Introduction and Vocabulary

- Principles of Design
  - modularity
  - separation of concerns
  - information hiding and data encapsulation
  - abstract data types, object orientation

Software Modelling

- views and viewpoints, the 4+1 view
- model-driven/-based software engineering
- Unified Modelling Language (UML)
- modelling structure
  - (simplified) class diagrams
  - (simplified) object diagrams
  - (simplified) object constraint logic (OCL)
- modelling behaviour
  - communicating finite automata
  - Uppaal query language
  - basic state-machines
  - an outlook on hierarchical state-machines

Design Patterns
Once Again: Concrete vs. Abstract Syntax

Concrete Syntax: Example

Alternative Notation: Attribute and Method

Abstract Syntax: Object System Signature
• $x = \text{Proto-OCL vs. OCL.}$
• $\emptyset$

Alternative notation:
• $\sigma$ semantics: system states.
• $C, x \mapsto \{ p \mapsto \emptyset \}$

We use $\in$.
$D \in \text{dom} \iff (D \setminus C) = D$. If and only if $D(wrt. S)\in C$. 

A system state is a partial function $\Sigma$. 

$\Sigma (\in, V, \tau) = (\in, \tau \in V)$. 

$\in$ means $\in \cup (D \cup \Sigma)$. 

$\in$ is the $\in$ of $\in$ on $\tau$. 

A structure $\tau$ is a set of pairwise disjoint finite sets.

Basic Object System Structure Example
Example: Data Structure

Object diagrams at work

Example: Illustrative Object Diagram

Object diagram representation:

- Complete or partial
- Dangling reference in parent

Special Case: Anonymous Objects

Special Case: Dangling Reference
The formula above in \( \bot \)

\[
\forall B \rightarrow B \\
\times C \tau \\
\times f \tau, \tau
\]

if \( v \rightarrow \tau \)

\[
F | v \tau \\
\times \tau
\]

\[
\forall c \in c:
\times \tau
\]

\[
\times \tau
\]

allInstances \in c:
\times \tau

\[
\times \tau
\]

should never have the value \( \tau \)

\[
\times \tau
\]

logical variable \( x \), \( F \), \( atr \), \( V \), \( C \), \( T \)

wrt. signature

Example:

\[
data = \begin{cases} \\
\text{Int} : 4 & : N \\
\text{M} : 1 \\
\text{ctime} = 5 & : M \\
\text{ctime} = 9 & : N \\
\end{cases}
\]

Object Diagrams for Analysis

Class Diagrams

Partial vs. complete, dangling references, • semantics, • object diagrams at work.
Example: Evaluate Formula for System State

\[
\begin{align*}
\phi & = (C_1 \cap C_2) \cup \{\bot\} \\
C_1 & = C_1 \cup \{\bot\} \\
C_2 & = C_2 \cup \{\bot\}
\end{align*}
\]
More General Software vs. Proto-OCL

Modelling Structure with Class Diagrams

Where To Put OCL Constraints?
Class Diagrams can be used to graphically visualize code, define an object system structure $S$. An Object System Structure $S$ (together with a structure $D$) defines a set of system states $\Sigma_{DS}$. A System State $\sigma \in \Sigma_{DS}$ can be visualized by an object diagram. Proto-OCL constraints can be evaluated on system states. A software over $\Sigma_{DS}$ satisfies a Proto-OCL constraint $F$ if and only if $F$ evaluates to true in all system states of all the software's computation paths.

References:


