Contents & Goals

Last Lecture:
- Hierarchical State Machines: partial order, “ls”, orthogonality, …

This Lecture:
- Educational Objectives: Capabilities for following tasks/questions.
  - 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:
  - Legal Transitions
  - Exit/Entry, internal transitions
  - History and others
  - Rhapsody Demo

Legal Transitions

A hierarchical state-machine \((S, \text{kind}, \text{region}, \rightarrow, \psi, \text{annot})\) is called well-formed if and only if for all transitions \(t \in \rightarrow\):
- source and destination are consistent, i.e. \(\downarrow \text{source}(t) \) and \(\downarrow \text{target}(t)\),
- for all \(s, s' \in \text{source}(t)\) \(s \perp s'\),
- the top state is neither source nor destination, i.e.
  \(\text{top} \not\in \text{source}(t) \bigcup \text{target}(t)\),
- Recall: final states are not source of transitions.

Example:

\[
\begin{array}{c}
\text{top} \\
\text{source} \\
\text{target} \\
\text{do} \\
\text{entry} \\
\text{exit}
\end{array}
\]

The Depth of States

- \(\text{depth}(\text{top}) = 0\),
- \(\text{depth}(s') = \text{depth}(s) + 1\), for all \(s' \in \text{do}(s)\)

Example:

Enabledness in Hierarchical State-Machines

- The scope (‘set of possibly affected states’) of a transition \(t\) is the least common region of \(\text{source}(t) \cup \text{target}(t)\).
- Two transitions \(t_1, t_2\) are called consistent if and only if their scopes are orthogonal (i.e. states in scopes pairwise orthogonal).
- The priority of transition \(t\) is the depth of its innermost source state, i.e.
  \(\text{prio}(t) := \max \{\text{depth}(s) \mid s \in \text{source}(t)\}\)
- A set of transitions \(T \subseteq \rightarrow\) is enabled in an object \(s\) if and only if:
  - \(T\) is consistent,
  - \(T\) is maximal w.r.t. priority,
  - all transitions in \(T\) share the same trigger,
  - all guards are satisfied by \(\sigma(s)\), and
  - for all \(t \in T\), the source states are active, i.e.
  \(\text{source}(t) \subseteq \sigma(s)(\text{do}(t)) \subseteq S\).
Some code generators assume that internal transitions have priority! Abbreviation may avoid confusion in context of hierarchical states (see later).

Internal Transitions

- For internal transitions, taking the one for $E_k$ for instance, still amounts to taking only $E_k$.
- Intention: The state is neither left nor entered, so: no exit, no entry.
  - adjust (2), (3), (5) 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!

Entry/Do/Exit Actions, Internal Transitions

- In general, with each state $s$, if $\tau$ is assigned
  - an entry, a do, and an exit action (default: skip)
  - a possibly empty set of trigger/action pairs called internal transitions.
  - (default: empty).
- Recall: each action’s supposed to have a transformer. Here: $T_{E_1}, T_{E_2}, \ldots$
- Taking the transition above then amounts to applying
  $T_{E_1} \circ \cdots \circ T_{E_n} \circ T_{a_1} \cdots T_{a_k}$

Instead of only $T_{a_1} \cdots T_{a_k}$
  - adjust (2), (3) accordingly.

Alternative View: Entry/Exit/Internal as Abbreviations

- ... as abbreviation for ...

Note: internal transitions also start a run-to-completion step.

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

- **Initialization**: after entering a state, start its do-action.
- If the do-action terminates,
  - then the state is considered completed,
  - 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...?

**Junction and Choice**

- **Junction** ("static conditional branch")
  - *good*: abbreviation
  - unfolds to as many similar transitions with different guards, the unfolded transitions are then checked for enabledness
  - *evil*: may get stuck
  - gets stuck without knowing whether there’s an enabled path
  - at least see “else” and convince yourself that it cannot get stuck
  - maybe even better: avoid

- **Choice** ("dynamic conditional branch")
  - *good*: abbreviation
  - unfolds to as many similar transitions with different guards, the unfolded transitions are then checked for enabledness
  - *evil*: may get stuck
  - gets stuck without knowing whether there’s an enabled path
  - at least see “else” and convince yourself that it cannot get stuck
  - maybe even better: avoid

Note: not so sure about naming and symbols, e.g., I’d guessed it was just the other way round...

**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, 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 that internal state’s entry action.
- Terminate Pseudo-State
  - When a terminate pseudo-state is reached, the object taking the transition is immediately killed.

---

**The Concept of History, and Other Pseudo-States**

**History and Deep History: By Example**

**Contemporary UML Modelling Tools**
References


