FOUNDATIONS OF THE UML

Lecture # 12 :

OCL Semantics

Joost-Pieter Katoen

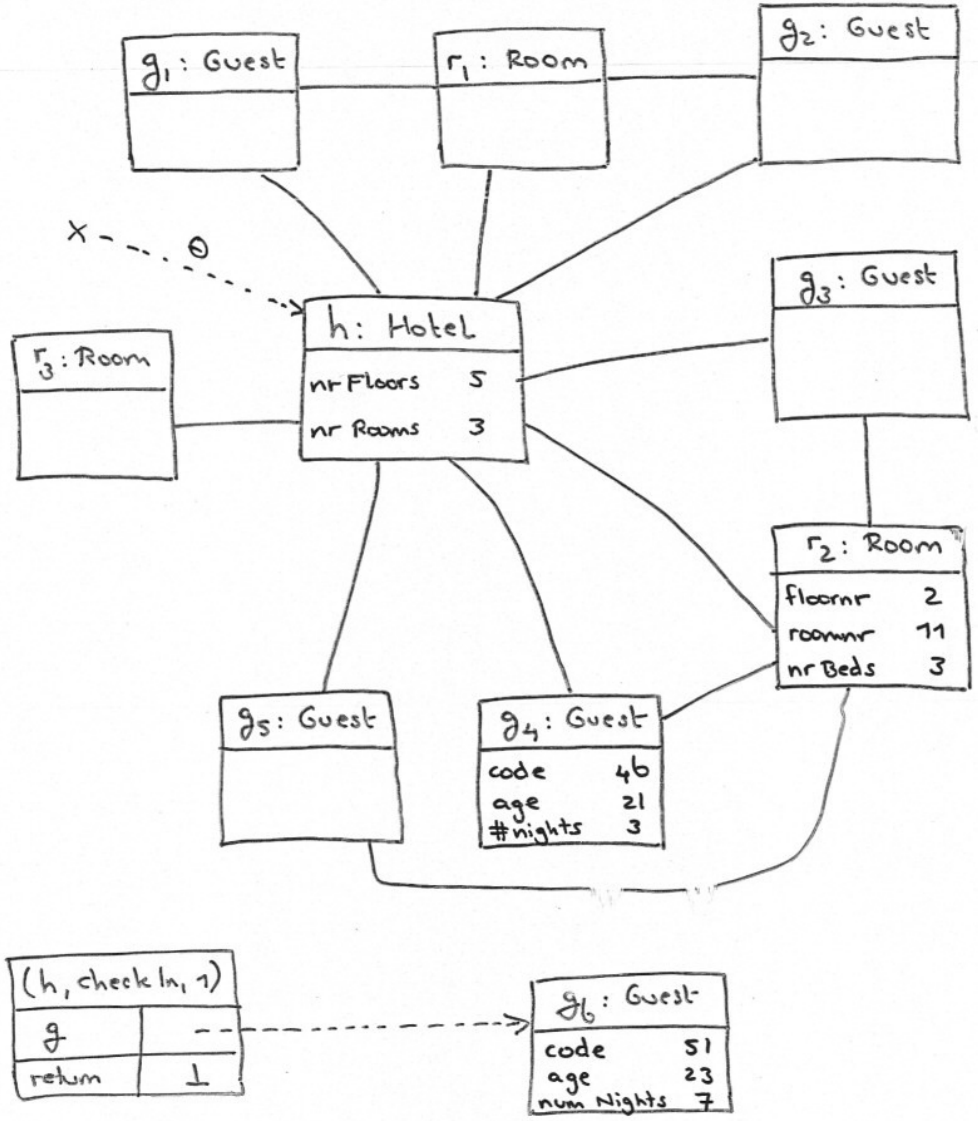
MOVES @ RWTH

moves.rwth-aachen.de / i2 / lg2

WS 2007 / 2008

January 28, 2008

CONFIGURATION EXAMPLE



EXAMPLE

Set of active objects:

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

set of active events:

$$E = \{ (h, \text{checkIn}, 1) \}$$

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

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

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

Local state of object g_6:

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

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

$\sigma(g_6)(\text{nrNights}) = 7$

state of event $(h, \text{checkIn}, 1)$:

$\tau(h, \text{checkIn}, 1) = (\ell, \bot)$ with $\ell(g) = g_6$

STATIC EXPRESSIONS

$$\xi ::= x \mid \xi.a \mid \xi.\text{owner} \mid \xi.\text{return} \mid \xi\ \text{new} \mid \xi\ \text{alive} \mid \omega(\xi, \ldots, \xi) \mid \underline{\text{with}}\ x_1 \in \xi\ \underline{\text{from}}\ x_2 := \xi\ \underline{\text{do}}\ x_2 := \xi$$

- x is a logical variable
- $\xi.a$ stands for parameter / attribute navigation
- $\xi.\text{owner}$ denotes object executing method ξ
- $\xi.\text{result}$ denotes the return value of method ξ
- ξ new is a predicate denoting that the object (or method) ξ is "fresh" in the current state
  - an object is new just after its creation
  - a method is new when it is just invoked
- ξ alive is a predicate denoting that the object (or method) ξ is currently alive
  - an object becomes alive when it is created and remains alive until it is deallocated (eg. by garbage collection)
- with ... is like the OCL iterate expression

QUANTIFICATION

Temporal expression $\exists x \in \tau : \phi$ expresses that ϕ holds for at least one alive instance x of type τ.

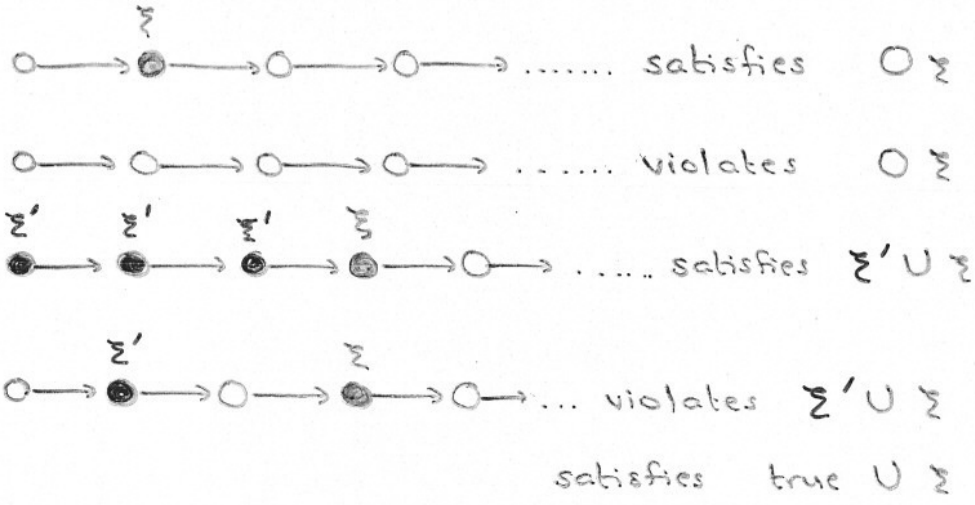
- if τ = void, nat, or bool an instance is always alive (as they are static)
- if $\tau = C.ref$ or $\tau = C.M\ ref$, however, instances are dynamic
  - object $x \in C\ ref$ is alive if it has been created and not yet deallocated;
  - method $x \in C.M\ ref$ is alive if occurrence M has been invocated and has not yet terminated.

TEMPORAL EXPRESSIONS

$$\phi ::= \xi \mid \neg\phi \mid \phi \vee \phi \mid \exists x \in \tau : \phi \mid \bigcirc\phi \mid \phi \cup \phi$$

$\bigcirc\phi$ expresses that in the next state ϕ holds

$\phi \cup \psi$ expresses that a ψ state can be reached via a path solely consisting of ϕ states



Derived operators:

$\Diamond\phi \equiv true \cup \phi$ "eventually"

$\Box\phi \equiv \neg(\Diamond\neg\phi)$ "always"

EXAMPLES

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

$$\Diamond\,(\exists\, m \in h.\text{checkIn ref} : m \text{ alive})$$

or, shortly:

$$\Diamond\,(\exists\, m \in h.\text{checkIn ref})$$

A guest cannot stay forever in hotel h:

$$\Box\,(\forall x \in \text{Guest} : \text{includes}(h.\text{guests}, x) \Longrightarrow \Diamond\,(\neg\, \text{includes}(h.\text{guests}, x)))$$

A guest cannot be hosted in more than one room:

$$\Box\,(\forall x \in \text{Guest} : \exists y \in \text{Room ref} : \exists z \in \text{Room ref} : \text{includes}(y.\text{guests}, x) \wedge \text{includes}(z.\text{guests}, x) \Longrightarrow y = z)$$

SEMANTICS STATIC EXPRESSIONS

Let $\Theta : \text{LVAR} \longrightarrow \text{VAL}$ assign values to logical variables

i.e., for $x \in \text{LVAR} \cap \text{dom}(\Theta)$, $\Theta(x)$ is the value of x.

Semantics of static expression Σ is given by:

$$[\![\,\Sigma\,]\!]_{\sigma, N, \Theta} \in \text{VAL}_{\perp}$$

where:

- $\sigma = (O_\sigma, E_\sigma, \tau_\sigma, \iota_\sigma)$ is a configuration
  - O_σ: alive objects
  - E_σ: alive method occurrences (= events)
  - τ_σ: state of object
  - ι_σ: variable valuation
- $N \subseteq O_\sigma \cup E_\sigma$ denotes the new objects and events in the configuration σ
- Θ is a valuation of the logical variables (in Σ) x, y, z

$$\Theta(x) = m \qquad \Theta(y) = 0$$

SEMANTICS

$$[\![x]\!]_{\sigma,N,\Theta} = \Theta(x)$$ logical valuation of x

$$[\![\Sigma.\text{owner}]\!]_{\sigma,N,\Theta} = o \quad \text{where} \quad \underbrace{[\![\Sigma]\!]_{\sigma,N,\Theta} = (o, M, j)}_{\text{object } o \text{ invoked } M}$$

$$[\![\Sigma.\text{return}]\!]_{\sigma,N,\Theta} = v \quad \text{where} \quad \underbrace{r_\sigma([\![\Sigma]\!]_{\sigma,N,\Theta}) = (\ell, v)}_{\text{evaluation of } \Sigma \text{ in configuration } \sigma}$$

$$[\![\Sigma \text{ new}]\!]_{\sigma,N,\Theta} = \underbrace{([\![\Sigma]\!]_{\sigma,N,\Theta} \in N)}_{\Sigma \text{ refers to a new object or event (in } \sigma)}$$

$$[\![\Sigma \text{ alive}]\!]_{\sigma,N,\Theta} = [\![\Sigma]\!]_{\sigma,N,\Theta} \in O_\sigma \cup E_\sigma$$

$$[\![\omega(\Sigma_1, \dots, \Sigma_n)]\!]_{\sigma,N,\Theta} = \underbrace{[\![\omega]\!]}_{\text{semantic counterpart (on } VAL_\perp) \text{ of operation } \omega} ([\![\Sigma_1]\!]_{\sigma,N,\Theta}, \dots, [\![\Sigma_n]\!]_{\sigma,N,\Theta})$$

SEMANTICS

For the semantics of $\Sigma.a$ distinguish two cases:

$[\![\Sigma]\!]$ is a reference <u>or</u> a list (of refs)

$$[\![\Sigma.a]\!]_{\sigma,N,\Theta} = \ell(a) \quad \text{where}$$

$[\![\Sigma]\!]_{\sigma,N,\Theta} \in C \text{ ref}$ and $\underbrace{\sigma_\sigma([\![\Sigma]\!]_{\sigma,N,\Theta}) = \ell}_{\text{state of object } \Sigma}$

or $[\![\Sigma]\!]_{\dots} \in C.M \text{ ref}$ and $\underbrace{r_\sigma([\![\Sigma]\!]_{\dots}) = (\ell, v)}_{\text{state of method } \Sigma}$

For lists:

$$[\![\Sigma.a]\!]_{\sigma,N,\Theta} = \vec{\ell}(a) \quad \text{where}$$

$[\![\Sigma]\!]_{\dots} \in C \text{ ref list}$ and $\sigma_\sigma([\![\Sigma]\!]) = \vec{\ell}$

or $[\![\Sigma.]\!]_{\dots} \in C.M \text{ ref list}$ and $r_\sigma([\![\Sigma]\!]) = (\vec{\ell}, v)$

SEMANTICS

$[\![\underline{\text{with}}\ x_1 \in \Sigma_1\ \underline{\text{from}}\ x_2 \in \Sigma_2\ \underline{\text{do}}\ x_2 := \Sigma_3]\!]_{\sigma,N,\theta}$

$= [\![\underline{\text{for}}\ x_1 \in [\![\Sigma_1]\!]_{\sigma,N,\theta}\ \underline{\text{do}}\ x_2 := \Sigma_3]\!]_{\sigma,N,\theta'}$

where $\theta' = \underbrace{\theta\,[x_2 := [\![\Sigma_2]\!]_{\sigma,N,\theta}]}$

θ but x_2 is bound to $[\![\Sigma_2]\!]$

and

$[\![\underline{\text{for}}\ x_1 \in \underbrace{[\,]}\ \underline{\text{do}}\ x_2 := \Sigma]\!]_{\sigma,N,\theta} = [\![x_2]\!]_{\sigma,N,\theta}$

empty list

$[\![\underline{\text{for}}\ x_1 \in h :: w\ \underline{\text{do}}\ x_2 := \Sigma]\!]_{\sigma,N,\theta} =$

$[\![\underline{\text{for}}\ x_1 \in w\ \underline{\text{do}}\ x_2 := \Sigma]\!]_{\sigma,N,\theta''}$

with $\theta'' = \theta\,[x_2 := [\![\Sigma]\!]_{\sigma,N,\theta[x_1 := h]}]$

EXAMPLE

Let $\sigma = (O, E, \sigma, \iota)$ as in previous example

$\theta(x) = h$ and $N = \emptyset$. Then:

$[\![(x.\ \text{rooms}).\text{guests}]\!]_{\sigma,N,\theta}$

$=$ (* $[\![x.\ \text{rooms}]\!] \in C\ \text{ref list}$ *)

$\sigma([\![x.\ \text{rooms}]\!]_{\sigma,N,\theta})\ (\text{guests})$

$=$ (* $[\![x]\!] \in C\ \text{ref list}$ *)

$\sigma(\sigma([\![x]\!]_{\sigma,N,\theta})(\text{rooms}))\ (\text{guests})$

$=$ (* $[\![x]\!] = \theta(x) = h$ *)

$\sigma(\underbrace{\sigma(h)(\text{rooms})})\ (\text{guests})$

$=$ (* $h.\ \text{rooms} = [r_1, r_2, r_3]$ *)

$\sigma([r_1, r_2, r_3])\ (\text{guests})$

$=$ (* $r_1.\ \text{guests} = [g_1, g_2]$ $\quad r_3.\ \text{guests} = []$

$r_2.\ \text{guests} = [g_3, g_4, g_5]$ *)

$[\,[g_1, g_2], [g_3, g_4, g_5], []\,]$

TEMPORAL EXPRESSIONS

Temporal expressions such as $\bigcirc\Phi$ and $\phi \cup \Psi$ are interpreted over paths in the automaton $(\text{Conf}, \longrightarrow, I)$

$\pi = c_0 c_1 c_2 \ldots\ldots$ is a path

if $c_i \in \text{Conf}$ and $c_i \longrightarrow c_{i+1}$, for all $i \geq 0$

Notation $\pi[i] = c_i$

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

- $N_0 = N \subseteq \underbrace{O_0 \cup E_0}$
  objects and events in configuration c_0
- $N_{i+1} = \underbrace{(O_{i+1} \setminus O_i)}_{\text{objects created in } c_i \to c_{i+1}} \cup \underbrace{(E_{i+1} \setminus E_i)}_{\text{events generated in } c_i \to c_{i+1}}$
- $\theta_i(x) = \begin{cases} \theta(x) & \text{if } \forall k \leq i : \theta(x) \in O_k \cup E_k \\ \text{undefined} & \text{else} \end{cases}$

TEMPORAL EXPRESSIONS

The semantics of ϕ is given by a relation $\models$.

$(\pi, N, \theta, \phi) \in \models$ should be read as:

for path π, initial set N of new objects/methods, and logical valuation θ, formula ϕ holds.

Write $\pi, N, \theta \models \phi$ instead of $(\pi, N, \theta, \phi) \in \models$.

By structural induction over ϕ:

$\pi, N, \theta \models \Sigma$ iff $[\![\Sigma]\!]_{\pi[0], N, \theta} = tt$

$\pi, N, \theta \models \neg\phi$ iff $\pi, N, \theta \not\models \phi$

$\pi, N, \theta \models \phi \vee \psi$ iff $\pi, N, \theta \models \phi$ or $\pi, N, \theta \models \psi$

$\pi, N, \theta \models \bigcirc\phi$ iff $\pi^1, N_1, \theta_1 \models \phi$
↳ $\pi[1]\ \pi[2] \ldots\ldots$

$\pi, N, \theta \models \phi \cup \psi$ iff $\exists j \geq 0 : \pi^j, N_j, \theta_j \models \psi$ and
$\forall k < j : \pi^k, N_k, \theta_k \models \phi$
↳ $\pi[k]\ \pi[k+1] \ldots\ldots$

$\pi, N, \theta \models \exists x \in \tau.\ \phi$ iff $\exists v \in VAL^{\tau}(O_0, E_0) :$
$\pi, N, \theta[x := v] \models \phi$

TEMPORAL EXPRESSIONS

$VAL^\tau \restriction (O,E)$ is the subset of VAL^τ alive in (O,E)

Formally:

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

The main remaining issue is how to Map OCL into our static and temporal expressions.

OCL expressions $\longmapsto$ static expressions

OCL constraints $\longmapsto$ temporal expressions

TRANSLATING OCL TYPES

OCL allows sets, bags, and lists, but no nested lists

OCL types are defined by:

$$\rho ::= \text{nat} \mid \text{bool} \mid C \text{ ref}$$

$$\tau ::= \rho \mid \rho \text{ list} \mid \rho \text{ set} \mid \rho \text{ bag}$$

Universe of values:

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

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

$$VAL^{C \text{ ref}} = \{null\} \cup OID^C$$

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

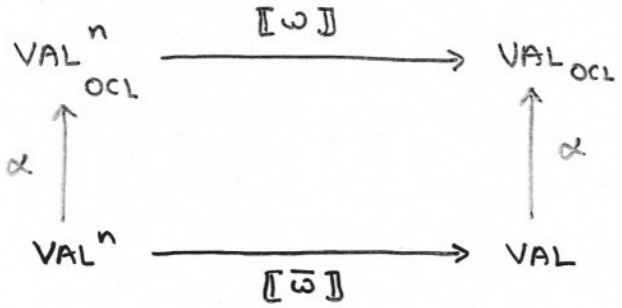
$$VAL^{\rho \text{ set}} = 2^{VAL^\rho}$$

$$VAL^{\rho \text{ bag}} = VAL^\rho \longrightarrow \mathbb{N}$$

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

TRANSLATING OCL TYPES

For each operation $\Sigma_1 \rightarrow \omega(\Sigma_2, \ldots, \Sigma_n)$ in OCL on sets or bags, there exists a corresponding operation $\overline{\omega}(\Sigma_1, \ldots, \Sigma_n)$ such that the following diagram commutes:



where α is an abstraction function. For sets:

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

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

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

EXAMPLE

- Consider the OCL expression $\Sigma_1 \rightarrow union(\Sigma_2)$

The corresponding operator $\overline{union}$ has semantics

$$[\![\,\overline{union}\,]\!] : VAL^{\tau\ list} \times VAL^{\tau\ list} \rightarrow VAL^{\tau\ list}$$

According to the commutativity diagram we have:

$$\alpha_{set}([\![\,\overline{union}(v_1, v_2)\,]\!]) = [\![\,union\,]\!](\alpha_{set}(v_1), \alpha_{set}(v_2))$$

union of lists — union of sets

e.g. $\overline{union}(w_1, w_2) = w_1 \mathbin{+\!\!+} w_2$ (list concatenation)

- OCL equality on sets: $\Sigma_1 = \Sigma_2$

$$[\![\,\overline{=_{set}}\,]\!] : VAL^{\tau\ list} \times VAL^{\tau\ list} \rightarrow VAL^{bool}$$

where $\overline{=_{set}}$ on lists is defined as follows:

$$\overline{=_{set}}(w_1, w_2) = Eq\,list\,(sort\,(del\_duplicates(w_1)),\ sort\,(del\_duplicates(w_2)))$$

TRANSLATING OCL EXPRESSIONS

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

$$\delta(\text{self}) = o$$

$$\delta(x) = \begin{cases} o.x & \text{if } o \in C\,\text{ref} \wedge x \in \text{dom}(C.\text{attrs}) \\ m.x & \text{if } x \in \vec{p} \\ x & \text{otherwise} \end{cases}$$

$$\delta(\text{result}) = m.\text{return}$$

$$\delta(\Sigma\ @_i\ \text{pre}) = u_i \quad (i\text{-th occurrence of } @\text{pre})$$

$$\delta(\Sigma.a) = \begin{cases} \text{flat}(\delta(\Sigma).a) & \text{if } \Sigma \in C\,\text{ref}_{\text{list}} \\ \delta(\Sigma).a & \text{otherwise} \end{cases}$$

$$\delta(\omega(\Sigma_1,\ldots,\Sigma_n)) = \overline{\omega}(\delta(\Sigma_1),\ldots,\delta(\Sigma_n))$$

$$\delta(\Sigma.\omega(\Sigma_1,\ldots,\Sigma_n)) = \overline{\omega}(\delta(\Sigma),\delta(\Sigma_1),\ldots,\delta(\Sigma_n))$$

$$\delta(\Sigma \to \omega(\Sigma_1,\ldots,\Sigma_n)) = \text{dito}$$

$$\delta(\Sigma_1 \to \text{iterate}(x_1 ; x_2 = \Sigma_2 \mid \Sigma_3)) =$$

with $x_1 \in \delta(\Sigma_1)$ from $x_2 := \delta(\Sigma_2)$

do $x_2 := \delta(\Sigma_3)$

TRANSLATING OCL INVARIANTS

context C inv Σ : the condition Σ 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 Σ.

Let $y \in \text{LVAR}$ and $\text{dom}(C.\text{meths}) = \{M_1,\ldots,M_k\}$

Then:

$$\Delta(\underline{\text{context}}\ C\ \underline{\text{inv}}\ \Sigma)$$

$$=$$

$$\Box\Big(\forall x \in C\,\text{ref} :$$

$$\big(\neg\exists m_1 \in x.M_1\,\text{ref} \wedge \ldots \wedge \neg\exists m_k \in x.M_k\,\text{ref}\big)$$

$$\text{implies}\ \underbrace{\delta_{x,y,[]}(\Sigma)}_{\text{translation of } \Sigma}\Big)$$

EXAMPLE

context Hotel

inv rooms.guests = guests (*)

we have:

$\delta((\text{rooms.guests}) = \text{guests}) =$

Eglist (sort (δ (rooms.guests)), sort (δ (guests)))

= Eglist (sort (flat (x.rooms.guests)), sort (x.guests))

Now we obtain for Δ (.... inv ....) :

$\Box$ ($\forall x \in$ Hotel ref :

($\neg \exists m \in x.$checkIn ref $\wedge$ $\neg \exists m' \in x.$checkOut)

implies Eglist (.....))

$\delta(*)$

TRANSLATING PRE- POST CONDITIONS

Main complication: for each $\Sigma@$pre expression in the postcondition, we should "remember" its value on evaluating the precondition.

This is done using auxiliary variables: for each $\Sigma@_i$ pre use auxiliary variable u_i.

Extended precondition = precondition Σ_{pre} + auxiliary variables $\{u_1, \ldots, u_n\}$

Formally: $\Sigma^{ext}_{pre} = \delta(\Sigma_{pre})$

$\wedge \bigwedge_{\Sigma@_i \text{ pre in } \Sigma_{post}} \underbrace{u_i = \delta(\Sigma)}_{\text{"freeze" value of } \Sigma \text{ in } u_i}$

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

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

$\Box$ (m new $\wedge$ Σ^{ext}_{pre} implies

m alive $\cup$ (term (m) $\wedge$ $\delta(\Sigma_{post})$))

m alive $\wedge$ $\bigcirc$ (m alive)

EXAMPLE

context Hotel: checkIn (g: Guest)

pre not guests → includes (g)

post guests → size = guests@pre → size + 1

and guests → includes (g)

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

m = occurrence of checkIn

Δ (context Hotel ... pre ..... post ...)

=

$\forall u_1 : \forall z \in$ Hotel ref $: \forall m \in z.$checkIn ref

$\Box (m \text{ new} \wedge \Sigma^{ext}_{pre}$

$\Rightarrow m \text{ alive } \mathcal{U} (\text{term}(m) \wedge \delta(\Sigma_{post}))$

It remains to consider Σ^{ext}_{pre} and $\delta(\Sigma_{post})$.

$\Sigma^{ext}_{pre} \equiv \neg \overline{\text{includes}} (z.\text{guests}, m.g) \wedge u_1 = \delta(\text{guests})$

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

$\delta(\Sigma_{post}) = \overline{\text{size}} (z.\text{guests}) = \overline{\text{size}} (u_1) + 1$

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

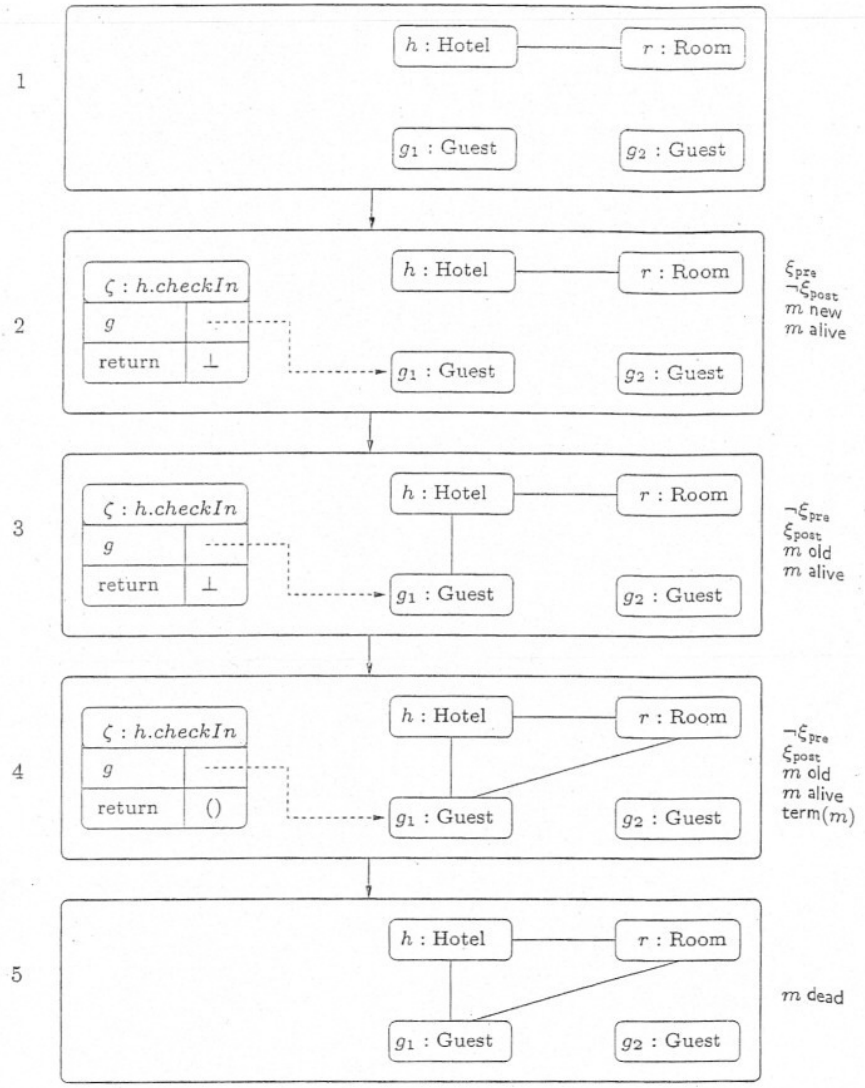


Figure 3.4: Configurations during the execution of checkIn(g_1).