Contents & Goals

Last Lecture:
- Live Sequence Charts Semantics

This Lecture:
- Educational Objectives: Capabilities for following tasks/questions.
  - What’s the Liskov Substitution Principle?
  - What is late/early binding?
  - What is the subset, what the uplink semantics of inheritance?
  - What’s the effect of inheritance on LSCs, State Machines, System States?
  - What’s the idea of Meta-Modelling?

- Content:
  - Inheritance in UML: concrete syntax
  - Liskov Substitution Principle — desired semantics
  - Two approaches to obtain desired semantics
Inheritance: Syntax
**Inheritance: Generalisation Relation**

![Diagram of inheritance relation]

- **Alternative renderings:**
  - ![Alternative diagrams A and B]
  - ![Alternative diagrams C and D]

- **Read:**
  - \( C \) generalises \( D_1 \) and \( D_2 \); \( C \) is a **generalisation** of \( D_1 \) and \( D_2 \),
  - \( D_1 \) and \( D_2 \) specialise \( C \); \( D_1 \) **is a** (specialisation of) \( C \),
  - \( D_1 \) is a \( C \); \( D_2 \) is a \( C \).

- **Well-formedness rule:** No cycles in the generalisation relation.

---

**Abstract Syntax**

**Recall:** a signature (with signals) is a tuple \( \mathcal{S} = (\mathcal{T}, \mathcal{E}, V, atr) \).

**Now** (finally): extend to

\[
\mathcal{S} = (\mathcal{T}, \mathcal{E}, V, atr, F, mth, \lhd)
\]

where \( F/mth \) are methods, analogously to attributes and

\[
\lhd \subseteq (\mathcal{E} \times \mathcal{E}) \cup (\mathcal{E} \times \mathcal{E})
\]

is a **generalisation** relation such that \( C \lhd D \) for no \( C \in \mathcal{E} \) (“acyclic”).

\( C \lhd D \) reads as
- \( C \) is a generalisation of \( D \),
- \( D \) is a specialisation of \( C \),
- \( D \) inherits from \( C \),
- \( D \) is a sub-class of \( C \),
- \( C \) is a super-class of \( D \),
- \( \ldots \)
Mapping Concrete to Abstract Syntax by Example

Defn. Given classes \( C_0, C_1, D \in \mathcal{C} \), we say \( D \) inherits from \( C_0 \) via \( C_1 \) if and only if there are \( C_1^0, \ldots, C_n^0 \), \( C_1^1, \ldots, C_m^1 \in \mathcal{C} \) such that

\[
\begin{align*}
C_0 & \preceq C_1^0 \preceq \ldots \preceq C_n^0 \preceq C_1 \preceq C_1^1 \preceq \ldots \preceq C_m^1 \preceq D
\end{align*}
\]

We use \( \leq \) to denote the reflexive, transitive closure of \( \prec \).

In the following, we assume

- that all attribute (method) names are of the form \( C::v, \ C \in \mathcal{C} \cup \mathcal{B} \) \( (C::f, \ C \in \mathcal{C}) \),
- that we have \( C::v \in \text{atr}(C) \) resp. \( C::f \in \text{mth}(C) \) if and only if \( v(f) \) appears in an attribute (method) compartment of \( C \) in a class diagram.

We still want to accept “context \( C \ inv: v < 0 \)”, which \( v \) is meant? Later!
References


