

- In a sense, composite states are about
 - abstraction,
 - structuring and
 - avoiding redundancy.
- Ideas instead of

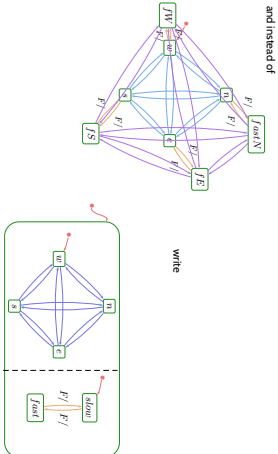
Content

- Hierarchical State Machines
 - Additional Well-Formedness Constraints
 - Adding **initial** for hierarchical states
 - Entry and Exit Actions
 - Initial and Final States
- Rhapsody Demo: Automated Tests
- Hierarchical State Machines: The Rest
 - History Connectors
 - Partition and Choice
 - Entry and Exit Points
 - Terminate
- Active vs. Passive Objects

Additional Well-Formedness Constraints

- Each non-empty region has **exactly one** initial pseudo-state and **at least one** transition from there to a state of the region, i.e.
 - for each $s \in S$ with $region(s) = \{S_1, \dots, S_n\}$,
 - for each $1 \leq i \leq n$, there exists exactly one initial pseudo-state $\{s_i\}$, $\{s_i\} \in S$, and
 - at least one transition $t \in T$ with s_i as source.
 - Initial pseudo-states are not targets of transitions.
- For simplicity:
- The target of a transition with initial pseudo-state source in S_i is also in S_i .
 - Transitions from initial pseudo-states have no trigger or guard, i.e. $t \in T$ from s with $kind(t) = \text{af}$ implies $guard(t) = \perp$, $trigger(t) = \perp$.
 - Final states are not sources of transitions.
-

An Intuition for "And-States"

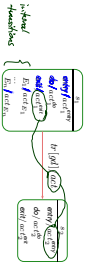


- In a sense, composite states are about
 - abstraction,
 - structuring and
 - avoiding redundancy.
- Ideas instead of

Entry and Exit Actions

Entry/Do/Exit Actions

- In general, with each state $s \in S$ there is associated:
 - an entry action and an exit
 - a possibly empty set of operations called **internal transitions** (default empty)



- Recall: each action is supposed to have a transformer: assume $f_{e_1, s_1}, f_{e_2, s_2}, \dots$
- Taking the transition above then amounts to applying:
 - $f_{e_1, s_1} \circ f_{x_1} \circ f_{e_2, s_2}$

$$f_{e_1, s_1} \circ f_{x_1} \circ f_{e_2, s_2}$$

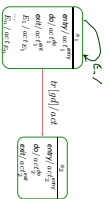
instead of just

$$f_{x_1}$$

→ adjust Rules (ii), (iii), and (v) accordingly.

7/29

Internal Transitions

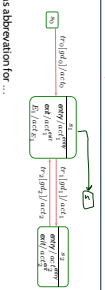


- Taking an **Internal Transition**, e.g. on E_1 , only executes f_{E_1, s_1} .
- **Initiation**: The state is neither left nor entered, so no exit, no entry action.
- **Note**: internal transitions also start as run-to-completion step.
- adjust Rules (ii), (iii), and (v) accordingly.

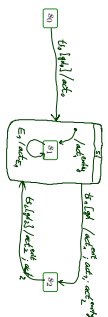
Note: this standard seems not to clarify whether internal transitions have **priority** over regular transitions with the same trigger at the same state.
Some code generators assume that internal transitions have priority!

8/29

Alternative View: Entry / Exit / Internal as Abbreviations



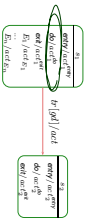
Can be viewed as abbreviation for ...



- That's Entry / Internal / Exit don't add expressive power to Core State Machines.
- If internal actions should have priority, it can be embedded in an On-state.
- The "abbreviation view" may avoid confusion in the context of hierarchical states.

9/29

Do Actions



- mutation: after entering a state, start its do-action.
- If the do-action terminates:
 - then the state is considered **completed** (like reaching a final state child → in a model).
 - then, **note (c2) (Contexts)** may apply
 - 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? "

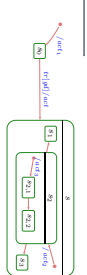
10/29

Initial and Final States

11/29

11/29

Initial Pseudostate



- Principle
 - when entering a non-simple state.
 - then go to the destination state of a transition with initial pseudo-state source.
 - execute the action of the chosen initiation transition(s) between exit and entry actions.
- Recall: For simplicity, we assume exactly one initiation transition per non-emptiness region.
Could also be: "at least one" and choosing one non-deterministically.

Special case: the region of h_1 .

- If class C has a state-machine then "create-C-transformer" is the concatenation of
 - the transformer of the "connector" of C (see not included explicitly) and
 - a transformer corresponding to over-entrance transition of the top region.

$$f_{h_1} \circ f_{e_1} \circ f_{x_1}$$

12/29

Final States



- If (σ, s) $\xrightarrow{event, state}$ (σ', s')
 - and all **simple states** s in $\sigma'(s)$ are **final** i.e. $final(s) = final$ then
 - stay **inside** if there is a common parent of the simple states in $\sigma'(s)$ which is source of a transition without trigger and satisfied guard.
 - otherwise **kill** (destroy) object σ .
- adjust Rules (I), (II), (III) and (V) accordingly.

Observation: a "newer" survives "reaching" a state $(s, final)$ with $s \in child(top)$.

Observation



vs.



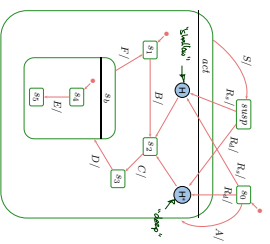
13/07

Rhapsody Demo: Automated Testing

14/07

15/07

History and Deep History: By Example

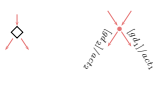
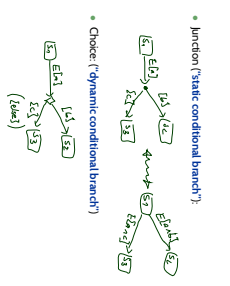


16/07

17/07

- What happens on...
- R1?
 - R2?
 - S0, S1
 - R1?
 - A, B, C, S, R1?
 - S0, S1, S2, S3, S4, S5
 - A, B, C, S, R1?
 - S1, S2, S3, S4, S5, S6, S7, S8, S9, S10
 - A, B, C, D, E, S, R1?

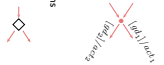
Junction and Choice



18/07

Junction and Choice

- Junction ("static conditional branch")
 - good abbreviation
 - the added transitions are then checked for enable-ness
 - at best start with trigger branch into conditions, then apply actions
- Choice ("dynamic conditional branch")
 - **evil**: may get stuck
 - enters the transition **without knowing** whether there's an enable path
 - at best use "clear" and convince yourself that it cannot get stuck
 - maybe even better: **avoid**



18/20

Entry and Exit Point, Submachine State, Terminate

- Hierarchical states can be "folded" for readability (but: this can also hinder readability)
- Can even be taken from a different state-machine for re-use.



S 7.8

19/20

Entry and Exit Point, Submachine State, Terminate

- Hierarchical states can be "folded" for readability (but: this can also hinder readability)
- Can even be taken from a different state-machine for re-use.
- **Entry/exit points**
- Provide connection points for finer integration into the current level, finer than just via initial state
- Semantically a bit tricky:
 - First the exit action of the exiting state
 - then the actions of the transition,
 - then the entry actions of the entered state
 - then action of the transition from the entry point to an internal state
 - and then the internal state's entry action.
- **Terminate Pseudo-State**
 - When a terminate pseudo-state is reached, the object taking the transition is immediately killed.

Remark: 

19/20

Tell Them What You've Told Them...

- OR- and AND-states could also be explained as an "unfolding" into core state machines.
- They add **concerns**, not **expressive power**.
- The remaining pseudo-states (history, junction, choice, etc.) are not so difficult
- **Modelling guideline: Avoid choice**
- **Rhapsody** also supports **non-active objects** - their instances share an event pool with an **active object**

27/20

References

28/20

References

- Harel, D. and Gery, E. (1997). Executable object modeling with statecharts. *IEEE Computer*, 3:07/31-42.
- OMG (2011a). Unified modeling language infrastructure version 2.1. Technical Report formal/2011-08-05.
- OMG (2011b). Unified modeling language Superstructure version 2.1. Technical Report formal/2011-08-06.

29/20