Software Design, Modelling and Analysis in UML

Lecture 04: OCL Cont'd, Object Diagrams

2011-10-31

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 is an object diagram? What are object diagrams good for?
  - When is an object diagram called partial? What are partial ones good for?
  - When is an object diagram an object diagram (wrt. what)?
  - Is this an object diagram wrt. to that other thing?
  - How are system states and object diagrams related?
  - 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?
- **Content:**
  - OCL Semantics
  - Object Diagrams
  - Example: Object Diagrams for Documentation
  - OCL: consistency, satisfiability

OCL Semantics [OMG, 2006]

The Task

OCL Syntax 1/4: Expressions

$$
\begin{array}{lll}
expr ::= & & \\
\quad w & : \tau(w) \\
\quad \mid expr_1 =_\tau expr_2 & : \tau \times \tau \to Bool \\
\quad \mid \mathsf{oclIsUndefined}_\tau(expr_1) & : \tau \to Bool \\
\quad \mid \{expr_1, \dots, expr_n\} & : \tau \times \dots \times \tau \to Set(\tau) \\
\quad \mid \mathsf{isEmpty}(expr_1) & : Set(\tau) \to Bool \\
\quad \mid \mathsf{size}(expr_1) & : Set(\tau) \to Int \\
\quad \mid \mathsf{allInstances}_C & : Set(\tau_C) \\
\quad \mid v(expr_1) & : \tau_C \to \tau(v) \\
\quad \mid r_1(expr_1) & : \tau_C \to \tau_D \\
\quad \mid r_2(expr_1) & : \tau_C \to Set(\tau_D)
\end{array}
$$

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

- $W \supseteq \{self\}$ is a set of typed logical variables, w has type $\tau(w)$
- τ is any type from $\mathscr{T} \cup T_B \cup T_\mathscr{C} \cup \{Set(\tau_0) \mid \tau_0 \in T_B \cup T_\mathscr{C}\}$
  - T_B is a set of basic types, in the following we use $T_B = \{Bool, Int, String\}$
  - $T_\mathscr{C} = \{\tau_C \mid C \in \mathscr{C}\}$ is the set of object types,
  - $Set(\tau_0)$ denotes the set-of-τ_0 type for $\tau_0 \in T_B \cup T_\mathscr{C}$ (sufficient because of "flattening" (cf. standard))
- $v : \tau(v) \in atr(C)$, $\tau(v) \in \mathscr{T}$,
- $r_1 : D_{0,1} \in atr(C)$,
- $r_2 : D_* \in atr(C)$,
- $C, D \in \mathscr{C}$.

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

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

  such that

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

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**).

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

(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

Recall:

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

We set:

"undefined"

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

We may omit index τ of $\perp_\tau$ 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

disjoint union, i.e. assume $\perp_{\tau_C} \notin \mathscr{D}(C)$ otherwise rename in $\mathscr{D}(C)$

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

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

powerset of $I(\tau)$, i.e. set of subsets of $I(\tau)$

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

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

– 04 – 2011-10-31 – Soclsem –

(iv) Interpretation of Arithmetic Operations

- **Literals** map to fixed values:

$$I(\mathsf{true}) := true, \quad I(\mathsf{false}) := false, \quad I(0) := 0, \quad I(1) := 1, \ldots$$

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

equality relation

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

symbol in OCL syntax

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

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

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

Note: There is a **common principle**.
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).

– 04 – 2011-10-31 – Soclsem –

(iv) Interpretation of OclIsUndefined

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

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

(v) Interpretation of Set Operations

Basically the same principle as with arithmetic operations...

Let $\tau \in T_B \cup T_{\mathscr{C}}$.

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

  $$I(\{\}_n^\tau)(x_1, \dots, x_n) := \{x_1, \dots, x_n\}$$

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

  cardinality

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

(vi) Putting It All Together

OCL Syntax 1/4: Expressions

$expr ::=$

w	$: \tau(w)$
$\mid expr_1 =_\tau expr_2$	$: \tau \times \tau \to Bool$
$\mid$ oclIsUndefined$_\tau(expr_1)$	$: \tau \to Bool$
$\mid \{expr_1, \dots, expr_n\}$	$: \tau \times \dots \times \tau \to Set(\tau)$
$\mid$ isEmpty$(expr_1)$	$: Set(\tau) \to Bool$
$\mid$ size$(expr_1)$	$: Set(\tau) \to Int$
$\mid$ allInstances$_C$	$: Set(\tau_C)$
$\mid v(expr_1)$	$: \tau_C \to \tau(v)$
$\mid r_1(expr_1)$	$: \tau_C \to \tau_D$
$\mid r_2(expr_1)$	$: \tau_C \to Set(\tau_D)$

Where, given $\mathscr{S} = (\mathscr{T}, \mathscr{C},$

- $W \supseteq \{self\}$ is a set of t… logical variables, w has …
- τ is any type from $\mathscr{T} \cup$ … $\cup \{Set(\tau_0) \mid \tau_0 \in T_B \cup$ …
  - T_B is a set of basic … the following we use … $T_B = \{Bool, Int, St$…
  - $T_{\mathscr{C}} = \{\tau_C \mid C \in \mathscr{C}\}$ … set of object types,
  - $Set(\tau_0)$ denotes the set-of-τ_0 type for $\tau_0 \in T_B \cup T_{\mathscr{C}}$ (sufficient because o… "flattening" (cf. star…
- $v : \tau(v) \in atr(C)$, $\tau(v)$ …
- $r_1 : D_{0,1} \in atr(C)$,
- $r_2 : D_* \in atr(C)$,
- $C, D \in \mathscr{C}$.

OCL Syntax 2/4: Constants, Arithmetical Operat…

For example:

$expr ::= \dots$	
$\mid$ true, false	$: Bool$
$\mid expr_1$ {and, or, implies} $expr_2$	$: Bool \times Bool$ –
$\mid$ not $expr_1$	$: Bool \to Bool$
$\mid 0, -1, 1, -2, 2, \dots$	$: Int$
$\mid$ OclUndefined	$: \tau$
$\mid expr_1 \; \{+, -, \dots\} \; expr_2$	$: Int \times Int \to I$
$\mid expr_1 \; \{<, \leq, \dots\} \; expr_2$	$: Int \times Int \to B$

Generalised notation:

$$expr ::= \omega(expr_1, \dots, expr_n) \qquad : \tau_1 \times \dots \times \tau_n -$$

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

OCL Syntax 3/4: Iterate

$$expr ::= \dots \mid expr_1\text{->iterate}(w_1 : \tau_1 \; ; \; w_2 : \tau_2 = expr_2 \mid expr_3)$$

or, with a little renaming,

$$expr ::= \dots \mid expr_1\text{->iterate}(iter : \tau_1; \; result : \tau_2 = expr_2 \mid expr_3)$$

OCL Syntax 4/4: Context

$$context ::= \text{context } w_1 : \tau_1, \dots, w_n : \tau_n \text{ inv : } expr$$

where $w \in W$ and $\tau_i \in T_{\mathscr{C}}$, $1 \leq i \leq n$, $n \geq 0$.

Valuations of Logical Variables

- **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 $w \in W$,

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

$$\beta : W \longrightarrow \bigcup_{w \in W} I(\tau(w))$$

e.g. if $w : \tau_C$ then

$\beta(w) \in I(\tau_C) = D(C) \cup \{\bot\}$

(vi) Putting It All Together...

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

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

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

– 04 – 2011-10-31 – Soclsem –

(vi) Putting It All Together...

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

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

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

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

– 04 – 2011-10-31 – Soclsem –

(vi) Putting It All Together...

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

assign to hlp the set denoted by $expr_1$

assign to v_2 the initial expression

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

modification of β at hlp and

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

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

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

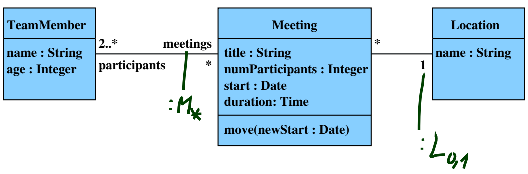
$$:= \begin{cases} I[\![expr_3]\!](\sigma, \beta'[v_1 \mapsto x]) & \text{, if } \beta'(hlp) = \{x\} \\ I[\![expr_3]\!](\sigma, \beta'') & \text{, if } \beta'(hlp) = X \,\dot\cup\, \{x\} \text{ and } X \neq \emptyset \end{cases}$$

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?

Example

... or: $\mapsto$



- `context TeamMember inv: age => 18`
- `context Meeting inv: duration > 0`

$\sigma = \{1_{TM} \mapsto \{name = \text{"Schulze"}, age = 27, meetings = \{3_M\}\},$
$3_M \mapsto \{title = \text{"Briefing"}, numPart = 2, start = 1.5.23, duration = 120, participants = \{1_{TM}, 8_{TM}\}, location = \{7_L\}\},$
$7_L \mapsto \{name = \text{"Hall"}, meeting = \{3_M\}\},$
$8_{TM} \mapsto \{name = \text{"Boss"}, age = 57, meetings = \{3_M\}\}\}$

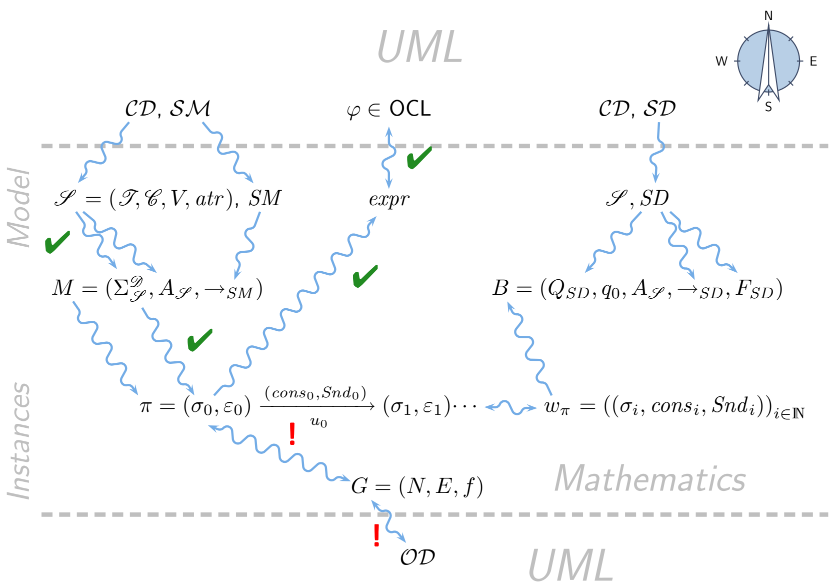
$\mathsf{allInstances}_{TM}\text{->}\mathsf{iterate}(self : TeamMember; res : Bool = true \mid res \text{ and } \geq(age(self), 18))$

$\beta = \{self \mapsto 1_{TM}\}$

- $I[\![self]\!](\sigma, \beta) = \beta(self) = 1_{TM}$
- $I[\![age(self)]\!](\sigma, \beta) = \sigma(1_{TM})(age) = 27$
- $I[\![\geq(age(self), 18)]\!](\sigma, \beta) = I(\geq)(I[\![age(self)]\!](\sigma, \beta), I(18)) = \geq(27, 18) = true \in I(Bool)$
- $I[\![\mathsf{allInstances}_{TM}]\!](\sigma, \beta) = dom(\sigma) \cap \mathscr{D}(TM) = \{1_{TM}, 7_L, 8_{TM}, 3_M\} \cap \{1_{TM}, 2_{TM}, \ldots\} = \{1_{TM}, 8_{TM}\}$
- $I[\![meetings(self)]\!](\sigma, \beta) = \{3_M\}$
- $I[\![location(x)]\!](\sigma, \{x \mapsto 3_M\}) = 7_L$
- something with $\bot$: exercises / tutorial

Where Are We?

You Are Here.



Object Diagrams

Graph

Definition. A node labelled graph is a triple

$$G = (N, E, f)$$

consisting of

- vertexes N,
- edges E,
- node labeling $f : N \to X$, where X is some label domain,

Definition. Let $\mathscr{D}$ be a structure of signature $\mathscr{S} = (\mathscr{T}, \mathscr{C}, V, atr)$ and $\sigma \in \Sigma_{\mathscr{S}}^{\mathscr{D}}$ a system state.

Then any graph $G = (N, E, f)$ with

- nodes are identities (not necessarily alive), i.e.

  $$N \subset \mathscr{D}(\mathscr{C}) \text{ finite},$$

- edges correspond to "links" of objects, i.e.

  source — attribute — destination — $:= V_{0,1,*}$

  $$E \subseteq N \times \{v : \tau \in V \mid \tau \in \{C_{0,1}, C_* \mid C \in \mathscr{C}\}\} \times N,$$

  source is alive in σ — source refers to destination via r

  $$\forall\, (u_1, r, u_2) \in E : u_1 \in \mathrm{dom}(\sigma) \wedge u_2 \in \sigma(u_1)(r),$$

- objects are labelled with attribute valuations and non-alive identities marked with "X", i.e.

  $$X = \{\mathsf{X}\} \,\dot\cup\, (V \nrightarrow (\mathscr{D}(\mathscr{T}) \cup \mathscr{D}(\mathscr{C}_*)))$$

  $$\forall\, u \in N \cap \mathrm{dom}(\sigma) : f(u) \subseteq \sigma(u)$$

  $$\forall\, u \in N \setminus \mathrm{dom}(\sigma) : f(u) = \{\mathsf{X}\}$$

  labelling of u is consistent with σ, may leave out some attributes

  note: we may have values of $V_{0,1,*}$ attributes in the labelling (maybe redundant with edges)

is called object diagram of σ.

$\sigma = \{ 1_{TM} \mapsto \{ name = \text{"Schulze"}, age = 27, meetings = \{3_M\}\},$
$3_M \mapsto \{ title = \text{"Briefing"},$
$numPart = 2,$
$start = 1.5.23,$
$duration = 120$
$participants = \{1_{TM}, 8_{TM}\} \cup \{5_{TM}\},$
$location = \{7_L\}\},$
$7_L \mapsto \{ name = \text{Hall}, meeting = \{3_M\}\}$
$8_{TM} \mapsto \{ name = \text{"Boss"}, age = 57, meetings = \{3_M\}\}\}$

(N, E, f)

$N = \{1_{TM}, 3_M, 5_{TM}\}$

$E = \{ (3_M, participants, 1_{TM}),$
$(3_M, participants, 5_{TM})\}$

$f = \{ 1_{TM} \mapsto \{age = 27\},$
$3_M \mapsto \{participants = \{1_{TM}, 8_{TM}, 5_{TM}\},$
$5_{TM} \mapsto X\}$

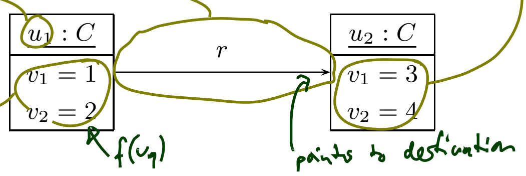
Graphical Representation of Object Diagrams

$N \subset \mathscr{D}(\mathscr{C})$ finite, $E \subset N \times V_{0,1,*} \times N$, $X = \{\mathsf{X}\} \dot\cup (V \nrightarrow (\mathscr{D}(\mathscr{T}) \cup \mathscr{D}(\mathscr{C}_*)))$
$u_1 \in \mathrm{dom}(\sigma) \wedge u_2 \in \sigma(u_1)(r)$, $f(u) \subseteq \sigma(u)$ or $f(u) = \{\mathsf{X}\}$

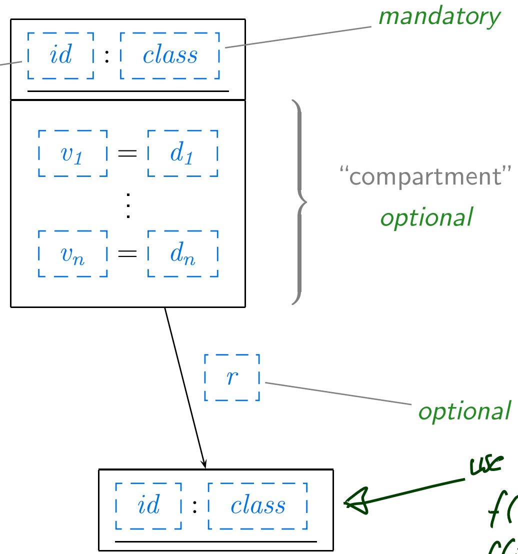
- Assume $\mathscr{S} = (\{Int\}, \{C\}, \{v_1 : Int, v_2 : Int, r : C_*\}, \{C \mapsto \{v_1, v_2, r\}\})$.
- Consider
  $\sigma = \{u_1 \mapsto \{v_1 \mapsto 1, v_2 \mapsto 2, r \mapsto \{u_2\}\}, u_2 \mapsto \{v_1 \mapsto 3, v_2 \mapsto 4, r \mapsto \emptyset\}\}$
- Then $G = (N, E, f)$

  $= (\{u_1, u_2\}, \{(u_1, r, u_2)\}, \{u_1 \mapsto \{v_1 \mapsto 1, v_2 \mapsto 2\}, u_2 \mapsto \{v_1 \mapsto 3, v_2 \mapsto 4\}\},$

  is an object diagram of σ wrt. $\mathscr{S}$ and any $\mathscr{D}$ with $\mathscr{D}(Int) \supseteq \{1, 2, 3, 4\}$.
- We may equivalently (!) **represent** G graphically as follows:



UML Notation for Object Diagrams



(we assume: different "boxes" represent different objects)

use this if $f(id) = \emptyset$ or $f(id) = X$

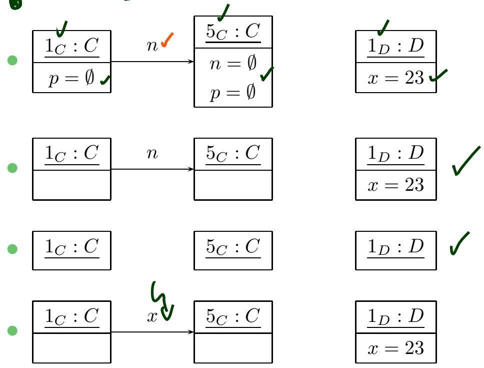
Object Diagrams: More Examples

$N \subset \mathscr{D}(\mathscr{C})$ finite, $E \subset N \times V_{0,1,*} \times N$, $X = \{\mathsf{X}\} \,\dot\cup\, (V \nrightarrow (\mathscr{D}(\mathscr{T}) \cup \mathscr{D}(\mathscr{C}_*)))$
$u_1 \in \mathrm{dom}(\sigma) \wedge u_2 \in \sigma(u_1)(r)$, $f(u) \subseteq \sigma(u)$ or $f(u) = \{\mathsf{X}\}$

$$\sigma = \{1_C \mapsto \{p \mapsto \emptyset, n \mapsto \{5_C\}\}, 5_C \mapsto \{p \mapsto \emptyset, n \mapsto \emptyset\}, 1_D \mapsto \{x \mapsto 23\}\}$$

vs.

- $(\emptyset, \emptyset, \emptyset)$ ✓ the empty "picture"



Complete vs. Partial Object Diagram

Definition. Let $G = (N, E, f)$ be an object diagram of system state $\sigma \in \Sigma_{\mathscr{S}}^{\mathscr{D}}$.

We call G complete wrt. σ if and only if

- G is object complete, i.e.
  - G consists of all alive objects, i.e. $N = \mathrm{dom}(\sigma)$,
- G is attribute complete, i.e.
  - G comprises all "links" between alive objects, i.e. if $u_2 \in \sigma(u_1)(r)$ for some $u_1, u_2 \in \mathrm{dom}(\sigma)$ and $r \in V$, then $(u_1, r, u_2) \in E$, and
  - each node is labelled with the values of all $\mathscr{T}$-typed attributes, i.e. for each $u \in \mathrm{dom}(\sigma)$,

    $$f(u) \supseteq \sigma(u)|_{V_\mathscr{T}} \cup \{r \mapsto (\sigma(u)(r) \setminus N) \mid r \in V : \sigma(u)(r) \setminus N \neq \emptyset\}$$

    where $V_\mathscr{T} := \{v : \tau \in V \mid \tau \in \mathscr{T}\}$.

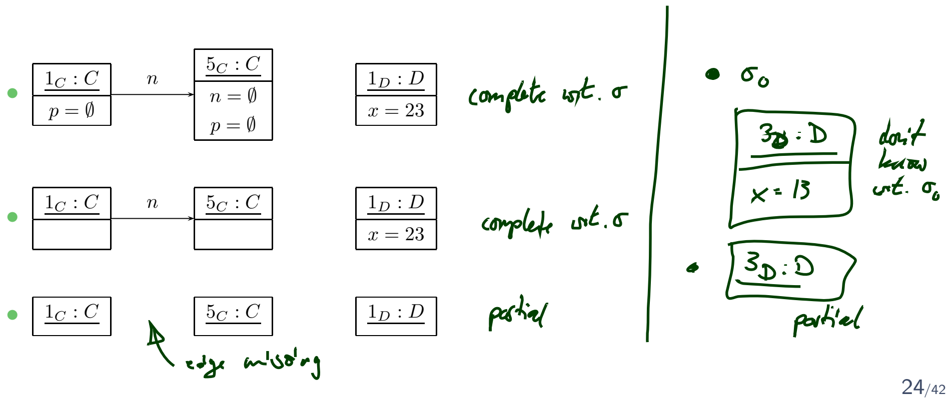
Otherwise we call G partial.

Complete vs. Partial Examples

- $N = \mathrm{dom}(\sigma)$, if $u_2 \in \sigma(u_1)(r)$, then $(u_1, r, u_2) \in E$,
- $f(u) = \sigma(u)|_{V_{\mathscr{T}}} \cup \{r \mapsto (\sigma(u)(r) \setminus N) \mid \sigma(u)(r) \setminus N\}$

Complete or partial?

$$\sigma = \{1_C \mapsto \{p \mapsto \emptyset, n \mapsto \{5_C\}\}, 5_C \mapsto \{p \mapsto \emptyset, n \mapsto \emptyset\}, 1_D \mapsto \{x \mapsto 23\}\}$$



Complete/Partial is Relative

- Claim:
  - Each finite system state has **exactly one complete** object diagram.
  - A finite system state can have **many partial** object diagrams.
- Each object diagram G represents a set of system states, namely

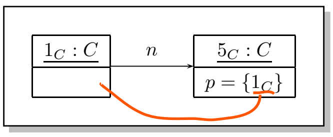
$$G^{-1} := \{\sigma \in \Sigma_{\mathscr{S}}^{\mathscr{D}} \mid G \text{ is an object diagram of } \sigma\}$$

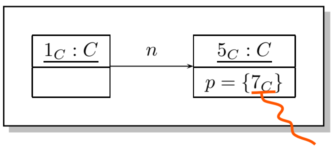
- **Observation**: If somebody **tells us**, that a given (consistent) object diagram G is **complete**, we can uniquely reconstruct the corresponding system state.
  In other words: G^{-1} is then a singleton.

Corner Cases

Closed Object Diagrams vs. Dangling References

Find the 10 differences! (Both diagrams shall be complete.)





Definition. Let σ be a system state. We say attribute $v \in V_{0,1,*}$ has a dangling reference in object $u \in \mathrm{dom}(\sigma)$ if and only if the attribute's value comprises an object which is not alive in σ, i.e. if

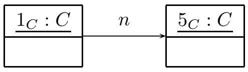
$$\sigma(u)(v) \not\subseteq \mathrm{dom}(\sigma).$$

We call σ closed if and only if no attribute has a dangling reference in any object alive in σ.

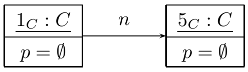
Observation: Let G be the (!) complete object diagram of a **closed** system state σ. Then the nodes in G are labelled with $\mathscr{T}$-typed attribute/value pairs only.

Special Notation

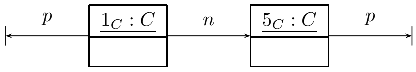
- $\mathscr{S} = (\{Int\}, \{C\}, \{n, p : C_*\}, \{C \mapsto \{n, p\}\})$.
- Instead of



we want to write



or

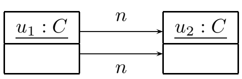


to **explicitly** indicate that attribute $p : C_*$ has value $\emptyset$ (also for $p : C_{0,1}$).

Aftermath

We slightly deviate from the standard (for reasons):

- In the course, $C_{0,1}$ and C_*-typed attributes **only** have **sets as values**. UML also considers multisets, that is, they can have



(This is not an object diagram in the sense of our definition because of the requirement on the edges E. Extension is straightforward but tedious.)

- We **allow** to give the valuation of $C_{0,1}$- or C_*-typed attributes in the **values compartment**.
  - Allows us to indicate that a certain r is not referring to another object.
  - Allows us to represent "dangling references", i.e. references to objects which are not alive in the current system state.
- We introduce a graphical representation of $\emptyset$ values.

References

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.

[Schumann et al., 2008] Schumann, M., Steinke, J., Deck, A., and Westphal, B. (2008).