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?
\[ G = (N, E, f) \]

\[ \mathcal{J} = (\mathcal{I}, \mathcal{C}, V, \text{atr}), \text{SM} \]

\[ B = (Q_{SD}, q_0, A_{\mathcal{I}}, \rightarrow_{SD}, F_{SD}) \]

\[ (\Sigma_{\mathcal{I}}, A_{\mathcal{I}}, \rightarrow_{SM}) = M \]

\[ \varphi \in \text{OCL} \]

\[ CD, SM, \varphi \in OCL, CD, SD \]

\[ (\sigma_0, \varepsilon_0) \xrightarrow{(\text{cons}_0, \text{Snd}_0)} (\sigma_1, \varepsilon_1) \xrightarrow{(\text{cons}_1, \text{Snd}_1)} \ldots \]

\[ CD, SD \]

\[ \mathcal{I}, SD \]

\[ \sigma_0, \varepsilon_0 \]

\[ \sigma_1, \varepsilon_1 \]
Object Diagrams
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 \( \mathcal{D} \) be a structure of signature \( \mathcal{J} = (\mathcal{T}, \mathcal{C}, V, \text{atr}) \) and \( \sigma \in \Sigma^\mathcal{D} \) a system state.

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

- nodes are identities (not necessarily alive), i.e.
  \( N \subset D(\mathcal{C}) \) finite,

- edges correspond to “links” of objects, i.e.
  \[ E \subseteq N \times \{v : \tau \in V \mid \tau \in \{C_{0,1}, C_\ast \mid C \in \mathcal{C}\}\} \times N, \]
  \( \forall (u_1, r, u_2) \in E : u_1 \in \text{dom}(\sigma) \land u_2 \in \sigma(u_1)(r), \)

- objects are labelled with attribute valuations and non-alive identities marked with “\( X \)”, i.e.
  \[ X = \{X\} \cup (V \rightarrow (D(\mathcal{T}) \cup D(\mathcal{C}_\ast))) \]
  \( \forall u \in N \cap \text{dom}(\sigma) : f(u) \subseteq \sigma(u) \)
  \( \forall u \in N \setminus \text{dom}(\sigma) : f(u) = \{X\} \)

is called **object diagram** of \( \sigma \).
Graphical Representation of Object Diagrams

\[ N \subset \mathcal{D}(C) \text{ finite, } E \subset N \times V_{0,1,*} \times N, \quad X = \{X\} \cup (V \mapsto (\mathcal{D}(T) \cup \mathcal{D}(C_*))) \]

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

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

- 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 \( \sigma \) wrt. \( \mathcal{I} \) and any \( \mathcal{D} \) with \( \mathcal{D}(\text{Int}) \supseteq \{1, 2, 3, 4\} \).

- \( \mathcal{G}_1 = (\emptyset, \emptyset, \emptyset) \)

- \( \mathcal{G}_2 = (\{u_1\}, \emptyset, \{u_1 \mapsto \{v_2 \mapsto 2\}\}) \)

- \( \mathcal{G}_3 = (\emptyset, \emptyset, \emptyset) \)

- \( \mathcal{G}_4 = (\{u_1, v_2\}, \emptyset, \emptyset) \)

- \( \mathcal{G}_5 = (\emptyset, \{v_1, v_2\}, \emptyset) \)

- \( \mathcal{G}_6 = (\emptyset, \emptyset, \emptyset) \)

- \( \mathcal{G}_7 = (\emptyset, \emptyset, \emptyset) \)

- \( u_2 \) is not alive.
\( N \subset \mathcal{D}(\mathcal{C}) \) finite, \( E \subset N \times V_{0,1,*} \times N \), \( X = \{X\} \cup (V \twoheadrightarrow (\mathcal{D}(\mathcal{I}) \cup \mathcal{D}(\mathcal{C}_*)) ) \)

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

- Assume \( \mathcal{I} = (\{\text{Int}\}, \{C\}, \{v_1 : \text{Int}, v_2 : \text{Int}, r : C_*\}, \{C \mapsto \{v_1, v_2, r\}\}) \).
- 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 \( \sigma \) wrt. \( \mathcal{I} \) and any \( \mathcal{D} \) with \( \mathcal{D}(\text{Int}) \supseteq \{1, 2, 3, 4\} \).
- We may equivalently (!) represent \( G \) graphically as follows:
UML Notation for Object Diagrams

(id : class)

\[ v_1 = [d_1] \]
\[ \vdots \]
\[ v_n = [d_n] \]

\[ [r] \]

mandatory

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

"compartment"

\[ \text{if } f(x) = \emptyset \]
\[ \text{or } f(x) = \Sigma x \]

optional

\[ \text{As we have} \]
Object Diagrams: More Examples

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

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

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

vs.

\[ \Psi = (\{\text{null}, \{C,D\}, \{x: \text{null}\}
\quad p:D_{0,7}, n:C_{0,7},
\quad C \leftarrow \text{null}, P, D \rightarrow 5_{x,3}\}) \]

1. \((\emptyset, \emptyset, \emptyset) \checkmark \checkmark \]
2. \[
\begin{array}{ccc}
  1_C : C & n & 5_C : C \\
  p = \emptyset & n = \emptyset & p = \emptyset \\
  & & \checkmark
\end{array}
\]
3. \[
\begin{array}{ccc}
  1_C : C & n & 5_C : C \\
  & & \checkmark
\end{array}
\]
4. \[
\begin{array}{ccc}
  1_C : C & 5_C : C & 1_D : D \\
  & x = 23 \checkmark
\end{array}
\]
5. \[
\begin{array}{ccc}
  1_C : C & 5_C : C & 1_D : D \\
  x & x = 23 \checkmark
\end{array}
\]
6. \[
\begin{array}{ccc}
  1_C : C & 5_C : C & 1_D : D \\
  x & x = 23 \checkmark
\end{array}
\]

7. \((\emptyset, \emptyset, \emptyset) \checkmark \checkmark \]

8. \[
\begin{array}{ccc}
  1_C : C & 5_C : C & 1_D : D \\
  x & x = 23 \checkmark
\end{array}
\]

9. \((\emptyset, \emptyset, \emptyset) \checkmark \checkmark \]

10. \[
\begin{array}{ccc}
  1_C : C & 5_C : C & 1_D : D \\
  x & x = 23 \checkmark
\end{array}
\]

11. \((\emptyset, \emptyset, \emptyset) \checkmark \checkmark \]

12. \[
\begin{array}{ccc}
  1_C : C & 5_C : C & 1_D : D \\
  x & x = 23 \checkmark
\end{array}
\]

13. \((\emptyset, \emptyset, \emptyset) \checkmark \checkmark \]

14. \[
\begin{array}{ccc}
  1_C : C & 5_C : C & 1_D : D \\
  x & x = 23 \checkmark
\end{array}
\]

15. \((\emptyset, \emptyset, \emptyset) \checkmark \checkmark \]

16. \[
\begin{array}{ccc}
  1_C : C & 5_C : C & 1_D : D \\
  x & x = 23 \checkmark
\end{array}
\]

17. \((\emptyset, \emptyset, \emptyset) \checkmark \checkmark \]

18. \[
\begin{array}{ccc}
  1_C : C & 5_C : C & 1_D : D \\
  x & x = 23 \checkmark
\end{array}
\]

19. \((\emptyset, \emptyset, \emptyset) \checkmark \checkmark \]

20. \[
\begin{array}{ccc}
  1_C : C & 5_C : C & 1_D : D \\
  x & x = 23 \checkmark
\end{array}
\]

21. \((\emptyset, \emptyset, \emptyset) \checkmark \checkmark \]

22. \[
\begin{array}{ccc}
  1_C : C & 5_C : C & 1_D : D \\
  x & x = 23 \checkmark
\end{array}
\]

23. \((\emptyset, \emptyset, \emptyset) \checkmark \checkmark \]

24. \[
\begin{array}{ccc}
  1_C : C & 5_C : C & 1_D : D \\
  x & x = 23 \checkmark
\end{array}
\]

25. \((\emptyset, \emptyset, \emptyset) \checkmark \checkmark \]

26. \[
\begin{array}{ccc}
  1_C : C & 5_C : C & 1_D : D \\
  x & x = 23 \checkmark
\end{array}
\]

27. \((\emptyset, \emptyset, \emptyset) \checkmark \checkmark \]

28. \[
\begin{array}{ccc}
  1_C : C & 5_C : C & 1_D : D \\
  x & x = 23 \checkmark
\end{array}
\]

29. \((\emptyset, \emptyset, \emptyset) \checkmark \checkmark \]

30. \[
\begin{array}{ccc}
  1_C : C & 5_C : C & 1_D : D \\
  x & x = 23 \checkmark
\end{array}
\]
**Definition.** Let $G = (N, E, f)$ be an object diagram of system state $\sigma \in \Sigma \mathcal{F}$.

We call $G$ **complete** wrt. $\sigma$ if and only if

- $G$ is **object complete**, i.e.
  - $G$ comprises all alive objects, i.e. $N \supseteq \text{dom}(\sigma)$,

- $G$ 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 $\mathcal{T}$-typed attributes, i.e. for each $u \in \text{dom}(\sigma)$,
    $$f(u) = \sigma(u)|_{V_{\mathcal{F}}} \cup \{r \mapsto (\sigma(u)(r)\setminus N) \mid r \in V : \sigma(u)(r)\setminus N \neq \emptyset\}$$
    where $V_{\mathcal{F}} := \{v : \tau \in V \mid \tau \in \mathcal{T}\}$.

Otherwise we call $G$ **partial**.
**Complete vs. Partial Examples**

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

**Complete or partial?**  (wrt. system state $\sigma$)

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

- $1_C : C$
  - $p = \emptyset$
- $5_C : C$
  - $n = \emptyset$
  - $p = \emptyset$
- $1_D : D$
  - $x = 23$

$\sigma' = \sigma \cup \{23 \mapsto 5 \times 70\}$

- **Object complete**
- **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
  \[
  G^{-1} := \{ \sigma \in \Sigma^G \mid G \text{ is an object diagram of } \sigma \}
  \]

- **Observation:** If somebody **tells us**, that a given object diagram $G$ is **complete**, we can uniquely reconstruct the corresponding system state. In other words: $G^{-1}$ is then a singleton.
Corner Cases
Find the 10 differences! (Both diagrams shall be complete.)

Definition. Let $\sigma$ 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 $\sigma$, i.e. if

$$\sigma(u)(v) \not\subset \text{dom}(\sigma).$$

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

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

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

- Instead of

\[
1_C : C \xrightarrow{n} 5_C : C
\]

we want to write

\[
\frac{1_C : C}{p = \emptyset} \xrightarrow{n} \frac{5_C : C}{p = \emptyset}
\]

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

\[ u_1 : C \xrightarrow{n} u_2 : C \]

(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: Illustrative Object Diagram [Schumann et al., 2008]
OCL Consistency
In the following, $\mathcal{S}$ denotes a signature and $\mathcal{D}$ a structure of $\mathcal{S}$.

**Definition (Satisfaction Relation).**
Let $\varphi$ be an OCL constraint over $\mathcal{S}$ and $\sigma \in \Sigma_{\mathcal{D}}$ a system state. We write
- $\sigma \models \varphi$ if and only if $I[\varphi](\sigma, \emptyset) = true$.
- $\sigma \not\models \varphi$ if and only if $I[\varphi](\sigma, \emptyset) = false$.

**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?)

\[
\begin{align*}
1_C : C & \quad n \quad 5_C : C \\
p = \emptyset & \quad n = \emptyset & \quad x = 23 \\
5_C : C & \quad n = \emptyset & \quad p = \emptyset \\
\end{align*}
\]

- context $C$ inv : $n \rightarrow $isEmpty() $\not\models false$
- context $C$ inv : $p \cdot n \rightarrow $isEmpty() $\not\models \bot$
- context $D$ inv : $x \neq 0 \models true$
**Definition (Consistency).** A set $\textit{Inv} = \{\varphi_1, \ldots, \varphi_n\}$ of OCL constraints over $\mathcal{I}$ is called consistent (or satisfiable) if and only if there exists a system state of $\mathcal{I}$ wrt. $\mathcal{B}$ which satisfies all of them, i.e. if

$$\exists \sigma \in \Sigma^\mathcal{B} : \sigma \models \varphi_1 \land \ldots \land \sigma \models \varphi_n$$

and inconsistent (or unrealizable) otherwise.
OCL Inconsistency Example

- context Location inv:
  
  \( \text{name} = '\text{Lobby}' \) implies \( \text{meeting} \rightarrow \text{isEmpty()} \)

- context Meeting inv:
  
  \( \text{title} = '\text{Reception}' \) implies \( \text{location} \cdot \text{name} = "\text{Lobby}" \)

- allInstances_{Meeting} \rightarrow \text{exists}(w : Meeting \mid w \cdot \text{title} = '\text{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**

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

Encoding in OCL:

\[
\text{allInstances}_C \rightarrow \exists (w : C \mid c_1 \times w.x_1^{n_1} + \cdots + c_m \times w.x_m^{n_m} = d).
\]
**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 undecided.

  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:

  \[
  \text{allInstances}_C \to \exists (w : C \mid c_1 \ast w.x_1^{n_1} + \cdots + c_m \ast w.x_m^{n_m} = d).
  \]

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

  [Cabot and Clarisó, 2008]
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


