Software Design, Modelling and Analysis in UML

Lecture 5: Object Diagrams

2016-11-10

Prof. Dr. Andreas Podelski, Dr. Bernd Westphal

Albert-Ludwigs-Universität Freiburg, Germany

Content

- **Object Constraint Language** completed:
  - Satisfaction Relation, Consistency
  - Decidability
  - OCL Critique

- **Object Diagrams**
  - Definition
  - Graphical Representation
  - Partial vs. Complete Object Diagrams

- **The Other Way Round**

- **Object Diagrams for Documentation**
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 $I[\varphi](\sigma, \emptyset) = \text{true}$.
- $\sigma \not\models \varphi$ 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.
**Definition (Consistency).** A set \( \mathcal{I} = \{ \varphi_1, \ldots, \varphi_n \} \) 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

\[
\exists \sigma \in \Sigma_D : \sigma \models \varphi_1 \land \ldots \land \sigma \models \varphi_n
\]

and **inconsistent** (or **unsatisfiable**) otherwise.

---

**Example: OCL Consistent?**

- **context** Location inv : name = 'Lobby' implies meetings \( \Rightarrow \) isEmpty()
- **context** Meeting inv : title = 'Reception' implies location . name = 'Lobby'
- allInstances Meeting \( \Rightarrow \) exists(w : Meeting | w . title = 'Reception')
- **context** Meeting inv : location \( \Rightarrow \) exists(m : Meeting | m . location = self)

\[
\text{not consistent}
\]
Deciding OCL Consistency

- Whether a set of OCL constraints is consistent 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.

OCL is as expressive as first-order logic over integers.

\[
\exists x, y \in \mathbb{R} \quad x + y > 2z \quad y \neq z
\]

\[
\forall z \in \mathbb{R} \quad \exists x \in \mathbb{R} \quad z = x^2 + 1
\]

all instance\(c_0\) \(\rightarrow\) exists (\(\langle c, x \rightarrow \text{size}(c) + c, y \rightarrow \text{size}(c) > 2z \rangle\))
Deciding OCL Consistency

- Whether a set of OCL constraints is consistent 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. OCL is as expressive as first-order logic over integers.

\[
\exists x, y \cdot x + y > 2 \exists \text{ } y = d
\]

\[
\Box = (\emptyset, \{c\}, \{x \cdot c, y \cdot c, y \cdot \{c \Rightarrow \{x, y\}\}\})
\]

\[
\forall \text{last} \text{new} \rightarrow \exists \text{last} (c | c \cdot \text{last} + c \cdot \text{new} \cdot \text{last} > 2d) \quad \text{last} \Rightarrow \text{new}
\]

- **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
OCL Critique

- **Concrete Syntax / Features**
  
  "The syntax of OCL has been criticized – e.g., by the authors of Catalysts [...] – 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)

- **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.
Where Are We?

You Are Here.
Object Diagrams

Object Constraint Language completed:
- Satisfaction Relation, Consistency
- Decidability
- OCL Critique

Object Diagrams
- Definition
- Graphical Representation
- Partial vs. Complete Object Diagrams

The Other Way Round
- Object Diagrams for Documentation
**Recall: Graph**

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 $\mathcal{P}$ be a structure of signature $\mathcal{S} = (\mathcal{R}, \mathcal{C}, V, \text{atr})$ and $\sigma \in \Sigma^\mathcal{P}$ a system state.

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

- nodes are alive objects, i.e. $N \subset \mathcal{P}(\mathcal{C}) \cap \text{dom}(\sigma)$,
- edges start are labelled with derived type attributes, i.e.
  $$E \subseteq N \times \{v : T \in V \mid T \in \{C_{0,1}, C_\ast \mid C \in \mathcal{C}\}\} \times N,$$
  $$:= V_{0,1,\ast} \text{ (derived type attributes in } \mathcal{S})$$
- edges correspond to “links” between objects, i.e.
  $$\forall u_1, u_2 \in \mathcal{P}(\mathcal{C}), r \in V_{0,1,\ast} : (u_1, r, u_2) \in E \implies u_2 \in \sigma(u_1)(r),$$
- nodes are labelled with an identity and attribute valuations, i.e.
  $$X = (V \cup \{id\} \rightarrow (\mathcal{R}(\mathcal{P}) \cup \mathcal{P}(\mathcal{C})))$$
  $$\forall u \in N : f(u) \subseteq \{id \rightarrow \{u\}\} \cup \sigma(u)_{V_\mathcal{R}} \cup \{r \rightarrow R \mid r \in V_{0,1,\ast}, R \subseteq \sigma(u)(r)\}$$
  where $V_\mathcal{R} := \{v : T \in V \mid T \in \mathcal{R}\}$ (basic type attributes in $\mathcal{S}$).

is called object diagram of $\sigma$. 
Object Diagram: Examples

\[ N \subseteq \mathcal{P}(\mathcal{E}) \cap \text{dom}(\sigma) \quad \mathcal{E} \subseteq N \times V_{0,1,\ast} \times N \quad (u_1, r, u_2) \in \mathcal{E} \implies u_2 \in \sigma(u_1)(r) \quad f : N \rightarrow X \]

\[ X = (V \cup \{id\}) \rightarrow (\mathcal{P}(\mathcal{E}) \cup \mathcal{P}(\sigma)) \quad f(u) \subseteq \{id \mapsto \{u\}\} \cup \sigma(u) \quad \mathcal{P}(\mathcal{E}) \cup \{r \mapsto R \mid R \subseteq \sigma(u)(r)\} \]

\[ \mathcal{E} = \{(i \mapsto \{x\}, r, j \mapsto \{y\}) \mid i, j \in \text{Int} \} \]

\[ \sigma = \{i \mapsto \{x \mapsto 1, y \mapsto 2, r \mapsto \{1c, 3c\}\} \mid i \in \text{Int} \} \]

\[ \mathcal{P}(\text{Int}) = Z \]

\[ \mathcal{E} = \{(i \mapsto \{x\}, r, j \mapsto \{y\}) \mid i, j \in \text{Int} \} \]

\[ \sigma = \{i \mapsto \{x \mapsto 1, y \mapsto 2, r \mapsto \{1c, 3c\}\} \mid i \in \text{Int} \} \]

\[ f : N \rightarrow X \]

\[ X = \{1c \mapsto \{x \mapsto 1, y \mapsto 2\} \} \]

\[ \mathcal{P}(\text{Int}) = Z \]

\[ X = \{1c \mapsto \{x \mapsto 1, y \mapsto 2\} \} \]

\[ f : N \rightarrow X \]

\[ X = \{1c \mapsto \{x \mapsto 1, y \mapsto 2\} \} \]

\[ \mathcal{P}(\text{Int}) = Z \]

\[ X = \{1c \mapsto \{x \mapsto 1, y \mapsto 2\} \} \]

\[ f : N \rightarrow X \]

\[ X = \{1c \mapsto \{x \mapsto 1, y \mapsto 2\} \} \]
Object Diagram: Examples

\* N \subseteq \mathcal{P}(\mathcal{F}) \cap \text{dom}(\sigma) \quad \* E \subseteq N \times V_{0,1,0} \times N \quad \* (u_1, r, u_2) \in E \implies u_2 \in \sigma(u_1)(r) \quad \* f : N \to X

\* X = (V \cup \{id\}) \to (\mathcal{P}(\mathcal{F}) \cup \mathcal{P}(\mathcal{E})) \quad \* f(u) \subseteq \{id \mapsto \{u\}\} \cup \sigma(u)_{Y \sigma} \cup \{r \mapsto R \mid R \subseteq \sigma(u)(r)\}

\> \mathcal{F} = (\{Int\}, \{C\}, \{x : \text{Int}, y : \text{Int}, r : C_x\}, \{C \mapsto \{x, y, r\}\}), \quad \mathcal{P}(\text{Int}) = \mathbb{Z}

\> \> \sigma = \{1_C \mapsto \{x \mapsto 1, y \mapsto 2, r \mapsto \{1_C, 3_C\}\}\}

\* \quad G = (N, E, f) with

\* \quad nodes N = \{1_C, 3_C\}

\* \quad edges E = \{(1_C, r, 1_C), (1_C, r, 3_C)\}

\* \quad node labelling f = \{1_C \mapsto \{id \mapsto \{1_C\}, x \mapsto 1, y \mapsto 2\}\}

is an object diagram of \sigma.

\* \quad Yes, and...? \( G \) can equivalently (!) be represented graphically:

\[ \text{UML Notation for Object Diagrams} \]
**Object Diagram: More Examples?**

1. $N \subset \mathcal{P}(\mathcal{E}) \cap \text{dom}(\sigma)$
2. $E \subset N \times V_{0,1,\ast} \times N$
3. $(u_1, r, u_2) \in E \implies u_2 \in \sigma(u_1)(r)$
4. $f : N \rightarrow X$
5. $X = \left( V \cup \{id\} \rightarrow (\mathcal{E}) \cup \mathcal{P}(\mathcal{E}) \right)$
6. $f(u) \subseteq \{id \rightarrow \{u\}\} \cup \sigma(u)_{\mathcal{V}_{\ast}} \cup \{r \rightarrow R \mid R \subseteq \sigma(u)(r)\}$

$\mathcal{F} = \{\{\text{Int}\}, \{C\}, \{x : \text{Int}, y : \text{Int}, r : C\ast\}, \{C \mapsto \{v_1, v_2, r\}\}\}$, \quad \mathcal{P}(\text{Int}) = \mathbb{Z}

$\sigma = \{1_C \mapsto \{x \mapsto 1, y \mapsto 2, r \mapsto \emptyset\}\}$, \quad 2_C = \{x \mapsto 13, y \mapsto 27, r \mapsto \emptyset\}$

**Complete vs. Partial Object Diagram**

**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.
  
  $N = \text{dom}(\sigma)$

- **$G$ is attribute complete**, i.e.
  
  $\forall u \in \text{dom}(\sigma) \bullet f(u) = \{id \mapsto u\} \cup \sigma(u)_{\mathcal{V}_{\ast}} \cup \{r \mapsto \sigma(u)(r) \setminus \text{dom}(\sigma) \mid \sigma(u)(r) \not\subseteq \text{dom}(\sigma)\}$
Otherwise we call $G$ partial.
Complete vs. Partial: Examples

- Each object diagram-like graph $G$ represents a set of system states, namely
  
  $$G^{-1} := \{ \sigma \in \Sigma^\mathcal{G} \mid G \text{ is an object diagram of } \sigma \}$$

- How many?

- Each system state has exactly one complete object diagram.

- A system state can have many partial object diagrams.

**Observation:**
If somebody tells us for a given object diagram $G$

- that it is meant to be complete, and

- if it is not inherently incomplete (e.g. missing attribute values).
then it uniquely denotes the corresponding system state, denoted by $\sigma(G)$.

Therefore we can use complete object diagrams exchangeably with system states.
Non-Standard Notation

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

- Instead of

  \[
  \text{Attribute } n : \mathcal{C}_* \mapsto C
  \]

  we want to write

  \[
  \text{Attribute } n : \mathcal{C}_* \mapsto C \quad \text{and} \quad \text{Attribute } p : \mathcal{C}_* \mapsto C
  \]

  or

  \[
  \text{Attribute } p : \mathcal{C}_* \mapsto C
  \]

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

UML Object Diagrams
Discussion

We slightly deviate from the standard (for reasons):

• We **allow** to show non-alive objects.
  • 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.
  • Easier to distinguish partial and complete object diagrams.

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

\[
\begin{array}{c}
\text{u}_1 : C \\
\text{n} \\
\text{u}_2 : C_1 \\
\end{array}
\]

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.

The Other Way Round
If we only have a diagram like

we typically assume that it is meant to be an object diagram wrt. some signature and structure.

In the example, we conclude that the author is referring to some signature $\mathcal{S} = (\mathcal{F}, \mathcal{E}, V, \text{atr})$ with at least

- $\{C, D\} \subseteq \mathcal{E}$
- $T \in \mathcal{F}$
- $\{\mathcal{C}, \mathcal{F}_1, \mathcal{C}_2, \mathcal{P}: \mathcal{C}_1\} \subseteq \mathcal{V}$
- $\text{atr}(\mathcal{C}) \geq \{\mathcal{C}\}$
- $\text{atr}(\mathcal{D}) \geq \{\mathcal{P}, \mathcal{Z}\}$

and a structure $\mathcal{S}$ with

- $\{\mathcal{A}_1, \mathcal{A}_2\} \subseteq \mathcal{D}(\mathcal{C})$
- $\mathcal{B}_1 \subseteq \mathcal{D}(\mathcal{D})$
- $\mathcal{B} \subseteq \mathcal{D}(\mathcal{T})$

Example: Object Diagrams for Documentation
Example: Data Structure (Schumann et al., 2008)

Example: Illustrative Object Diagram (Schumann et al., 2008)
Tell Them What You’ve Told Them...

- When using an OCL constraint $F$ to formalise requirements, we typically ask to ensure $\sigma \models F$.

- System states can graphically be represented using Object Diagrams.

- Our notation is slightly non-standard (for reasons) – mind the syntax (to not confuse Object and Class Diagrams)!

- Object diagrams can be partial or complete, the author's got to tell us.

- An Object Diagram for a typical system state can be used as a starting point to design a signature.

- Object Diagrams can be used to illustrate/document how a structure is supposed to be used.

References
References


