapsto \{wen, dd\}, DD \mapsto \{wis\}\})$$




$$\sigma_1 = \{7_{VM} \mapsto \{dd \mapsto \{1_{DD}\}, cp \mapsto \{3_{DD}, 5_{DD}\}\}, \quad 1_{DD} \mapsto \{wis \mapsto 13\},$$
$$3_{CP} \mapsto \{dd \mapsto \{1_{DD}\}, wen \mapsto \mathit{true}\}, \quad 5_{CP} \mapsto \{dd \mapsto \{1_{DD}\}, wen \mapsto \mathit{false}\}\}$$

$F :=$ context $CP$ inv : $wen$ implies $dd\,.\,wis > 0$ $\quad expr_1$

allInstances$_{CP}$ -> iterate $(self; r : Bool = \mathit{true} \mid \mathsf{and}(r, \mathsf{implies}(wen(self), >(wis(dd(self)), 0))))$

$\mathcal{I}[\![\text{allInstances}_{CP}]\!](\sigma_1, \emptyset) = \{3_{CP}, 5_{CP}\}$

$=: \beta_2$

$\mathcal{I}[\![expr]\!](\sigma_1, \{self \mapsto 3_{CP}, r \mapsto true\}) = \mathcal{I}(and)\left(\mathcal{I}[\![r]\!](\sigma_1, \beta_2), \mathcal{I}[\![expr_1]\!](\sigma_1, \beta_2)\right) \overset{(*)}{=} \mathcal{I}(and)(true, true) = true$

$\mathcal{I}[\![expr_1]\!](\sigma_1, \beta_2) = \mathcal{I}(implies)\left(\mathcal{I}[\![wen(self)]\!](\sigma_1, \beta_2), \mathcal{I}(>)\left(\mathcal{I}[\![wis(dd(self))]\!](\sigma_1, \beta_2), 0\right)\right)$ $\quad (**)$

$\mathcal{I}[\![dd(self)]\!](\sigma_1, \beta_2) = 1_{DD}$

$\mathcal{I}[\![wis(dd(self))]\!](\sigma_1, \beta_2) = 13$

$\vdots$

$= \mathcal{I}(implies)(true, true) = true \quad (*)$

$\mathcal{I}[\![expr]\!](\sigma_1, \{self \mapsto 5_{CP}, r \mapsto true\}) = true$ $\quad \mathcal{I}[\![F]\!](\sigma_1, \beta_1) = true$



## *Tell Them What You've Told Them...*

- **Given**
  - an OCL expression $expr$,
  - and a system state $\sigma$,
  - and a valuation $\beta$ of the logical variables
- we can **compute** the value

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

of $expr$ in $\sigma$ under $\beta$

- using the **interpretation function**

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

## *User's Guide*

- **App**

  The

  It is

- **Inte**

  Abs

**Example:**

**Task**: Given a square with side length $a = 19.1$. What is the length of the longest straight line fully inside the square?

**Submission A:**

27

**Submission B:**

The length of the longest straight line fully inside the square with side length $a = 19.1$ is 27.01 (rounded).

The longest straight line inside the square is the diagonal. By Pythagoras, its length is $\sqrt{a^2 + a^2}$. Inserting $a = 19.1$ yields 27.01 (rounded).

- **Exercise submissions:**

  Each task is a **tiny little scientific work**:

  (i) Briefly rephrase the task in your own words.
  (ii) State your claimed solution.
  (iii) Convince your reader that your proposal is a solution (proofs are very convincing).



## *User's Guide*

- **App**

  The

  It is

- **Inte**

  Abs

**Example:**

**Task**: Given a square with side length $a = 19.1$. What is the length of the longest straight line fully inside the square?

**Submission A:**

**Submission B:**

The length of the longest straight line fully inside the square with side length $a = 19.1$ is 27.01 (rounded).

The longest straight line inside the square is the diagonal. By Pythagoras, its length is $\sqrt{a^2 + a^2}$. Inserting $a = 19.1$ yields 27.01 (rounded).

- **Exercise submissions:**

  Each task is a **tiny little scientific work**:

  (i) Briefly rephrase the task in your own words.
  (ii) State your claimed solution.
  (iii) Convince your reader that your proposal is a solution (proofs are very convincing).

- 1 - 2016-10-18 - Sformalia -
- 4 - 2016-11-03 - main -
31/34
25/29

## *Formalia: Exercises and Tutorials*

- You should work in groups of **approx. 3**, clearly give **names** on submission.
- Please submit via ILIAS (cf. homepage); **paper submissions** are **tolerated**.
- **Schedule:**

| | | | |
|---|---|---|---|
| Week $N$, | Thursday, | 8–10 **Lecture A1** | (exercise sheet $A$ **online**) |
| Week $N+1$, | Tuesday | 8–10 **Lecture A2** | |
| | Thursday | 8–10 **Lecture A3** | |
| Week $N+2$, | Monday, | 12:00 | (exercises $A$ **early submission**) |
| | Tuesday, | 8:00 | (exercises $A$ **late submission**) |
| | | 8–10 **Tutorial A** | |
| | Thursday | 8–10 **Lecture B1** | (exercise sheet $B$ **online**) |
| | … | | |

- **Rating system:** "most complicated rating system **ever**"
  - **Admission points** (good-will rating, upper bound)
    ("reasonable proposal given student's knowledge **before** tutorial")
  - **Exam-like points** (evil rating, lower bound)
    ("reasonable proposal given student's knowledge **after** tutorial")

  **10% bonus** for **early** submission.
- **Tutorial:** Plenary, **not recorded**.
  - Together develop **one good solution** based on selection of early submissions (anonymous) – there is no "Musterlösung" for modelling tasks.









- E.g.
  - give a syst. state as pos. example
  - system state
    $\sigma_1 = \{ \ldots \}$
    satisfies the req. because ....
- 18 submissions
  – ~10 singleton groups

# *References*



*References*

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

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

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