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?

You Are Here.
Object Diagrams

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 $\mathcal{D}$ be a structure of signature $\mathcal{S} = (\mathcal{F}, \mathcal{C}, V, atr)$ 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. $N \subseteq \mathcal{D}(\mathcal{F})$ finite,
- edges correspond to "links" of objects, i.e. $E \subseteq N \times \{ \tau : \tau \in V \mid \tau \in \{ C_0, 1, C_* \mid C \in \mathcal{C} \} \} \times N$,
- objects are labelled with attribute valuations and non-alive identities marked with "X", i.e.

$$X = \{ X \} \cup (V \rightarrow (\mathcal{D}(\mathcal{F}) \cup \mathcal{D}(\mathcal{C}_0)))$$

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

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

is called **object diagram** of $\sigma$.

---

**Graphical Representation of Object Diagrams**

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

- Consider

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

- 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. $\mathcal{S}$ and any $\mathcal{D}$ with $\mathcal{D}(Int) \supseteq \{ 1, 2, 3, 4 \}$.

- $G = \{ \{ v_1, v_2 \}, \emptyset \}$

- $G = \{ \{ v_1 \}, \emptyset \}$

- $G = \{ \{ v_2 \}, \emptyset \}$

- $G = \{ \{ v_1, v_2 \}, \emptyset \}$

- $G = \{ \{ v_1, v_2 \}, \emptyset \}$

- $G = \{ \{ v_1, v_2 \}, \emptyset \}$

- $G = \{ \{ v_1, v_2 \}, \emptyset \}$

- $G = \{ \{ v_1, v_2 \}, \emptyset \}$

- $G = \{ \{ v_1, v_2 \}, \emptyset \}$
Graphical Representation of Object Diagrams

\[ N \subset \mathcal{P}(\mathcal{C}) \text{ finite, } E \subset N \times V_{0,1,2} \times N, \quad X = \{X\} \cup (V \rightsquigarrow (\mathcal{P}(\mathcal{C}) \cup \mathcal{P}(\mathcal{C}))) \]

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

- Assume \( \mathcal{P} = \{\{\text{Int}\}, \{C\}, \{v_1 : \text{Int}, v_2 : \text{Int}, r : C\}, \{C \mapsto \{v_1, v_2, r\}\}\} \).
- 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. \( \mathcal{P} \) and any \( \mathcal{P} \) with \( \mathcal{P}(\text{Int}) \supseteq \{1, 2, 3, 4\} \).
- We may equivalently (!) represent \( G \) graphically as follows:

UML Notation for Object Diagrams

- \( \{\text{id}\} : \{\text{class}\} \)
- \( \{v_1\} = \{d_1\} \)
- \( \{v_n\} = \{d_n\} \)
- \( \{r\} \)
Object Diagrams: More Examples

\[ N \subseteq \mathcal{P}(\mathcal{E}) \text{ finite}, \quad E \subseteq N \times V_{0,1,*} \times N, \quad X = \{X\} \cup (V \rightarrow (\mathcal{P}(\mathcal{F}) \cup \mathcal{P}(\mathcal{E}))) \]

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

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

vs.

\[ \mathcal{J} = (\{D, C\}, \{x: \mathcal{U}, \quad p: \mathcal{E}, n: C_{\mathcal{E}}, \quad C = \{\bullet, \circ, \diamond\} \}) \]

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

\[ \text{the empty picture} \]

\[ \bullet \quad \checkmark \]

\[ \text{vs.} \]

\[ \bullet \quad \checkmark \]

\[ \bullet \quad \checkmark \]

\[ \bullet \quad \checkmark \]

\[ \bullet \quad \checkmark \]

\[ \text{Complete vs. Partial Object Diagram} \]

**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.
  - \( G \) **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 \( \mathcal{F} \)-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 \( V_\mathcal{F} := \{v : \tau \in V \mid \tau \in \mathcal{F}\} \).

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_{\eta}} \cup \{ r \mapsto (\sigma(u)(r) \setminus N) \} \setminus (\sigma(u)(r) \setminus N) \}

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

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

- \( 1_C : C, n \)
- \( 5_C : C, (n = 0) \)
- \( 1_D : D, x = 23 \)

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

- 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

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

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

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 $\mathcal{S}$-typed attribute/value pairs only.
**Special Notation**

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

- Instead of

  ![Diagram 1](attachment:image1.png)

  we want to write

  ![Diagram 2](attachment:image2.png)

  or

  ![Diagram 3](attachment:image3.png)

  to explicitly indicate that attribute \( p : C_s \) 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 \( C^*_s \)-typed attributes only have sets as values. UML also considers multisets, that is, they can have

  ![Diagram 4](attachment:image4.png)

  (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.
The Other Way Round

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, \( \mathcal{X} \) denotes a signature and \( \mathcal{D} \) a structure of \( \mathcal{X} \).

---

**Definition (Satisfaction Relation).**

Let \( \varphi \) be an OCL constraint over \( \mathcal{X} \) and \( \sigma \in \Sigma^\mathcal{X} \) a system state. We write

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

---

**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 $S$ wrt. structure $D$.
  Let $expr$ be an OCL expression over $S$.
  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 "$\nmid$".
  (Otherwise better not to avoid confusion: $G^{-1}$ could comprise different system states in which $expr$ evaluates to true, false, and $\bot$.)
- Example: (complete — what if not complete wrt. object/attribute/both?)

<table>
<thead>
<tr>
<th>$\lambda C : C$</th>
<th>$n$</th>
<th>$\lambda C : C$</th>
<th>$n = 0$</th>
<th>$\lambda D : D$</th>
<th>$x = 23$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p = 0$</td>
<td></td>
<td>$n = 0$</td>
<td>$p = 0$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- context $C$ inv: $n \rightarrow $isEmpty() \(\rightarrow\) false
- context $C$ inv: $p \cdot n \rightarrow $isEmpty() \(\rightarrow\) true
- context $D$ inv: $x \neq 0 \rightarrow $ true

OCL Consistency

**Definition (Consistency).** A set $Inv = \{ \varphi_1, \ldots, \varphi_n \}$ of OCL constraints over $S$ is called consistent (or satisfiable) if and only if there exists a system state of $S$ wrt. $D$ which satisfies all of them, i.e. if
\[ \exists \sigma \in \Sigma^D_S : \sigma \models \varphi_1 \land \ldots \land \sigma \models \varphi_n \]
and inconsistent (or unrealizable) otherwise.
OCL Inconsistency Example

- context `Location` inv:  
  `name = 'Lobby'` implies `meeting -> isEmpty()`

- context `Meeting` inv:  
  `title = 'Reception'` implies `location.name = "Lobby"

- allInstances `Meeting` -> exists(w : `Meeting` | `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

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

  Encoding in OCL:

  \[ \text{allInstances}_C -> \exists (w : C | c_1 * w.x_1^{n_1} + \cdots + c_m * w.x_m^{n_m} = d). \]
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 undecidable.

  Otherwise we could, for instance, solve diophantine equations

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

  Encoding in OCL:

  \[
  \text{allInstances}_C \rightarrow \exists (w : C | c_1 \ast w. x^{n_1} + \cdots + c_m \ast w. x^{n_m} = d).
  \]

- **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

- **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


