Software Design, Modelling and Analysis in UML

Lecture 05: OCL Semantics Cont’d, Object Diagrams

2014-11-06

Prof. Dr. Andreas Podelski, Dr. Bernd Westphal

Albert-Ludwigs-Universität Freiburg, Germany

Contents & Goals

Last Lecture:
- OCL Semantics (nearly complete)

This Lecture:
- **Educational Objectives:** Capabilities for following tasks/questions.
  - What does it mean that an OCL expression is satisfiable?
  - When is a set of OCL constraints said to be consistent?
  - 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)?
  - How are system states and object diagrams related?
  - Can you think of an object diagram which violates this OCL constraint?

- **Content:**
  - OCL: consistency, satisfiability
  - Object Diagrams
  - Example: Object Diagrams for Documentation
(vi) Putting It All Together

**OCL Syntax 1/4: Expressions**

Where, given \( \mathcal{J} = (\mathcal{F}, \mathcal{V}) \)

- \( W \models \{ \text{set of logical variables} \} \)
- \( \tau \) is a type from \( \mathcal{J} \)
- \( \{ \text{set of object types} \} \)
- \( \text{iterate}(\tau) \) denotes the set-of-\( \tau \) type for \( \mathcal{J} \)

**OCL Syntax 2/4: Constants, Arithmetical Operators**

For example:

**OCL Syntax 3/4: Iterate**

or, with a little meaning:

- \( \text{iterate}(\tau_1 \times \tau_2) = \text{iterate}(\tau_1) \times \text{iterate}(\tau_2) \)

**OCL Syntax 4/4: Context**

where \( w \in W \) and \( \tau_i \in T_{\mathcal{J}_i}, 1 \leq i \leq n, n \geq 0. \)

- \( \text{iterate}(\tau) \) is a collection type (here: a set \( \text{Set}(\tau_i) \) for some \( \tau_i \).)
(vi) Putting It All Together...

\[ \beta: W \rightarrow \bigcup \mathcal{I}(\tau) \]

- \( I[w](\sigma, \beta) := \beta(w) \) \quad \( I[\omega](\sigma, \beta) := \mathcal{I}_i \times \mathcal{I}_j \rightarrow \mathcal{I}_n \)
- \( I[\omega(\text{expr}_1, \ldots, \text{expr}_n)](\sigma, \beta) := \mathcal{I}(\omega) \left( \mathcal{I}[\text{expr}_1](\sigma, \beta), \ldots, \mathcal{I}[\text{expr}_n](\sigma, \beta) \right) \)
- \( I[\text{allInstances}_C](\sigma, \beta) := \text{dom}(\sigma) \cap \mathcal{D}(C) \)

**Note:** in the OCL standard, \( \text{dom}(\sigma) \) is assumed to be **finite**.

Again: doesn’t scare us.

---

(vi) Putting It All Together...

\[ \text{expr} := w \mid \omega(\text{expr}_1, \ldots, \text{expr}_n) \mid \text{allInstances}_C \mid v(\text{expr}_1) \mid r_1(\text{expr}_1) \mid r_2(\text{expr}_1) \mid \text{expr}_1 \rightarrow \text{iterate}(v_1 : \tau_1; v_2 : \tau_2 = \text{expr}_2 \mid \text{expr}_3) \]

Assume \( \text{expr}_1 : \tau_C \) for some \( C \in \mathcal{C} \). Set \( u_1 := I[\text{expr}_1](\sigma, \beta) \in \mathcal{D}(\tau_C) \).

- \( I[v(\text{expr}_1)](\sigma, \beta) := \begin{cases} \sigma(u) & \text{if } u \in \text{dom}(\sigma) \\ \bot_{\mathcal{D}_\tau} & \text{otherwise} \end{cases} \)
- \( I[r_1(\text{expr}_1)](\sigma, \beta) := \begin{cases} u & \text{if } u \in \text{dom}(\sigma) \text{ and } \sigma(u)(r_1) = \text{expr}_1 \\ \bot_{\mathcal{D}_\tau} & \text{otherwise} \end{cases} \)
- \( I[r_2(\text{expr}_1)](\sigma, \beta) := \begin{cases} \sigma(u_1)(r_2) & \text{if } u_1 \in \text{dom}(\sigma) \\ \bot_{\mathcal{D}_\tau} & \text{otherwise} \end{cases} \)

(Recall: \( \sigma \) evaluates \( r_2 \) of type \( C_\ast \) to a set)
(vi) Putting It All Together...

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

- \text{Quiz: Is (our) } I \text{ a function?}

Example

\[
\sigma ::=
\begin{align*}
\text{TeamMember} : \text{Name} & : \text{String} \quad \text{age} : \text{Integer} \\
\text{Meeting} : \text{title} & : \text{String} \quad \text{numParticipants} : \text{Integer} \\
& \quad \text{start} : \text{Date} \quad \text{duration} : \text{Time}
\end{align*}
\]

\[
\begin{align*}
& \text{context TeamMember inv: age} \Rightarrow 18 \\
& \text{context Meeting inv: duration} > 0
\end{align*}
\]
OCL Satisfaction Relation

In the following, \( \mathcal{S} \) denotes a signature and \( \mathcal{D} \) a structure of \( \mathcal{S} \).

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

\[\text{Note:} \] In general we can’t conclude from \( \neg(\sigma \models \varphi) \) to \( \sigma \not\models \varphi \) or vice versa.
**OCL Consistency**

**Definition (Consistency).** A set \( \text{Inv} = \{ \varphi_1, \ldots, \varphi_n \} \) 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

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

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 \( \text{location}.\text{name} = \text{''Lobby''} \)

- allInstances **Meeting** \( \rightarrow \) exists \((w: \text{Meeting} | w.\text{title} = \text{'Reception'})\)
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

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

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

**OCL Critique**

- **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]
You Are Here.
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

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

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

- nodes are identities (not necessarily alive), i.e. $N \subseteq \mathcal{D}(\mathcal{C})$ finite,
- edges correspond to “links” of objects, i.e. $E \subseteq N \times \{v : \tau \in V | \tau \in \{C_{0,1}, C_* | C \in \mathcal{C}\}\} \times N$,
- objects are labelled with attribute valuations and non-alive identities with “X”, i.e.
  
  $X = \{X\} \cup (V \rightarrow (\mathcal{D}(\mathcal{T}) \cup \mathcal{D}(\mathcal{C_*})))$

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

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

is called object diagram of $\sigma$.

Object Diagram: Example

\[ N \subseteq \mathcal{D}(\mathcal{C}) \text{ finite}, \quad E \subseteq N \times \{v : \tau \in V | \tau \in \{C_{0,1}, C_* | C \in \mathcal{C}\}\} \times N, \quad X = \{X\} \cup (V \rightarrow (\mathcal{D}(\mathcal{T}) \cup \mathcal{D}(\mathcal{C_*}))) \]

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

$\mathcal{S} = (\{Int\}, \{C\}, \{v_1 : Int, v_2 : Int, r : C_*\}, \{C \mapsto \{v_1, v_2, r\}\}), \quad \mathcal{D}(Int) = \mathbb{Z}$

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

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

\[ N = \{u_1, u_2\}, \quad \{u_1 \mapsto \{v_1 \mapsto 1, v_2 \mapsto 2\}, u_2 \mapsto \{v_1 \mapsto 3, v_2 \mapsto 4\}\} \]

\[ E = \{(u_1, v_2)\} \]

\[ f = \{u_1 \mapsto \{v_1 \mapsto 1, v_2 \mapsto 2\}, v_2 \mapsto \{v_1 \mapsto 3, v_2 \mapsto 4\}\} \]
\( \forall (u_1, r, u_2) \in E : u_1 \in \text{dom}(\sigma) \land u_2 \in \sigma(u_1)(r), \quad f(u) \subseteq \sigma(u) \text{ or } f(u) = \{X\} \)

\( \mathcal{J} = (\{\text{Int}\}, \{C\}, \{v_1 : \text{Int}, v_2 : \text{Int}, r : C\}, \{C \mapsto \{v_1, v_2, r\}\}), \quad \mathcal{D}(\text{Int}) = \mathbb{Z} \)

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

- Then \( G = (N, E, f) \) with

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

is an object diagram of \( \sigma \) wrt. \( \mathcal{J} \) and any structure \( \mathcal{D} \) with \( \mathcal{D}(\text{Int}) \supseteq \{1, 2, 3, 4\} \).

Node: we may equivalently (!) represent \( G \) graphically as follows:
Object Diagrams: More Examples?

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

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

\[ \mathcal{S} = (\{\text{Int}\}, \{C, D\}, \{x : \text{Int}, p : C_{0,1}, n : C_\ast\}, \{C \mapsto \{p, n\}, D \mapsto \{x\}\}), \mathcal{P}(\text{Int}) = Z \]

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

Complete vs. Partial Object Diagram

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

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

- \(G\) is **object complete**, i.e.
  - \(G\) consists of all alive objects, i.e. \(N \supset \text{dom}(\sigma)\),
  - \(G\) 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 \(u \in \text{dom}(\sigma)\),
      \(f(u) = \sigma(u)|_{\mathcal{T}} \cup \{r \mapsto (\sigma(u)(r) \setminus N) \mid r \in V : \sigma(u)(r) \setminus N \neq \emptyset\}\)
      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 $(u_1, r, u_2) \in E$.
- $f(u) = \sigma(u)|_{\forall r} \cup \{ r \mapsto (\sigma(u)(r) \setminus N) \mid \sigma(u)(r) \setminus N \}$

Complete or partial?

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

Special Notation

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

- Instead of

we want to write

or

to explicitly indicate that attribute $p : C^*$ has value $\emptyset$ (also for $p : C_{0,1}$).
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
  
  $$G^{-1} := \{ \sigma \in \Sigma \mid G \text{ is an object diagram of } \sigma \}$$

- **Observation:**
  If somebody **tells us**, that a given (consistent) object diagram $G$
  - is **meant to be complete**, and if it is not inherently incomplete (e.g. missing attribute values),
    then we can uniquely reconstruct the corresponding system state.
    In other words: $G^{-1}$ is then a singleton.

Closed Object Diagrams vs. Dangling References

**Find the 10 differences!** (Both diagrams are meant to 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 $T$-typed attribute/value pairs only.
UML Object Diagrams

UML Notation for Object Diagrams

id: class

v_1 = d_1

\ldots

v_n = d_n

we assume: different "boxes" different instances

optional

mandatory

"compartment" optional

optional

id: class
Discussion

We slightly deviate from the standard (for reasons):

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

\[
\begin{array}{c}
C_1 \quad n \quad C_2
\end{array}
\]

(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 $C_s$-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


