

NOTE: next lecture, Mon 11.11.,
in the "old" room
51-0-0034

Software Design, Modelling and Analysis in UML

Lecture 05: Object Diagrams, OCL Consistency

2013-11-06

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

Albert-Ludwigs-Universität Freiburg, Germany

Contents & Goals

Last Lecture:

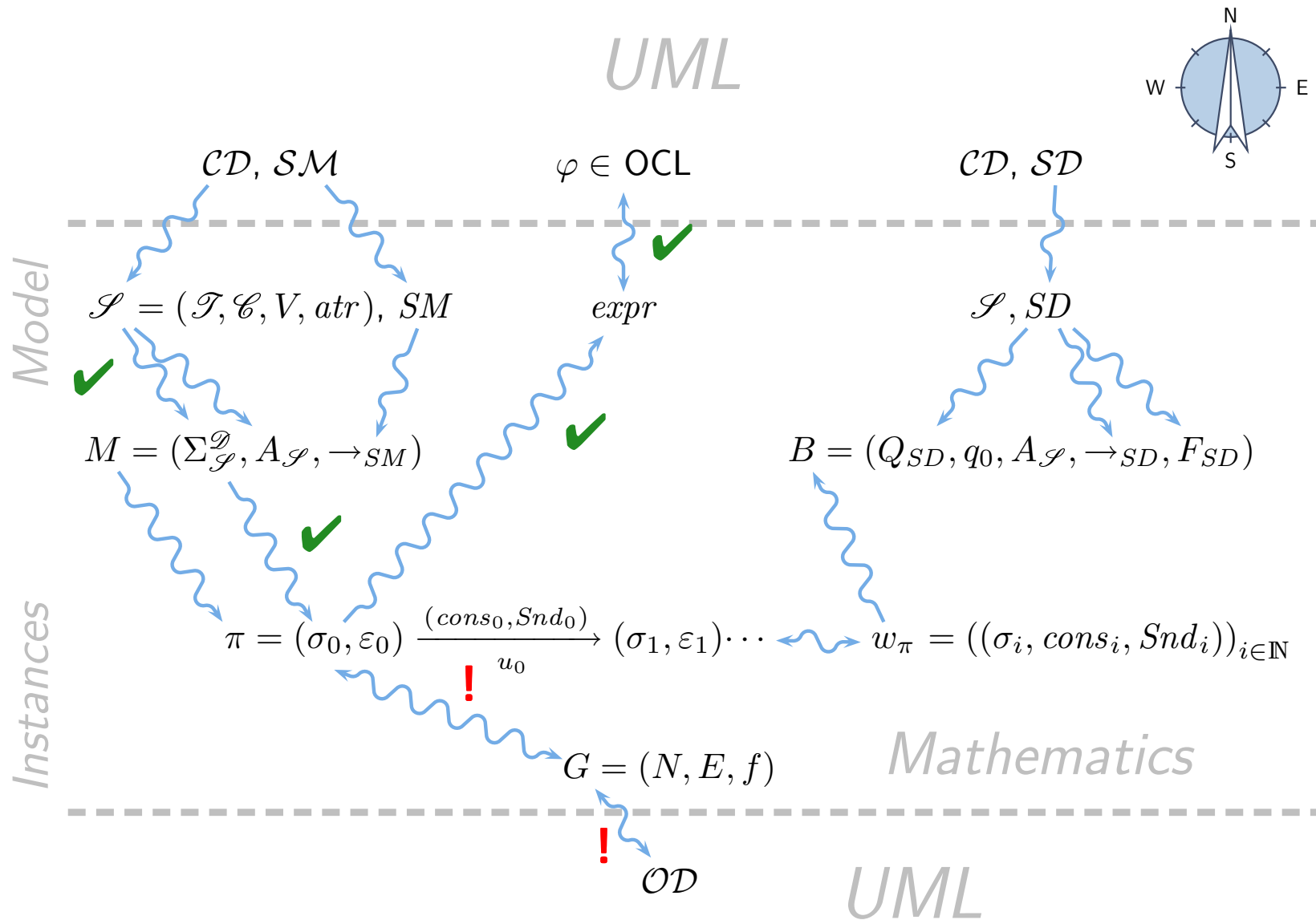
- OCL Semantics

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

Definition. A node labelled **graph** is a triple

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

consisting of

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

Object Diagrams

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

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

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

$N \subset \mathcal{D}(\mathcal{C})$ finite, $=: V_{0,1,*}$

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

source object is alive in σ

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

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

source refers to the destination via r

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

$\alpha: \textcircled{X}$

labelling of u is consistent with σ , we may leave out some attributes is called object diagram of σ .

$$X = \{X\} \dot{\cup} (V \rightarrow (\mathcal{D}(\mathcal{I}) \cup \mathcal{D}(\mathcal{C}_*)))$$

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

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

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

Graphical Representation of Object Diagrams

$$N \subset \mathcal{D}(\mathcal{C}) \text{ finite, } E \subset N \times V_{0,1,*} \times N, \quad X = \{X\} \dot{\cup} (V \rightarrow (\mathcal{D}(\mathcal{T}) \cup \mathcal{D}(\mathcal{C}_*)))$$

$$u_1 \in \text{dom}(\sigma) \wedge u_2 \in \sigma(u_1)(r), \quad f(u) \subseteq \sigma(u) \text{ or } f(u) = \{X\}$$

- Assume $\mathcal{S} = (\{Int\}, \{C\}, \{v_1 : Int, v_2 : Int, r : C_*\}, \{C \mapsto \{v_1, v_2, r\}\})$.

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

$$= (\underbrace{\{u_1, u_2\}}_N, \underbrace{\{(u_1, r, u_2)\}}_E, \underbrace{f: N \rightarrow X}_{f: N \rightarrow X})$$

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

is an object diagram of σ wrt. \mathcal{S} and any \mathcal{D} with $\mathcal{D}(Int) \supseteq \{1, 2, 3, 4\}$.

$$G_1 = (\underbrace{\{u_1, u_3\}}_{u_3 \notin \text{dom}(\sigma)}, \emptyset, \{u_3 \mapsto X, u_1 \mapsto \{v_1 \mapsto 1\}\})$$

Graphical Representation of Object Diagrams

$$N \subset \mathcal{D}(\mathcal{C}) \text{ finite, } E \subset N \times V_{0,1,*} \times N, \quad X = \{\mathbf{X}\} \dot{\cup} (V \rightarrow (\mathcal{D}(\mathcal{T}) \cup \mathcal{D}(\mathcal{C}_*)))$$

$$u_1 \in \text{dom}(\sigma) \wedge u_2 \in \sigma(u_1)(r), \quad f(u) \subseteq \sigma(u) \text{ or } f(u) = \{\mathbf{X}\}$$

- Assume $\mathcal{S} = (\{Int\}, \{C\}, \{v_1 : Int, v_2 : Int, r : C_*\}, \{C \mapsto \{v_1, v_2, r\}\})$.

- Consider

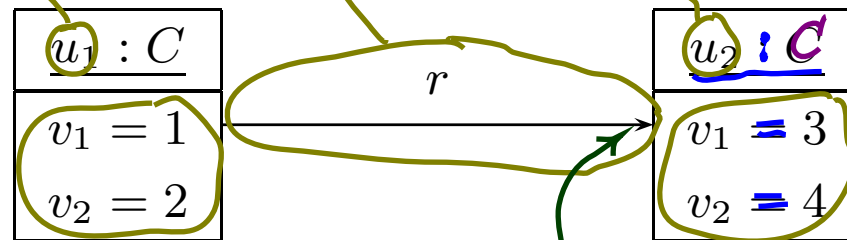
$$\sigma = \{u_1 \mapsto \{v_1 \mapsto 1, v_2 \mapsto 2, r \mapsto \{u_2\}\}, u_2 \mapsto \{v_1 \mapsto 3, v_2 \mapsto 4, r \mapsto \emptyset\}\}$$

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

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

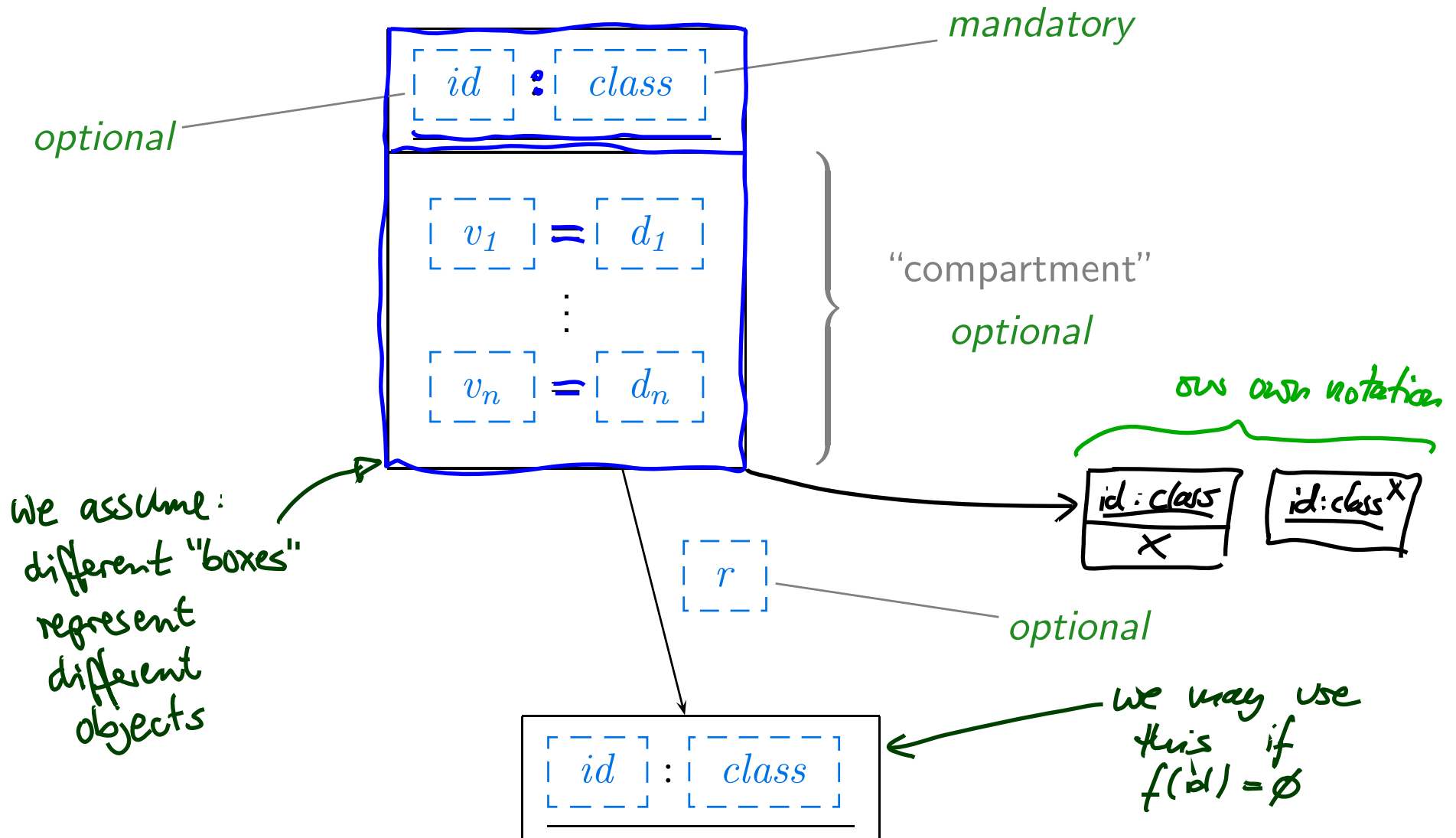
is an object diagram of σ wrt. \mathcal{S} and any \mathcal{D} with $\mathcal{D}(Int) \supseteq \{1, 2, 3, 4\}$.

- We may equivalently (!) **represent** G graphically as follows:



points
to
destination

UML Notation for Object Diagrams



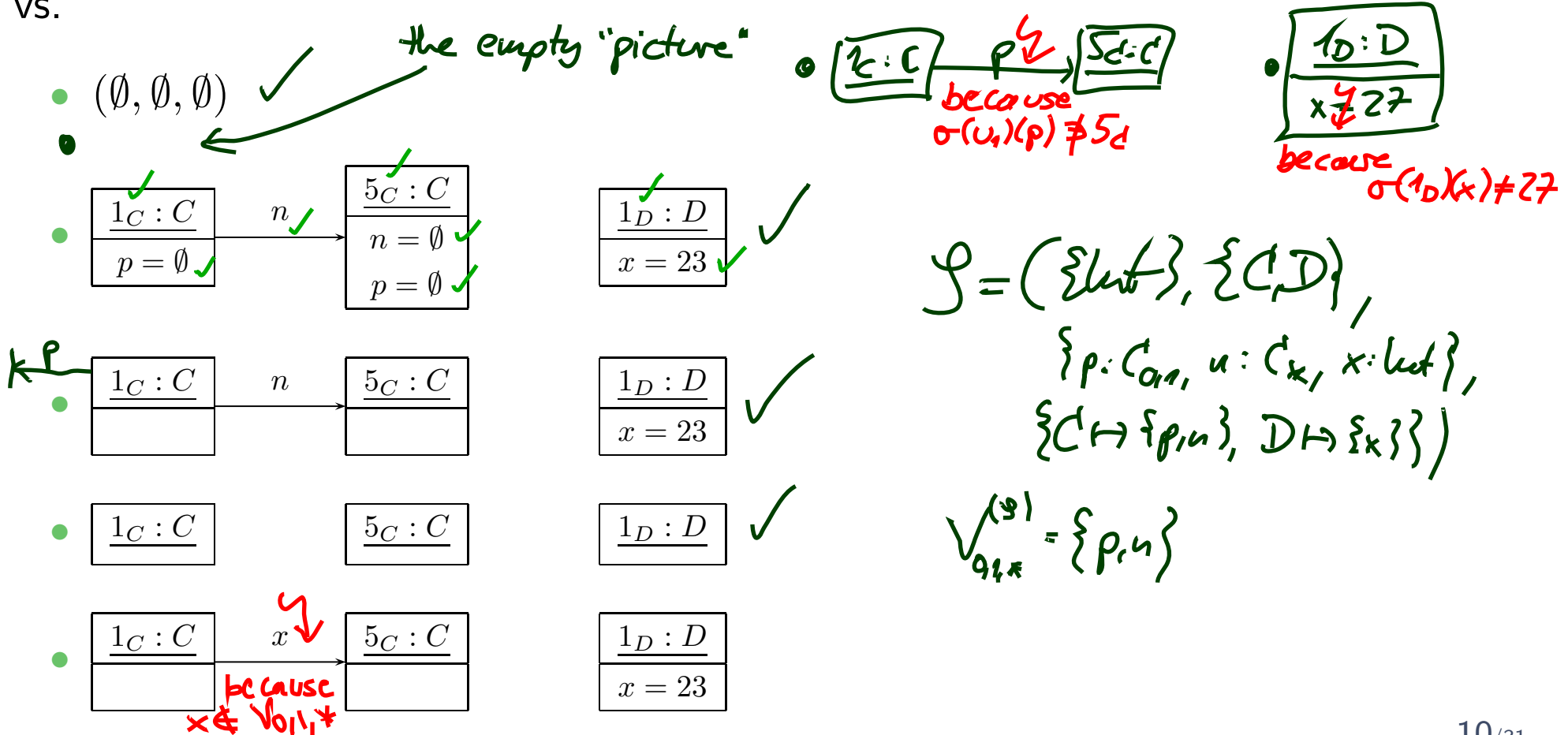
Object Diagrams: More Examples

$$N \subset \mathcal{D}(\mathcal{C}) \text{ finite, } E \subset N \times \underline{V_{0,1,*}} \times N, \quad X = \{X\} \dot{\cup} (V \rightarrow (\mathcal{D}(\mathcal{I}) \cup \mathcal{D}(\mathcal{C}_*)))$$

$$u_1 \in \text{dom}(\sigma) \wedge u_2 \in \sigma(u_1)(r), \quad f(u) \subseteq \sigma(u) \text{ or } f(u) = \{X\}$$

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

VS.



Complete vs. Partial Object Diagram

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

We call G **complete** wrt. σ if and only if

- G is **object complete**, i.e.
 - G consists of all alive objects, i.e. $N = \text{dom}(\sigma)$,
- 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 $u_1, u_2 \in \text{dom}(\sigma)$ and $r \in V$, then $(u_1, r, u_2) \in E$, and
 - each node is labelled with the values of all \mathcal{I} -typed attributes, i.e. for each $u \in \text{dom}(\sigma)$,

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

$$\text{where } V_{\mathcal{I}} := \{v : \tau \in V \mid \tau \in \mathcal{I}\}.$$

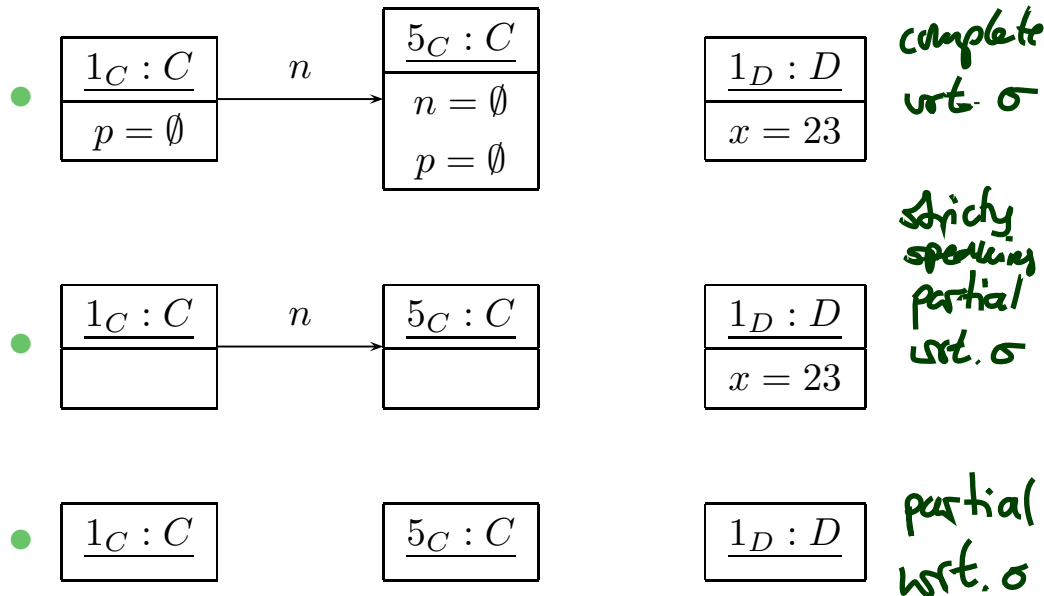
Otherwise we call G **partial**.

Complete vs. Partial Examples

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

Complete or partial?

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



o $\sigma_1 \in \Sigma_{\mathcal{F}}^{\mathcal{D}}$, $3_D \in \text{dom}(\sigma_1)$

•

$3_D : D$
$x = 13$

don't know wrt. σ_1

•

$3_D : D$

partial wrt. σ_1

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

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

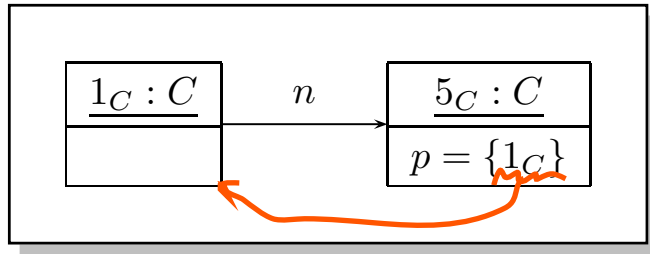
- **Observation:** If somebody **tells us**, that a given (consistent) object diagram G is **complete**, we can uniquely reconstruct the corresponding system state.

In other words: G^{-1} is then a singleton.

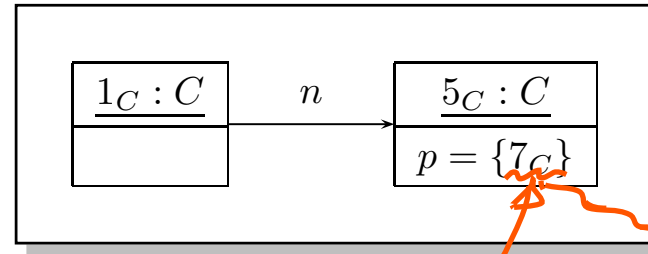
Corner Cases

Closed Object Diagrams vs. Dangling References

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



closed



dangling reference

7_C not alive

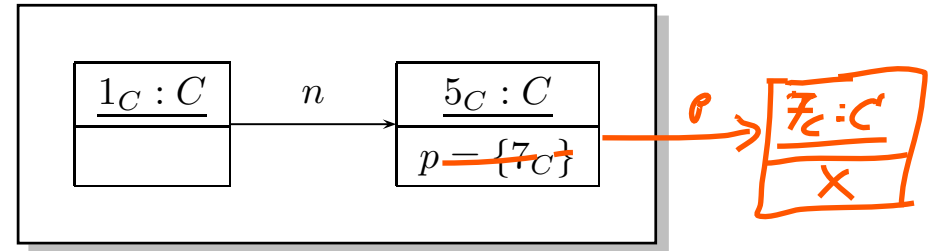
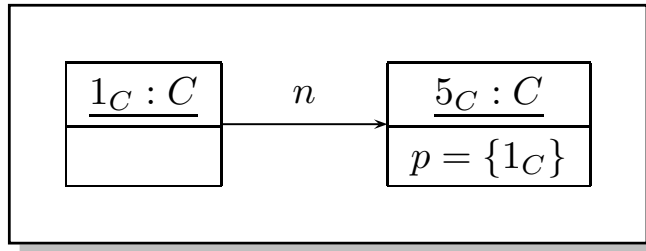
Definition. Let σ be a system state. We say attribute $v \in V_{0,1,*}$ has a **dangling reference** in object $u \in \text{dom}(\sigma)$ if and only if the attribute's value comprises an object which is not alive in σ , i.e. if

$$\sigma(u)(v) \notin \text{dom}(\sigma).$$

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

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 $u \in \text{dom}(\sigma)$ if and only if the attribute's value comprises an object which is not alive in σ , i.e. if

$$\sigma(u)(v) \notin \text{dom}(\sigma).$$

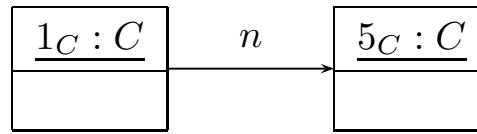
We call σ **closed** if and only if no attribute has a dangling reference in any object alive in σ .

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

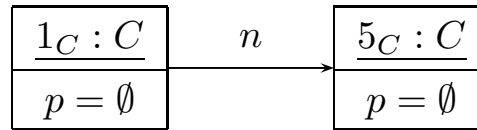
Special Notation

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

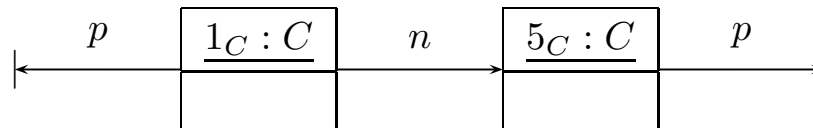
- Instead of



we want to write



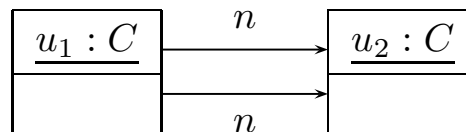
or



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

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



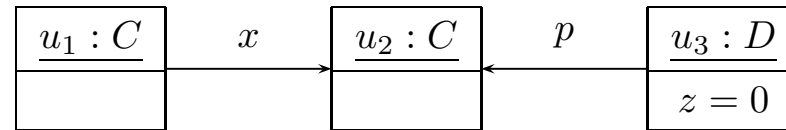
(This is not an object diagram in the sense of our definition because of the requirement on the edges E . Extension is straightforward but tedious.)

- 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

The Other Way Round

- If we **only** have a picture as below, we typically assume that it's **meant to be** an object diagram wrt. **some** signature and structure.



- In the example, we can conclude (by “**good will**”) that the author is referring to **some** signature $\mathcal{S} = (\mathcal{I}, \mathcal{C}, V, atr)$ with at least

- $\{c.D\} \subseteq \mathcal{C}$
- $T \in \mathcal{I}$
- $\{x : C_x, z : T, p : C_x\}$
- $\{x\} \subseteq atr(C)$
- $\{p, z\} \subseteq atr(D)$

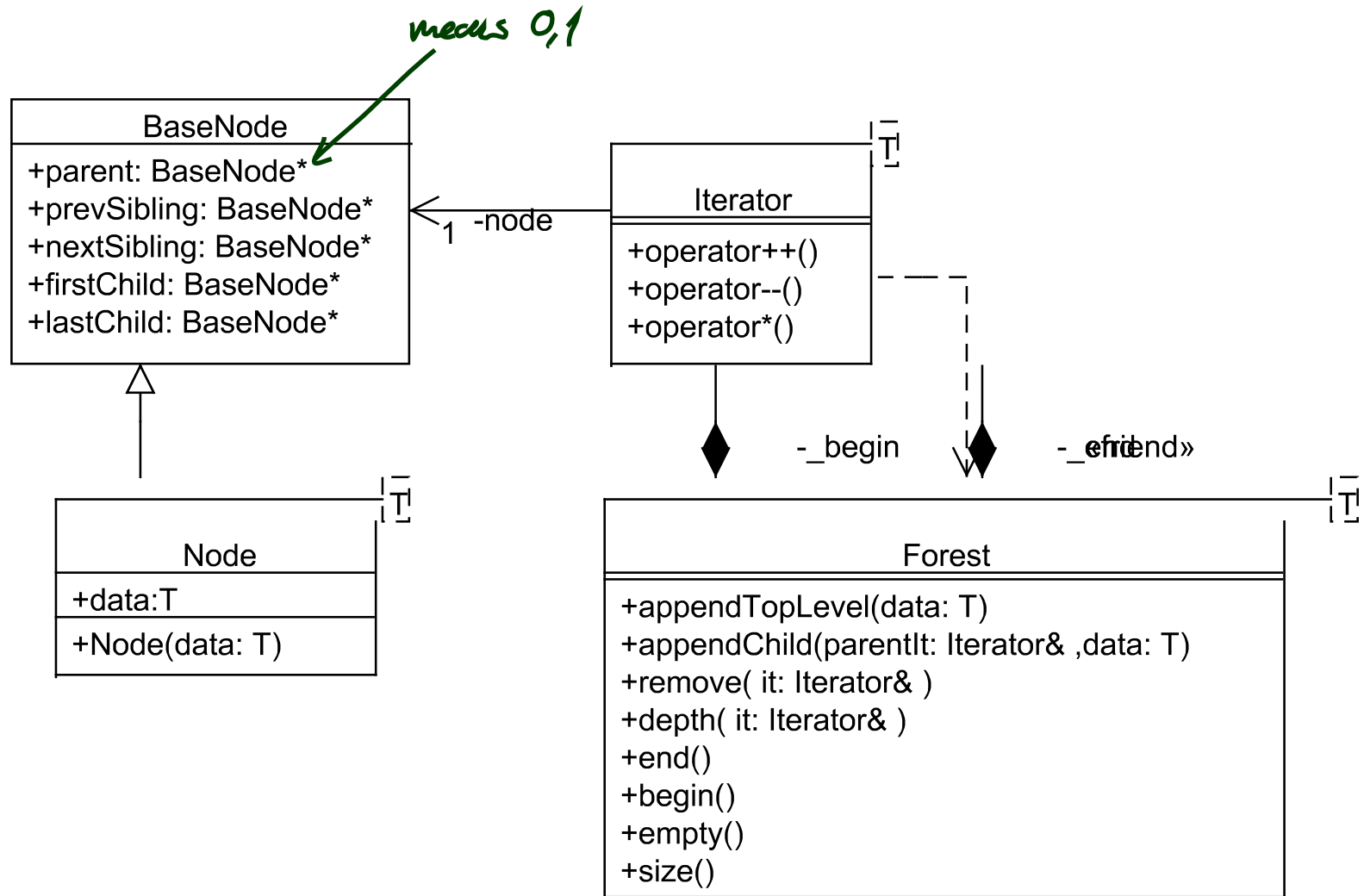
$$\forall a \in \mathbb{N}_0 \cdot \sigma(u_3)(z) \leq a$$

and a structure with

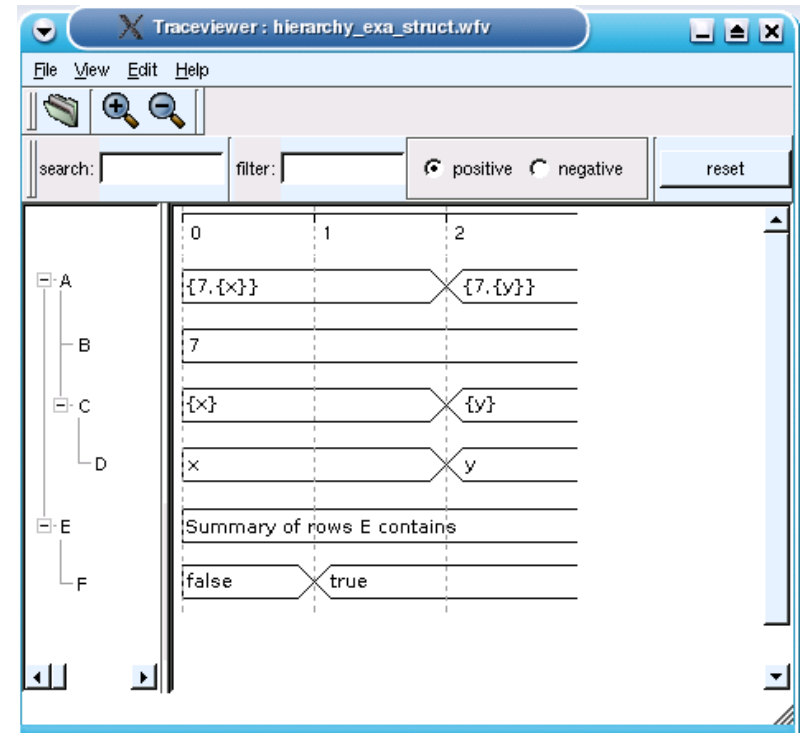
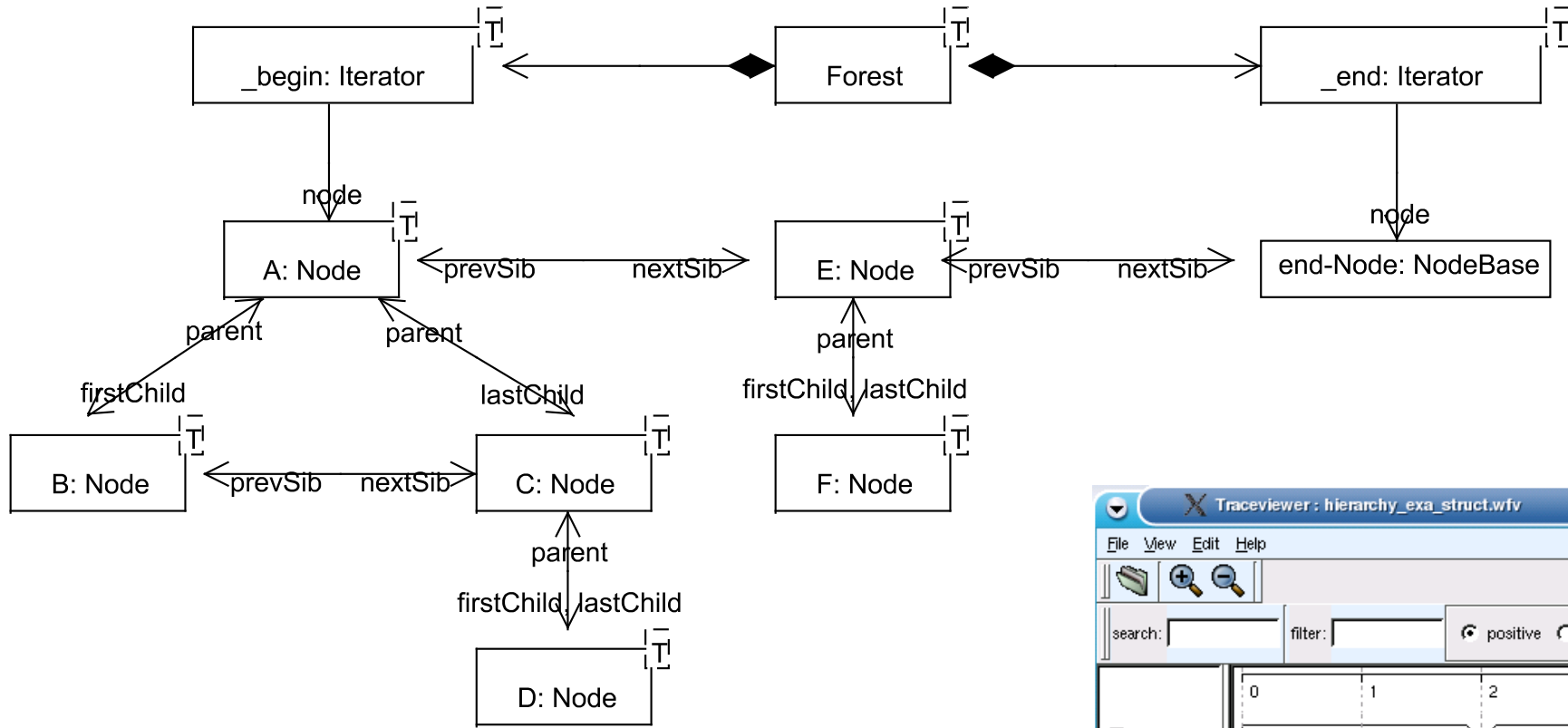
- $\{v_1, v_2\} \subseteq \mathcal{D}(C)$
- $v_3 \in \mathcal{D}(D)$
- $0 \in \mathcal{D}(T)$

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, \mathcal{S} denotes a signature and \mathcal{D} a structure of \mathcal{S} .

Definition (Satisfaction Relation).

Let φ be an OCL constraint over \mathcal{S} and $\sigma \in \Sigma_{\mathcal{D}}^{\mathcal{S}}$ a system state.

We write

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

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 \mathcal{S} wrt. structure \mathcal{D} .
Let $expr$ be an OCL expression over \mathcal{S} .

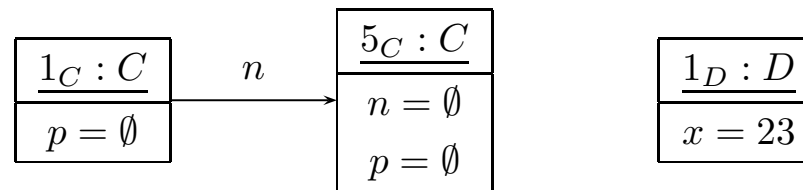
We say G **satisfies** $expr$, denoted by $G \models expr$, if and only if

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

- 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?)



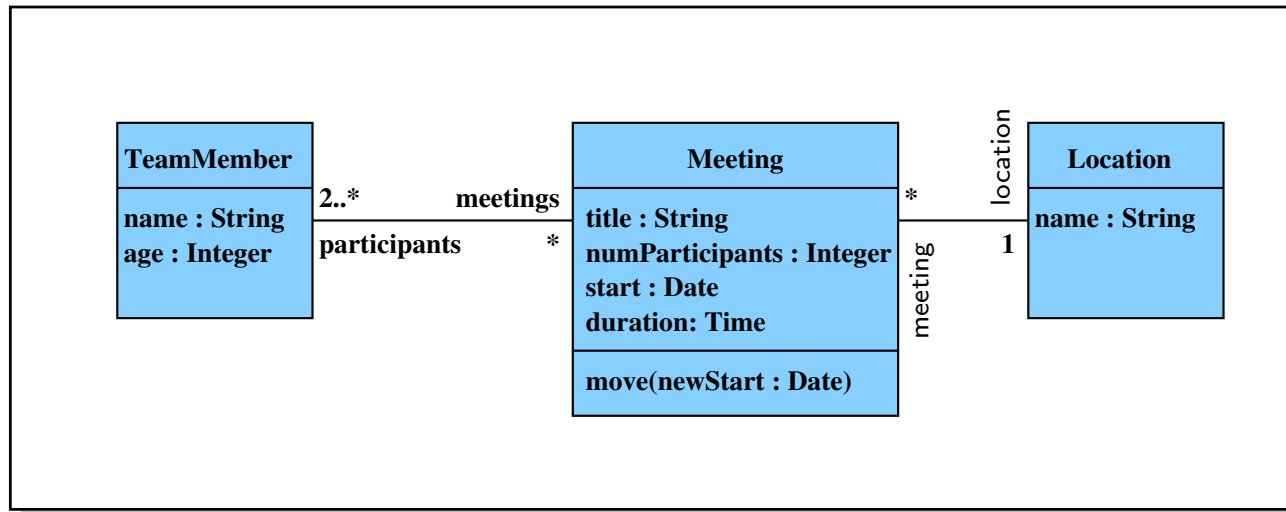
- context C inv : $n \rightarrow \text{isEmpty}()$
- context C inv : $p.n \rightarrow \text{isEmpty}()$
- context D inv : $x \neq 0$

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

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

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 :
 $name = 'Lobby' \text{ implies } meeting \rightarrow isEmpty()$
- context *Meeting* inv :
 $title = 'Reception' \text{ implies } location . name = "Lobby"$
- $allInstances_{Meeting} \rightarrow exists(w : Meeting \mid w . title = 'Reception')$

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**

$$c_1x_1^{n_1} + \dots + c_mx_m^{n_m} = d.$$

Encoding in OCL:

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

attributes of C

- **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.

- **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.