

# *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

- The **Object Constraint Language (OCL)**:

- Semantics**

- Overview
    - OCL Types
    - Arithmetic / Logical Operators
    - OCL Expressions
    - Iterate

- **A Complete Example**

## OCL Syntax 1/4: Expressions

$expr ::=$

$w$  :  $\tau(w)$   
 $expr_1 = expr_2$  :  $\tau \times \tau \rightarrow Bool$   
 $oclUndefined_?(expr_1)$  :  $\tau \rightarrow Bool$   
 $\{expr_1, \dots, expr_n\}$  :  $\tau \times \dots \times \tau \rightarrow Set(\tau)$   
 $isEmpty(expr_1)$  :  $Set(\tau) \rightarrow Bool$   
 $size(expr_1)$  :  $Set(\tau) \rightarrow Int$   
 $allInstances_C$  :  $Set(\tau_C)$

$v(expr_1)$  :  $\tau_C \rightarrow \tau$  where  $v : \tau \in atr(C), \tau \in \mathcal{T}$ ,  
 $r_1(expr_1)$  :  $\tau_C \rightarrow \tau_D$  where  $r_1 : D_{0,1} \in atr(C), C, D \in \mathcal{C}$ ,  
 $r_2(expr_1)$  :  $\tau_C \rightarrow Set(\tau_D)$  where  $r_2 : D_* \in atr(C), C, D \in \mathcal{C}$ .

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

- $w \in W \supseteq \{self_C : \tau_C \mid C \in \mathcal{C}\}$  is a set of typed logical variables,  $w$  has type  $\tau(w)$
- $\tau$  is any type from  $\mathcal{T} \cup T_B \cup T_{\mathcal{C}} \cup \{Set(\tau_0) \mid \tau_0 \in \mathcal{T} \cup T_B \cup T_{\mathcal{C}}\}$
- $T_B$  is a set of (OCL) basic types, in the following we use  $T_B = \{Bool, Int, String\}$
- $T_{\mathcal{C}} = \{\tau_C \mid C \in \mathcal{C}\}$  is the set of object types,
- $Set(\tau_0)$  denotes the set-of- $\tau_0$  type for  $\tau_0 \in T_B \cup T_{\mathcal{C}}$  (sufficient because of "flattening" (cf. standard)).

7/24

## OCL Syntax 2/4: Constants & Arithmetics

For example:

$expr ::=$  ...  
 $true \mid false$  :  $Bool$   
 $expr_1 \{and, or, implies\} expr_2$  :  $Bool \times Bool \rightarrow Bool$   
 $not expr_1$  :  $Bool \rightarrow Bool$   
 $0 \mid -1 \mid 1 \mid -2 \mid \dots$  :  $Int$   
 $expr_1 \{+, -, \dots\} expr_2$  :  $Int \times Int \rightarrow Int$   
 $expr_1 \{<, \leq, \dots\} expr_2$  :  $Int \times Int \rightarrow Bool$   
 $OclUndefined_?$  :  $\tau$

Generalised notation: (prefix normal form)

$expr ::= \omega(expr_1, \dots, expr_n)$  :  $\tau_1 \times \dots \times \tau_n \rightarrow \tau$

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

$1 + 2 \rightsquigarrow +(1, 2)$   
 $\omega \quad expr_1 \quad expr_2$

10/24

## OCL Syntax 3/4: Iterate

$expr ::= \dots \mid expr_1 \rightarrow iterate(w_1 : T_1; w_2 : T_2 = expr_2 \mid expr_3)$

or, with a little renaming,

$expr ::= \dots \mid expr_1 \rightarrow iterate(iter : T_1; result : T_2 = expr_2 \mid expr_3)$

where

- $expr_1$  is of a **collection type** (here: a set  $Set(\tau_0)$  for some  $\tau_0$ ),
- $iter \in W$  is called **iterator**, of the type denoted by  $T_1$  (if  $T_1$  is omitted,  $\tau_0$  is assumed as type of  $iter$ )
- $result \in W$  is called **result variable**, gets type  $\tau_2$  denoted by  $T_2$ ,
- $expr_2$  is an expression of type  $\tau_2$  giving the **initial value** for  $result$ , ( $OclUndefined_{\tau_2}$ , if omitted)
- $expr_3$  is an expression of type  $\tau_2$ , in particular  $iter$  and  $result$  may appear in  $expr_3$ .

12/24

## OCL Syntax 4/4: Context

**Syntax:** (Assuming signature  $\mathcal{S} = (\mathcal{T}, \mathcal{C}, V, atr)$ )

$context ::= context \ w_1 : T_1, \dots, w_n : T_n \ inv : expr$

where  $T_i \in \mathcal{C}$  and  $w_i : \tau_{T_i} \in W$  for all  $1 \leq i \leq n, n \geq 0$ .

**Semantics:**

$context \ w_1 : C_1, \dots, w_n : C_n \ inv : expr$

is (just) an **abbreviation** for

$allInstances_{C_1} \rightarrow forAll(w_1 : \#C_1 \mid$   
 $\dots$   
 $allInstances_{C_n} \rightarrow forAll(w_n : \#C_n \mid$   
 $expr$   
 $)$   
 $\dots$   
 $)$

17/24

# OCL Semantics: The Task

- Given

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

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

- and a system state

$$\sigma_1 = \{7_{VM} \mapsto \{dd \mapsto \{1_{DD}\}, cp \mapsto \{3_{DD}, 5_{DD}\}\}, 1_{DD} \mapsto \{wis \mapsto 13\},$$

$$3_{CP} \mapsto \{dd \mapsto \{1_{DD}\}, wen \mapsto true\}, 5_{CP} \mapsto \{dd \mapsto \{1_{DD}\}, wen \mapsto false\}\} \in \Sigma_{\mathcal{S}}^{\mathcal{D}}$$

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

*Handwritten:  $\{set \in C \mid C \in \mathcal{C}\}$*

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

*Handwritten:  $\uparrow$  three-valued logic*

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

$$I[\cdot](\cdot, \cdot) : \underbrace{OCLExpressions(\mathcal{S})}_{\text{OCL expressions}} \times \underbrace{\Sigma_{\mathcal{S}}^{\mathcal{D}}}_{\text{system states}} \times \underbrace{(W \rightarrow I(\mathcal{T} \cup T_B \cup T_{\mathcal{C}}))}_{\text{valuations}} \rightarrow I(Bool)$$



# *OCL Semantics OMG (2006)*

# Basically business as usual...

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

$$I_{(t)} \text{ with } \text{dom}(I_{(t)}) = \mathcal{T} \cup T_B \cup T_{\mathcal{C}}$$

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

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

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

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

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

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

$$I_{(e)}: Expr \times \Sigma_{\mathcal{D}} \times (W \rightarrow I_{(t)}(\mathcal{T} \cup T_B \cup T_{\mathcal{C}})) \rightarrow I_{(b)}(Bool)$$

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

$$I := I_{(t)} \cup I_{(s)} \cup I_{(f)} \cup I_{(g)} \cup I_{(e)}$$

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

---

### Recall: OCL basic types

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

### We set:

- $I(Bool) := \{true, false, \perp_{Bool}\}$
  - $I(Int) := \mathbb{Z} \dot{\cup} \{\perp_{Int}\}$
  - $I(String) := \dots \dot{\cup} \{\perp_{String}\}$
- three-valued* (handwritten note with arrow pointing to  $\perp_{Bool}$ )
- disjoint union* (handwritten note with arrow pointing to  $\dot{\cup}$ )

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

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

$$I(T) := \mathcal{D}(T) \dot{\cup} \{\perp_T\}$$

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

# OCL and Model Types?! An Example.

$$\mathcal{S} = (\{\text{Bool}_M, \text{Nat}\}, \{\text{VM}, \text{CP}, \text{DD}\}, \\ \{cp : \text{CP}_*, dd : \text{DD}_{0,1}, wen : \text{Bool}, wis : \text{Nat}\}, \\ \{\text{VM} \mapsto \{cp, dd\}, \text{CP} \mapsto \{wen, dd\}, \text{DD} \mapsto \{wis\}\})$$

## Model Types:

$$\mathcal{D}(\text{Bool}_M) = \{0, 1\}$$

$$\mathcal{D}(\text{Nat}) = \{0, \dots, 255\}$$

$$\mathcal{D}(\text{VM}) = \mathbb{N} \times \{\text{VM}\} \\ = \{1_{\text{VM}}, 2_{\text{VM}}, \dots\}$$

## OCL Types:

$$\begin{aligned} \mathcal{I}(\text{Bool}) &= \{\text{true}, \text{false}, \perp\} \\ \mathcal{I}(\text{Int}) &= \mathbb{Z} \cup \{\perp_{\text{int}}\} \end{aligned} \left. \vphantom{\begin{aligned} \mathcal{I}(\text{Bool}) \\ \mathcal{I}(\text{Int}) \end{aligned}} \right\} \begin{array}{l} \text{fixed for} \\ \text{OCL } \mathcal{T}_B \end{array}$$

$$\begin{aligned} \mathcal{I}(\text{Bool}_M) &= \mathcal{D}(\text{Bool}_M) \cup \{\perp_{\text{Bool}_M}\} \\ &= \{0, 1, \perp_{\text{Bool}_M}\} \end{aligned}$$

$$\begin{aligned} \mathcal{I}(\text{Nat}) &= \mathcal{D}(\text{Nat}) \cup \{\perp_{\text{Nat}}\} \\ &= \{0, \dots, 255\} \cup \{\perp_{\text{Nat}}\} \end{aligned}$$

$$\mathcal{I}(\tau_{\text{VM}}) = \mathcal{D}(\text{VM}) \cup \{\perp_{\text{VM}}\}$$





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

---

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

$$I(\tau_C) := \mathcal{D}(C) \dot{\cup} \{\perp_{\tau_C}\}$$

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

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

**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:
 
$$\begin{array}{ccc}
 I(\text{Bool}) & I(\text{Bool}) & I(\text{Int}) \\
 \downarrow & \downarrow & \downarrow \\
 I(\text{true}) := \text{true}, & I(\text{false}) := \text{false}, & I(0) := 0, \quad I(1) := 1, \dots \\
 \uparrow & & \\
 \mathcal{Q}(\text{Expr}(S)) & I(\text{OclUndefined}_\tau) := \perp_\tau &
 \end{array}$$

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

$$\begin{array}{c}
 I(\tau) \quad I(\tau) \\
 \downarrow \quad \downarrow \\
 I(=_\tau)(x_1, x_2) := \begin{cases} \text{true} & , \text{ if } x_1 \neq \perp_\tau \neq x_2 \text{ and } x_1 = x_2 \\ \text{false} & , \text{ if } x_1 \neq \perp_\tau \neq x_2 \text{ and } x_1 \neq x_2 \\ \perp_{\text{Bool}} & , \text{ otherwise} \end{cases} \\
 I(=_\tau) : I(\tau) \times I(\tau) \rightarrow I(\text{Bool})
 \end{array}$$

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

$x_1$	true	true	true	false	false	false	$\perp$	$\perp$	$\perp$
$x_2$	true	false	$\perp$	true	false	$\perp$	true	false	$\perp$
$I(\text{and})(x_1, x_2)$	true	false	$\perp$	false	false	false	$\perp$	false	$\perp$

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  $x \in I(\tau)$ ):

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

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

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

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

**Note:** There is a **common principle**.

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

$$\omega : \tau_1 \times \dots \times \tau_n \rightarrow \tau, \quad n \geq 0$$

is a function

$$I(\omega) : I(\tau_1) \times \dots \times I(\tau_n) \rightarrow I(\tau)$$

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

## (iv) Interpretation of Set Operations

Basically the same principle as with arithmetic operations...

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

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

$$I(\{\}_n^\tau)(x_1, \dots, x_n) := \{x_1, \dots, x_n\}$$

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

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

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

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

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

*numbers of elements in x*

# (v) Interpretation of OCL Expressions

## OCL Syntax 1/4: Expressions

$expr ::=$

$w$  :  $\tau(w)$

$expr_1 = expr_2$  :  $\tau \times \tau \rightarrow Bool$

$oclUndefined_?(expr_1)$  :  $\tau \rightarrow Bool$

$\{expr_1, \dots, expr_n\}$  :  $\tau \times \dots \times \tau \rightarrow Set(\tau)$

$isEmpty(expr_1)$  :  $Set(\tau) \rightarrow Bool$

$size(expr_1)$  :  $Set(\tau) \rightarrow Int$

$allInstances_C$  :  $Set(\tau_C)$

Where, given  $\mathcal{S} = (\mathcal{F}, \mathcal{C}, V, atr)$ ,

- $w \in W \supseteq \{self_C : \tau_C \mid C \in \mathcal{C}\}$  is a set of typed logical variables,  $w$  has type  $\tau(w)$
- $\tau$  is any type from  $\mathcal{F} \cup T_B \cup T_{\mathcal{C}} \cup \{Set(\tau_0) \mid \tau_0 \in \mathcal{F} \cup T_B \cup T_{\mathcal{C}}\}$
- $T_B$  is a set of (OCL) basic types, in the following we use  $T_B = \{Bool, Int, String\}$
- $T_{\mathcal{C}} = \{\tau_C \mid C \in \mathcal{C}\}$  is the set of object types,
- $Set(\tau_0)$  denotes the set-of- $\tau_0$  type for  $\tau_0 \in T_B \cup T_{\mathcal{C}}$  (sufficient because of "flattening" (cf. standard)).

$v(expr_1)$  :  $\tau_C \rightarrow \tau$  where  $v : \tau \in atr(C), \tau \in \mathcal{F}$ ,

$r_1(expr_1)$  :  $\tau_C \rightarrow \tau_D$  where  $r_1 : D_{0,1} \in atr(C), C, D \in \mathcal{C}$ ,

$r_2(expr_1)$  :  $\tau_C \rightarrow Set(\tau_D)$  where  $r_2 : D_* \in atr(C), C, D \in \mathcal{C}$ .

## OCL Syntax 2/4: Constants & Arithmetics

For example:

$expr ::=$  ...

$true \mid false$  :  $Bool$

$expr_1 \{and, or, implies\} expr_2$  :  $Bool \times Bool \rightarrow Bool$

$not \ expr_1$  :  $Bool \rightarrow Bool$

$0 \mid -1 \mid 1 \mid -2 \mid 2 \mid \dots$  :  $Int$

$expr_1 \{+, -, \dots\} expr_2$  :  $Int \times Int \rightarrow Int$

$expr_1 \{<, \leq, \dots\} expr_2$  :  $Int \times Int \rightarrow Bool$

$OclUndefined_?$  :  $\tau$

Generalised notation: (prefix normal form)

$expr ::= \omega(expr_1, \dots, expr_n)$  :  $\tau_1 \times \dots \times \tau_n \rightarrow \tau$

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

$1 + 2 \rightsquigarrow +(1, 2)$   
 $\omega \quad expr_1 \quad expr_2$

## OCL Syntax 3/4: Iterate

$expr ::= \dots \mid expr_1 \rightarrow iterate(w_1 : T_1 ; w_2 : T_2 = expr_2 \mid expr_3)$

or, with a little renaming,

$expr ::= \dots \mid expr_1 \rightarrow iterate(iter : T_1 ; result : T_2 = expr_2 \mid expr_3)$

where

- $expr_1$  is of a **collection type** (here: a set  $Set(\tau_0)$  for some  $\tau_0$ ),
- $iter \in W$  is called **iterator**, of the type denoted by  $T_1$  (if  $T_1$  is omitted,  $\tau_0$  is assumed as type of  $iter$ )
- $result \in W$  is called **result variable**, gets type  $\tau_2$  denoted by  $T_2$ ,
- $expr_2$  is an expression of type  $\tau_2$  giving the **initial value** for  $result$ , ( $OclUndefined_{\tau_2}$ , if omitted)
- $expr_3$  is an expression of type  $\tau_2$ , in particular  $iter$  and  $result$  may appear in  $expr_3$ .

## OCL Syntax 4/4: Context

**Syntax:** (Assuming signature  $\mathcal{S} = (\mathcal{F}, \mathcal{C}, V, atr)$ )

$context ::= context \ w_1 : T_1, \dots, w_n : T_n \ inv : expr$

where  $T_i \in \mathcal{C}$  and  $w_i : \tau_{T_i} \in W$  for all  $1 \leq i \leq n, n \geq 0$ .

**Semantics:**

$context \ w_1 : C_1, \dots, w_n : C_n \ inv : expr$

is (just) an **abbreviation** for

$allInstances_{C_1} \rightarrow forAll(w_1 : \#C_1 \mid$   
 $\dots$   
 $allInstances_{C_n} \rightarrow forAll(w_n : \#C_n \mid$   
 $expr$   
 $)$   
 $\dots$   
 $)$

# 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  $w \in W$ ,

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

- $\text{self}_{v_m} \in W$
- $\text{self}_{v_m} : \tau_{v_m}$  is an OCL expression
- $I[\llbracket \text{self}_{v_m} \rrbracket](\sigma, \beta) := \beta(\text{self}_{v_m})$
- $\beta_1 = \{ \text{self}_{v_m} \mapsto 1_{v_m} \}$   
↳  $I[\llbracket \text{self}_{v_m} \rrbracket](\sigma, \beta_1) = \beta_1(\text{self}_{v_m}) = 1_{v_m}$
- $\beta : W \longrightarrow I(\tau_B \cup \tau_C \cup \mathcal{J})$

## (v) Interpretation of OCL Expressions

$$\text{expr} ::= w \mid \omega(\text{expr}_1, \dots, \text{expr}_n) \mid \text{allInstances}_C \mid v(\text{expr}_1) \mid r_1(\text{expr}_1) \\ \mid r_2(\text{expr}_1) \mid \text{expr}_1 \text{->iterate}(v_1 : \tau_1 ; v_2 : \tau_2 = \text{expr}_2 \mid \text{expr}_3)$$

- $I[\overset{w}{\underbrace{w}}](\sigma, \beta) := \beta(w)$
- $I[\omega(\text{expr}_1, \dots, \text{expr}_n)](\sigma, \beta) := I(w) \left( I[\text{expr}_1](\sigma, \beta), \dots, I[\text{expr}_n](\sigma, \beta) \right)$
- $I[\text{allInstances}_C](\sigma, \beta) := \underbrace{\text{dom}(\sigma)}_{\text{all alive objects in } \sigma} \cap \underbrace{\mathcal{D}(C)}_{\text{objects of class } C}$

**Note:** in the OCL standard,  $\text{dom}(\sigma)$  is assumed to be **finite**.

Again: doesn't scare us.

# Example



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

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

- $I[\omega](\sigma, \beta) := \beta(\omega)$
- $I[\text{allInstances}_C](\sigma, \beta) := \text{dom}(\sigma) \cap \mathcal{D}(C)$
- $I[\omega(\text{expr}_1, \dots, \text{expr}_n)](\sigma, \beta) := I(\omega)(I[\text{expr}_1](\sigma, \beta), \dots, I[\text{expr}_n](\sigma, \beta))$

- $I[\text{allInstances}_{CP}](\sigma_1, \beta) = \text{dom}(\sigma_1) \cap \mathcal{D}(CP) = \{7_{VM}, 1_{DD}, 3_{CP}, 5_{CP}\} \cap \mathcal{D}(CP) \\ = \{3_{CP}, 5_{CP}\}$
- $I[\text{allInstances}_{CP} \rightarrow \text{size}](\sigma_1, \beta) = I[\text{size}(\text{allInstances}_{CP})](\sigma_1, \beta) \\ = I(\text{size})(I[\text{allInstances}_{CP}](\sigma_1, \beta)) = I(\text{size})(\{3_{CP}, 5_{CP}\}) = 2$
- $\beta_1 := \{3_{CP}\}, \quad I[\text{self}](\sigma_1, \beta_1) = \beta_1(\text{self}) = 3_{CP}$



## (v) Interpretation of OCL Expressions

$$\begin{aligned} \text{expr} ::= & w \mid \omega(\text{expr}_1, \dots, \text{expr}_n) \mid \text{allInstances}_C \mid v(\text{expr}_1) \mid r_1(\text{expr}_1) \\ & \mid r_2(\text{expr}_1) \mid \text{expr}_1 \text{->iterate}(v_1 : \tau_1 ; v_2 : \tau_2 = \text{expr}_2 \mid \text{expr}_3) \end{aligned}$$

Assume  $\text{expr}_1 : \tau_C$  for some  $C \in \mathcal{C}$ . Set  $u_1 := I[\![\text{expr}_1]\!](\sigma, \beta) \in \mathcal{D}(\tau_C) \cap I(\tau_C)$

- $I[\![v(\text{expr}_1)]\!](\sigma, \beta) := \begin{cases} \sigma(u_1)(v) & , \text{ if } u_1 \in \text{dom}(\sigma) \\ \perp & , \text{ otherwise} \end{cases}$

# Example

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



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

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

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

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

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

# (v) Interpretation of OCL Expressions

$$\text{expr} ::= w \mid \omega(\text{expr}_1, \dots, \text{expr}_n) \mid \text{allInstances}_C \mid v(\text{expr}_1) \mid r_1(\text{expr}_1) \mid r_2(\text{expr}_1) \mid \text{expr}_1 \text{->iterate}(v_1 : \tau_1 ; v_2 : \tau_2 = \text{expr}_2 \mid \text{expr}_3)$$

Assume  $\text{expr}_1 : \tau_C$  for some  $C \in \mathcal{C}$ . Set  $u_1 := I[\text{expr}_1](\sigma, \beta) \in \mathcal{D}(\tau_C)$ .

- $I[v(\text{expr}_1)](\sigma, \beta) := \begin{cases} \sigma(u_1)(v) & , \text{if } u_1 \in \text{dom}(\sigma) \\ \perp & , \text{otherwise} \end{cases}$
- $I[r_1(\text{expr}_1)](\sigma, \beta) := \begin{cases} u & , \text{if } v_1 \in \text{dom}(\sigma) \text{ and } \sigma(v_1)(r_1) = \{u\} \\ \perp & , \text{otherwise} \end{cases}$   
 $r_1 : \mathcal{C}_{0,1}$
- $I[r_2(\text{expr}_1)](\sigma, \beta) := \begin{cases} \sigma(v_1)(r_2) & , \text{if } v_1 \in \text{dom}(\sigma) \\ \perp & , \text{otherwise} \end{cases}$   
 $r_2 : \mathcal{C}_*$

Recall:  $\sigma$  evaluates  $r_2$  of type  $\mathcal{C}_*$  to a set.

# Iterate: Intuitive Semantics

```
expr ::= expr1 -> iterate(iter : T1;  
                                result : T2 = expr2 | expr3)
```

*Set*( $\tau_0$ ) *hlp* := *expr*<sub>1</sub>;

$\tau_1$  *iter*;

$\tau_2$  *result* := *expr*<sub>2</sub>;

while (!*hlp*.empty()) do

*iter* := *hlp*.pop();

*result* := *expr*<sub>3</sub>;

od;

return *result*;

pick and remove  
one element

may comprise  
itv and result

context CP inv : wem iter  
    ↓  
all hst<sub>CP</sub> → forall(self | wem(self))

# Iterate: Intuitive Semantics

```
expr ::= expr1 -> iterate(iter : T1;  
                                result : T2 = expr2 | expr3)
```

```
Set( $\tau_0$ ) hlp := expr1;  
 $\tau_1$  iter;  
 $\tau_2$  result := expr2;  
while (!hlp.empty()) do  
    iter := hlp.pop();  
    result := expr3;  
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

$$\begin{aligned} \text{expr} ::= & w \mid \omega(\text{expr}_1, \dots, \text{expr}_n) \mid \text{allInstances}_C \mid v(\text{expr}_1) \mid r_1(\text{expr}_1) \\ & \mid r_2(\text{expr}_1) \mid \text{expr}_1 \text{->iterate}(v_1 : \tau_1 ; v_2 : \tau_2 = \text{expr}_2 \mid \text{expr}_3) \end{aligned}$$

- $I[\text{expr}_1 \text{->iterate}(v_1 : \tau_1 ; v_2 : \tau_2 = \text{expr}_2 \mid \text{expr}_3)](\sigma, \beta)$

$$:= \begin{cases} I[\text{expr}_2](\sigma, \beta) & , \text{ if } I[\text{expr}_1](\sigma, \beta) = \emptyset \\ \textit{iterate}(hlp, v_1, v_2, \text{expr}_3, \sigma, \beta') & , \text{ otherwise} \end{cases}$$

where  $\beta' = \beta[hlp \mapsto I[\text{expr}_1](\sigma, \beta), v_2 \mapsto I[\text{expr}_2](\sigma, \beta)]$  and

- $\textit{iterate}(hlp, v_1, v_2, \text{expr}_3, \sigma, \beta')$

$$:= \begin{cases} I[\text{expr}_3](\sigma, \beta'[v_1 \mapsto x]) & , \text{ if } \beta'(hlp) = \{x\} \\ I[\text{expr}_3](\sigma, \beta'') & , \text{ if } \beta'(hlp) = X \dot{\cup} \{x\} \text{ and } X \neq \emptyset \end{cases}$$

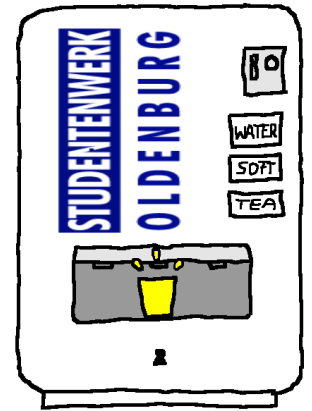
where  $\beta'' = \beta'[v_1 \mapsto x, v_2 \mapsto \textit{iterate}(hlp, v_1, v_2, \text{expr}_3, \sigma, \beta'[hlp \mapsto X])]$

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

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

# Example

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

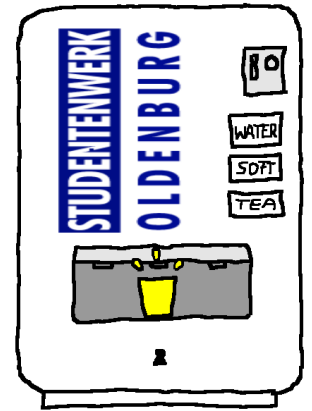


$$\sigma_1 = \{7_{VM} \mapsto \{dd \mapsto \{1_{DD}\}, cp \mapsto \{3_{DD}, 5_{DD}\}\}, 1_{DD} \mapsto \{wis \mapsto 13\}, \\ 3_{CP} \mapsto \{dd \mapsto \{1_{DD}\}, wen \mapsto true\}, 5_{CP} \mapsto \{dd \mapsto \{1_{DD}\}, wen \mapsto false\}\}$$

context  $CP$  inv :  $wen$  implies  $dd.wis > 0$   
     $\{$  unabbr.  
     $\}$

all instances  $CP \rightarrow \text{forall}(\text{self} / wen \text{ implies } dd.wis > 0)$

# Example

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

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

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

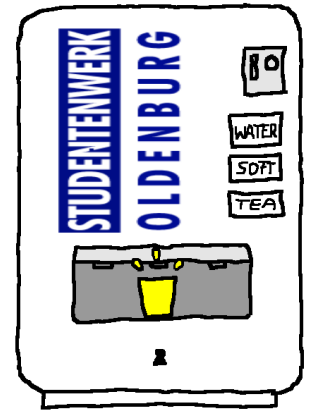
$\text{allInstances}_{CP} \rightarrow \text{forall} (self \mid self.wen \text{ implies } self.dd.wis > 0)$

$\Downarrow$   
 $\text{allInstances}_{CP} \rightarrow \text{iterate} (self; \uparrow : Bool = true \mid \uparrow \text{ and } \dots)$



# Example

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



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

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

$allInstances_{CP} \rightarrow iterate (self; r : Bool = true \mid \underbrace{and(r, implies(wen(self), >(wis(dd(self)), 0)))}_{=: expr_1})$

$$\mathbb{I} allInstances_{CP} \mathbb{J}(\sigma_1, \emptyset) = \{3_{CP}, 5_{CP}\}$$

$\underbrace{\quad}_{=: \beta_1}$

$$\mathbb{I} expr_1 \mathbb{J}(\sigma_1, \{self \mapsto 3_{CP}, r \mapsto true\}) = \mathbb{I} and \mathbb{J}(\mathbb{I} true \mathbb{J}(\sigma_1, \beta_1), \mathbb{I} expr_2 \mathbb{J}(\sigma_1, \beta_1)) \stackrel{(*)}{=} \mathbb{I} and \mathbb{J}(true, true) = true$$

$$\mathbb{I} expr_2 \mathbb{J}(\sigma_1, \beta_1) = \mathbb{I} implies \mathbb{J}(\underbrace{\mathbb{I} wen(self) \mathbb{J}(\sigma_1, \beta_1)}_{true}, \mathbb{I} > \mathbb{J}(\underbrace{\mathbb{I} wis(dd(self)) \mathbb{J}(\sigma_1, \beta_1)}_{13}, 0)) \stackrel{(**)}{=} true$$

$$\mathbb{I} dd(self) \mathbb{J}(\sigma_1, \beta_1) = 1_{DD}$$

$$\mathbb{I} wis(dd(self)) \mathbb{J}(\sigma_1, \beta_1) = 13$$

$$\stackrel{(***)}{=} \mathbb{I} implies \mathbb{J}(true, true) = true \quad (*)$$

$$\mathbb{I} expr_1 \mathbb{J}(\sigma_1, \{self \mapsto 5_{CP}, r \mapsto true\}) = true \quad (**)$$

$$\mathbb{I} F \mathbb{J}(\sigma_1, \beta_1) = true$$

# 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

$$I\llbracket expr \rrbracket(\sigma, \beta) \in \{true, false, \perp_{Bool}\}$$

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

- using the **interpretation function**

$$I\llbracket \cdot \rrbracket(\cdot, \cdot) : OCLExpressions(\mathcal{S}) \times \Sigma_{\mathcal{D}}^{\mathcal{D}} \times (W \rightarrow I(\mathcal{T} \cup T_B \cup T_C)) \\ \rightarrow I(Bool).$$

# User's Guide

- **App**  
The  
It is  
**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).
- **Inte**  
Absc  
27

- **Exercise submissions:**

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

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

# User's Guide

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

**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:**

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

## 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  
 $\sigma_1 = \{ \dots \}$   
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.