Software Design, Modelling and Analysis in UML

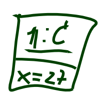
Lecture 04: Object Diagrams

2011-11-09

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

Albert-Ludwigs-Universität Freiburg, Germany

Contents & Goals



Last Lecture:

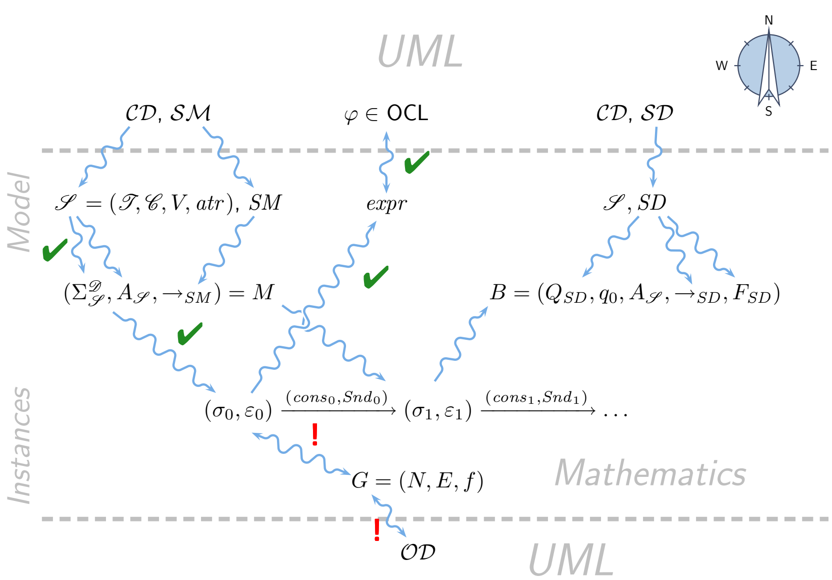
- OCL Syntax and Semantics

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:**
  - Object Diagrams
  - Example: Object Diagrams for Documentation
  - OCL: consistency, satisfiability

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,

Object Diagrams

note: we are allowed to have edges, we choose which to put

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.

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

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

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

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

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

$$E \subseteq \underset{\text{source}}{N} \times \underbrace{\{\overset{\text{attribute}}{v} : \tau \in V \mid \tau \in \{C_{0,1}, C_* \mid C \in \mathscr{C}\}\}}_{V_{0,1,*} :=} \times \underset{\text{destination}}{N},$$

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

source object alive

set of objects referred to via r by u_1

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

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

label of $u \in N$ gives values acc. to $\sigma(u)$, not necessarily all $: V \nrightarrow \mathscr{D}(\mathscr{T}) \cup \mathscr{D}(\mathscr{C}_*)$

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

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

is called object diagram of σ.

Graphical Representation of Object Diagrams

$N \subset \mathscr{D}(\mathscr{C})$ finite, $\quad E \subset N \times V_{0,1,*} \times N$, $\quad X = \{\mathsf{X}\} \,\dot\cup\, (V \nrightarrow (\mathscr{D}(\mathscr{T}) \cup \mathscr{D}(\mathscr{C}_*)))$
$u_1 \in \mathrm{dom}(\sigma) \wedge u_2 \in \sigma(u_1)(r)$, $\qquad f(u) \subseteq \sigma(u)$ 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
  $\sigma = \{u_1 \mapsto \{v_1 \mapsto 1, v_2 \mapsto 2, r \mapsto \{u_2\}\}, u_2 \mapsto \{v_1 \mapsto 3, v_2 \mapsto 4, r \mapsto \emptyset\}\}$ $\quad u_1 \neq u_2$
- 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 $\mathscr{D}(Int) \supseteq \{1, 2, 3, 4\}$.

- $G = (\emptyset, \emptyset, \emptyset)$
  $G = (\{u_1\}, \emptyset, \{u_1 \mapsto \{v_2 \mapsto 2\}\}$
  $G = \{\{u_1\}, \emptyset, u_1 \mapsto \emptyset\}$
  $G = \{\{u_1\}, \emptyset, u_1 \mapsto \{r \mapsto \{u_2\}\}\}$
- $\sigma_2 = \{u_1 \mapsto \{v_1 \mapsto 1, v_2 \mapsto 2, r \mapsto \{u_2\}\}\}$
  $G = \{\{u_1, u_2\}, \{(u_1, r, u_2)\},$
  $\{u_1 \mapsto \{v_1 \mapsto 1, v_2 \mapsto 2\},$
  $u_2 \mapsto \{\mathsf{X}\}\}$ ← u_2 is not alive!

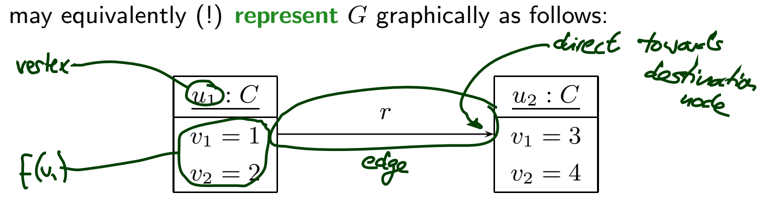
Graphical Representation of Object Diagrams

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

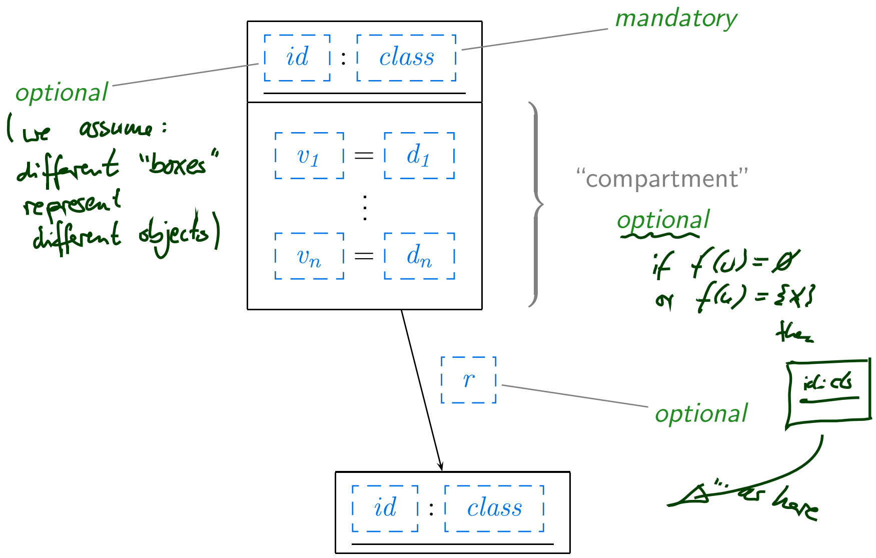
- Assume $\mathscr{S} = (\{Int\}, \{C\}, \{v_1 : Int, v_2 : Int, r : C_*\}, \{C \mapsto \{v_1, v_2, r\}\})$.
- Consider
  $\sigma = \{u_1 \mapsto \{v_1 \mapsto 1, v_2 \mapsto 2, r \mapsto \{u_2\}\}, u_2 \mapsto \{v_1 \mapsto 3, v_2 \mapsto 4, r \mapsto \emptyset\}\}$
- Then $G = (N, E, f)$

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

  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



Object Diagrams: More Examples

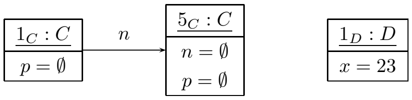
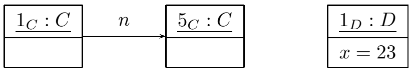
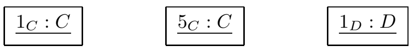
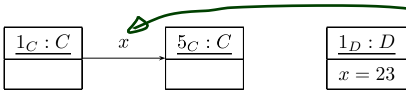
$N \subset \mathscr{D}(\mathscr{C})$ finite, $E \subset N \times V_{0,1,*} \times N$, $X = \{\mathsf{X}\} \,\dot\cup\, (V \nrightarrow (\mathscr{D}(\mathscr{T}) \cup \mathscr{D}(\mathscr{C}_*)))$
$u_1 \in \mathrm{dom}(\sigma) \wedge u_2 \in \sigma(u_1)(r)$, $f(u) \subseteq \sigma(u)$ or $f(u) = \{\mathsf{X}\}$

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

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

vs.

the empty picture

- $(\emptyset, \emptyset, \emptyset)$ ✓
- ✓
-  ✓
-  ✓
-  ✓
- 

NO, $x \notin V_{0,1,*}$

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 ~~comprises~~ consists of all alive objects, i.e. $N \boxed{\cancel{\subset}} \mathrm{dom}(\sigma)$, $=$
- 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 $u \in \mathrm{dom}(\sigma)$,

the non-alive objects refered to via r by u...

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

... if u does refer to some non-alive objects

function restriction

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

Otherwise we call G partial.

Complete vs. Partial Examples

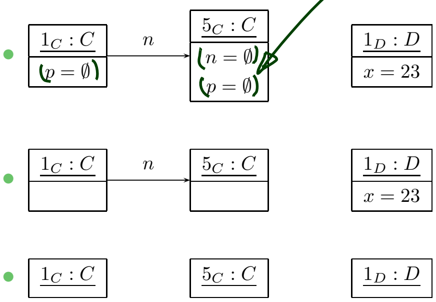
- $N = \mathrm{dom}(\sigma)$, if $u_2 \in \sigma(u_1)(r)$, then $(u_1, r, u_2) \in E$,
- $f(u) = \sigma(u)|_{V_\mathscr{T}} \cup \{r \mapsto (\sigma(u)(r) \setminus N) \mid \sigma(u)(r) \setminus N\}$

Complete or partial? (wrt. system state σ)

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

complete acc. to def. if we remove those

complete: /



$\sigma' = \sigma \cup \{2_D \mapsto \{x \mapsto 0\}\}$

complete

- object-complete
- not attribute complete

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

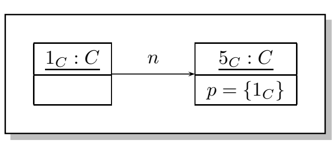
$$G^{-1} := \{\sigma \in \Sigma_\mathscr{S}^\mathscr{D} \mid G \text{ is an object diagram of } \sigma\}$$

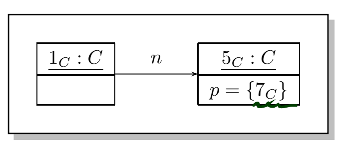
- **Observation**: If somebody **tells us**, that a given (consistent) object diagram G is **complete**, we can uniquely reconstruct the corresponding system state. In other words: G^{-1} is then a singleton.

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

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

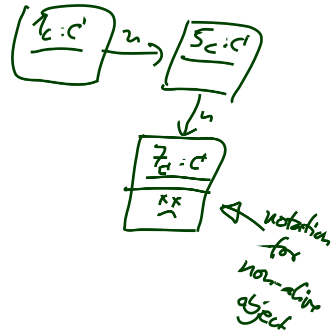
alive objects

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

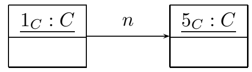
Observation: Let G be the (!) complete object diagram of a **closed** system state σ. Then the nodes in G are labelled with $\mathscr{T}$-typed attribute/value pairs only.

Special Notation

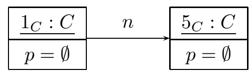
possible extension:



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

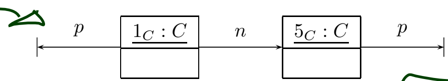


we want to write
can



or

not OMG UML

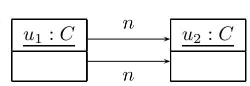


to **explicitly** indicate that attribute $p : C_*$ has value $\emptyset$ (also for $p : C_{0,1}$).

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

"


"

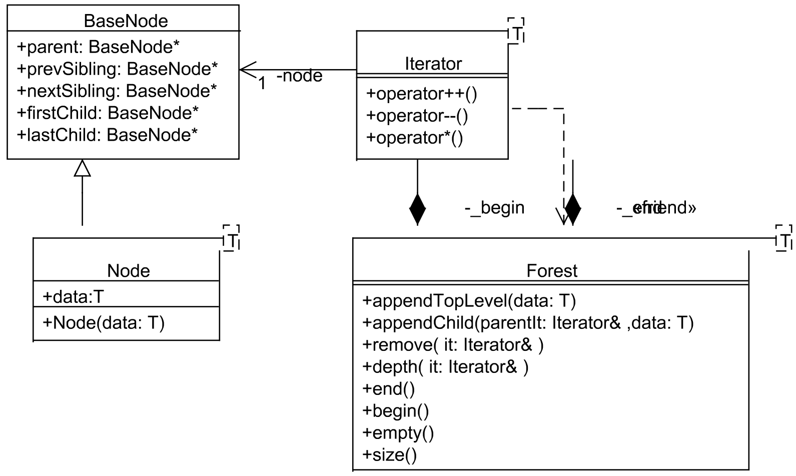
(This is not an object diagram in the sense of our definition because of the requirement on the edges E. Extension is straightforward but tedious.)

- 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.

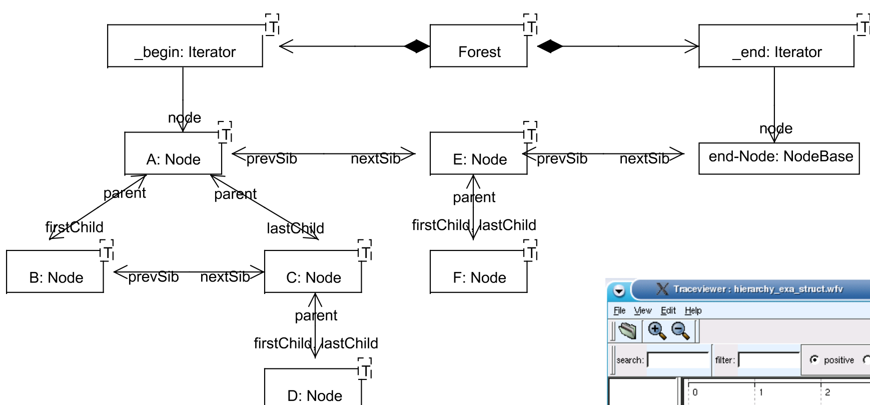
The Other Way Round

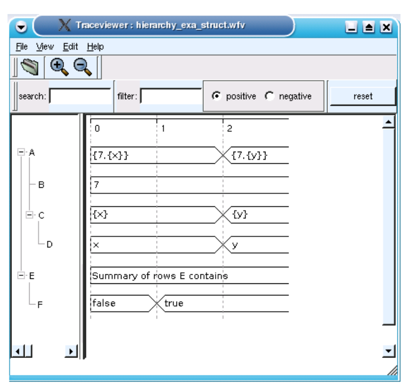
Example: Object Diagrams for Documentation

Example: Data Structure [Schumann et al., 2008]



Example: Illustrative Object Diagram [Schumann et al., 2008]





OCL Consistency

OCL Satisfaction Relation

In the following, $\mathscr{S}$ denotes a signature and $\mathscr{D}$ a structure of $\mathscr{S}$.

Definition (Satisfaction Relation).
Let φ be an OCL constraint over $\mathscr{S}$ and $\sigma \in \Sigma_{\mathscr{S}}^{\mathscr{D}}$ a system state.
We write

- $\sigma \models \varphi$ if and only if $I[\![\varphi]\!](\sigma, \emptyset) = \mathit{true}$.
- $\sigma \not\models \varphi$ if and only if $I[\![\varphi]\!](\sigma, \emptyset) = \mathit{false}$.

Note: In general we **can't** conclude from $\neg(\sigma \models \varphi)$ to $\sigma \not\models \varphi$ or vice versa.

Object Diagrams and OCL

- Let G be an object diagram of signature $\mathscr{S}$ wrt. structure $\mathscr{D}$.
  Let $expr$ be an OCL expression over $\mathscr{S}$.

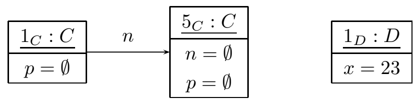
  We say G **satisfies** $expr$, denoted by $G \models expr$, if and only if

  $$\forall \sigma \in G^{-1} : \sigma \models expr.$$

- If G is **complete**, we can also talk about "$\not\models$".

  (Otherwise better not to avoid confusion: G^{-1} could comprise different system states in which $expr$ evaluates to *true*, *false*, and $\perp$.)

- **Example**: (complete — what if not complete wrt. object/attribute/both?)



  - context C inv : n -> isEmpty() $\rightsquigarrow$ ~~true~~ false
  - context C inv : p . n -> isEmpty() $\rightsquigarrow$ $\perp_{Bool}$
  - context D inv : $x \neq 0$ $\rightsquigarrow$ true

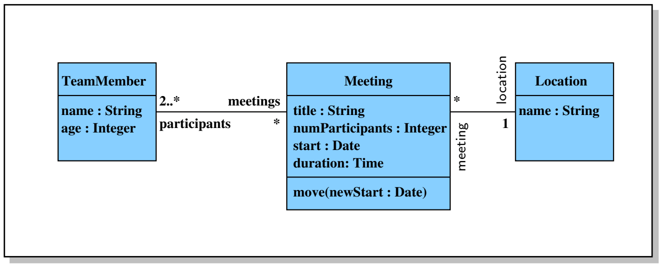
OCL Consistency

Definition (Consistency). A set *Inv* $= \{\varphi_1, \dots, \varphi_n\}$ of OCL constraints over $\mathscr{S}$ is called consistent (or satisfiable) if and only if there exists a system state of $\mathscr{S}$ wrt. $\mathscr{D}$ which satisfies all of them, i.e. if

$$\exists \sigma \in \Sigma_{\mathscr{S}}^{\mathscr{D}} : \sigma \models \varphi_1 \wedge \ldots \wedge \sigma \models \varphi_n$$

and inconsistent (or unrealizable) otherwise.

OCL Inconsistency Example



((C) Prof. Dr. P. Thiemann, http://proglang.informatik.uni-freiburg.de/teaching/swt/2008/)

- context $Location$ inv :
  $name =$ 'Lobby' implies $meeting$ -> $isEmpty()$

- context $Meeting$ inv :
  $title =$ 'Reception' implies $location\ .\ name =$ "$Lobby$"

- allInstances$_{Meeting}$ -> exists($w : Meeting \mid w\ .\ title =$ 'Reception')

Deciding OCL Consistency

- Whether a set of OCL constraints is satisfiable or not is **in general not as obvious** as in the made-up example.

- **Wanted**: A procedure which decides the OCL satisfiability problem.

- **Unfortunately**: in general **undecidable**.

  Otherwise we could, for instance, solve **diophantine equations**

  $$c_1 x_1^{n_1} + \cdots + c_m x_m^{n_m} = d.$$

  constants

  Encoding in OCL:

  allInstances$_{\mathsf{C}}$ -> exists($w : C \mid c_1 * w.x_1^{n_1} + \cdots + c_m * w.x_m^{n_m} = d$).

Deciding OCL Consistency

- Whether a set of OCL constraints is satisfiable or not is **in general not as obvious** as in the made-up example.
- **Wanted**: A procedure which decides the OCL satisfiability problem.
- **Unfortunately**: in general **undecidable**.

  Otherwise we could, for instance, solve **diophantine equations**

  $$c_1 x_1^{n_1} + \cdots + c_m x_m^{n_m} = d.$$

  Encoding in OCL:

  allInstances$_C$ -> exists$(w : C \mid c_1 * w.x_1^{n_1} + \cdots + c_m * w.x_m^{n_m} = d)$.

- **And now**? Options: [Cabot and Clarisó, 2008]
  - Constrain OCL, use a **less rich** fragment of OCL.
  - Revert to **finite domains** — basic types vs. number of objects.

OCL Critique

- **Expressive Power**:
  - "Pure OCL expressions only compute primitive recursive functions, but not recursive functions in general." [Cengarle and Knapp, 2001]
  - **Evolution over Time**: "finally $self.x > 0$"

    Proposals for fixes e.g. [Flake and Müller, 2003]. (Or: sequence diagrams.)
  - **Real-Time**: "Objects respond within 10s"

    Proposals for fixes e.g. [Cengarle and Knapp, 2002]
  - **Reachability**: "After insert operation, node shall be reachable."

    Fix: add transitive closure.
- **Concrete Syntax**
  "The syntax of OCL has been criticized – e.g., by the authors of Catalysis [...] – for being hard to read and write.
  - OCL's expressions are stacked in the style of Smalltalk, which makes it hard to see the scope of quantified variables.
  - Navigations are applied to atoms and not sets of atoms, although there is a collect operation that maps a function over a set.
  - Attributes, [...], are partial functions in OCL, and result in expressions with undefined value." [Jackson, 2002]

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, 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). Traceviewer technical documentation, version 1.0. Technical report, Carl von Ossietzky Universität Oldenburg und OFFIS.