Software Design, Modelling and Analysis in UML

Lecture 04: OCL Semantics

2014-10-30

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

Albert-Ludwigs-Universität Freiburg, Germany

Contents & Goals

Last Lecture:

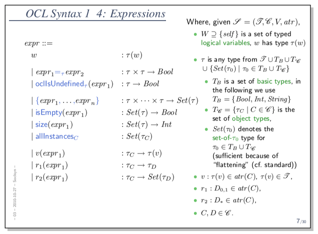
- OCL Syntax

This Lecture:

- **Educational Objectives:** Capabilities for following tasks/questions.
  - What 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? *(next time)*
- **Content:**
  - OCL Semantics
  - maybe: OCL consistency and satisfiability

OCL Semantics [OMG, 2006]

The Task



$I(\cdot,\cdot)$

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

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

$= \{true, false, \bot_{Bool}\}$

such that

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

Basically business as usual...

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

$$I \text{ with } \mathrm{dom}(I) = T_B \text{ } = \{Int, Bool, String\}$$

e.g. $I(Bool) = \{true, false, \bot_{Bool}\}$

Basically business as usual...

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

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

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

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

(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

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

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

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

$expr_1 + expr_2 : Int \times Int \to Int$

Basically business as usual...

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

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

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

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

(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

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

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

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

(v) **Set operations** similar: I with $\mathrm{dom}(I) = \{\text{isEmpty}, \dots\}$

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

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

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

(i) Domains of Basic Types of OCL

Recall:

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

We set:

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

assume both sets disjoint; read "bottom" or "undefined"; 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

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

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

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

powerset of $I(\tau)$

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:

$$I(\textsf{true}) := true, \quad I(\textsf{false}) := false, \quad I(0) := 0, \quad I(1) := 1, \dots$$
$$I(\textsf{OclUndefined}_\tau) := \bot_\tau$$

$true \in I(Bool)$, $0 \in I(Int) = \mathbb{Z} \dot\cup \{\bot_{Int}\}$

(iv) Interpretation of Arithmetic Operations

- **Literals** map to fixed values:

$$I(\text{true}) := true, \quad I(\text{false}) := false, \quad I(0) := 0, \quad I(1) := 1, \dots$$
$$I(\text{OclUndefined}_\tau) := \bot_\tau$$

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

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

$=_\tau : \tau \times \tau \to Bool$; $I(=_\tau) : I(\tau) \times I(\tau) \to I(Bool) = \{true, false, \bot\}$

(iv) Interpretation of Arithmetic Operations

- **Literals** map to fixed values:

$$I(\text{true}) := true, \quad I(\text{false}) := false, \quad I(0) := 0, \quad I(1) := 1, \dots$$
$$I(\text{OclUndefined}_\tau) := \bot_\tau$$

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

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

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

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

$+ : Int \times Int \to Int$; $I(+) : I(Int) \times I(Int) \to I(Int) = \mathbb{Z} \dot\cup \{\bot\}$

(iv) Interpretation of Arithmetic Operations

- **Literals** map to fixed values:

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

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

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

- **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**. $\omega(expr_1, \ldots, expr_n)$ e.g. $+(expr_1, expr_2)$

Namely, the **interpretation** of an operation $\omega : \tau_1 \times \ldots \tau_n \to \tau$ is a function $I(\omega) : I(\tau_1) \times \cdots \times I(\tau_n) \to I(\tau)$ on corresponding semantical domain(s).

$0 + 27 = 13$

$= (+(0, 27), 13)$

$I(0) = 0$

$I(27) = 27$

$I(13) = 13$

$I(+) : I(Int) \times I(Int) \to I(Int)$

$I(=) : I(Int) \times I(Int) \to I(Bool)$

actually $=_{Int}$ here — see previous slide

$I[\![\omega(expr_1, \ldots, expr_n)]\!](\sigma, \beta) := (I(\omega))(I[\![expr_1]\!](\sigma, \beta), \ldots, I[\![expr_n]\!](\sigma, \beta))$

$I[\![+(0, 27)]\!](\sigma, \beta) = (I(+))(I(0), I(27)) = 27$

(iv) Interpretation of OclIsUndefined

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

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

(v) Interpretation of Set Operations

Basically the same principle as with arithmetic operations...

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

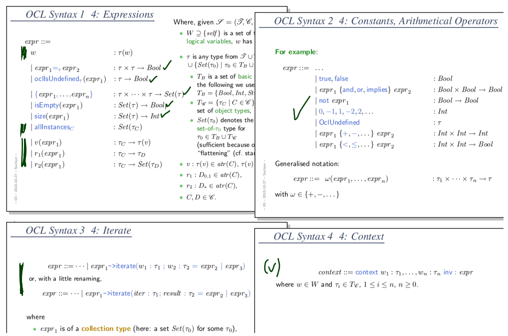
$$I(\text{isEmpty}^\tau)(x) := \begin{cases} true & \text{, if } x = \emptyset \\ \bot_{Bool} & \text{, if } x = \bot_{Set(\tau)} \\ false & \text{, otherwise} \end{cases}$$

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

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

(cardinality)

(vi) Putting It All Together



Valuations of Logical Variables

$\{self_C \mid C \in \mathscr{C}\}$, $self_C : \tau_C$

- **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 $w \in W$,

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

$\beta : W \to I(Int) \cup I(Bool) \cup \ldots \cup I(\tau_C) \cup \ldots = \bigcup_{w \in W} I(\tau(w))$

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

$\beta : W \to I(Int) \cup I(\tau_C) = \mathbb{Z} \cup \{\bot\} \cup \mathscr{D}(C)$

Examples:

- $\beta(x) = 27 \in I(Int)$, $\beta(self_C) = 1_C \in I(\tau_C) = \mathscr{D}(C)$
- $\beta(x) = \bot_{Int}$, $\beta(self_C) = 5_C$

(vi) Putting It All Together...

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

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

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

(vi) Putting It All Together...

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

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

- $I[\![v(expr_1)]\!](\sigma, \beta) := \begin{cases} \sigma(u_1)(v) & \text{, if } u_1 \in dom(\sigma) \\ \bot & \text{, otherwise} \end{cases}$ ($\mathcal{T}$)
- $I[\![r_1(expr_1)]\!](\sigma, \beta) := \begin{cases} u & \text{, if } u_1 \in dom(\sigma) \text{ and } \sigma(u_1)(r_1) = \{u\} \\ \bot & \text{, otherwise} \end{cases}$ ($D_{0,1}$)
- $I[\![r_2(expr_1)]\!](\sigma, \beta) := \begin{cases} \sigma(u_1)(r_2) & \text{, if } u_1 \in dom(\sigma) \\ \bot & \text{, otherwise} \end{cases}$ (D_*)

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

$\mathscr{S} = (\{Nat\}, \{TeamMember, Meeting\}, \{age : Nat, m : M_{0,1}, p : TM_*\}, \{TM \mapsto \{age, m\}, M \mapsto \{p\}\})$

$\mathscr{D}(Nat) = \mathbb{N}_0$

$W = \{x, self_M, self_{TM}\}$

$\sigma = \{1_{TM} \mapsto (27, 5_M), 2_{TM} \mapsto (17, 5_M), 5_M \mapsto (\{1_{TM}, 2_{TM}\})\}$

- $I[\![allInstances_{TM}]\!](\sigma,\beta) = dom(\sigma) \cap \mathscr{D}(TM) = \{1_{TM}, 2_{TM}, 5_M\} \cap \mathscr{D}(TM) = \{1_{TM}, 2_{TM}\}$
- $\beta_1 : x \mapsto 10, \ldots$

  $I[\![x > allInstances_{TM}\text{->}size]\!](\sigma,\beta)$
  $= I[\![>(x, size(allInstances_{TM}))]\!](\sigma,\beta) = (I(>))\big(\underbrace{I[\![x]\!](\sigma,\beta_1)}_{\beta_1(x)=10}, \underbrace{I(size)(\underbrace{I[\![allInst_{TM}]\!](\sigma,\beta_1)}_{\{1_{TM},2_{TM}\}})}_{2}\big)$ — true
- $\beta_2 : self_C \mapsto 2_{TM}, \ldots$ (and some value for x)

  $I[\![self_C . age]\!](\sigma,\beta_2) = I[\![age(self_C)]\!](\sigma,\beta_2)$
  $= \sigma(u_1)(age) = \sigma(2_{TM})(age) = 17$
  $u_1 = I[\![self_C]\!](\sigma,\beta_2) = \beta_2(self_C) = 2_{TM}$
- $\beta_3 : self_C \mapsto 7_{TM}, \ldots$ $I[\![self_C . age]\!](\sigma,\beta_3) = \bot$ because $7_{TM} \notin dom(\sigma)$

(vi) Putting It All Together...

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

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

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

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

(vi) Putting It All Together...

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

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

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

  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?

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.