Software Design, Modelling and Analysis in UML

Lecture 16: Hierarchical State Machines I

2015-01-15

Prof. Dr. Andreas Podelski, Dr. Bernd Westphal

Albert-Ludwigs-Universität Freiburg, Germany
Contents & Goals

Last Lecture:
- Missing transformers: create and destroy
- Step and run-to-completion (RTC) step, divergence

This Lecture:
- Educational Objectives: Capabilities for following tasks/questions.
  - What does this State Machine mean? What happens if I inject this event?
  - Can you please model the following behaviour.
  - What does this hierarchical State Machine mean? What may happen if I inject this event?
  - What is: AND-State, OR-State, pseudo-state, entry/exit/do, final state, ...

- Content:
  - Putting it all together: UML model semantics (so far)
  - State Machines and OCL
  - Hierarchical State Machines Syntax
  - Initial and Final State
Putting It All Together
Recall: a labelled transition system is \((S, \rightarrow, S_0)\). We have

- \(S\): system configurations \((\sigma, \varepsilon)\)
- \(\rightarrow\): labelled transition relation \((\sigma, \varepsilon) \xrightarrow{(\text{cons}, \text{Snd})} (\sigma', \varepsilon')\).

Wanted: initial states \(S_0\).

Proposal:
Require a (finite) set of object diagrams \(OD\) as part of a UML model \((CD, SM, OD)\).

And set
\[
S_0 = \{ (\sigma, \varepsilon) \mid \sigma \in G^{-1}(OD), OD \in OD, \varepsilon \text{ empty} \}.
\]

Other Approach: (used by Rhapsody tool) multiplicity of classes. We can read that as an abbreviation for an object diagram.
Semantics of UML Model — So Far

The **semantics** of the **UML model**

\[ M = (C_D, S_M, O_D) \]

where

- some classes in \( C_D \) are stereotyped as ‘signal’ (standard), some signals and attributes are stereotyped as ‘external’ (non-standard),
- there is a 1-to-1 relation between classes and state machines,
- \( O_D \) is a set of object diagrams over \( C_D \),

is the **transition system** \( (S, \rightarrow, S_0) \) constructed on the previous slide.

The **computations of** \( M \) are the computations of \( (S, \rightarrow, S_0) \).
State Machines and OCL
Let $\mathcal{M} = (\mathcal{C} \mathcal{D}, \mathcal{S} \mathcal{M}, \mathcal{O} \mathcal{D})$ be a UML model.

We call $\mathcal{M}$ consistent iff, for each OCL constraint $expr \in \text{Inv}(\mathcal{C} \mathcal{D}) \cup \text{Inv}(\mathcal{S} \mathcal{M})$,

$$\sigma \models expr$$

for each “reasonable point” $(\sigma, \varepsilon)$ of computations of $\mathcal{M}$.

(Cf. exercises and tutorial for discussion of “reasonable point”.)

**Note:** we could define $\text{Inv}(\mathcal{S} \mathcal{M})$ similar to $\text{Inv}(\mathcal{C} \mathcal{D})$. 

![Diagram of OCL constraints and behaviour]

Context $\mathcal{C}$ inv.:

- $st = s_1$ implies $x = 27$

Attribute $st$ — allow OCL over $s_1$, not only $s_0$. 

(Variable $x$.):
OCL Constraints and Behaviour

• Let $\mathcal{M} = (\mathcal{CD}, \mathcal{SM}, \mathcal{OD})$ be a UML model.
• We call $\mathcal{M}$ consistent iff, for each OCL constraint $\text{expr} \in \text{Inv}(\mathcal{CD})$,
  $$\sigma \models \text{expr} \text{ for each “reasonable point” } (\sigma, \varepsilon) \text{ of computations of } \mathcal{M}.$$  
  (Cf. exercises and tutorial for discussion of “reasonable point”.)

**Note:** we could define $\text{Inv}(\mathcal{SM})$ similar to $\text{Inv}(\mathcal{CD})$.

**OUR CHOICE:** check for each $(\sigma, \varepsilon)$ in a computation (skip similarity)

**Pragmatics:**

• In UML-as-blueprint mode, if $\mathcal{SM}$ doesn’t exist yet, then $\mathcal{M} = (\mathcal{CD}, \emptyset, \mathcal{OD})$ is typically asking the developer to provide $\mathcal{SM}$ such that $\mathcal{M}’ = (\mathcal{CD}, \mathcal{SM}, \mathcal{OD})$ is consistent.
  If the developer makes a mistake, then $\mathcal{M}’$ is inconsistent.

• **Not common:** if $\mathcal{SM}$ is given, then constraints are also considered when choosing transitions in the RTC-algorithm. In other words: even in presence of mistakes, the $\mathcal{SM}$ never move to inconsistent configurations.
Pragmatics: Example

(1) $\mathcal{M}$ is not consistent ("broken") because there is a comp. path. leading to a $(0, \varepsilon)$ s.t. $0 \notin \text{inv}(\mathcal{M})$

(2) "Dear developer, I want:
- states $s_0, s_1$
- $E$ must move me from $s_0$ to $s_1$
- in $s_1$, $x$ must not be 0
  do THAT!"
In each system state \( \sigma \), for each alive object \( u \in \text{dom}(\sigma), u \in D(c) \), each of \( u \)'s attributes has a (definite) value!

\[
\forall v \in \text{ak}(c) \quad \sigma(u)(v) \in D(kp(v))
\]
Rhapsody Demo II
Rhapsody

\[ z \in \text{generate} \]

\[ \text{build/compile} \rightarrow \text{DEFCup.exe} \]

"D just stupid from \( s_0 \) to \( s_z \) by transform \( t \)"