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?
\[ \mathcal{I} = (\mathcal{F}, \mathcal{E}, V, \text{atr}), \ SM \]
\[ M = (\Sigma^\mathcal{I}, A^\mathcal{I}, \rightarrow_{SM}) \]
\[ \pi = (\sigma_0, \varepsilon_0) \xrightarrow{(\text{cons}_0, \text{Snd}_0)} (\sigma_1, \varepsilon_1) \ldots \]
\[ \varphi \in \text{OCL} \]
\[ CD, SM \]
\[ \varphi \in \text{OCL} \]
\[ CD, SD \]
\[ w_\pi = ((\sigma_i, \text{cons}_i, \text{Snd}_i))_{i \in \mathbb{N}} \]
\[ G = (N, E, f) \]
\[ OD \]
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,
**Definition.** Let $\mathcal{D}$ be a structure of signature $\mathcal{S} = (\mathcal{T}, \mathcal{C}, V, \text{atr})$ and $\sigma \in \Sigma^{\mathcal{D}}$ a system state.

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

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

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

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

is called object diagram of $\sigma$. 

Graphical Representation of Object Diagrams

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

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

- 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 \( G = (N, E, f) \) is an object diagram of \( \sigma \) wrt. \( \mathcal{I} \) and any \( \mathcal{D} \) with \( \mathcal{D}(\text{Int}) \supseteq \{1, 2, 3, 4\} \).
Graphical Representation of Object Diagrams

\[
\begin{align*}
N & \subseteq \mathcal{D}(\mathcal{C}) \text{ finite, } \\
E & \subseteq N \times V_{0,1,*}\times N, \\
X & = \{X\} \cup (V \mapsto (\mathcal{D}(\mathcal{T}) \cup \mathcal{D}(\mathcal{C}_*)))) \\
u_1 & \in \text{dom}(\sigma) \land u_2 \in \sigma(u_1)(r), \\
f(u) & \subseteq \sigma(u) \text{ or } f(u) = \{X\}
\end{align*}
\]

- 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

- We assume: different “boxes” represent different objects.
- "compartment" is optional.
- The notation "id: class" indicates an object.
- An "x" can be used if there are no objects with the same ID.
Object Diagrams: More Examples

\[ N \subset \mathcal{D}(C) \text{ finite, } E \subset N \times V_{0,1,*} \times N, \quad X = \{X\} \cup (V \rightarrow (\mathcal{D}(F) \cup \mathcal{D}(C_*))) \]
\[ u_1 \in \text{dom}(\sigma) \land u_2 \in \sigma(u_1)(r), \quad f(u) \subseteq \sigma(u) \text{ or } f(u) = \{X\} \]

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

vs.

\[ \begin{array}{c}
(\emptyset, \emptyset, \emptyset) \\
\begin{array}{ccc}
1_C : C & n & 5_C : C \\
| & n = \emptyset & n = \emptyset \\
p = \emptyset & p = \emptyset
\end{array} & \begin{array}{c}
1_D : D \\
x = 23
\end{array} \\
\begin{array}{c}
5_C : C \\
\end{array} & \begin{array}{c}
1_D : D \\
x = 23
\end{array}
\end{array} \]

the empty "picture"

\[ \begin{array}{c}
1_C : C & n & 5_C : C \\
\begin{array}{c}
1_D : D \\
x = 23
\end{array} \\
\begin{array}{c}
5_C : C \\
\end{array}
\end{array} \]

\[ \begin{array}{c}
1_C : C & n & 5_C : C \\
\begin{array}{c}
1_D : D \\
x = 23
\end{array} \\
\begin{array}{c}
5_C : C \\
\end{array}
\end{array} \]

\[ \begin{array}{c}
1_C : C \\
\begin{array}{c}
1_D : D \\
x = 23
\end{array} \\
\begin{array}{c}
5_C : C \\
\end{array}
\end{array} \]

\[ \begin{array}{c}
1_C : C \\
\begin{array}{c}
1_D : D \\
x = 23
\end{array} \\
\begin{array}{c}
5_C : C \\
\end{array}
\end{array} \]

\[ S = (C \ni 1_C \ni D, \{C, D\}, \{p:C_{1,4}, u:C_{3,1}, x:u\ni t\}, \{C \ni f_{1,4}, D \ni f_{3,1}\}) \]

\[ \chi(x) = \{p, u\} \]

\[ \chi(x) = \{p, u\} \]
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. $\sigma$ 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 $\mathcal{T}$-typed attributes, i.e. for each $u \in \text{dom}(\sigma),
    $$f(u) = \sigma(u)|_{V_{\mathcal{T}}} \cup \{ r \mapsto (\sigma(u)(r) \setminus N) | r \in V : \sigma(u)(r) \setminus N \neq \emptyset \}$$

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

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

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

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

  $$G^{-1} := \{\sigma \in \Sigma^D | G \text{ is an object diagram of } \sigma\}$$

- **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.)

\begin{align*}
1_C & : C \\
\begin{array}{c}
\quad n \\
\end{array} & \text{closed} & \begin{array}{c}
5_C : C \\
p = \{1_C\}
\end{array}
\end{align*}

\begin{align*}
1_C & : C \\
\begin{array}{c}
\quad n \\
\end{array} & \text{closed} & \begin{array}{c}
5_C : C \\
p = \{7_C\}
\end{array}
\end{align*}

**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

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

We call $\sigma$ **closed** if and only if no attribute has a dangling reference in any object alive in $\sigma$. 
Closed Object Diagrams vs. Dangling References

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

\[
\sigma(u)(v) \not\subseteq \text{dom}(\sigma).
\]

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

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

- Instead of

\[
\begin{array}{c}
1_C : C \\
\downarrow n
\end{array}
\rightarrow
\begin{array}{c}
5_C : C \\
\downarrow n
\end{array}
\]

we want to write

\[
\begin{array}{c}
1_C : C \\
p = \emptyset
\end{array}
\rightarrow
\begin{array}{c}
5_C : C \\
p = \emptyset
\end{array}
\]

or

\[
\begin{array}{c}
p \\
1_C : C
\end{array}
\rightarrow
\begin{array}{c}
5_C : C \\
\downarrow n
\end{array}
\rightarrow
\begin{array}{c}
p
\end{array}
\]

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

![Diagram](image)

(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
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.

\[
\begin{array}{c}
\text{\( u_1 : C \)} & \text{\( x \)} & \text{\( u_2 : C \)} & \text{\( p \)} & \text{\( u_3 : D \)} \\
\end{array}
\]

\[
z = 0
\]

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

- \( \{ \mathcal{C}, \mathcal{D} \} \subseteq \mathcal{C} \)
- \( T \in \mathcal{I} \)
- \( \{ x : \mathcal{C}_x, x : T, p : \mathcal{C}_x \} \)
- \( \{ x \} \subseteq \text{atr} (\mathcal{C}) \)
- \( \{ p, z \} \subseteq \text{atr} (\mathcal{D}) \)

and a structure with

- \( \{ u_1, u_2 \} \subseteq \mathcal{D}(\mathcal{C}) \)
- \( u_3 \in \mathcal{D}(\mathcal{D}) \)
- \( 0 \in \mathcal{D}(\mathcal{T}) \)
Example: Object Diagrams for Documentation
Example: Data Structure [Schumann et al., 2008]

BaseNode
+parent: BaseNode*
+prevSibling: BaseNode*
+nextSibling: BaseNode*
+firstChild: BaseNode*
+lastChild: BaseNode*

Node
+data: T
+Node(data: T)

Forest
+appendTopLevel(data: T)
+appendChild(parentIt: Iterator&, data: T)
+remove(it: Iterator&)
+depth(it: Iterator&)
+end()
+begin()
+empty()
+size()

Iterator
+operator++()
+operator--()
+operator*()
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 $\varphi$ 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 $\mathcal{I}$ wrt. structure $\mathcal{D}$. Let $expr$ be an OCL expression over $\mathcal{I}$.

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?)

$$\begin{array}{c}
1_C : C \\
p = \emptyset
\end{array} \quad n \quad \begin{array}{c}
5_C : C \\
n = \emptyset \\
p = \emptyset
\end{array} \quad \begin{array}{c}
1_D : D \\
x = 23
\end{array}$$

- context $C$ inv : $n \rightarrow \text{isEmpty}()$
- context $C$ inv : $p \cdot n \rightarrow \text{isEmpty}()$
- context $D$ inv : $x \neq 0$
**Definition (Consistency).** A set \( \text{Inv} = \{\varphi_1, \ldots, \varphi_n\} \) of OCL constraints over \( \mathcal{I} \) is called consistent (or satisfiable) if and only if there exists a system state of \( \mathcal{I} \) wrt. \( \mathcal{D} \) which satisfies all of them, i.e. if

\[
\exists \sigma \in \Sigma_{\mathcal{D}} : \sigma \models \varphi_1 \land \ldots \land \sigma \models \varphi_n
\]

and inconsistent (or unrealizable) otherwise.
OCL Inconsistency Example

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

- **context** *Meeting* inv:
  
  \[ \text{title} = '\text{Reception}' \text{ implies } \text{location} \cdot \text{name} = "\text{Lobby}" \]

- **allInstances**\(_{\text{Meeting}}\) \rightarrow \text{exists}(w : \text{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:

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

- **And now? Options**: [Cabot and Clarísó, 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


