

Software Design, Modelling and Analysis in UML

Lecture 03: Object Constraint Language

2016-10-27

Prof. Dr. Andreas Podelski, Dr. Bernd Westphal

Albert-Ludwigs-Universität Freiburg, Germany

- 03 - 2016-10-27 - main -

Content

- The **Object Constraint Language (OCL)**:
 - Syntax**
 - Running Example
 - Overview
 - Expressions
 - Notational Conventions
("." (OCL-Dot) and "->" (OCL-Arrow))
 - Constants & Arithmetics
 - Iterate
 - Context
 - More Notational Conventions
 - The Running Example Revisited
 - **"Not Interesting"**

- 03 - 2016-10-27 - content -

(Core) OCL Syntax OMG (2006)

-03-2016-10-27-main-

3/24

Overview

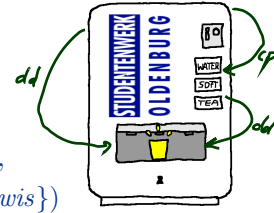
$expr ::=$	w	$: \tau(w)$
	$expr_1 =_{\tau} expr_2$	$: \tau \times \tau \rightarrow Bool$
	$oclIsUndefined_{\tau}(expr_1)$	$: \tau \rightarrow Bool$
	$\{expr_1, \dots, expr_n\}$	$: \tau \times \dots \times \tau \rightarrow Set(\tau)$
	$size(expr_1)$	$: Set(\tau) \rightarrow Int$
	$allInstances_C$	$: Set(\tau_C)$
	$v(expr_1)$	$: \tau_C \rightarrow \tau(v)$
	$r_1(expr_1)$	$: \tau_C \rightarrow \tau_D$
	$r_2(expr_1)$	$: \tau_C \rightarrow Set(\tau_D)$
	$true, false$	$: Bool$
	$not\ expr_1$	$: Bool \rightarrow Bool$
	$expr_1 \{and, or, implies\} expr_2$	$: Bool \times Bool \rightarrow Bool$
	\dots	
	$OclUndefined_{\tau}$	$: \tau$
	$expr_1 \rightarrow iterate(w_1 : T_1 ; w_2 : T_2 = expr_2 \mid expr_3)$	$: Set(\tau_0) \rightarrow \tau_{T_2}$
$context ::=$	$context\ w_1 : T_1, \dots, w_n : T_n\ inv : expr$	$: Bool$

-03-2016-10-27-SocSyn-

4/24

Recall: Vending Machine Structure

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



Claim: this is a proper OCL constraint over \mathcal{S} :

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

string $\tau \rightarrow Int$ $\tau \times \tau \rightarrow Bool$

$T ::= \emptyset \mid \Delta(T) \mid \nabla(T_1, T_2)$! τ

$\Delta(\emptyset) : Bool \checkmark$ $\nabla(\emptyset, \Delta(\emptyset)) \times$ (not well-typed)

$\nabla(\emptyset, \emptyset) : Bool \checkmark$ string

...

-03-2016-10-27-Scalyn-

5/24

Plan

$expr ::=$	w	$: \tau(w)$
1/4	$ expr_1 =_{\tau} expr_2$	$: \tau \times \tau \rightarrow Bool$
	$ oclIsUndefined_{\tau}(expr_1)$	$: \tau \rightarrow Bool$
	$ \{expr_1, \dots, expr_n\}$	$: \tau \times \dots \times \tau \rightarrow Set(\tau)$
	$ size(expr_1)$	$: Set(\tau) \rightarrow Int$
	$ allInstances_C$	$: Set(\tau_C)$
	$ v(expr_1)$	$: \tau_C \rightarrow \tau(v)$
2/4	$ r_1(expr_1)$	$: \tau_C \rightarrow \tau_D$
	$ r_2(expr_1)$	$: \tau_C \rightarrow Set(\tau_D)$
	$ true, false$	$: Bool$
	$ not expr_1$	$: Bool \rightarrow Bool$
3/4	$ expr_1 \{and, or, implies\} expr_2$	$: Bool \times Bool \rightarrow Bool$
	$ \dots$	
	$ OclUndefined_{\tau}$	$: \tau$
4/4	$\{ expr_1 \rightarrow iterate(w_1 : T_1 ; w_2 : T_2 = expr_2 \mid expr_3)$	$: Set(\tau_0) \rightarrow \tau_{T_2}$
$context ::=$	$context w_1 : T_1, \dots, w_n : T_n \text{ inv} : expr$	$: Bool$

-03-2016-10-27-Scalyn-

6/24

OCL Syntax 1/4: Expressions

expr ::=

w	$: \tau(w)$
$ \text{expr}_1 =_{\tau} \text{expr}_2$	$: \tau \times \tau \rightarrow Bool$
$ \text{ocllsUndefined}_{\tau}(\text{expr}_1)$	$: \tau \rightarrow Bool$
$ \{ \text{expr}_1, \dots, \text{expr}_n \}$	$: \tau \times \dots \times \tau \rightarrow Set(\tau)$
$ \text{isEmpty}(\text{expr}_1)$	$: Set(\tau) \rightarrow Bool$
$ \text{size}(\text{expr}_1)$	$: Set(\tau) \rightarrow Int$
$ \text{allInstances}_C$	$: Set(\tau_C)$

$ v(\text{expr}_1)$	$: \tau_C \rightarrow \tau$	where $v : \tau \in \text{atr}(C), \tau \in \mathcal{T}$,
$ r_1(\text{expr}_1)$	$: \tau_C \rightarrow \tau_D$	where $r_1 : D_{0,1} \in \text{atr}(C), C, D \in \mathcal{C}$,
$ r_2(\text{expr}_1)$	$: \tau_C \rightarrow Set(\tau_D)$	where $r_2 : D_* \in \text{atr}(C), C, D \in \mathcal{C}$.

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

- $w \in W \supseteq \{ \text{self}_C : \tau_C \mid C \in \mathcal{C} \}$ is a set of typed logical variables, w has type $\tau(w)$
- τ is any type from $\mathcal{T} \cup T_B \cup T_{\mathcal{C}} \cup \{ Set(\tau_0) \mid \tau_0 \in \mathcal{T} \cup T_B \cup T_{\mathcal{C}} \}$
- 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- τ_0 type for $\tau_0 \in T_B \cup T_{\mathcal{C}}$ (sufficient because of "flattening" (cf. standard)).

-03-2016-10-27-SetExpr-

Expression Examples

expr ::=		
w	$: \tau(w)$	$ \text{size}(\text{expr}_1) : Set(\tau) \rightarrow Int$
$ \text{expr}_1 =_{\tau} \text{expr}_2$	$: \tau \times \tau \rightarrow Bool$	$ \text{allInstances}_C : Set(\tau_C)$
$ \text{ocllsUndefined}_{\tau}(\text{expr}_1)$	$: \tau \rightarrow Bool$	$ v(\text{expr}_1) : \tau_C \rightarrow \tau(v)$
$ \{ \text{expr}_1, \dots, \text{expr}_n \}$	$: \tau \times \dots \times \tau \rightarrow Set(\tau)$	$ r_1(\text{expr}_1) : \tau_C \rightarrow \tau_D$
$ \text{isEmpty}(\text{expr}_1)$	$: Set(\tau) \rightarrow Bool$	$ r_2(\text{expr}_1) : \tau_C \rightarrow Set(\tau_D)$

$$\mathcal{S}_0 = (\{ Int \}, \{ \bar{C}, \bar{D} \}, \{ x : Int, p : \bar{C}_{0,1}, n : \bar{C}_* \}, \{ \bar{C} \mapsto \{ p, n \}, \bar{D} \mapsto \{ x \} \})$$

- $\bullet \text{self}_{\bar{D}} : \tau_{\bar{D}} \checkmark$
- $\bullet x(\text{self}_{\bar{D}}) : Int \checkmark$
- $\bullet p(\text{self}_{\bar{D}}) \times p \in \text{atr}(\bar{D})$
- $\bullet p(\text{self}_{\bar{C}}) : \tau_{\bar{C}} \rightarrow \tau_{\bar{C}} \checkmark$
- $\bullet n(\text{self}_{\bar{C}}) : \tau_{\bar{C}} \rightarrow Set(\tau_{\bar{C}}) \checkmark$
- $\bullet n(p(\text{self}_{\bar{C}})) : \tau_{\bar{C}} \rightarrow Set(\tau_{\bar{C}}) \checkmark$
 $\quad \quad \quad \tau_{\bar{C}} \quad \quad \quad [\text{self}_{\bar{C}} . p . n]$
- $\bullet p(n(\text{self}_{\bar{C}})) \downarrow$
 $\quad \quad \quad \tau_{\bar{C}} \rightarrow Set(\tau_{\bar{C}})$
 $\quad \quad \quad [\text{self}_{\bar{C}} . n . p]$
- $\bullet \text{size}(n(\text{self}_{\bar{C}})) : Set(\tau_{\bar{C}}) \rightarrow Int$

-03-2016-10-27-SetExpr-

Expression Examples

$expr ::=$			
w	$: \tau(w)$	$ size(expr_1)$	$: Set(\tau) \rightarrow Int$
$ expr_1 =_{\tau} expr_2$	$: \tau \times \tau \rightarrow Bool$	$ allInstances_C$	$: Set(\tau_C)$
$ oclIsUndefined_{\tau}(expr_1)$	$: \tau \rightarrow Bool$	$ v(expr_1)$	$: \tau_C \rightarrow \tau(v)$
$ \{expr_1, \dots, expr_n\}$	$: \tau \times \dots \times \tau \rightarrow Set(\tau)$	$ r_1(expr_1)$	$: \tau_C \rightarrow \tau_D$
$ isEmpty(expr_1)$	$: Set(\tau) \rightarrow Bool$	$ r_2(expr_1)$	$: \tau_C \rightarrow Set(\tau_D)$

$$\mathcal{S}_0 = (\{Int\}, \{C, D\}, \{x : Int, p : C_{0,1}, n : C_*\}, \{C \mapsto \{p, n\}, D \mapsto \{x\}\})$$

-03-2016-10-27-SetSym-

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

Notational Conventions for Expressions

- Each expression

$$\omega(\overbrace{expr_1, expr_2, \dots, expr_n}^{\forall}) : \tau_1 \times \dots \times \tau_n \rightarrow \tau$$

may alternatively be written ("abbreviated as")

- $expr_1 . \omega(expr_2, \dots, expr_n)$ if τ_1 is an **object type**, i.e. if $\tau_1 \in T_{\mathcal{O}}$.
- $expr_1 \rightarrow \omega(expr_2, \dots, expr_n)$ if τ_1 is a **collection type** (here: only sets), i.e. if $\tau_1 = Set(\tau_0)$ for some $\tau_0 \in T_B \cup T_{\mathcal{O}}$.

- Examples:** $(\{Int\}, \{C, D\}, \{x : Int, p : C_{0,1}, n : C_*\}, \{C \mapsto \{p, n\}, D \mapsto \{x\}\})$

- $self_C . p \rightsquigarrow p(self_C)$
- $self_C . p . n \rightsquigarrow n(p(self_C))$
- $self_C . p . n \rightarrow isEmpty \rightsquigarrow isEmpty(n(p(self_C)))$
- context CP inv : wen implies $dd.wis > 0$ ($atr(CP) = \{wen : Bool, dd : DD_{0,1}\}$)

-03-2016-10-27-SetSym-

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 | -2 | 2 | ...                   : Int
| expr1 {+, -, ...} expr2                 : Int × Int → Int
| expr1 {<, ≤, ...} expr2                 : Int × Int → Bool
| OclUndefinedτ                             : τ
  
```

Generalised notation: (prefix normal form)

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

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

Handwritten diagram:
 $1 + 2 \rightsquigarrow + (1, 2)$
 Below the plus sign is ω . Below 1 is $expr_1$. Below 2 is $expr_2$.

- 03 - 2016-10-27 - Selskyr -

Constants & Arithmetics Examples

```

expr ::= ...
| true, false                               : Bool
| expr1 {and, or, implies} expr2         : Bool × Bool → Bool
| not expr1                                 : Bool → Bool
| 0, -1, 1, -2, 2, ...                       : Int
| expr1 {+, -, ...} expr2                 : Int × Int → Int
| expr1 {<, ≤, ...} expr2                 : Int × Int → Bool
| OclUndefinedτ                             : τ
  
```

$$\mathcal{S}_0 = (\{Int\}, \{C, D\}, \{x : Int, p : C_{0,1}, n : C^*\}, \{C \mapsto \{p, n\}, D \mapsto \{x\}\})$$

Handwritten diagram:
 A dashed box contains the expression $implies(wen, >(wis(dd), 0))$.
 Inside the box, $wis(dd)$ is underlined and labeled $expr_1$.
 0 is underlined and labeled $expr_2$.
 An arrow points from $wis(dd)$ down to the boxed context below.

context CP inv : wen implies dd.wis > 0

- 03 - 2016-10-27 - Selskyr -

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,

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

where

- $expr_1$ is of a **collection type** (here: a set $Set(\tau_0)$ for some τ_0),
- $ite\!r \in W$ is called **iterator**, of the type denoted by T_1
(if T_1 is omitted, τ_0 is assumed as type of $ite\!r$)
- $result \in W$ 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 $result$,
($OclUndefined_{\tau_2}$, if omitted)
- $expr_3$ is an expression of type τ_2 ,
in particular $ite\!r$ and $result$ may appear in $expr_3$.

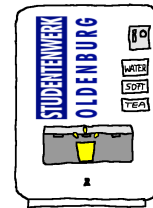
-03-2016-10-27 - Selskyr -

12/24

Iterate Example

$$\mathcal{S} = (\{Bool, \overset{Nat}{\mathbb{N}}, \{VM, CP, DD\},$$

$$\{cp : CP^*, dd : DD_{0,1}, wen : Bool, wis : Nat\},$$

$$\{VM \mapsto \{cp, dd\}, CP \mapsto \{wen, dd\}, DD \mapsto \{wis\}\})$$


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

- $cp(\text{self}_{VM}) \rightarrow iterate(\underbrace{ite\!r}_{: T_{cp}} ; res : \text{Int} = 0 \mid res + \underbrace{ite\!r.dd.wis}_{: Nat \in T_{\mathbb{N}}})$
- $cp(\text{self}_{VM}) \rightarrow iterate(ite\!r ; res : Bool = true \mid res \text{ and } itv.wen)$

-03-2016-10-27 - Selskyr -

13/24

Abbreviations on Top of Iterate

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

- $expr_1 \rightarrow forAll(w_1 : T_1 \mid expr_3)$ (" $\forall w_1 \in expr_1 \bullet expr_3$ ")

is an abbreviation for

$$expr_1 \rightarrow iterate(w_1 : T_1; w_2 : Bool = true \mid w_2 \text{ and } expr_3).$$

- $expr_1 \rightarrow Exists(w : T_1 \mid expr_3)$

is an abbreviation for

$$expr_1 \rightarrow iterate(w : T_1; w_2 : Bool = false \mid w_2 \text{ or } expr_3)$$

To ensure confusion, we may again omit all kinds of things, cf. [OMG \(2006\)](#).

-03-2016-10-27-56k5yn-

14/24

Recall: Overview

$expr ::=$	w	$: \tau(w)$
	$ expr_1 =_{\tau} expr_2$	$: \tau \times \tau \rightarrow Bool$
	$ oclIsUndefined_{\tau}(expr_1)$	$: \tau \rightarrow Bool$
	$ \{expr_1, \dots, expr_n\}$	$: \tau \times \dots \times \tau \rightarrow Set(\tau)$
	$ size(expr_1)$	$: Set(\tau) \rightarrow Int$
	$ \underline{allInstances}_C$	$: Set(\tau_C)$
	$ v(expr_1)$	$: \tau_C \rightarrow \tau(v)$
	$ r_1(expr_1)$	$: \tau_C \rightarrow \tau_D$
	$ r_2(expr_1)$	$: \tau_C \rightarrow Set(\tau_D)$
	$ true, false$	$: Bool$
	$ not expr_1$	$: Bool \rightarrow Bool$
	$ expr_1 \{and, or, implies\} expr_2$	$: Bool \times Bool \rightarrow Bool$
	$ \dots$	
	$ OclUndefined_{\tau}$	$: \tau$
	$ expr_1 \rightarrow iterate(w_1 : T_1; w_2 : T_2 = expr_2 \mid expr_3)$	$: Set(\tau_0) \rightarrow \tau_{T_2}$
$context ::=$	$context w_1 : T_1, \dots, w_n : T_n \text{ inv } : expr$	$: Bool$

-03-2016-10-27-56k5yn-

15/24

More Iterate Examples

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



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

$\text{allInstances}_{CP} \rightarrow \text{iterate}(\text{self}_{CP} : CP; \text{res} : Bool = true \mid$
 $\text{res and } (\text{self}_{CP}.wen \text{ implies } \text{self}_{CP}.dd.wis > 0))$
 or:
 $\text{allInstances}_{CP} \rightarrow \text{forall}(\text{self}_{CP} \mid \text{self}_{CP}.wen \text{ implies } \text{self}_{CP}.dd.wis > 0)$

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

- 03 - 2016-10-27 - Selsyn -

16/24

OCL Syntax 4/4: Context

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

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

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:

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

is (just) an **abbreviation** for

$\text{allInstances}_{C_1} \rightarrow \text{forall}(w_1 : \#C_1 \mid$
 \dots
 $\text{allInstances}_{C_n} \rightarrow \text{forall}(w_n : \#C_n \mid$
 $expr$
 $)$
 \dots
 $)$

- 03 - 2016-10-27 - Selsyn -

17/24

Context: More Notational Conventions

- For

context $self : T \text{ inv} : expr$

we may **alternatively** write (“**abbreviate as**”)

context $T \text{ inv} : expr$

- **Within** the latter abbreviation, we may omit the “ $self$ ” in expression $expr$, i.e. for

context $T \text{ inv} : self.v$

(which is an abbreviation for context $T \text{ inv} : v(self)$)

we may alternatively write (“**abbreviate as**”)

context $T \text{ inv} : v$

- 03 - 2016-10-27 - ScalaSyn -

18/24

The Running Example

context $CP \text{ inv} : wem \text{ implies } dd.wis > 0$

\Downarrow
context $self : CP \text{ inv} : wem \text{ implies } dd.wis > 0$

\Downarrow
context $self : CP \text{ inv} : self.wem \text{ implies } self.dd.wis > 0$

\Downarrow
allInstances_{CP} \rightarrow forAll (self | self.wem implies self.dd.wis > 0)

\Downarrow
- 1 - iterate (....)

- 03 - 2016-10-27 - ScalaSyn -

19/24

Recall: Overview

$expr ::=$	w	$: \tau(w)$
	$expr_1 =_{\tau} expr_2$	$: \tau \times \tau \rightarrow Bool$
	$oclIsUndefined_{\tau}(expr_1)$	$: \tau \rightarrow Bool$
	$\{expr_1, \dots, expr_n\}$	$: \tau \times \dots \times \tau \rightarrow Set(\tau)$
	$size(expr_1)$	$: Set(\tau) \rightarrow Int$
	$allInstances_C$	$: Set(\tau_C)$
	$v(expr_1)$	$: \tau_C \rightarrow \tau(v)$
	$r_1(expr_1)$	$: \tau_C \rightarrow \tau_D$
	$r_2(expr_1)$	$: \tau_C \rightarrow Set(\tau_D)$
	$true, false$	$: Bool$
	$not\ expr_1$	$: Bool \rightarrow Bool$
	$expr_1 \{and, or, implies\} expr_2$	$: Bool \times Bool \rightarrow Bool$
	...	
	$OclUndefined_{\tau}$	$: \tau$
	$expr_1 \rightarrow iterate(w_1 : T_1 ; w_2 : T_2 = expr_2 \mid expr_3)$	$: Set(\tau_0) \rightarrow \tau_{T_2}$
$context ::=$	$context\ w_1 : T_1, \dots, w_n : T_n\ inv : expr$	$: Bool$

-03-2016-10-27-565kyn-

20/24

“Not Interesting”

Among others:

- Enumeration types
- Type hierarchy
- Complete list of arithmetical operators
- The two other collection types Bag and Sequence
- Casting
- Runtime type information
- Pre/post conditions
(maybe later, when we officially know what an operation is)

• ... context f pre: expr₁
post: expr₂

-03-2016-10-27-565kyn-

22/24

References

-03-2016-10-27-main-

23/24

References

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

OMG (2011a). Unified modeling language: Infrastructure, version 2.4.1. Technical Report formal/2011-08-05.

OMG (2011b). Unified modeling language: Superstructure, version 2.4.1. Technical Report formal/2011-08-06.

Warmer, J. and Kleppe, A. (1999). *The Object Constraint Language*. Addison-Wesley.

-03-2016-10-27-main-

24/24