Software Design, Modelling and Analysis in UML

Lecture 4: OCL Semantics

2015-11-03

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

Albert-Ludwigs-Universität Freiburg, Germany

Contents & Goals

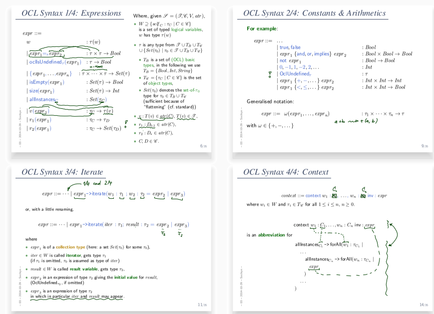
Last Lecture:

- OCL Syntax

This Lecture:

- **Educational Objectives:** Capabilities for these tasks/questions:
  - Please un-abbreviate all abbreviations in this OCL expression. ✓
  - Please explain this OCL constraint.
  - Please formalise this constraint in OCL.
  - Does this OCL constraint hold in this system state?
  - Give a system state satisfying this constraint?
  - In what sense is OCL a three-valued logic? For what purpose?
  - How are $\mathscr{D}(C)$ and T_C related?
- **Content:**
  - OCL Semantics
  - OCL Consistency and Satisfiability

Recall



OCL Semantics: The Task

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

$$expr_1 = \text{context } DD \text{ inv} : wen \text{ implies } win > 0$$

- and a system state $\sigma \in \Sigma_{\mathscr{S}}^{\mathscr{D}}$, e.g.

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

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

- **define** the **interpretation of** $expr$ **in** σ **under** β

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

i.e.

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

OCL Semantics OMG (2006)

Basically business as usual...

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

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

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

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

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

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

(iv) **Set operations** similar: $I_{(iv)}$ with $\operatorname{dom}(I) = \{\text{isEmpty}, \dots\}$

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

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

Basically business as usual...

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

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

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

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

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

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

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

(v) 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 OCL and (!) Model Basic Types

Recall: **OCL basic types**

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

We set:

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

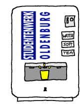
We may omit index τ of $\bot_\tau$ if it is clear from context.

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

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

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

OCL and Model Types?! An Example.



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

Model Types:

$\mathscr{D}_1(Bool_M) = \{0, 1\}$

$\mathscr{D}_1(Nat) = \{0, \dots, 255\}$

$\mathscr{D}_1(VM) = \{1_{VM}, 2_{VM}, \dots\}$

OCL Types:

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

$I(Nat) = \{0, \dots, 255\} \mathbin{\dot\cup} \{\bot_{Nat}\}$

$I(\tau_{VM}) = \{1_{VM}, 2_{VM}, \dots\} \mathbin{\dot\cup} \{\bot_{\tau_{VM}}\}$

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

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

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

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

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

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

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

(iii) Interpretation of Arithmetic Operations

- **Literals** map to fixed values:

$$\underset{OCLExpr}{I(\mathsf{true})} := \underset{I(Bool)}{true}, \quad I(\mathsf{false}) := false, \quad I(0) := \overset{I(Int)}{0}, \quad I(1) := 1, \dots$$

$$I(\mathsf{OclUndefined}_\tau) := \bot_\tau$$

OCLExpr $\ni$ OclUndefined, $\bot_\tau \in I(\tau)$; $0 \in$ OCLExpr, $0 \in I(Int)$

(iii) Interpretation of Arithmetic Operations

- **Literals** map to fixed values:

$$\underset{OCLExpr}{I(\mathsf{true})} := \underset{I(Bool)}{true}, \quad I(\mathsf{false}) := false, \quad I(0) := \overset{I(Int)}{0}, \quad I(1) := 1, \dots$$

$$I(\mathsf{OclUndefined}_\tau) := \bot_\tau$$

OCLExpr $\ni$ OclUndefined, $\bot_\tau \in I(\tau)$; $0 \in$ OCLExpr, $0 \in I(Int)$

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

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

$x_1, x_2 \in I(\tau)$, $true \in I(Bool)$

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

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

but:
$I(\vee)(false, \bot) = \bot$
$I(\wedge)(false, \bot) = false$
and?

Note: There is a **common principle**.
The **interpretation** of an operation (symbol) $\omega : \tau_1 \times \dots \tau_n \to \tau$
is a function $I(\omega) : I(\tau_1) \times \dots \times I(\tau_n) \to I(\tau)$ on corresponding semantical domain(s).

(iii) Interpretation of OclIsUndefined

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

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

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

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

$$I(\{\}_n^\tau)(\underset{\in I(\tau)}{x_1}, \dots, \underset{\in I(\tau)}{x_n}) := \underbrace{\{x_1, \dots, x_n\}}_{\in I(Set(\tau))}$$

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

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

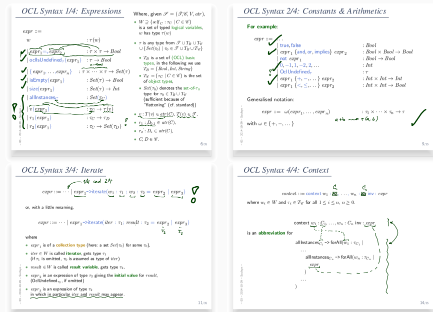
$$I(\text{isEmpty}^\tau)(x) := \begin{cases} \mathit{true} & \text{, if } x = \emptyset \\ \bot_{Bool} & \text{, if } x = \bot_{Set(\tau)} \\ \mathit{false} & \text{, otherwise} \end{cases}$$

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

number of elements in x

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

(v) Putting It All Together



Valuations of Logical Variables

- **Recall**: we have typed logical variables $(\underbrace{w \in) W}_{\ni \{self_C \mid C \in \mathscr{C}\}}$, $\tau(w)$ is the type of w.

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

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

$\beta : W \longrightarrow I(T_B \cup T_\mathscr{C} \cup \mathscr{T})$

$\beta = \{ \text{result} \mapsto \text{true},\ self_{VM} \mapsto 27_{VM} \}$

$\text{result} \in W$, $I(\tau(\text{result})) = I(Bool)$

$self_{VM} \in W$, $27_{VM} \in I(\tau(self_{VM})) = I(\tau_{self_{VM}})$

(v) Putting It All Together...

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

double-w

- $I[\![w]\!](\sigma, \beta) := \beta(w)$ e.g. self

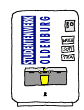
- $I[\![\omega(expr_1, \dots, expr_n)]\!](\sigma, \beta) := I(\omega)\big(I[\![expr_1]\!](\sigma,\beta), \dots, I[\![expr_n]\!](\sigma,\beta)\big)$ omega, e.g. $+(\beta, \text{result})$; e.g. $I(+)(I[\![3]\!](\sigma,\beta), I[\![result]\!](\sigma,\beta))$

- $I[\![\text{allInstances}_C]\!](\sigma, \beta) := \underbrace{\text{dom}(\sigma)}_{\text{all alive objects in } \sigma} \cap \underbrace{\mathscr{D}(C)}_{\text{objects of class } C}$

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

Example

$\beta = \{ self_{VM} \mapsto 7_{VM},\ x \mapsto 27 \}$



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

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

- $I[\![w]\!](\sigma, \beta) := \beta(w)$ (1)
- $I[\![\text{allInstances}_C]\!](\sigma, \beta) := \text{dom}(\sigma) \cap \mathscr{D}(C)$ (2)
- $I[\![\omega(expr_1, \dots, expr_n)]\!](\sigma, \beta) := I(\omega)(I[\![expr_1]\!](\sigma, \beta), \dots I[\![expr_n]\!](\sigma, \beta))$ (3)

- $I[\![self_{VM}]\!](\sigma_1, \beta) \overset{(1)}{=} \beta(self_{VM}) = 7_{VM}$
- $I[\![x]\!](\sigma_1, \beta) \overset{(1)}{=} \beta(x) = 27$ (*)
- $I[\![\text{allInstances}_{CP}]\!](\sigma_1, \beta) \overset{(2)}{=} \text{dom}(\sigma_1) \cap \mathscr{D}(CP)$
  $= \{7_{VM}, 1_{DD}, 3_{CP}, 5_{CP}\} \cap \{1_{CP}, 2_{CP}, \dots\} = \underbrace{\{3_{CP}, 5_{CP}\}}_{(**)}$

- $I[\![x + \text{allInstances}_{CP} \to \text{size}]\!](\sigma_1, \beta) = 29$
  $(I[\![+(x, \text{size}(\text{allInstances}_{CP}))]\!](\sigma_1, \beta))$
  $\overset{(3)}{=} I(+)\big(I[\![x]\!](\sigma_1, \beta),\ I[\![\text{allInstances}_{CP} \to \text{size}]\!](\sigma_1, \beta)\big)$
  $\overset{(*)}{=} I(+)\big(27, I(\text{size})(I[\![\text{allInstances}_{CP}]\!](\sigma_1, \beta))\big)$
  $\overset{(**)}{=} I(+)\big(27, I(\text{size})(\{3_{CP}, 5_{CP}\})\big)$
  $= I(+)(27, 2) = 29$ ✓

(v) Putting It All Together...

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

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

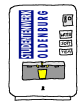
- $I[\![v(expr_1)]\!](\sigma, \beta) := \begin{cases} (\sigma(u_1))(v) & \text{, if } u_1 \in \text{dom}(\sigma) \\ \bot & \text{, otherwise} \end{cases}$

  τ_C

  Eat v.T. TeT

Example

$\beta = \{ self_{VM} \mapsto 7_{VM}, self_{CP} \mapsto 3_{CP}, y \mapsto 1_{CP} \}$



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

$\sigma_1 = \{7_{VM} \mapsto \{dd \mapsto \{1_{DD}\}, cp \mapsto \{3_{DD}, 5_{DD}\}\}, \quad 1_{DD} \mapsto \{win \mapsto 13, wen \mapsto \boxed{true}\ 1\},$
$3_{CP} \mapsto \{wen \mapsto \boxed{true}\ 1\}, \quad 5_{CP} \mapsto \{wen \mapsto \boxed{false}\ 0\}\}$

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

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

- $I[\![wen(\underbrace{self_{CP}}_{expr_1})]\!](\sigma_1, \beta) = (\sigma_1(u_1))(wen) = 1$

  $u_1 = I[\![self_{CP}]\!](\sigma, \beta) = 3_{CP}$ alive!

- $I[\![wen(y)]\!](\sigma_1, \beta) = \bot$

  $I[\![y]\!](\sigma, \beta) = 1_{CP}$ not alive!

(v) Putting It All Together...

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

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

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

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

  $r_1 : D_{0,1}$

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

  $r_2 : D_*$

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

(v) Putting It All Together...

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

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

  $:= \begin{cases} \bot_{\tau_2} & \text{, if } I[\![expr_1]\!](\sigma, \beta) = \bot_{\tau_1} \\ I[\![expr_2]\!](\sigma, \beta) & \text{, if } I[\![expr_1]\!](\sigma, \beta) = \emptyset \\ iterate(hlp, v_1, v_2, expr_3, \sigma, \beta') & \text{, otherwise} \end{cases}$

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

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

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

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

Quiz: Is (our) I a function?

Example

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

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

expr = context DD inv : wen implies $win > 0$

$I[\![\text{allInstances}_{DD} \to \text{forAll}(self_{DD} : \tau_{DD} \mid \text{implies}(wen(self_{DD}), >(win(self_{DD}), 0))]\!](\sigma, \beta)$

iterate(... ; result = true | result and ...)

= true

Another Example

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

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

$I[\![$ context VM inv : $cp \to \text{forAll}(c \mid wen(c) = true)$

$\quad$ or $cp \to \text{forAll}(c \mid wen(c) = false)]\!](\sigma, \beta) = false$

References

References

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

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

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

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.