Foundations of the UML

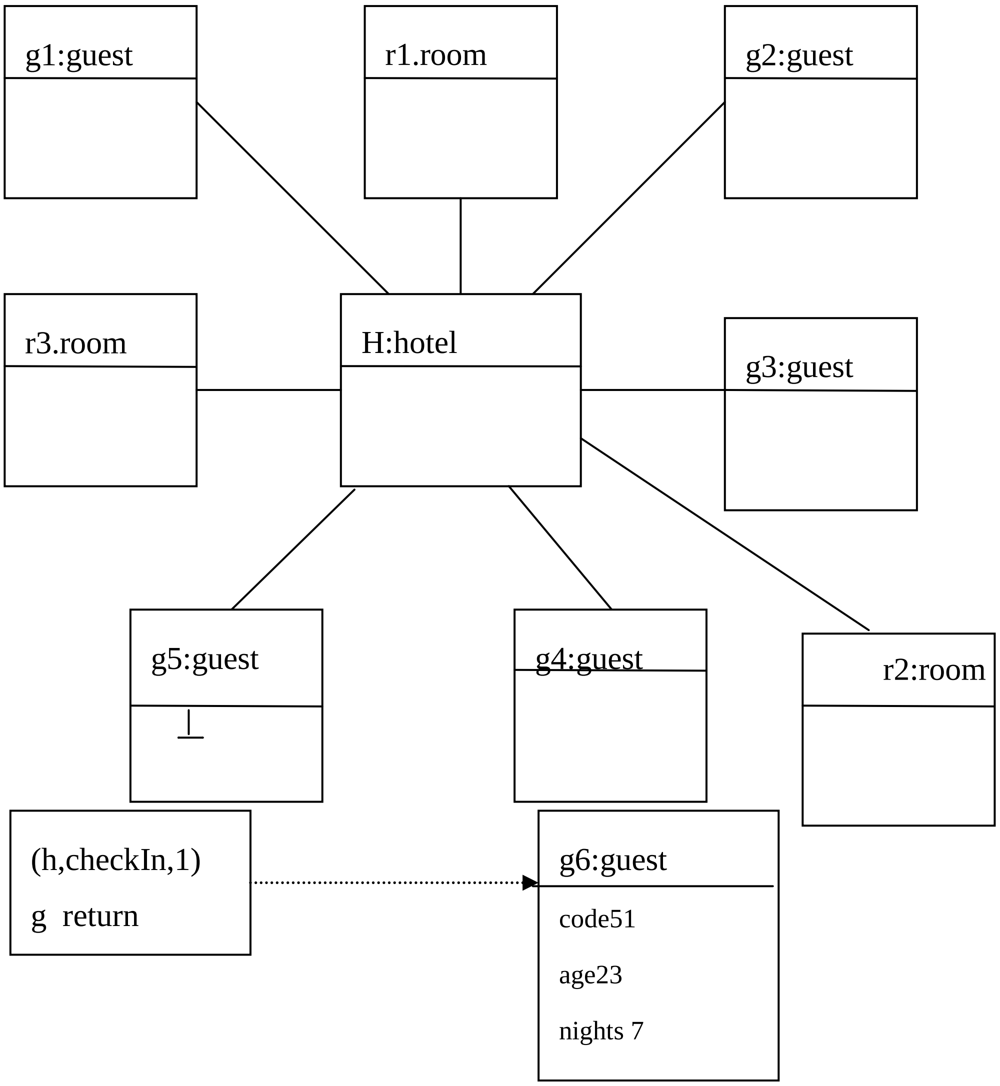
Winter Term 07/08

– Lecture number 8 –

(Date (28th Jan 2008))

summarized by *Ratna Widyastuti(277010)* and *Teena Mary Mihan(284125)*

1.1 Configuration example



1.2Example

Set of active objects in the above eg are

O = { h, r_1, r_2, r_3, g_1, g_2, g_3, g_4, g_5, g_6 }

Set of active events can be written as:

E = { h, check In, 1 }

where

$h = (\text{Hotel}, 1)$

$g_1 = (\text{Guest}, i)$

$r_1 = (\text{Room}, i)$

In the above case, local set of object g_6 is:

$\sigma (g_6) (\text{code}) = 51$

$\sigma (g_6) (\text{age}) = 23$

$\sigma (g_6) (\text{Nr. nights}) = 7$

State of event (h, Check In, 1):

γ (h, Check In, 1) = (l, $\perp$) with l(g) = g_6

1.2 Static expressions

$\xi ::= X \mid \xi . a \mid \xi . \text{owner} \mid \xi . \text{return} \mid \xi . \text{new} \mid \xi . \text{alive} \mid \omega . (\xi , \ldots\ldots, \xi)$

| <u>with</u> X_1 € ξ <u>from</u> $X_2 := \xi$ <u>do</u> $X_2 := \xi$

X = logical variable

ξ . a = Parameter / attribute navigation

ξ . owner = Object executing method ξ

ξ . result = Return value of method ξ

ξ . new = The object (or method) ξ is "fresh" in the current state

An **object is usually new just after its creation and the method is new when it is just invoked.**

ξ . alive = Object (method) ξ is currently alive.

An object here becomes alive when it is created and remains alive until it is deallocated (eg. By garbage collection).

<u>with</u> = Like the OCL **iterate** expression

1.3 Quantification

Temporal expression Э X € Γ : Φ expresses that Φ holds for at least one <u>alive</u> instance X of type Γ.

Case 1: If **Γ = void, not, or bool** an instance is always **alive** (as they are **<u>static</u>**).

Case 2: If **Γ = C.ref or Γ = C.M ref**, however instances are **<u>dynamic</u>.**

Object X € C ref is <u>alive</u> if it has been created and not yet deallocated;

Method X € C.M ref is <u>alive</u> if occurred

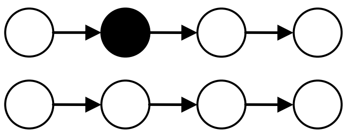
M has been invocated and has not yet terminated.

2. Temporal expressions

$\Phi ::= \xi \mid \neg \Phi \mid \Phi \vee \Phi \mid \exists X \in \Gamma : \Phi \mid O\,\Phi \mid \Phi\, U\, \Phi \mid$

O Φ = <u>next</u> state Φ holds

Φ U Φ = Φ state can be reached via a path solely consisting of Φ states



satisfies

violates

Derived operators:

$\Diamond \Phi \equiv \text{true}\ U\ \Phi$ - - - - - - -► eventually

$\Box \Phi \equiv \neg (\Diamond \neg \Phi)$ - - - - - -► always

Example:

Along the computation of hotel h, eventually at least one guest will check in:

◊ (Э m € h. Check In ref: m alive)

2.1. Semantics static expressions

Let $\Theta : LVAR \longrightarrow VAL$ assign values to logical variables.

i.e. for X € LVAR ∩ dom (Θ) , Θ (X) is the value of X.

Semantics of static expression ξ is given by

$\mathbf{[|\xi|]_{g,N,\Theta} \in VAL_{\perp}}$

where:

$g = (O_g, E_g, \sigma_g, t_g)$ is a configuration

$N \in O_g \cup E_g$, denotes new objects and events in configuration g

Θ = Valuation of logical variables (in ξ)

2.2. Semantics

$[|X|]_{g,N,\Theta} = \Theta(X)$; logical valuation of X.

$[|\xi \,.\, owner|]_{g,N,\Theta} = 0$; where $[|\xi|]_{g,N,\Theta} = (O, M, J)$(object O invoked M)

$[|\xi \,.\, return|]_{g,N,\Theta} = v$; where $t_g\ ([|\xi|]_{g,N,\Theta} = (\rho, v)$(evaluation of ξ in config $_{g)}$

$[|\xi\ new|]_{g,N,\Theta} = ([|\xi|]_{g,N,\Theta}$ € N)(refers to a new object or event)

$[|\xi\ alive|]_{g,N,\Theta} = [|\xi|]_{g,N,\Theta}$ € O_g U E_g

$[|\ \omega\ (\xi\ , \ldots\ldots, \xi_n)\ |]_{g,N,\Theta} = [|\omega|]\ ([|\omega|]_{\ g,N,\Theta}, \ldots\ldots, [|\xi_n\ |]_{g,N,\Theta}$

For the semantics of ξ . we distinguish two cases:

[|ξ|] is a reference or a list (of refs)

$[|\xi \,.\, a|]_{g,N,\Theta} = l\ (a)$

where:

$[|\xi|]_{g,N,\Theta}$ € c ref and $\sigma_g\ ([|\xi|]_{g,N,\Theta}) = l$

or [|ξ|]…..€ C.M ref and $\gamma_g\ (([|\xi|]_{\ldots}) = (l,v)$

For lists:

$[|\xi \,.\, a|]_{g,N,\Theta} = \overrightarrow{l}(a)$ where

or [|ξ|]…..€ C ref list and $\sigma_g\ ([|\xi|]) = \overrightarrow{l}$

or [|ξ|]…..€ C.M ref list $\gamma_g\ ([|\xi|]) = (\overrightarrow{l}, v)$

[|<u>with</u> X_1 € ξ_1 <u>from</u> X_2 € ξ_2 do $X_2 := \xi_3|]_{\ g,N,\Theta}$ = [|<u>for</u> X_1 € $[|\xi_1|]_{\ g,N,\Theta}$ do $X_2 := \xi_3|]_{\ g,N,\Theta'}$

where:

$\Theta' = \Theta\ [X2 := [|\xi_2|]_{\ g,N,\Theta}]$

3. Temporal Expressions

Temporal expressions such as Oϕ and ϕ ∪ φ are interpreted over paths in the automaton (Conf, →, I), whereas:

$\pi = c_0\ c_1\ c_2 \ldots.$ is a path
if c_i ∈Conf and $c_i = c_{i+1,}$ for all $i \geq 0$
Notation $\pi[i] = c_i$

For $\pi = c_0\ c_1\ c_2 \ldots.$ let:

- $N_0 = N \subseteq O_0 \cup E_0$
  O_0 are objects in configuration C_0
  E_0 are events in configuration C_0
- $N_{i+1} = (O_{i+1}\ |\ O_i) \cup (E_{i+1}\ |\ E_i)$
  $(O_{i+1}\ |\ O_i)$ are objects created in $C_i \rightarrow C_{i+1}$
  $(E_{i+1}\ |\ E_i)$ are events generated in $C_i \rightarrow C_{i+1}$
- $\theta_i(x) = \begin{cases} \theta(x) if\ \forall k \leq i :\ \theta(x) \epsilon \theta_k\ \cup\ E_k \\ \quad undefined\ otherwise \end{cases}$

The semantics of ϕ is given by a relation $\vDash$(satisfaction relation, a $\vDash$b means a satisfies b).

(π, N, θ, ϕ) $\in \vDash$, should be read as: for path π, initial set N of new objects/methods, and logical valuation θ, formula ϕ holds. Write $\pi, N, \theta, \phi \vDash \phi$ instead of $(\pi, N, \theta, \phi) \in \vDash$.

By structural induction over ϕ:

- $\pi, N, \theta \vDash \xi$ iff $[\![\xi]\!]_{\pi[o],N,\theta} = tt$
- $\pi, N, \theta \vDash \neg\theta$ iff $\pi, N, \theta \nvDash \phi$
- $\pi, N, \theta \vDash \phi \cup \varphi$ iff $\pi, N, \theta \vDash \phi$ or $\pi, N, \theta \vDash \varphi$
- $\pi, N, \theta \vDash O\phi$ iff $\pi^1, N_1, \theta_1 \vDash \phi$, where $\pi^1 = \pi[1], \pi[2], \ldots$
- $\pi, N, \theta \vDash \phi \cup \psi$ iff $\exists j \geqq 0$: $\pi^{j,} N_j, \theta_j \vDash \psi$ and $\forall k < j$:$\pi^k, N_k, \theta_k \vDash \phi$ where $\pi^k = \pi[k]\ \pi[k+1]\ldots$
- $\pi, N, \theta \vDash \exists x \epsilon \tau.\phi$ iff $\exists v \in VAL^\tau \restriction (O_0, E_0)$: $\pi, N, \theta[x:=v] \vDash \phi$ where $VAL^\tau \restriction (O, E)$ is the subset of VAL^τ alive in (O,E), formally:

$$VAL^\tau \restriction (O, E) = \begin{cases} VAL^\tau \cap O & if\ \tau = C\ ref \\ VAL^\tau \cap O & if\ \tau = C.M\ ref \\ VAL^\tau \cap O & otherwise \end{cases}$$

4. Translating OCL Types

OCL allows sets, bags, and lists, but no nested lists. OCL types are defined by:

ρ ::= not | bool | C ref (means ρ could be not, bool or/and C ref)

τ ::= ρ | ρ list | ρ set | ρ bag

Universe of values:

$VAL^{nat} = \mathbb{N}$

$VAL^{bool} = \{tt, ff\}$

$VAL^{C\ ref} = \{null\} \cup OD^C$

$VAL^{\rho\ list} = \{[\ ]\} \cup \{h :: w |\ h \in VAL^\rho, w \in VAL^{\rho\ list}\}$

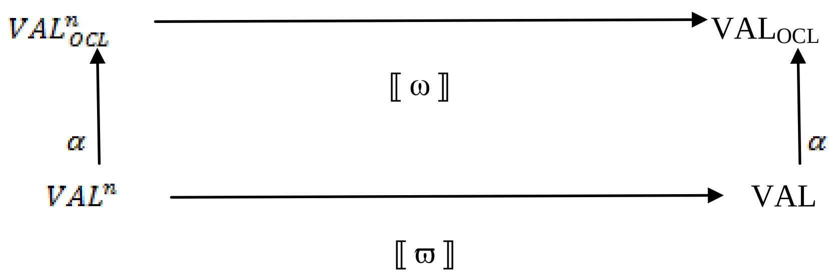
$VAL^{\rho\ set} = 2^{VAL^\rho}$

$VAL^{\rho\ bag} = VAL^\rho \rightarrow \mathbb{N}$

The set of value in OCL: $VAL_{OCL} = \bigcup_\tau VAL^\tau$

bag is type of multi set consists of the set itself and the occurrence frequency of the set, example:{|1,2,2,3|}, in this set 1 appears 1 times, 2 appears 2 times, 3 appears 1 time.

For each operation $\xi_1 \rightarrow \omega(\xi_{1,\ldots,}\ \xi_n)$ in OCL on sets or bags, there exists a corresponding operation $\varpi\ (\xi_{1,\ldots,}\ \xi_n)$ such that the following diagram commutes:



where α is an abstraction function.

For sets: $\alpha_{set}(v) = \begin{cases} \emptyset & if\ v = [] \\ \{h\} \cup \alpha_{SET(w)} & if\ v = h :: w \\ v & otherwise \end{cases}$

For bags: $\alpha_{bag} : VAL \rightarrow VAL_{OCL}$ defined by

$$\alpha_{bag}(v) = \begin{cases} \{|\ |\} & if\ v = [] \\ \{|h|\} \cup \alpha_{bag}(w) & if\ v = h :: w \\ v & otherwise \end{cases}$$

Example:

Consider the OCL expression $\xi_1 \rightarrow$ union(ξ_2). The corresponding operator $\overline{union}$ has semantics $[\![\overline{union}]\!] : VAL^{\tau\ list} \times VAL^{\tau\ list} \rightarrow VAL^{\tau\ list}$, i.e. $[\![\overline{union}]\!]$ is the concatenation of two $VAL^{\tau\ list}$

According to the commutativity diagram, we have:

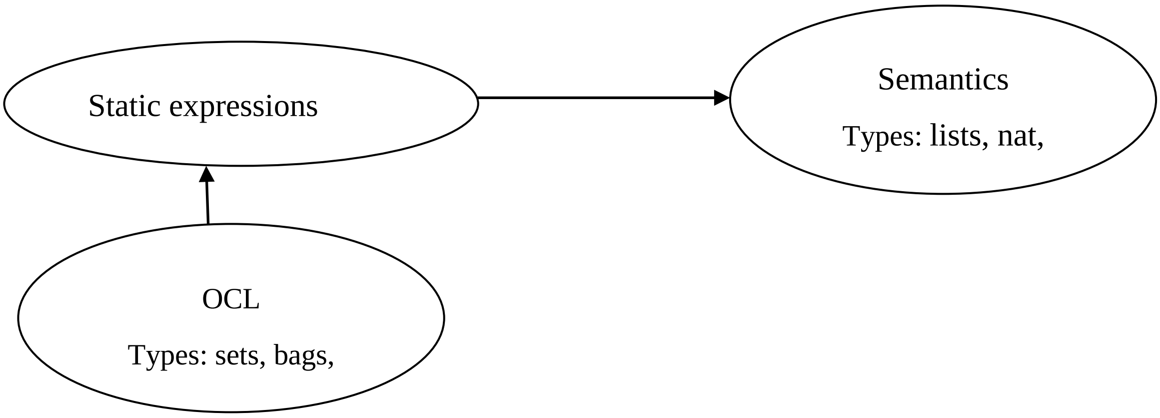
$$\alpha_{set}\underbrace{([\![\overline{union}(v1, v2)]\!])}_{\text{union of lists}} = \underbrace{[\![union]\!]}_{\text{union of sets}}\ (\alpha_{set}(v_1), \alpha_{set}(v_2)),$$

e.g. $\overline{union}(w_1, w_2) = w_1 \# w_2$ where # is list concatenation.

- OCL equality on sets: $\xi_1 = \xi_2$
  $[\![\ \equiv\overline{set}\ ]\!] : VAL^{\tau\ list} \times VAL^{\tau\ list} \rightarrow VAL^{bool}$

Where $\equiv\overline{set}$ on lists is defined as follows:

$$\equiv\overline{set}(w_1,w_2) = \text{Eqlist}\ \underbrace{(sort(del\_duplicates(w_1)), sort(del\_duplicates(w_2)))}_{\text{delete duplicate value in W and sort}}$$



If we want to transform *bags* (in *OCL*) to *nat* (in *semantics*), then do *type translation*.

5. Translating OCL Expressions

$\delta_{o,m,\vec{p}}$ (ξ) is the translation of expression ξ wrt. object *o*, method occurrence m with formal parameters $\vec{p}$.

- δ(self) = 0
- $\delta(x) = \begin{cases} o.x & if\ o \in C\ ref\ \wedge x \in dom(C.attr) \\ m.x & if\ x \in \vec{p} \\ x & otherwise \end{cases}$
- δ(result) = m.return

- $\delta(\underbrace{\xi@ipre}_{\text{the i-th occurrence of @pre}}) = u_i$
- $\delta(\omega(\xi_1,\ldots,\xi_n)) = \varpi(\delta(\xi_1),\ldots,\delta(\xi_n))$
- $\delta(\xi.\omega(\xi_1,\ldots,\xi_n)) = \varpi(\delta(\xi),\delta(\xi_1),\ldots,\delta(\xi_n))$
- $\delta(\xi\rightarrow\omega(\xi_1,\ldots,\xi_n))$ = ditto
- $\delta(\xi\rightarrow$iterate $(x_1; x_2= \xi_2 \mid \xi_3)) =$ with $x_1 \in \delta(\xi_1)$ from $x_2 = \delta(\xi_2)$ do $= x_2 = \delta(\xi_3)$

6. Translating OCL Invariants

Context C inv ξ : the condition ξ must hold in any state where no method in dom(C.meths) is active. During the execution of such method, some configuration may even violate ξ. Let y ∈ LVAR and dom(C.meths) = $\{m_1,\ldots,m_k\}$.Then:

δ(context C inv ξ) = □(∀x∈C ref:

$$(\neg\exists m_1 \in x.M_1 \text{ ref} \wedge \ldots \wedge \neg\exists m_k \in x.M_k \text{ ref}) \text{ implies } \underbrace{\delta x, y, [](\xi)}_{\text{translation of } \xi})$$

Example:
Context Hotel
Inv rooms.guests = guests

The OCL translation of the example above:
□(∀x ∈ Hotel ref:
(¬∃m ∈ x.checkIn ref ∧ ¬∃m` ∈ x.checkOut) implies Eqlist(δ(rooms.guests) = guests)
= □(∀x ∈ Hotel ref:
(¬∃m ∈ x.checkIn ref ∧ ¬∃m` ∈ x.checkOut) implies Eqlist(sort(δ(rooms.guests)), sort(δ(guests)))
= □(∀x ∈ Hotel ref:
(¬∃m ∈ x.checkIn ref ∧ ¬∃m` ∈ x.checkOut) implies
Eqlist(sort(flat(x.rooms.guests)), sort(x.guests))

7. Translating Pre- and Post Condition

Main complication: for each ξ@pre expression in the post condition, we should "remember" its value on evaluating the precondition. This is done using auxiliary variables: for each $\xi@_i$ pre use auxiliary variable u_i.

Extended precondition = $\underbrace{precondition}_{\xi_{pre}} + \underbrace{auxiliary\ variables}_{\{u_1,\ldots,u_n\}}$

Formally: $\xi_{pre}^{ext} = \delta(\xi_{pre}) \wedge \bigwedge_{\xi@_i pre \text{ in } \xi_{post}} u_i = \delta(\xi)$, where $u_i = \delta(\xi)$ means "freeze" value of ξ in u_i

Then: Δ(context C: : $M(\vec{p})$ pre ξ_{pre} post ξ_{post})

$= \forall u_1 \in \tau_1, \ldots, u_n \in \tau_n : \forall z \in C \text{ ref} : \forall m \in z.M \text{ ref}:$

$\square(m \text{ new} \wedge \xi_{pre}^{ext} \text{ implies } m \text{ alive } \underbrace{U}_{until} (\text{term}(m) \wedge \delta(\xi_{post})))$

Example:

<u>context</u> Hotel : checkIn (g:Guest)

<u>pre</u> not guests→includes(g)

<u>post</u> guests→size=guests@pre→size+1 and guests→includes(g)

Let z,m ∈ LVAR, z = object class Hotel

m = occurrence of method checkIn

▵(<u>context</u> Hotel …. <u>pre</u> …. <u>post</u> …)

$= \forall\, u_1 : \forall z \in \text{Hotel ref} : \forall m \in z.\text{checkIn ref}\ \Box(\, m \text{ new} \wedge \xi^{ext}_{pre} \text{ and } \delta(\xi_{post}))$

It remains to consider: $\xi^{ext}_{pre} \equiv \neg\, \overline{includes}(z.\text{guests}, m.g) \wedge u_1 = \delta(\text{guests})$

$\equiv \neg\, \overline{includes}(z.\text{guests}, m.g) \wedge u_1 = z.\text{guests}$

$\Delta(\xi_{post}) = \overline{size}\,(z.\text{guests}) = \overline{size}\,(u_1) + 1 \wedge \overline{includes}(z.\text{guests}, m.g)$

Configuration during the execution of checkIn(g_1):