

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

-4-2016-11-03-math-

Content

- The **Object Constraint Language** (OCL):
 - **Semantics**
 - Overview
 - OCL Types
 - Arithmetic / Logical Operators
 - OCL Expressions
 - Iterate
 - **A Complete Example**

-4-2016-11-03-5content-

Recall

OCL Syntax 1/4: Expressions

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

- $w \in W \supset \{\text{set } \tau : \tau \in \mathcal{C} \in \mathcal{V}\}$ is a set of typed logical variables, w has type $\tau(w)$.
- τ is any type from $\mathcal{F} \cup T_B \cup T_{\mathcal{C}} \cup \{\text{Set}(t_0) \mid t_0 \in \mathcal{F} \cup T_B \cup T_{\mathcal{C}}\}$ in the following use.
- T_B is a set of (OCL) basic types.
- $T_{\mathcal{C}} = \{\text{Bool}, \text{Int}, \text{String}\}$
- $T_{\mathcal{V}} = \{\tau \in \mathcal{C} \mid \mathcal{C} \in \mathcal{V}\}$ is the set of object types.
- $\text{Set}(t_0)$ denotes the set-of- t_0 type for $t_0 \in T_B \cup T_{\mathcal{C}}$ (sufficient because of "flattening" [cf. standard]).

```

expr ::= ...
w      : τ(w)
| expr1 == expr2 : τ × τ → Bool
| oclIsUndefined(expr1) : τ → Bool
| { expr1, ..., expr_n } : τ × ... × τ → Set(τ)
| isEmpty(expr1) : Set(τ) → Bool
| size(expr1) : Set(τ) → Int
| allInstances_C : Set(τ_C)

where
  τ : τ ∈ \mathcal{F} \in \text{atr}(\mathcal{C}), \tau \in \mathcal{F},
  \tau_1 : \tau_1 \in \text{atr}(\mathcal{C}), \mathcal{C}, D \in \mathcal{V},
  \tau_2 : \tau_2 \in \text{atr}(\tau_1), \mathcal{C}, D \in \mathcal{V}.
  
```

OCL Syntax 2/4: Constants & Arithmetics

For example:

```

expr ::= ...
| true | false : Bool
| expr1 {and, or, implies} expr2 : Bool × Bool → Bool
| not expr1 : Bool → Bool
| 0 | 1 | -1 | ... : Int
| expr1 {+, -, *} expr2 : Int × Int → Int
| expr1 {<, <=, ...} expr2 : Int × Int → Bool
| OclUndefined : τ
  
```

Generalised notation: (not a normal form)

$$\text{expr} ::= \omega(\text{expr}_1, \dots, \text{expr}_n) : \tau_1 \times \dots \times \tau_n \rightarrow \tau$$

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

Handwritten note: $1 + 2, 4 \rightarrow + (1, 2)$
 ω expr₁ expr₂

OCL Syntax 3/4: Iterate

```

expr ::= ... | expr1 -> iterate(w1 : T1; w2 : T2 = expr2 | expr3)
  
```

or, with a little renaming,

```

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

where

- expr_1 is of a collection type (here a set $\text{Set}(t_0)$ for some t_0).
- $\text{iter} \in W$ is called **iterator**, of the type denoted by T_1 (if T_1 is omitted, τ_0 is assumed as type of iter).
- $\text{result} \in W$ is called **result variable**, gets type τ_2 denoted by T_2 .
- expr_2 in an expression of type τ_2 giving the **initial value** for result , ($\text{OclUndefined}_{\tau_2}$ if omitted).
- expr_3 is an expression of type τ_2 , in particular iter and result may appear in expr_3 .

OCL Syntax 4/4: Context

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

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

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

Semantics:

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

is (just) an abbreviation for

$$\text{allInstances}_{C_1} \rightarrow \text{forAll}(w_1 : \bullet_{C_1} | \dots \text{ allInstances}_{C_n} \rightarrow \text{forAll}(w_n : \bullet_{C_n} | \text{expr} \dots)$$

-4- 2016-11-03 - Shikhar -

OCL Semantics: The Task

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

$$\text{expr}_1 = \text{context } CP \text{ inv } : \text{wen implies } dd . \text{wis} > 0$$
 - and a system state

$$\sigma_1 = \{7VM \mapsto \{dd \mapsto \{1DD\}, cp \mapsto \{3DD, 5DD\}\}, 1DD \mapsto \{\text{wis} \mapsto 13\}, 3CP \mapsto \{dd \mapsto \{1DD\}, \text{wen} \mapsto \text{true}\}, 5CP \mapsto \{dd \mapsto \{1DD\}, \text{wen} \mapsto \text{false}\}\} \in \Sigma_{\mathcal{S}}$$
 - and a valuation of the logical variables $\beta_1 : W \rightarrow I(\mathcal{F} \cup T_B \cup T_{\mathcal{C}})$,

Handwritten note: $\tau \in \{\text{set } \tau \mid \tau \in \mathcal{C} \in \mathcal{V}\}$
- compute** the value $I[\llbracket \text{expr}_1 \rrbracket](\sigma_1, \beta_1) \in \{\text{true}, \text{false}, \perp_{\text{Bool}}\}$ of expr_1 in σ_1 under β_1 .

Handwritten note: \uparrow three-valued logic
- More general: **Define** the **interpretation** $I[\llbracket \text{expr} \rrbracket](\sigma, \beta)$ of expr in σ under β :

$$I[\llbracket \cdot \rrbracket](\cdot, \cdot) : \text{OCLExpressions}(\mathcal{S}) \times \Sigma_{\mathcal{S}} \times (W \rightarrow I(\mathcal{F} \cup T_B \cup T_{\mathcal{C}})) \rightarrow I(\text{Bool})$$

Handwritten arrows: Green arrows point from the arguments of the function to the corresponding parts of the function signature.

-4- 2016-11-03 - Shikhar -

OCL Semantics OMG (2006)

-4-2016-11-03 - math -

5/29

Basically business as usual...

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

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

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

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

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

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

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

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

$$I_{\tau} : Expr \times \Sigma_{\mathcal{S}} \times (W \rightarrow I_{\tau}(\mathcal{T} \cup T_B \cup T_{\mathcal{E}})) \rightarrow I_{\tau}(Bool)$$

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

$$I : \mathcal{I}_{\tau_1} \cup \mathcal{I}_{\tau_2} \cup \mathcal{I}_{\tau_3} \cup \dots \cup \mathcal{I}_{\tau_n} \cup \mathcal{I}_{\tau_{n+1}}$$

-4-2016-11-03 - SoSem -

6/29

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

Recall: OCL basic types

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

We set:

- $I(Bool) := \{true, false, \perp_{Bool}\}$
 - $I(Int) := \mathbb{Z} \dot{\cup} \{\perp_{Int}\}$
 - $I(String) := \dots \dot{\cup} \{\perp_{String}\}$
- \swarrow three-valued
 \uparrow disjoint union

We may omit index τ of \perp_τ 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 $T \in \mathcal{T}$.

-4-2016-11-03 - SoSeMTypes -

7/29

OCL and Model Types?! An Example.

$$\begin{aligned} \mathcal{S} = & \{ \{Bool, Nat\}, \{VM, CP, DD\}, \\ & \{cp : CP^*, dd : DD_{0,1}, wen : Bool, wis : Nat\}, \\ & \{VM \mapsto \{cp, dd\}, CP \mapsto \{wen, dd\}, DD \mapsto \{wis\} \} \end{aligned}$$

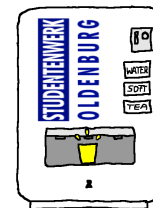
Model Types:

$$\begin{aligned} \mathcal{D}(Bool_M) &= \{0, 1\} \\ \mathcal{D}(Nat) &= \{0, \dots, 255\} \\ \mathcal{D}(VM) &= \mathbb{N} \times \{VM\} \\ &= \{1_{VM}, 2_{VM}, \dots\} \end{aligned}$$

OCL Types:

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

$$\begin{aligned} I(Bool_M) &= \mathcal{D}(Bool_M) \dot{\cup} \{\perp_{Bool_M}\} \\ &= \{0, 1, \perp_{Bool_M}\} \\ I(Nat) &= \mathcal{D}(Nat) \dot{\cup} \{\perp_{Nat}\} \\ &= \{0, \dots, 255\} \dot{\cup} \{\perp_{Nat}\} \\ I(\tau_{VM}) &= \mathcal{D}(VM) \dot{\cup} \{\perp_{VM}\} \end{aligned}$$



-4-2016-11-03 - SoSeMTypes -

8/29

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

- Let τ_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 τ 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:

$I(\text{Bool})$	$I(\text{Bool})$	$I(\text{Int})$
↓	↓	↓
$I(\text{true}) := \text{true}$	$I(\text{false}) := \text{false}$	$I(0) := 0, \quad I(1) := 1, \dots$
↑	↑	
$OclExpr(\mathcal{S})$	$I(\text{OclUndefined}_{\tau}) := \perp_{\tau}$	

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

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

$$I(=_{\tau}) : I(\tau) \times I(\tau) \rightarrow I(\text{Bool})$$

- 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.

-4-2016-11-03 - SoSeSemantik -

11/29

(iv) Interpretation of Set Operations

Basically the same principle as with arithmetic operations...

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

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

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

↑
number of elements in x

-4-2016-11-03 - SoSeSemantik -

12/29

(v) Interpretation of OCL Expressions

OCL Syntax 1/4: Expressions

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

- $w \in W \supset \{\text{self}_c : \tau_c \mid C \in \mathcal{C}\}$ is a set of typed logical variables, w has type $\tau(w)$
- τ is any type from $\mathcal{F} \cup T_B \cup T_E \cup \{\text{Set}(T_0) \mid T_0 \in \mathcal{F} \cup T_B \cup T_E\}$ in the following use
- T_B is a set of (OCL) basic types, in the following use
- $T_E = \{\text{Bool}, \text{Int}, \text{String}\}$
- $T_0 = \{\tau_c \mid C \in \mathcal{C}\}$ is the set of object types.
- $\text{Set}(T_0)$ denotes the set-of- T_0 type for $T_0 \in T_B \cup T_E$ (sufficient because of "flattening" [cf. standard])

w : $\tau(w)$
 self_c : τ_c
 OclUndefined_c : τ_c
 $\{ \text{expr}_1, \dots, \text{expr}_n \}$: $\tau \times \dots \times \tau \rightarrow \text{Set}(\tau)$
 $\text{isEmpty}(\text{expr}_1)$: $\text{Set}(\tau) \rightarrow \text{Bool}$
 $\text{size}(\text{expr}_1)$: $\text{Set}(\tau) \rightarrow \text{Int}$
 allInstances_c : $\text{Set}(\tau_c)$

$\forall (\text{expr}_1)$: $\tau_0 \rightarrow \tau$ where $w : \tau \in \text{atr}(C), \tau \in \mathcal{F}$
 $\exists (\text{expr}_1)$: $\tau_0 \rightarrow \tau$ where $r_1 : D_{0,1} \in \text{atr}(C), C, D \in \mathcal{C}$
 $\exists (\text{expr}_1)$: $\tau_0 \rightarrow \text{Set}(\tau_0)$ where $r_2 : D_2 \in \text{atr}(C), C, D \in \mathcal{C}$

OCL Syntax 2/4: Constants & Arithmetics

For example:

$\text{true} | \text{false}$: Bool
 $\text{expr}_1 \{ \text{and, or, implies} \} \text{expr}_2$: $\text{Bool} \times \text{Bool} \rightarrow \text{Bool}$
 not expr_1 : $\text{Bool} \rightarrow \text{Bool}$
 $! | - | + | - | \dots$: Int
 $\text{expr}_1 \{ +, -, \dots \} \text{expr}_2$: $\text{Int} \times \text{Int} \rightarrow \text{Int}$
 $\text{expr}_1 \{ <, \leq, \dots \} \text{expr}_2$: $\text{Int} \times \text{Int} \rightarrow \text{Bool}$
 OclUndefined_c : τ

Generalised notation: (not a normal form)

$\text{expr} ::= \omega(\text{expr}_1, \dots, \text{expr}_n)$: $\tau_1 \times \dots \times \tau_n \rightarrow \tau$
 with $\omega \in \{+, -, \dots\}$

$1 + 2 \text{ and } 2 - 1$
 $\omega \text{ expr}_1 \text{ expr}_2$

OCL Syntax 3/4: Iterate

$\text{expr} ::= \dots | \text{expr}_1 \rightarrow \text{iterate}(w_1 : T_1; w_2 : T_2 = \text{expr}_2 | \text{expr}_3)$

or, with a little renaming,

$\text{expr} ::= \dots | \text{expr}_1 \rightarrow \text{iterate}(\text{iter} : T_1; \text{result} : T_2 = \text{expr}_2 | \text{expr}_3)$

where

- expr_1 is of a collection type (here: a set $\text{Set}(\tau_0)$ for some τ_0).
- $\text{iter} \in W$ is called iterator, of the type denoted by T_1 (if T_1 is omitted, τ_0 is assumed as type of iter).
- $\text{result} \in W$ is called result variable, gets type τ_2 denoted by T_2 .
- expr_2 in an expression of type τ_2 giving the initial value for result, ($\text{OclUndefined}_{\tau_2}$ if omitted).
- expr_3 is an expression of type τ_2 , in particular iter and result may appear in expr_3 .

OCL Syntax 4/4: Context

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

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

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

Semantics:

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

is (just) an abbreviation for

$\text{allInstances}_{C_1} \rightarrow \text{forAll}(w_1 : \bullet_{C_1} | \dots$
 \dots
 $\text{allInstances}_{C_n} \rightarrow \text{forAll}(w_n : \bullet_{C_n} |$
 expr
 $)$
 $)$

-4- 2016-11-03 - SoSeSemestr -

Valuations of Logical Variables

- **Recall:** we have typed logical variables ($w \in W$), $\tau(w)$ is the type of w .
- By β , 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
- $I[\text{self}_{V_M}](\sigma, \beta) := \beta(\text{self}_{V_M})$
- $\beta_0 = \{ \text{self}_{V_M} \mapsto 1_{V_M} \}$
 $\hookrightarrow I[\text{self}_{V_M}](\sigma, \beta_0) = \beta_0(\text{self}_{V_M}) = 1_{V_M}$
- $\beta : W \rightarrow I(T_B \cup T_C \cup \mathcal{J})$

-4- 2016-11-03 - SoSeSemestr -

(v) Interpretation of OCL Expressions

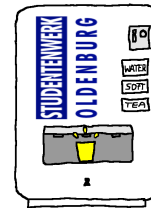
$$\text{expr} ::= w \mid \omega(\text{expr}_1, \dots, \text{expr}_n) \mid \text{allInstances}_C \mid v(\text{expr}_1) \mid r_1(\text{expr}_1) \mid r_2(\text{expr}_1) \mid \text{expr}_1 \rightarrow \text{iterate}(v_1 : \tau_1 ; v_2 : \tau_2 = \text{expr}_2 \mid \text{expr}_3)$$

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

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

Example

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

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


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

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

(v) Interpretation of OCL Expressions

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

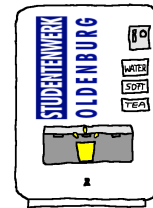
Assume $\text{expr}_1 : \tau_C$ for some $C \in \mathcal{C}$. Set $u_1 := I[\text{expr}_1](\sigma, \beta) \in \mathcal{D}(\tau_C) \mid \mathcal{I}(\tau_C)$

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

Example

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

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



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

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

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

(v) Interpretation of OCL Expressions

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

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

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

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

Iterate: Intuitive Semantics

$$\text{expr} ::= \text{expr}_1 \rightarrow \text{iterate}(\text{iter} : T_1 ; \text{result} : T_2 = \text{expr}_2 \mid \text{expr}_3)$$

```

Set( $\tau_0$ ) hlp :=  $\text{expr}_1$ ;
 $\tau_1$  iter;
 $\tau_2$  result :=  $\text{expr}_2$ ;
while (!hlp.empty()) do
    iter := hlp.pop();
    result :=  $\text{expr}_3$ ;
od;
return result;
    
```

pick and remove one element

may comprise iter and result

context CP inv : len
 $\{$
 all hst_{CP} \rightarrow forall (self | $\text{len}(\text{self})$)

Iterate: Intuitive Semantics

$$\text{expr} ::= \text{expr}_1 \text{->iterate}(\text{iter} : T_1; \\ \text{result} : T_2 = \text{expr}_2 \mid \text{expr}_3)$$

```

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 $\text{expr}_1 : \text{Set}(\tau_0)$ and $\tau_1 = \tau_0$. In the type hierarchy of full OCL with inheritance and `oclAny`, τ_0 and τ_1 may be different and still type consistent.

-4-2016-11-03 - SoSeWinter -

20/29

(v) Interpretation of OCL Expressions

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

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

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

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

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

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

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

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

Quiz: Is (our) I a function?

-4-2016-11-03 - SoSeWinter -

21/29

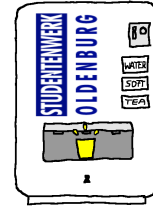
Example

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

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

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

allInstances_{CP} → forall (self / ^{undbri.} wen implies dd.wis > 0)



Example

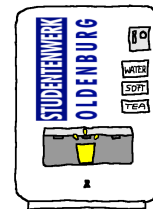
$$\mathcal{S} = (\{Bool, Nat\}, \{VM, CP, DD\}, \\ \{cp : CP_*, dd : DD_{0,1}, wen : Bool, wis : Nat\}, \\ \{VM \mapsto \{cp, dd\}, CP \mapsto \{wen, dd\}, DD \mapsto \{wis\}\})$$

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

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

allInstances_{CP} → forall (self | self.wen implies self.dd.wis > 0)

allInstances_{CP} → iterate (self ; [↓] f = Bool = true | f and ...)



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 ($self; r : Bool = true \mid$ and($r, implies(wen(self), >(wis(dd(self)), 0))$))

$I \llbracket allInstances_{CP} \rrbracket (\sigma, \emptyset) = \{3_{CP}, 5_{CP}\}$

$I \llbracket expr_1 \rrbracket (\sigma, \beta) = I \llbracket and \rrbracket (I \llbracket r \rrbracket (\sigma, \beta), I \llbracket implies \rrbracket (\sigma, \beta)) = I \llbracket and \rrbracket (true, true) = true$
 $I \llbracket implies \rrbracket (\sigma, \beta) = I \llbracket implies \rrbracket (I \llbracket wen \rrbracket (\sigma, \beta), I \llbracket > \rrbracket (I \llbracket wis \rrbracket (dd(self)) (\sigma, \beta), 0)) = true$
 $I \llbracket dd \rrbracket (self) (\sigma, \beta) = 1_{DD}$
 $I \llbracket wis \rrbracket (dd(self)) (\sigma, \beta) = 13$

$\stackrel{**}{=} I \llbracket implies \rrbracket (true, true) = true$

$I \llbracket F \rrbracket (\sigma, \beta) = true$ $I \llbracket F \rrbracket (\sigma, \beta) = true$

-4-2016-11-03 - Schemata -

22/29

Tell Them What You've Told Them...

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

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

of $expr$ in σ under β

- using the **interpretation function**

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

-4-2016-11-03 - Schemata -

23/29

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:

27

- **Inte**
Abs

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).

-1-2016-10-18 - 5formath-

31/34

24/29

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:



- **Inte**
Abs

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).

-1-2016-10-18 - 5formath-

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.

29/34

26/29

- 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.