

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

– 17 – 2015-01-20 – main –

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

– 17 – 2015-01-20 – Prelim –

Hierarchical State Machines

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

[Störrle, 2005]

Wenn der **Endzustand** eines Zustandsautomaten erreicht wird, wird die Region beendet, in der der Endzustand liegt.

Die Zustandsübergänge von Protokoll-Zustandsautomaten verfügen über eine **Vorbedingung**, einen **Auslöser** und eine **Nachbedingung** (alle optional) – jedoch nicht über einen Effekt.

Ein **Eintrittspunkt** definiert, dass ein komplexer Zustand an einer anderen Stelle betreten wird, als durch den Anfangszustand definiert ist.

ZA Boarding/

Ein **komplexer Zustand** mit einer Region.

Der **Anfangszustand** markiert den voreingestellten Startpunkt von „Boarding“ bzw. „Bordkarte einlesen“.

Das **Zeitereignis** `after(10s)` löst einen Abbruch von „Bordkarte einlesen“ aus.

Der **Gedächtniszustand** sorgt dafür, dass nach dem Wiederaufnehmen der gleiche Zustand wie vor dem Aussetzen eingenommen wird.

Der **Austrittspunkt** erlaubt es, von einem definierten inneren Zustand aus den Oberzustand zu verlassen.

Ein Zustand löst sich aus bestimmte Ereignisse aus:

- **entry** beim Betreten;
- **do** während des Aufenthaltes;
- **completion** beim Erreichen des Endzustandes einer Unter-Zustandsmaschine
- **exit** beim Verlassen.

Diese und andere Ereignisse können als Auslöser für Aktivitäten herangezogen werden.

Ein Zustand kann eine oder mehrere **Regionen** enthalten, die wiederum Zustandsautomaten enthalten können. Wenn ein Zustand mehrere Regionen enthält, werden diese in verschiedenen Aktivitäten angezeigt, die durch gestrichelte Linien voneinander getrennt sind. Regionen können benannt werden. Alle Regionen werden parallel zueinander abgearbeitet.

Wenn ein **Regionsend-**zustand erreicht wird, wird der gesamte komplexe Zustand beendet, also auch alle parallelen Regionen.

Ein **verfeinerter Zustand** verweist auf einen Zustandsautomaten (angedeutet von dem Symbol unten links), der das Verhalten des Zustandes definiert.

ZA Kartenleser

Auch Zeit- und Änderungsereignisse können Zustandsübergänge auslösen:

- **after** definiert das Verstreichen eines Intervalls;
- **when** definiert einen Zustandswechsel.

Zustände und zeitlicher Bezugsrahmen werden über den umgebenden Classifier definiert, hier die Werte der Ports, siehe das Montage-diagramm „Abfertigung“ links oben.

Ereignisse können innerhalb eines Zustandes Aktionen auslösen.

ZA Boardingautomat (HW)

Ein **Regionsend-**zustand erreicht wird, wird der gesamte komplexe Zustand beendet, also auch alle parallelen Regionen.

Ein **verfeinerter Zustand** verweist auf einen Zustandsautomaten (angedeutet von dem Symbol unten links), der das Verhalten des Zustandes definiert.

The Full Story

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

	example		example
simple state	<p>received keywords $E \ni$</p>	pseudo-state	<ul style="list-style-type: none"> initial: (shallow) history: deep history: fork/join: junction, choice: entry point: exit point: terminate:
final state		submachine state	
composite state			
OR			
AND			

Representing All Kinds of States

• **Until now:**

(S, s_0, \rightarrow) , $s_0 \in S, \rightarrow \subseteq S \times (\mathcal{E} \cup \{-\}) \times Expr_{\mathcal{G}} \times Act_{\mathcal{G}} \times S$

set of states \uparrow S , *initial state* \uparrow s_0 , *edges* \uparrow \rightarrow , *source state* \uparrow S , *trigger* \uparrow $\mathcal{E} \cup \{-\}$, *guard* \uparrow $Expr_{\mathcal{G}}$, *action* \uparrow $Act_{\mathcal{G}}$, *destination state* \uparrow S

$[s_1] \xrightarrow{E[x>0]/x++} [s_2] \approx (s_1, E, x>0, x++, s_2)$

transition

NEW:

$(\{s_1, s_2, s_3, s_4\}, \{\text{transition}_2, \text{transition}_3\}, \{\text{transition}_2 \mapsto (\{s_1\}, \{s_2, s_3\}), \dots\})$

states \uparrow $\{s_1, s_2, s_3, s_4\}$, *transitions* \uparrow $\{\text{transition}_2, \text{transition}_3\}$, *incidence function* \uparrow $\{\text{transition}_2 \mapsto (\{s_1\}, \{s_2, s_3\}), \dots\}$

Representing All Kinds of States

- **Until now:**

$$(S, s_0, \rightarrow), \quad s_0 \in S, \rightarrow \subseteq S \times (\mathcal{E} \cup \{-\}) \times \text{Expr}_{\mathcal{G}} \times \text{Act}_{\mathcal{G}} \times S$$

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

$$(S, \text{kind}, \text{region}, \rightarrow, \psi, \text{annot})$$

where

- $S \supseteq \{\text{top}\}$ is a finite set of states (as before),
- $\text{kind} : S \rightarrow \{\text{st}, \text{init}, \text{fin}, \text{shst}, \text{dhist}, \text{fork}, \text{join}, \text{junc}, \text{choi}, \text{ent}, \text{exi}, \text{term}\}$ is a function which labels states with their **kind**, (new)
- $\text{region} : S \rightarrow 2^{2^S}$ is a function which characterises the **regions** of a state, (new)
- \rightarrow is a set of transitions (or: edges) – just names (changed)
- $\psi : (\rightarrow) \mapsto 2^S \times 2^S$ is an **incidence function**, and (new)
- $\text{annot} : (\rightarrow) \rightarrow (\mathcal{E} \cup \{-\}) \times \text{Expr}_{\mathcal{G}} \times \text{Act}_{\mathcal{G}}$ provides an annotation for each transition. (new)

- 17 - 2015-01-20 - Shiersyn -

6/37

From UML to Hierarchical StM: By Example

$$(S, \text{kind}, \text{region}, \rightarrow, \psi, \text{annot})$$

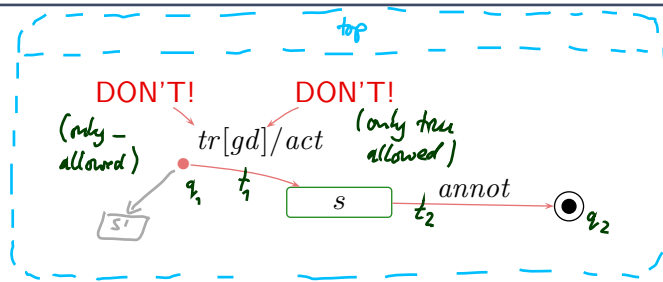
	example	$\in S$	kind	region
simple state (nothing nested within)		s	st	\emptyset
final state		q	fin	\emptyset
composite state				
OR		s	st	$\{\{s_1, s_2, s_3\}\}$
AND		s	st	$\{\{s_1, s_1'\}, \{s_3, s_3'\}, \{s_2, s_2'\}\}$
submachine state	(later) -	-	-	-
pseudo-state		q	init, shst, ...	\emptyset

$(s, \text{kind}(s))$ for short

- 17 - 2015-01-20 - Shiersyn -

7/37

From UML to Hierarchical StM: By Example



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

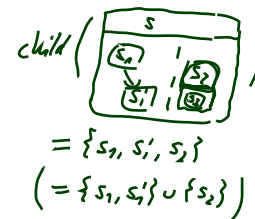
$$\begin{aligned}
 & \underbrace{(\{s, st\}, (q_1, init), (q_2, fin), (top, st))}_{S, kind} \\
 & \underbrace{\{q_2 \mapsto \emptyset, q_1 \mapsto \emptyset, s \mapsto \emptyset, top \mapsto \{\{q_1, s, q_2\}\}\}}_{region} \\
 & \underbrace{\{t_1, t_2\}, \{t_1 \mapsto (\{t_1\}, \{s\}), t_2 \mapsto (\{s\}, \{q_2\})\}}_{\psi} \\
 & \underbrace{\{t_1 \mapsto (tr, gd, act), t_2 \mapsto annot\}}_{annot}
 \end{aligned}$$

- 17 - 2015-01-20 - Shiersyn -

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	$\{S_1, \dots, S_n\}, n \geq 1$	$S_1 \cup \dots \cup S_n$
pseudo-state	s	$init, \dots$	\emptyset	\emptyset
implicit top state	top	st	$\{S_1\}$	S_1

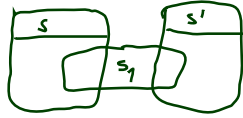
Def.



- 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.
 - Two or more regions: AND-state. } *composite states*
- Final and pseudo states **don't comprise** regions.
- The region function induces a **child** function.

- 17 - 2015-01-20 - Shiersyn -

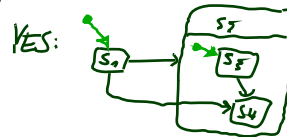
- each state lies in exactly one region



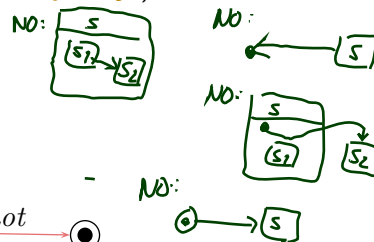
$$\left. \begin{array}{l} \text{region}(s) = \{\{s_1\}\} \\ \text{region}(s') = \{\{s_1, s_2\}\} \end{array} \right\} \begin{array}{l} \bullet \text{ typing ok} \\ \bullet \text{ not well-formed} \end{array}$$

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

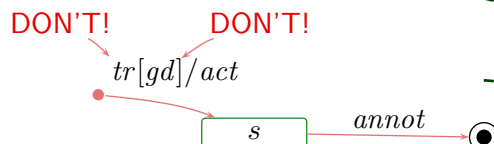
- Each non-empty region has exactly one initial pseudo-state and at least one transition from there, i.e.
 - for each $s \in S$ with $\text{region}(s) = \{S_1, \dots, S_n\}$, $n \geq 1$, for each $1 \leq i \leq n$,
 - there exists exactly one initial pseudo-state $(s_1^i, \text{init}) \in S_i$ and at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



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



- Recall:



Plan

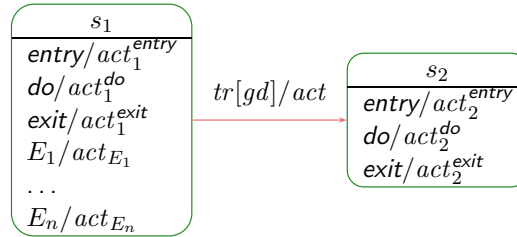
	example		example
simple state	<pre> s1 entry/act1^entry do/act1^do exit/act1^exit ... E1/actE1 ... En/actEn </pre>	pseudo-state	
final state		initial	
composite state		(shallow) history	
OR		deep history	
AND		fork/join	
		junction, choice	
		entry point	
		exit point	
		terminate	
		submachine state	

- 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:** $E_1, \dots, E_n \in \mathcal{E}$, 'entry', 'do', 'exit' are reserved names!



- Recall: each action's supposed to have a transformer. Here: $t_{act_1^{entry}}, t_{act_1^{exit}}, \dots$
- Taking the transition above then amounts to applying

$$t_{act_{s_2}^{entry}} \circ t_{act} \circ t_{act_{s_1}^{exit}}$$

instead of only

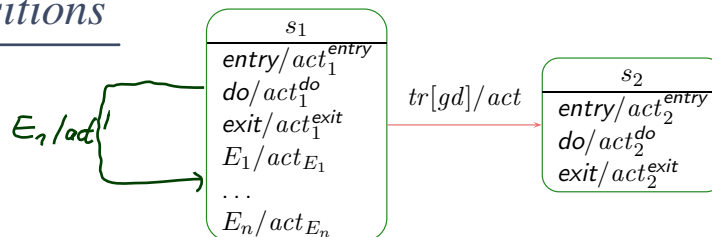
$$t_{act}$$

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

13/37

- 17 - 2015-01-20 - Sentryexit -

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.

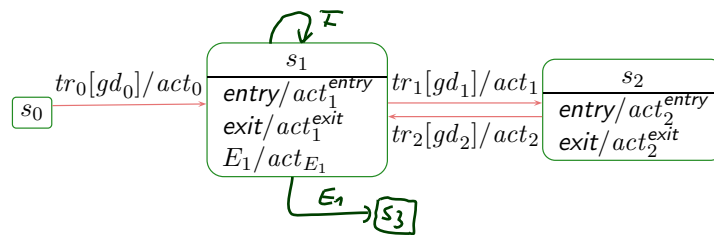
\rightsquigarrow adjust (2.) accordingly.

- 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!

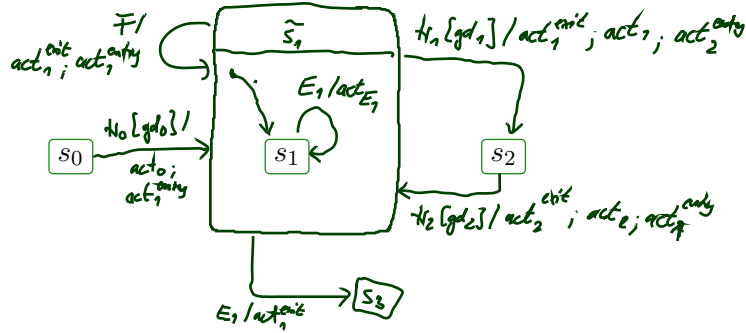
14/37

- 17 - 2015-01-20 - Sentryexit -

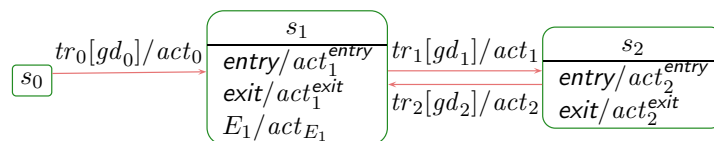
Alternative View: ... as Abbreviations



- ... as abbreviation for ...



Alternative View: ... as Abbreviations



- ... as abbreviation for ...

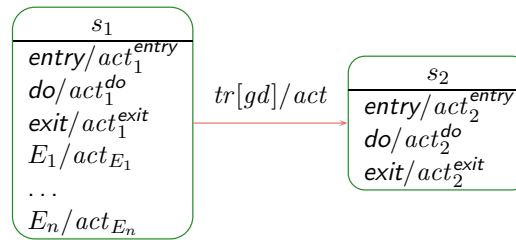
s0

s1

s2

- 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** (\rightarrow 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.