Software Design, Modelling and Analysis in UML

Lecture 17: Hierarchical State Machines Ib

2015-01-20

Prof. Dr. Andreas Podelski, **Dr. Bernd Westphal**

Albert-Ludwigs-Universität Freiburg, Germany

Contents & Goals

Last Lecture:

- State Machines and OCL
- Rhapsody Demo II

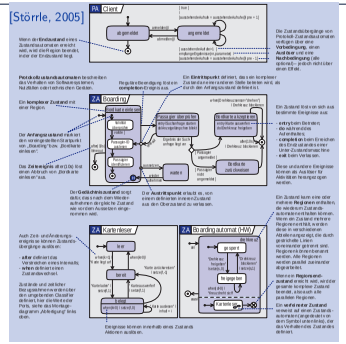
This Lecture:

- **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:**
  - Hierarchical State Machines Syntax
  - Initial and Final State
  - Composite State Semantics
  - The Rest

Hierarchical State Machines

UML State-Machines: What do we have to cover?

[Störrle, 2005]



The Full Story

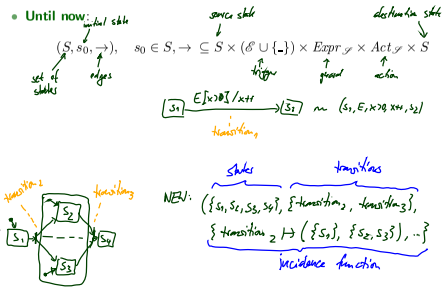
UML distinguishes the following **kinds of states**:

	example
reserved keywords	
simple state	
final state	
composite state	
OR	
AND	

	example
pseudo-state	
initial	
(shallow) history	H
deep history	H*
fork/join	
junction, choice	
entry point	
exit point	
terminate	
submachine state	$S:s$

Representing All Kinds of States

- **Until now:** $(S, s_0, \to)$, $s_0 \in S, \to \subseteq S \times (\mathcal{E} \cup \{\_\}) \times Expr_\mathcal{Y} \times Act_\mathcal{Y} \times S$



Representing All Kinds of States

- **Until now**:

$$(S, s_0, \to), \quad s_0 \in S, \to \subseteq S \times (\mathscr{E} \cup \{\_\}) \times Expr_{\mathscr{D}} \times Act_{\mathscr{D}} \times S$$

- **From now on**: **(hierarchical) state machines**

$$(S, kind, region, \to, \psi, annot)$$

where

- $S \supseteq \{top\}$ is a finite set of states **(as before)**,
- $kind : S \to \{st, init, fin, shist, dhist, fork, join, junc, choi, ent, exi, term\}$ is a function which labels states with their **kind**, **(new)**
- $region : S \to 2^{2^S}$ is a function which characterises the **regions** of a state, **(new)**
- $\to$ is a set of transitions **(changed)**
- $\psi : (\to) \overset{1}{\mapsto} 2^S \times 2^S$ is an **incidence function**, and **(new)**
- $annot : (\to) \to (\mathscr{E} \cup \{\_\}) \times Expr_{\mathscr{D}} \times Act_{\mathscr{D}}$ provides an annotation for each transition. **(new)**

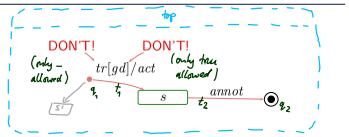
From UML to Hierarchical StM: By Example

$(S, kind, region, \to, \psi, annot)$

	example	$\in S$	$kind$	$region$
simple state (nothing nested within)	s	s	st	$\emptyset$
final state		q	fin	$\emptyset$
composite state				
OR		s	st	$\{\{s_1, s_2, s_3\}\}$
AND		s	st	$\{\{s_1, s_1'\}, \{s_3, s_3'\}, \{s_2, s_2'\}\}$
submachine state	(later)			
pseudo-state	$\bullet$, $\textcircled{H}$, ...	q	$init, shist, \ldots$	$\emptyset$

$(s, kind(s))$ for short

From UML to Hierarchical StM: By Example



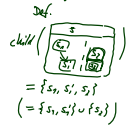
... translates to $(S, kind, region, \to, \psi, annot) =$

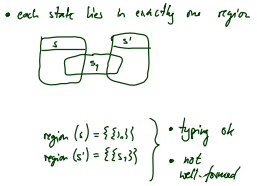
$$\Big(\underbrace{\{(s, st), (q_1, init), (q_2, fin), (top, st)\}}_{S,kind},$$
$$\underbrace{\{q_2 \mapsto \emptyset, q_1 \mapsto \emptyset, s \mapsto \emptyset, top \mapsto \{\{q_1, s, q_2\}\}\}}_{region},$$
$$\underbrace{\{t_1, t_2\}}_{\to}, \quad \underbrace{\{t_1 \mapsto (\{q_1\}, \{s\}), t_2 \mapsto (\{s\}, \{q_2\})\}}_{\psi},$$
$$\underbrace{\{t_1 \mapsto (\_, gd, act), t_2 \mapsto annot\}}_{annot}\Big)$$

Well-Formedness: Regions (follows from diagram)

	$\in S$	$kind$	$region \subseteq 2^S, S_i \subseteq S$	$child \subseteq S$
simple state	s	st	$\emptyset$	$\emptyset$
final state	s	fin	$\emptyset$	$\emptyset$
composite state	s	st	$\{S_1, \ldots, S_n\}, n \geq 1$	$S_1 \cup \cdots \cup S_n$
pseudo-state	s	$init, \ldots$	$\emptyset$	$\emptyset$
implicit top state	top	st	$\{S_1\}$	S_1

- Each state (except for top) lies in exactly one region,
- States $s \in S$ with $kind(s) = st$ **may comprise** regions.
  - No region: simple state.
  - One region: OR-state. } composite state
  - Two or more regions: AND-state.
- Final and pseudo states **don't comprise** regions.
- The region function induces a **child** function.

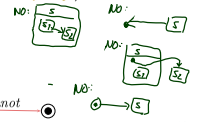




Well-Formedness: Initial State (requir. on diagram)

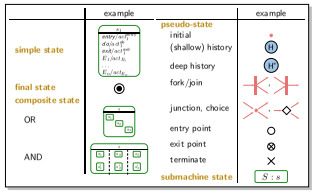
- Each non-empty region has exactly one initial pseudo-state and at least one transition from there, i.e.
  - for each $s \in S$ with $region(s) = \{S_1, \ldots, S_n\}$, $n \geq 1$, for each $1 \leq i \leq n$,
  - there exists exactly one initial pseudo-state $(s_1^i, init) \in S_i$ and at least one transition $t \in \to$ with s_1^i as source,
  - and such transition's target s_2^i is in S_i, and (**for simplicity!**) $kind(s_2^i) = st$, and $annot(t) = (\_, true, act)$.

- No ingoing transitions to initial states.
- No outgoing transitions from final states (**for simplicity!**).



- Recall:

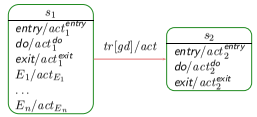
Plan



- Entry/do/exit actions, internal transitions.
- Initial pseudostate, final state.
- Composite states.
- History and other pseudostates, the rest.

Entry/Do/Exit Actions, Internal Transitions

Entry/Do/Exit Actions



- In general, with each state $s \in S$ there is associated
  - an **entry**, a **do**, and an **exit** action (default: **skip**)
  - a possibly empty set of trigger/action pairs called **internal transitions**, (default: empty).
    Note: $E_1, \ldots, E_n \in \mathscr{E}$, 'entry', 'do', 'exit' are reserved names!
- Recall: each action's supposed to have a transformer. Here: $t_{act_1^{entry}}$, $t_{act_1^{exit}}$, ...
- Taking the transition above then amounts to applying

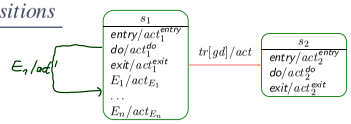
$$t_{act_2^{entry}} \circ t_{act} \circ t_{act_1^{exit}}$$

instead of only

$$t_{act}$$

$\rightsquigarrow$ adjust (2.), (3.) accordingly.

Internal Transitions

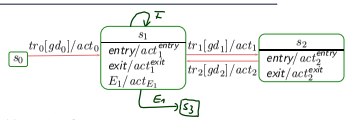


- For **internal transitions**, taking the one for E_1, for instance, **still** amounts to taking **only** $t_{act_{E_1}}$.
- Intuition: The state is neither left nor entered, so: no exit, no entry.

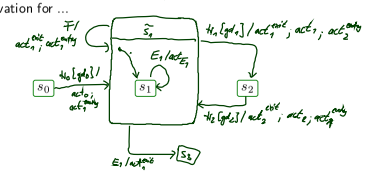
  $\rightsquigarrow$ adjust (2.) accordingly.
- **Note**: internal transitions also start a run-to-completion step.
- **Note**: the standard seems not to clarify whether internal transitions have **priority** over regular transitions with the same trigger at the same state.

  Some code generators assume that internal transitions have priority!

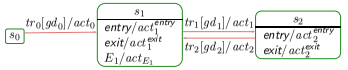
Alternative View: ... as Abbreviations



- ... as abbreviation for ...



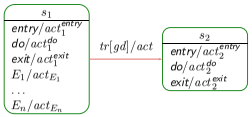
Alternative View: ... as Abbreviations



- ... as abbreviation for ...

- That is: Entry/Internal/Exit don't add expressive power to Core State Machines.
  If internal actions should have priority, s_1 can be embedded into an OR-state (later).
- Abbreviation view may avoid confusion in context of hierarchical states (later).

Do Actions



- **Intuition**: after entering a state, start its do-action.
- If the do-action terminates,
  - then the state is considered **completed** (→ later),
- otherwise,
  - if the state is left before termination, the do-action is stopped.

- Recall the overall UML State Machine philosophy:

  "An object is either idle or doing a run-to-completion step."

- Now, what is it exactly while the do action is executing...?

References

[Crane and Dingel, 2007] Crane, M. L. and Dingel, J. (2007). UML vs. classical vs. rhapsody statecharts: not all models are created equal. *Software and Systems Modeling*, 6(4):415–435.

[Damm et al., 2003] Damm, W., Josko, B., Votintseva, A., and Pnueli, A. (2003). A formal semantics for a UML kernel language 1.2. IST/33522/WP 1.1/D1.1.2-Part1, Version 1.2.

[Fecher and Schönborn, 2007] Fecher, H. and Schönborn, J. (2007). UML 2.0 state machines: Complete formal semantics via core state machines. In Brim, L., Haverkort, B. R., Leucker, M., and van de Pol, J., editors, *FMICS/PDMC*, volume 4346 of *LNCS*, pages 244–260. Springer.

[Harel and Kugler, 2004] Harel, D. and Kugler, H. (2004). The rhapsody semantics of statecharts. In Ehrig, H., Damm, W., Große-Rhode, M., Reif, W., Schnieder, E., and Westkämper, E., editors, *Integration of Software Specification Techniques for Applications in Engineering*, number 3147 in LNCS, pages 325–354. Springer-Verlag.

[OMG, 2007] OMG (2007). Unified modeling language: Superstructure, version 2.1.2. Technical Report formal/07-11-02.

[Störrle, 2005] Störrle, H. (2005). *UML 2 für Studenten*. Pearson Studium.