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
The Missing Piece: Initial States

Recall: a labelled transition system is $(S, \rightarrow, S_0)$. We have
- $S$: system configurations $(\sigma, \varepsilon)$
- $\rightarrow$: labelled transition relation $(\sigma, \varepsilon) \xrightarrow{\text{cons,Snd}} (\sigma', \varepsilon')$.

Wanted: initial states $S_0$.

Proposal:
Require a (finite) set of object diagrams $OD$ as part of a UML model

$$(CD, IM, OD).$$

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.
The semantics of the UML model

\[ \mathcal{M} = (\mathcal{C}, \mathcal{M}, \mathcal{O}) \]

where
- some classes in \( \mathcal{C} \) 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,
- \( \mathcal{O} \) is a set of object diagrams over \( \mathcal{C} \),

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

The computations of \( \mathcal{M} \) are the computations of \( (S, \rightarrow, S_0) \).
OCL Constraints and Behaviour

• Let $M = (C, S, O)$ be a UML model.
• We call $M$ consistent iff, for each OCL constraint $expr \in Inv(C)$, \[ \sigma \models expr \] for each “reasonable point” $(\sigma, \varepsilon)$ of computations of $M$.

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

Note: we could define $Inv(S)$ similar to $Inv(C)$.

---

OCL Constraints and Behaviour

• Let $M = (C, S, O)$ be a UML model.
• We call $M$ consistent iff, for each OCL constraint $expr \in Inv(C)$,

\[ \sigma \models expr \] for each “reasonable point” $(\sigma, \varepsilon)$ of computations of $M$.

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

Note: we could define $Inv(S)$ similar to $Inv(C)$.

Pragmatics:

• In UML-as-blueprint mode, if $S$ doesn’t exist yet, then $M = (C, \emptyset, O)$ is typically asking the developer to provide $S$ such that $M' = (C, S, O)$ is consistent.

If the developer makes a mistake, then $M'$ is inconsistent.

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

IN EACH SYSTEM STATE $\sigma_i$, FOR EACH ALIVE OBJECT $u \in dom(\sigma_i), u \in D(\sigma_i)$, EACH OF $u$'S ATTRIBUTES HAS A (DEFINITE) VALUE!

\[ \forall v \in atk(\sigma_i) \cdot \sigma(u)(v) \in D(\theta_q(u)) \]