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
OCL Satisfaction Relation
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.
**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

$$\exists \sigma \in \Sigma_D : \sigma \models \varphi_1 \land \ldots \land \sigma \models \varphi_n$$

and **inconsistent** (or **unsatisfiable**) otherwise.
Example: OCL Consistent?

\[ s = \{ \{ \text{Integer}, \text{String} \} \} \]

\(-\text{TeamMembers, } \ldots \)?

\(-\text{Name : String, } \ldots \), \(-\text{TeamMember} \}

\(-\text{name, } \ldots \)

- 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')

- context **Meeting** inv : location --> exists(i: Meeting | i = self)

\[ \text{"self.loc = self Self loc . meeting"} \]

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.
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.

\[
\exists x, y \cdot x + y > 2^7
\]

\[
y = 2^7
\]

\[
\forall \bigcup = ( \emptyset, \{ C \}, \{ \exists x : C_x, y : C_y \}, \{ C' \rightarrow \{ x, y \} \})
\]

all instances \( c \rightarrow \exists x \left( c \mid c.x \rightarrow \text{size}(c) + c.y \rightarrow \text{size}(c) > 2^7 \right) \)
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 > 27 \quad \text{\(x^2 = 27\)} \quad \text{\(y = 1\)}
  \]

  \[
  \gamma = (\emptyset, \{C\}, \{x : C, y : C, \{x \rightarrow \{k, y\}\}\})
  \]

  all instances \(\gamma \rightarrow \exists x (c | c.x \rightarrow \text{size}() + c.y \rightarrow \text{size}() > 27)\)

- **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
**OCL Critique**

- **Concrete Syntax / Features**
  
  “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)
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.
Where Are We?
You Are Here.

Instances

Model

G = (N, E, f)

M = Σφ, Aφ, →SM

π = (σ0, ε0, cons0, Snd0)

→

w0 (σ1, ε1, cons1, Snd1)

→

wN (σN, εN, consN, SndN)

B = (Qsd, q0, Aφ, →sd, Fsd)

ϕ ∈ OCL

CD, SM

Mathematics

UML

CD, SD

UML

You Are Here.
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
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,
**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 alive objects, i.e. $N \subseteq \mathcal{D}(\mathcal{C}) \cap \text{dom}(\sigma)$,
- edges start are labelled with derived type attributes, i.e.

  $$E \subseteq N \times \{v : T \in V \mid T \in \{C_{0,1}, C_* \mid C \in \mathcal{C}\}\} \times N,$$

  $$=: V_{0,1;*} \text{ (derived type attributes in } \mathcal{S})$$

- edges correspond to “links” between objects, i.e.

  $$\forall u_1, u_2 \in \mathcal{D}(\mathcal{C}), r \in V_{0,1;*} : (u_1, r, u_2) \in E \implies u_2 \in \sigma(u_1)(r),$$

- nodes are labelled with an identity and attribute valuations, i.e.

  $$X = (V \cup \{id\} \leftrightarrow (\mathcal{D}(\mathcal{T}) \cup \mathcal{D}(\mathcal{C}_*))$$

  $$\forall u \in N : f(u) \subseteq \{id \mapsto \{u\}\} \cup \sigma(u)|_{V_{\mathcal{T}}} \cup \{r \mapsto R \mid r \in V_{0,1;*}, R \subseteq \sigma(u)(r)\}$$

  where $V_{\mathcal{T}} := \{v : T \in V \mid T \in \mathcal{T}\}$ (basic type attributes in $\mathcal{S}$).

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

- $N \subseteq \mathcal{D}(C) \cap \text{dom}(\sigma)$
- $E \subseteq N \times V_{0,1;\ast} \times N$
- $(u_1, r, u_2) \in E \implies u_2 \in \sigma(u_1)(r)$
- $f: N \rightarrow X$
- $X = (V \cup \{id\}) \hookrightarrow (\mathcal{D}(T) \cup \mathcal{D}(C_*))$
- $f(u) \subseteq \{id \mapsto \{u\}\} \cup \sigma(u)|_{\mathcal{V}} \cup \{r \mapsto R \mid R \subseteq \sigma(u)(r)\}$

\[\mathcal{I} = (\{\text{Int}\}, \{C\}, \{x : \text{Int}, y : \text{Int}, r : C_*\}, \{C \mapsto \{x, y, r\}\}), \quad \mathcal{D}(\text{Int}) = \mathbb{Z}\]

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

\[\mathcal{G} = (N, E, f) \text{ with}
\]
- nodes $N = \{1_C\}_{i = 0}^{\infty}$
- edges $E = \{(1_C, r, 1_C)_{i = 0}^{\infty}, \}
- node labelling $f = \{1_C \mapsto \{id \mapsto \{1_C\}, x \mapsto 1, r \mapsto \{3_C\}\} 3_C \mapsto \{id \mapsto \{3_C\}\}\}

is an object diagram of $\sigma$. 

\( \mathcal{I} = (\{\text{Int}\}, \{C\}, \{x : \text{Int}, y : \text{Int}, r : C_*\}, \{C \mapsto \{x, y, r\}\}), \quad \mathcal{D}(\text{Int}) = \mathbb{Z} \)

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

- \( G = (N, E, f) \) with
  - nodes \( N = \{1_C\} \)
  - edges \( E = \{(1_C, r, 1_C)\} \)
  - node labelling \( f = \{1_C \mapsto \{id \mapsto \{1_C\}, x \mapsto 1, y \mapsto 2\}\} \)

is an object diagram of \( \sigma \).

- Yes, and...?
Object Diagram: Examples

- \( N \subset \mathcal{D}(\mathcal{C}) \cap \text{dom}(\sigma) \)
- \( E \subset N \times V_{0,1;\ast} \times N \)
- \((u_1, r, u_2) \in E \implies u_2 \in \sigma(u_1)(r) \)
- \( f : N \to X \)
- \( X = (V \cup \{id\}) \to (\mathcal{D}(\mathcal{I}) \cup \mathcal{D}(\mathcal{C}_\ast)) \)
- \( f(u) \subseteq \{id \mapsto \{u\}\} \cup \sigma(u)|_{V_{\mathcal{I}}} \cup \{r \mapsto R \mid R \subseteq \sigma(u)(r)\} \)

\[
\mathcal{I} = (\{\text{Int}\}, \{C\}, \{x : \text{Int}, y : \text{Int}, r : C_\ast\}, \{C \mapsto \{x, y, r\}\}) \quad \mathcal{D}(\text{Int}) = \mathbb{Z} \\
\sigma = \{1_C \mapsto \{x \mapsto 1, y \mapsto 2, r \mapsto \{1_C, 3_C\}\}\} 
\]

- \( G = (N, E, f) \) with
  - nodes \( N = \{1_C\} \)
  - edges \( E = \{(1_C, r, 1_C)\} \)
  - node labelling \( f = \{1_C \mapsto \{id \mapsto \{1_C\}, x \mapsto 1, y \mapsto 2\}\} \)

is an object diagram of \( \sigma \).

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

\[
\begin{array}{c}
1_C : C \\
\hline
x = 1 \\
y = 2 \\
r \\
1_C : C \\
\hline
x = 1 \\
y = 2 \\
r \\
3_C : C \\
\hline
\end{array}
\]
UML Notation for Object Diagrams

Diagram showing the notation for object diagrams with mandatory and optional elements.
**Object Diagram: More Examples?**

- $N \subseteq \mathcal{D}(C) \cap \text{dom}(\sigma)$
- $E \subseteq N \times V_{0,1;*} \times N$
- $(u_1, r, u_2) \in E \implies u_2 \in \sigma(u_1)(r)$
- $f : N \to X$
- $X = (V \cup \{id\}) \mapsto (\mathcal{D}(\mathcal{I})) \cup \mathcal{D}(C_*)$
- $f(u) \subseteq \{id \mapsto \{u\}\} \cup \sigma(u)|_{\mathcal{I}} \cup \{r \mapsto R \mid R \subseteq \sigma(u)(r)\}$

$\mathcal{I} = (\{\text{Int}\}, \{C\}, \{x : \text{Int}, y : \text{Int}, r : C_*\}, \{C \mapsto \{v_1, v_2, r\}\})$, $\mathcal{D}(\text{Int}) = \mathbb{Z}$

$\sigma = \{1_C \mapsto \{x \mapsto 1, y \mapsto 2, r \mapsto \{2_C\}\}, \ 2_C \mapsto \{x \mapsto 13, y \mapsto 27, r \mapsto \emptyset\}\}$,
Complete vs. Partial Object Diagram

**Definition.** Let $G = (N, E, f)$ be an object diagram of system state $\sigma \in \Sigma^\mathcal{D}$. We call $G$ complete wrt. $\sigma$ if and only if

- $G$ is **object complete**, i.e.
  - $G$ consists of all alive and “linked” non-alive objects, i.e.
    $$N = \text{dom}(\sigma)$$
  - $G$ is **attribute complete**, i.e.
    - $G$ comprises all “links” between objects, i.e.
      $$\forall u_1, u_2 \in N, r \in V_{0,1,*} : (u_1, r, u_2) \in E \iff u_2 \in \sigma(u_1)(r),$$
    - each node is labelled with the values of all $\mathcal{T}$-typed attributes and the dangling references, i.e.
      $$\forall u \in \text{dom}(\sigma) \cdot f(u) = \{id \mapsto u\} \cup (\sigma(u)|_{V_{\mathcal{T}}} \cup \{r \mapsto \sigma(u)(r) \setminus \text{dom}(\sigma) \mid \sigma(u)(r) \not\subseteq \text{dom}(\sigma)\}).$$

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

\[ N \subset \mathcal{D}(C) \cap \text{dom}(\sigma) \quad E \subset N \times V_{0,1;*} \times N \quad (u_1, r, u_2) \in E \implies u_2 \in \sigma(u_1)(r) \quad f : N \rightarrow X \]

\[ X = (V \cup \{id\}) \rightarrow (\mathcal{D}(\mathcal{I}) \cup \mathcal{D}(C_*)) \quad f(u) \subseteq \{id \mapsto \{u\}\} \cup \sigma(u)|_{\mathcal{I}} \cup \{r \mapsto R \mid R \subseteq \sigma(u)(r)\} \]

\[ \mathcal{I} = (\{\text{Int}\}, \{C\}, \{x : \text{Int}, y : \text{Int}, r : C_*\}, \{C \mapsto \{v_1, v_2, r\}\}), \quad \mathcal{D}(\text{Int}) = \mathbb{Z} \]

\[ \sigma = \{1_C \mapsto \{x \mapsto 1, y \mapsto 2, r \mapsto \{2_C, 3_C\}\}, \quad 2_C \mapsto \{x \mapsto 13, y \mapsto 27, r \mapsto \emptyset\}\}, \]

\[ \begin{array}{c}
3_C : C \\
\hline
x
\end{array} \quad r \quad \begin{array}{c}
1_C : C \\
\hline
x = 1 \quad y = 2
\end{array} \quad r \quad \begin{array}{c}
2_C : C \\
\hline
x = 13 \quad y = 27
\end{array} \quad \checkmark \quad \text{complete} \]
Each object diagram-like graph $G$ represents a set of system states, namely

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

- 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

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

- Instead of

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

we want to write

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

or

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

to explicitly indicate that attribute \( p : C_* \) has value \( \emptyset \) (also for \( p : C_{0,1} \)).
UML Object Diagrams
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 $C_*$-typed attributes only have **sets** as values.
  UML also considers multisets, that is, they can have
  \[
  u_1 : C \quad \text{and} \quad u_2 : C
  \]
  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.
The Other Way Round
If we **only** have a diagram like

```
1 : C  \( n \) \( \rightarrow \) 2 : C  \( \rightarrow \) 3 : D
```

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** \( S = (\mathcal{I}, \mathcal{C}, V, \text{atr}) \) with at least

- \( \{C, D\} \subseteq \mathcal{C} \)
- \( T \subseteq \mathcal{I} \)
- \( \{ z : T, n : C, p : C, z \} \subseteq V \)
- \( \text{atr}(C) \supseteq \{ n \} \)
- \( \text{atr}(D) \supseteq \{ p, z \} \)

and a structure \( D \) with

- \( \{1 : C, 2 : C\} \subseteq D(C) \)
- \( 3 : D \subseteq D(D) \)
- \( 0 \in D(T) \)
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


