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

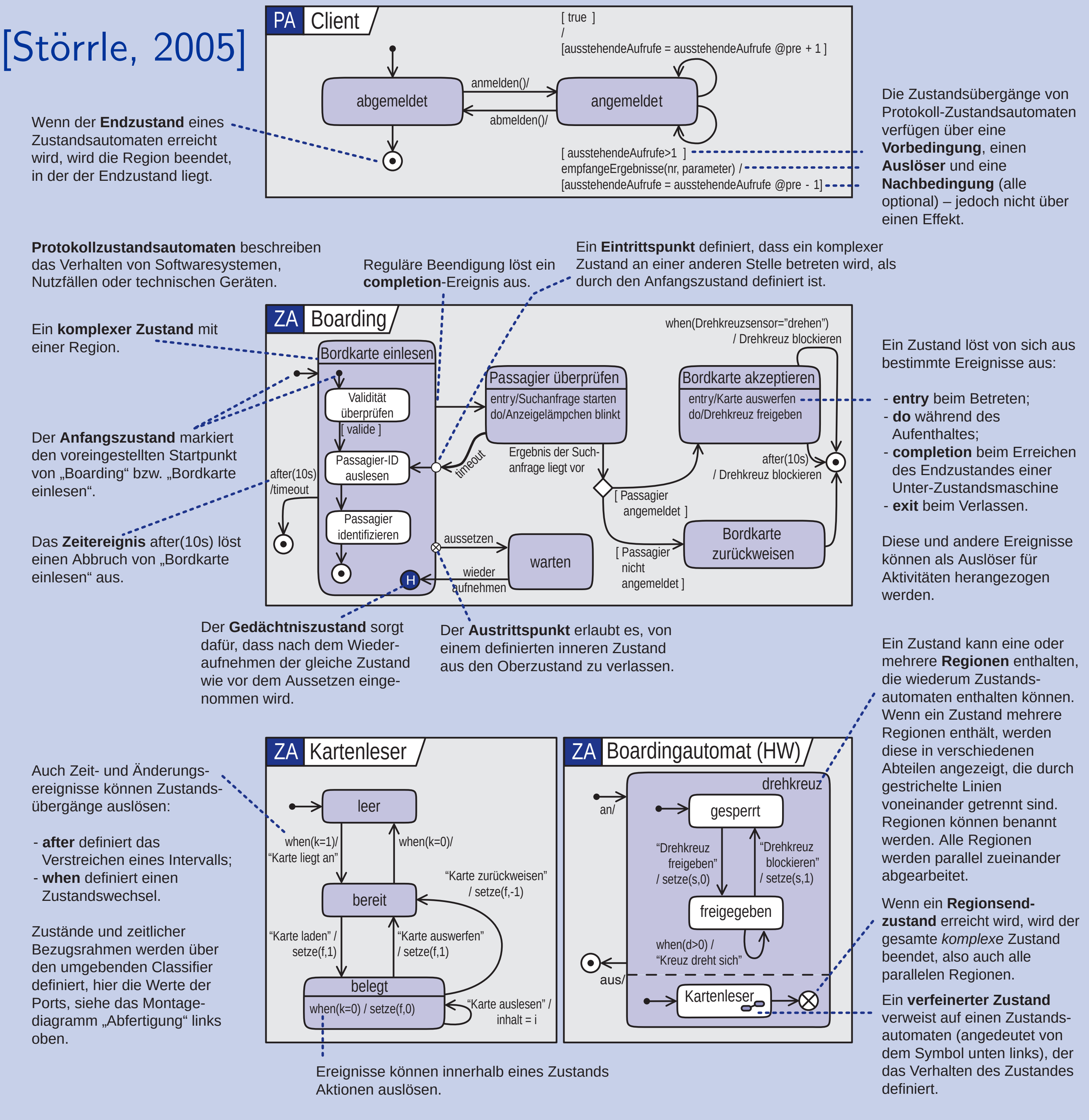
This Lecture:

- **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]

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.

Protokollzustandsautomaten beschreiben das Verhalten von Softwaresystemen, Nutzfällen oder technischen Geräten.

Reguläre Beendigung löst ein **completion**-Ereignis aus.

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

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.

Ein Zustand löst von 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.

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

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 Montagediagramm „Abfertigung“ links oben.

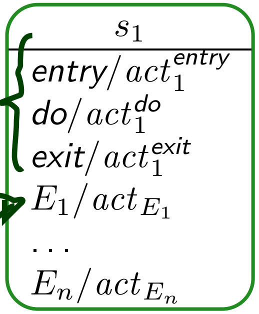
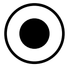
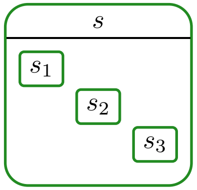
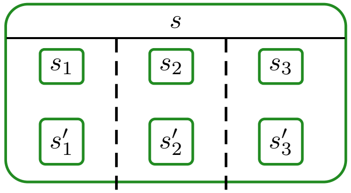
Wenn ein **Regionsendzustand** erreicht wird, wird der gesamte *komplexe* Zustand beendet, also auch alle parallelen Regionen.

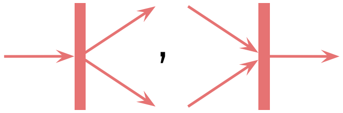
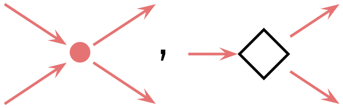
Ein **verfeinerter Zustand** verweist auf einen Zustandsautomaten (angedeutet von dem Symbol unten links), der das Verhalten des Zustandes definiert.

Ereignisse können innerhalb eines Zustands Aktionen auslösen.

The Full Story

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

	example
reserved keywords **simple state** E ∋	 s_1 entry/act_1^{entry} do/act_1^{do} exit/act_1^{exit} E_1/act_{E_1} ... E_n/act_{E_n}
final state	
composite state	
OR	
AND	 s s_1 s_2 s_3 s_1' s_2' s_3'

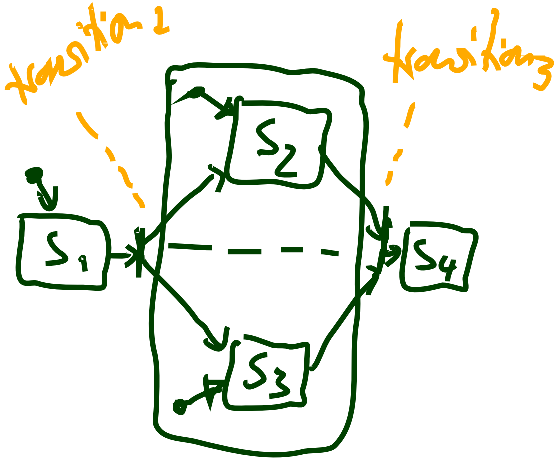
	example
pseudo-state	
initial	●
(shallow) history	H
deep history	H^*
fork/join	
junction, choice	
entry point	○
exit point	⊗
terminate	✕
submachine state	$S : s$

Representing All Kinds of States

- **Until now:**

$$(\underset{\text{set of states}}{S}, \overset{\text{initial state}}{s_0}, \underset{\text{edges}}{\to}), \quad s_0 \in S, \to \subseteq \overset{\text{source state}}{S} \times (\underset{\text{trigger}}{\mathscr{E} \cup \{\_\}}) \times \underset{\text{guard}}{Expr_{\mathscr{S}}} \times \underset{\text{action}}{Act_{\mathscr{S}}} \times \overset{\text{destination state}}{S}$$

$$\boxed{s_1} \xrightarrow[\text{transition}_1]{E[x>0]/x{+}{+}} \boxed{s_2} \rightsquigarrow (s_1, E, x>0, x{+}{+}, s_2)$$



$$\text{NEW: } (\overbrace{\{s_1, s_2, s_3, s_4\}}^{\text{states}}, \overbrace{\{\text{transition}_2, \text{transition}_3\}}^{\text{transitions}}, \underbrace{\{\text{transition}_2 \mapsto (\{s_1\}, \{s_2, s_3\}), \dots\}}_{\text{incidence function}})$$

Representing All Kinds of States

- **Until now**:

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

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

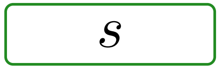
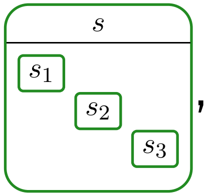
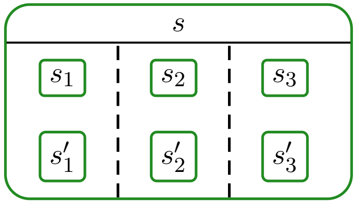
$$(S, kind, region, \to, \psi, annot)$$

where

- $S \supseteq \{\overset{\text{reserved name}}{top}\}$ 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, sets of sets of states **(new)**
- $\to$ is a set of transitions (or: edges) – just names **(changed)**
- $\psi : (\to) \to 2^S \times 2^S$ is an **incidence function**, and **(new)**
- $annot : (\to) \to (\mathscr{E} \cup \{\_\}) \times Expr_{\mathscr{S}} \times Act_{\mathscr{S}}$ provides an annotation for each transition. **(new)**

From UML to Hierarchical StM: By Example

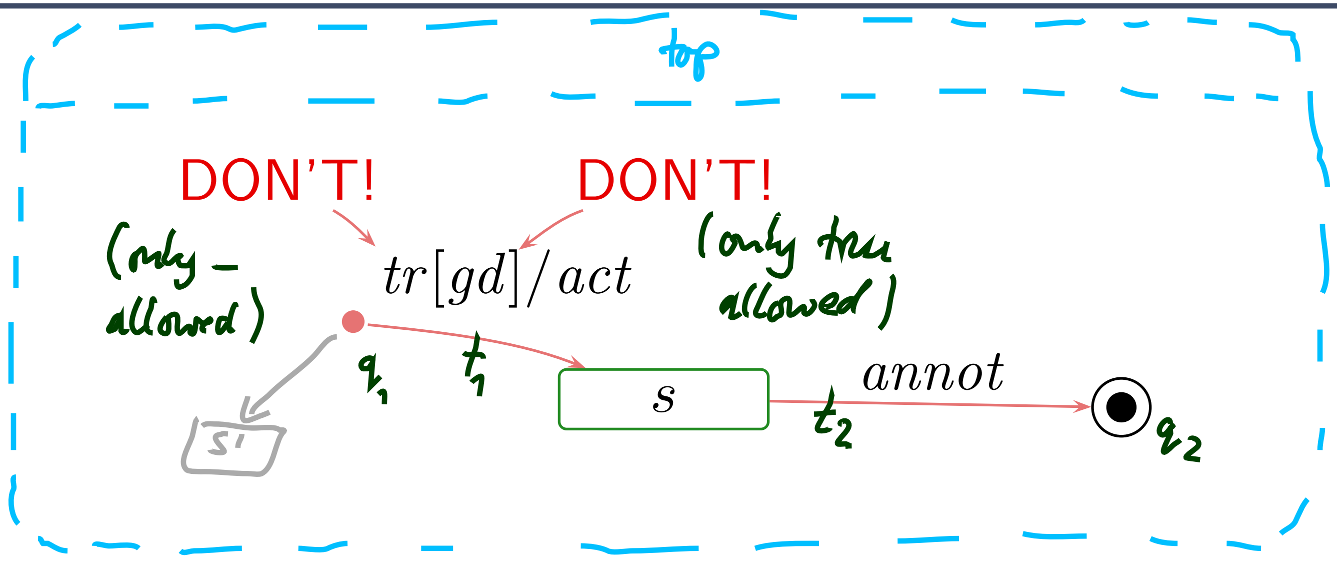
$(S, kind, region, \to, \psi, annot)$

	example	$\in S$	$kind$	$region$
simple state (nothing nested within)		s	st	$\emptyset$
final state	◉	fresh name → 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	$init, shist, \ldots$	$\emptyset$

region

$\underbrace{\in S \quad kind}_{(s, kind(s)) \text{ for short}}$

From UML to Hierarchical StM: By Example



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

$$\Big(\underbrace{\{(s, st), (q_1, init), (q_2, fin), (top, st)\}}_{S, kind},$$

$$\underbrace{\{q_2 \mapsto \emptyset, q_1 \mapsto \emptyset, s \mapsto \emptyset, top \mapsto \{\{q_1, s, q_2\}\}}_{region},$$

$$\underbrace{\{t_1, t_2\}}_{\to}, \quad \underbrace{\{t_1 \mapsto (\{q_1\}, \{s\}), t_2 \mapsto (\{s\}, \{q_2\})\}}_{\psi},$$

$$\underbrace{\{t_1 \mapsto (tr, gd, act), t_2 \mapsto annot \}}_{annot}\Big)$$

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

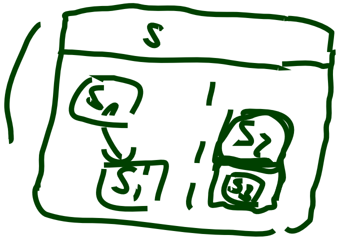
Def.

- 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.

child (

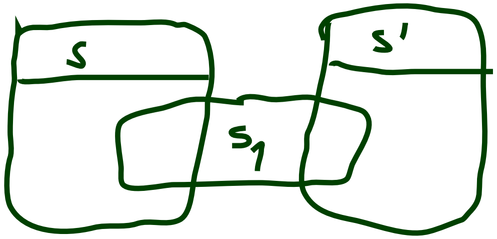


)

$= \{s_1, s_1', s_2\}$

$(= \{s_1, s_1'\} \cup \{s_2\})$

- each state lies in exactly one region

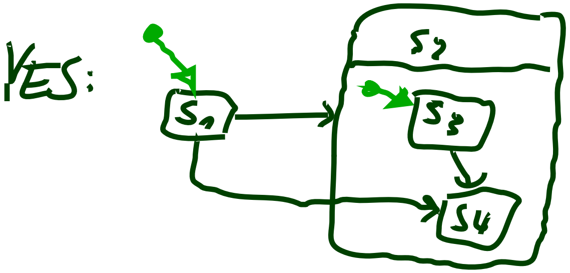


$$\left.\begin{array}{l} \text{region}(s) = \{\{s_1\}\} \\ \text{region}(s') = \{\{s_1\}\} \end{array}\right\}$$

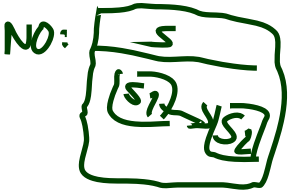
- typing ok
- not well-formed

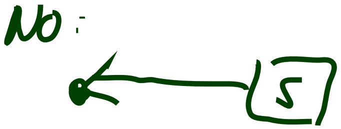
Well-Formedness: Initial State (requir. on diagram)

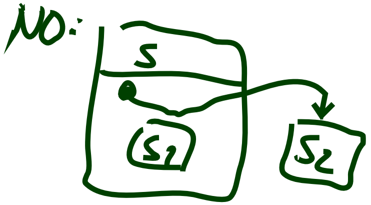
- Each non-empty region has exactly one initial pseudo-state and at least one transition from there, i.e.
  - for each $s \in S$ with $region(s) = \{S_1, \dots, S_n\}$, $n \geq 1$, for each $1 \leq i \leq n$,
  - there exists exactly one initial pseudo-state $(s_1^i, \mathit{init}) \in S_i$ and at least 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!**) $kind(s_2^i) = \mathit{st}$, and $annot(t) = (\_, \mathit{true}, act)$.

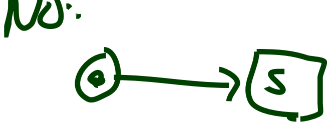


- No ingoing transitions to initial states.
- No outgoing transitions from final states (**for simplicity!**).

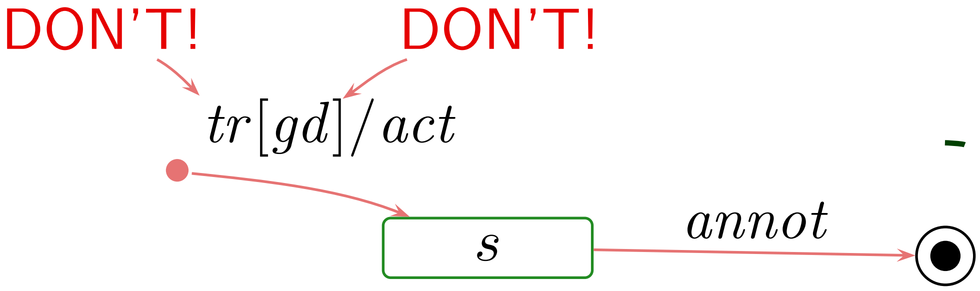




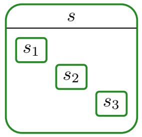
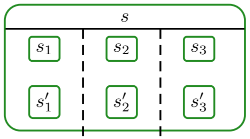


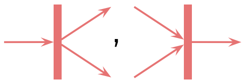
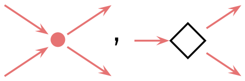


- Recall:



Plan

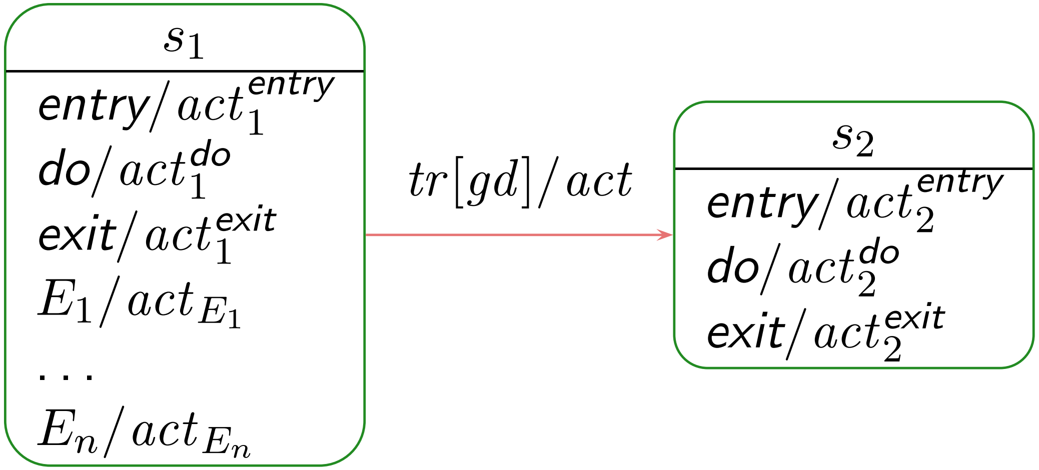
	example
simple state	s_1 entry/act_1^{entry} do/act_1^{do} exit/act_1^{exit} E_1/act_{E_1} ... E_n/act_{E_n}
final state	◉
composite state	
OR	
AND	

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

- 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



s_1

entry$/act_1^{entry}$
do$/act_1^{do}$
exit$/act_1^{exit}$
E_1/act_{E_1}
$\dots$
E_n/act_{E_n}

$tr[gd]/act$

s_2

entry$/act_2^{entry}$
do$/act_2^{do}$
exit$/act_2^{exit}$

- 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: $E_1, \dots, E_n \in \mathscr{E}$, '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

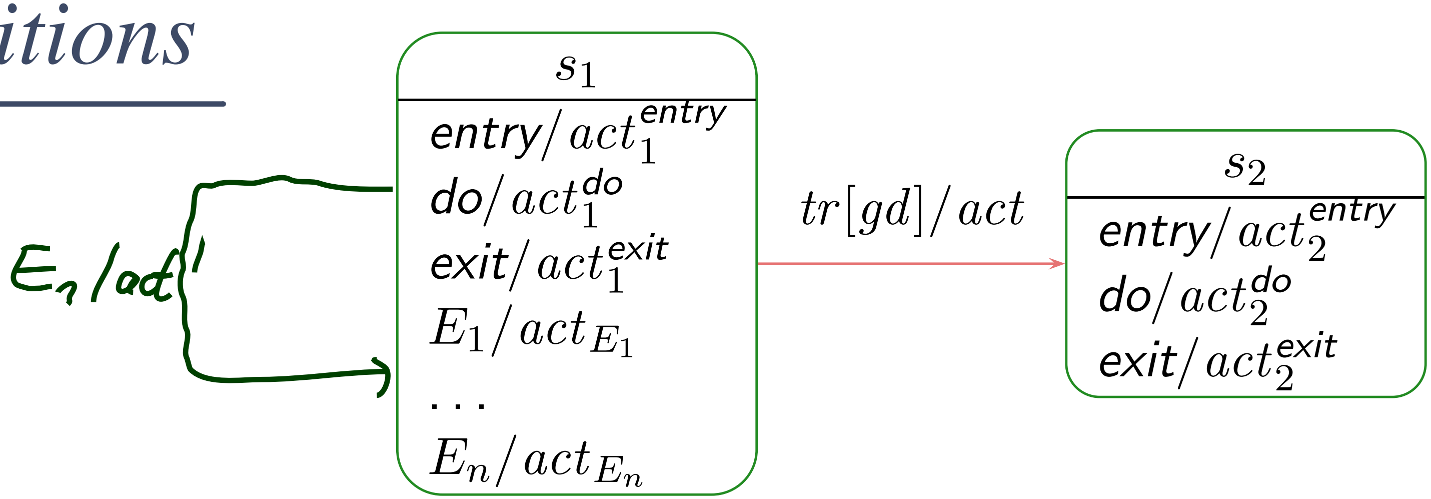
$$t_{act_{s_2}^{entry}} \circ t_{act} \circ t_{act_{s_1}^{exit}}$$

instead of only

$$t_{act}$$

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

Internal Transitions



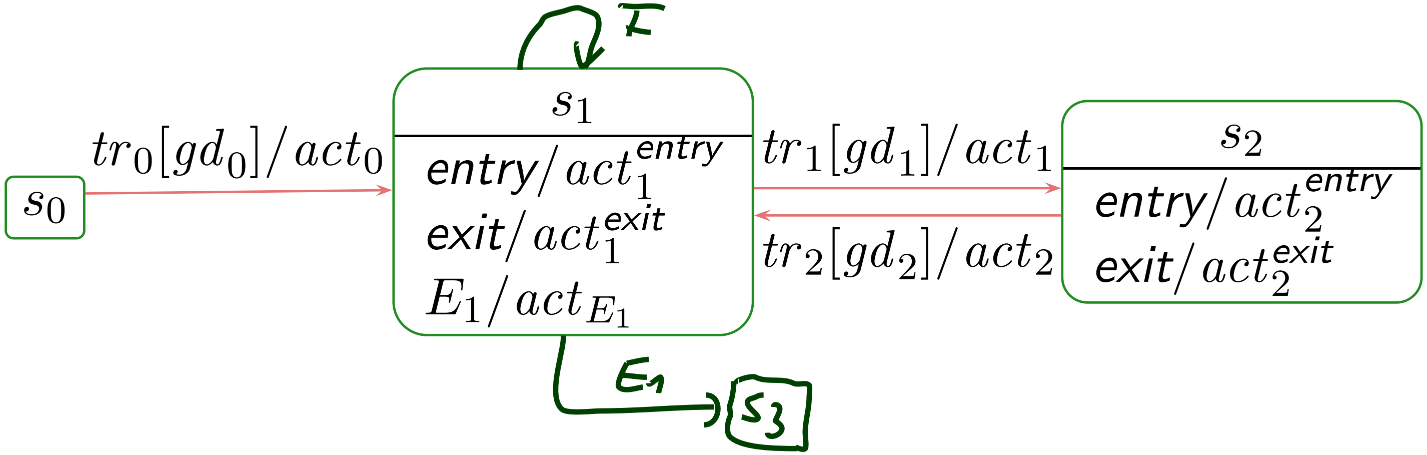
E_1/act'
s_1
$\mathsf{entry}/act_1^{\mathsf{entry}}$
$\mathsf{do}/act_1^{\mathsf{do}}$
$\mathsf{exit}/act_1^{\mathsf{exit}}$
E_1/act_{E_1}
$\dots$
E_n/act_{E_n}
$tr[gd]/act$
s_2
$\mathsf{entry}/act_2^{\mathsf{entry}}$
$\mathsf{do}/act_2^{\mathsf{do}}$
$\mathsf{exit}/act_2^{\mathsf{exit}}$

- 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



s_0

$tr_0[gd_0]/act_0$

s_1

entry$/act_1^{entry}$

exit$/act_1^{exit}$

E_1/act_{E_1}

$tr_1[gd_1]/act_1$

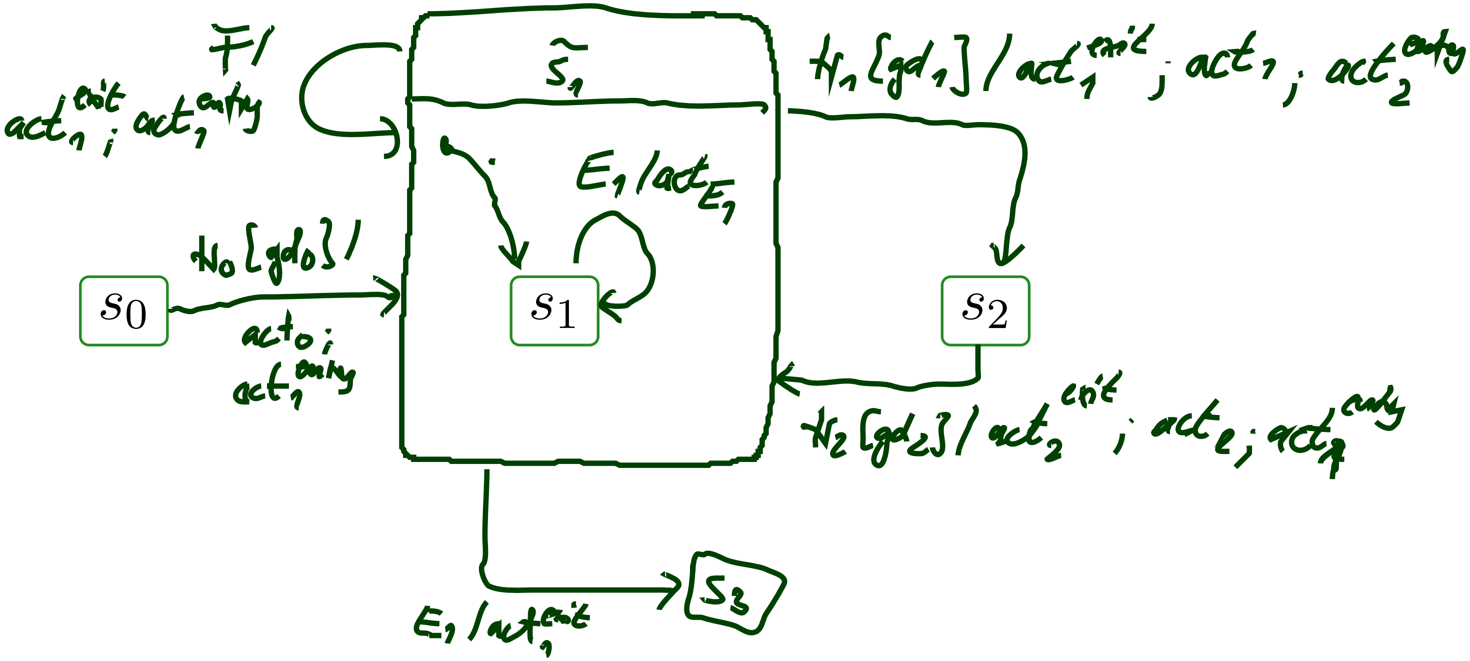
$tr_2[gd_2]/act_2$

s_2

entry$/act_2^{entry}$

exit$/act_2^{exit}$

- ... as abbrevation for ...



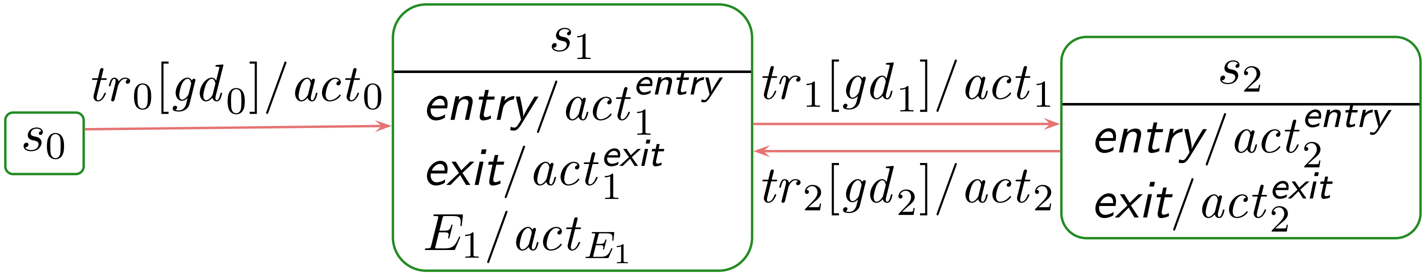
s_0

s_1

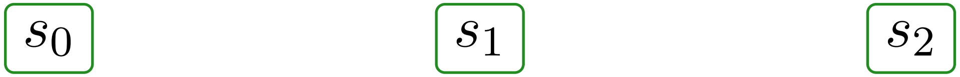
s_2

F/ act1exit; act1entry
s̃1
E1/actE1
tr0[gd0]/ act0; act1entry
tr1[gd1]/act1exit; act1; act2entry
tr2[gd2]/act2exit; act2; act1entry
E1/act1exit
S3

Alternative View: ... as Abbreviations

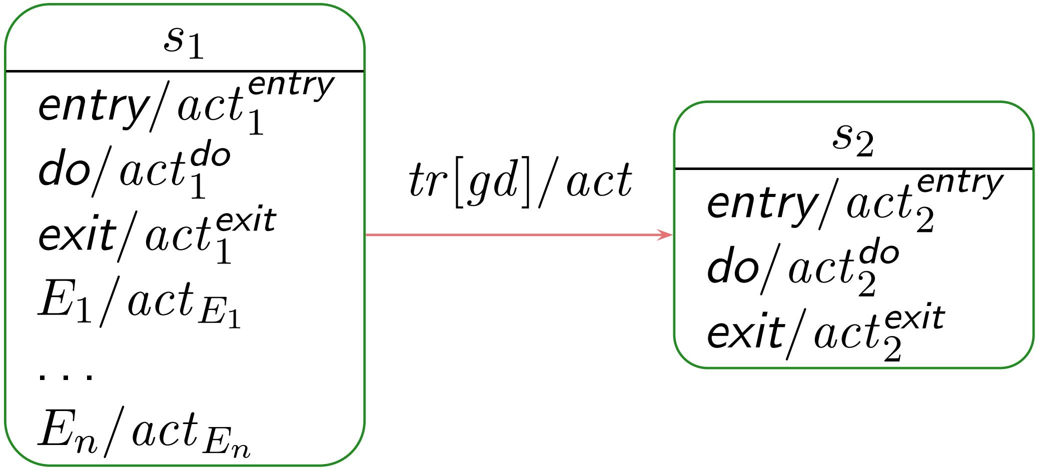


- ... as abbrevation for ...



- 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.