Software Design, Modelling and Analysis in UML

Lecture 18: Inheritance I

2012-02-01

Prof. Dr. Andreas Podelski, Dr. Bernd Westphal

Albert-Ludwigs-Universität Freiburg, Germany
**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

- Alternative renderings:

- 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 $\mathcal{S} = (\mathcal{T}, \mathcal{C}, V, atr)$.  

**Now** (finally): extend to  

$$\mathcal{S} = (\mathcal{T}, \mathcal{C}, V, atr, F, mth, \sqsubseteq)$$  

where $F/mth$ are methods, analogously to attributes and  

$$\sqsubseteq \subseteq (\mathcal{C} \times \mathcal{C}) \cup (\mathcal{C} \times \mathcal{S}(\mathcal{C}))$$  

is a **generalisation** relation such that $C \sqsubseteq^+ C$ for no $C \in \mathcal{C}$ ("acyclic").

$C \sqsubseteq D$ 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 $D$,  
- ...
Mapping Concrete to Abstract Syntax by Example

$$\gamma = (\exists \lambda \in \mathcal{E},$$
$$\{ C_0, C_1, D, C_2 \},$$
$$\{ C_0 : x : \text{Int},$$
$$D : x : \text{Int} \},$$
$$\{ C_0 \vdash \sigma \times \chi, \quad D \vdash \sigma \times \chi, \quad C_1 \vdash \sigma \times \chi \},$$
$$\{ C_0 \triangleleft C_1, \quad C_1 \triangleleft C_2, \quad D \triangleleft C_2 \})$$

**Note**: we can have multiple inheritance.
**Definition.** 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 $C_0^1, \ldots, C_0^n, C_1^1, \ldots, C_1^m \in \mathcal{C}$ such that

$$C_0 \triangleright C_0^1 \triangleright \ldots \triangleright C_0^n \triangleright C_1 \triangleright C_1^1 \triangleright \ldots \triangleright C_1^m \triangleright D.$$ 

We use ‘$\preceq$’ to denote the reflexive, transitive closure of ‘$\triangleright$’.

In the following, we assume

- that all attribute (method) names are of the form
  $$C::v, \quad C \in \mathcal{C} \cup \mathcal{E} \quad (C::f, \quad C \in \mathcal{C}),$$
- that we have $C::v \in atr(C)$ resp. $C::f \in mth(C)$ 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


