Software Design, Modelling and Analysis in UML

Lecture 15: Hierarchical State Machines I

2016-01-14

Prof. Dr. Andreas Podelski, Dr. Bernd Westphal

Albert-Ludwigs-Universität Freiburg, Germany
Contents & Goals

Last Lecture:
- step, RTC-step, divergence
- initial state, UML model semantics (so far)
- create, destroy actions

This Lecture:
- **Educational Objectives:** Capabilities for following tasks/questions.
  - What is simple state, OR-state, AND-state?
  - What is a legal state configuration?
  - What is a legal transition?
  - How is enabledness of transitions defined for hierarchical state machines?

- **Content:**
  - Legal state configurations
  - Legal transitions
  - Rules (i) to (v) for hierarchical state machines
Hierarchical State-Machines
UML distinguishes the following kinds of states:

<table>
<thead>
<tr>
<th>Category</th>
<th>Example</th>
<th>pseudo-state</th>
<th>example</th>
</tr>
</thead>
<tbody>
<tr>
<td>simple state</td>
<td></td>
<td>initial (shallow) history</td>
<td><img src="#" alt="H" /></td>
</tr>
<tr>
<td>final state</td>
<td><img src="#" alt="s" /></td>
<td>deep history</td>
<td><img src="#" alt="H*" /></td>
</tr>
<tr>
<td>composite state</td>
<td><img src="#" alt="s" /></td>
<td>fork/join</td>
<td><img src="#" alt="junction" /></td>
</tr>
<tr>
<td>OR</td>
<td><img src="#" alt="s" /></td>
<td>junction, choice</td>
<td><img src="#" alt="entry point" />, <img src="#" alt="exit point" /></td>
</tr>
<tr>
<td>AND</td>
<td><img src="#" alt="s" /></td>
<td>entry point</td>
<td><img src="#" alt="terminate" /></td>
</tr>
<tr>
<td>OR</td>
<td><img src="#" alt="s" /></td>
<td>exit point</td>
<td><img src="#" alt="terminate" /></td>
</tr>
<tr>
<td>AND</td>
<td><img src="#" alt="s" /></td>
<td>terminate</td>
<td><img src="#" alt="terminate" /></td>
</tr>
<tr>
<td>submachine state</td>
<td><img src="#" alt="s" /></td>
<td></td>
<td><img src="#" alt="S : s" /></td>
</tr>
</tbody>
</table>

- **entry/act**: action entry
- **do/act**: action
- **exit/act**: action exit
Until now: \( (S, s_0, \rightarrow) \), \( s_0 \in S, \rightarrow \subseteq S \times (\mathcal{E} \cup \{\_\}) \times \text{Expr } \mathcal{Q} \times \text{Act } \mathcal{Q} \times S \)

NEW: \( (\{s_1, s_2, s_3, s_4, s, \_\}, \{t_7, t_2\}, \{t_4 \mapsto (\{s_7\}, \{s_2, s_3\}), \ldots\}) \)

\( s_1 \rightarrow s_2 \)

\( s_1 \)
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),

- $\text{kind} : S \to \{\text{st}, \text{init}, \text{fin}, \text{shist}, \text{dhist}, \text{fork}, \text{join}, \text{junc}, \text{choi}, \text{ent}, \text{exi}, \text{term}\}$ is a function which labels states with their kind,
  - (new)

- $\text{region} : S \to 2^S$ is a function which characterises the regions of a state,
  - (new)

- $\rightarrow$ is a set of transitions,
  - (changed)

- $\psi : (\rightarrow) \to 2^S \times 2^S$ is an incidence function, and
  - (new)

- $\text{annot} : (\rightarrow) \to (\mathcal{E} \cup \{\_\}) \times \text{Expr}_\mathcal{G} \times \text{Act}_\mathcal{G}$ provides an annotation for each transition.
  - (new)

($s_0$ is then redundant — replaced by proper state (!) of kind ‘init’.)

### From UML to Hierarchical State Machine: By Example

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

<table>
<thead>
<tr>
<th>Example</th>
<th>∈ (S)</th>
<th>kind</th>
<th>region</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>simple state</strong></td>
<td>(s)</td>
<td>(\text{st})</td>
<td>(\emptyset)</td>
</tr>
<tr>
<td><strong>final state</strong></td>
<td>(q)</td>
<td>(\text{fin})</td>
<td>(\emptyset)</td>
</tr>
<tr>
<td><strong>composite state</strong></td>
<td>(s)</td>
<td>(\text{st})</td>
<td>({{s_1, s_2, s_3}})</td>
</tr>
</tbody>
</table>

**submachine state**

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

**pseudo-state**

\(\rho, \ldots\)
\[
\text{... denotes } (S, \text{kind}, \text{region}, \rightarrow, \psi, \text{annot}) = \\
\left( \left\{ (q, \text{init}), (s, \text{st}), (p, \text{fin}), (\text{top}, \text{st}) \right\} \right), \\
\left( S, \text{kind} \rightarrow \left\{ (q, \text{init}), (s, \text{st}), (p, \text{fin}), (\text{top}, \text{st}) \right\} \right), \\
\left( \text{region} \rightarrow \left\{ (t_0, t_2), (t_0 \rightarrow (s_1, s_3), t_2 \rightarrow (s_5, s_3)) \right\} \right), \\
\left( \psi \rightarrow \left\{ (t_0 \rightarrow (\text{st}, \text{gd}, \text{act}), t_2 \rightarrow \text{annot}) \right\} \right), \\
\left( \text{annot} \rightarrow \left\{ \right\} \right)
\]
## Well-Formedness: Regions

<table>
<thead>
<tr>
<th></th>
<th>$\in S$</th>
<th>kind</th>
<th>region $\subseteq 2^S$, $S_i \subseteq S$</th>
<th>child $\subseteq S$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>final state</strong></td>
<td>$s$</td>
<td>fin</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td><strong>pseudo-state</strong></td>
<td>$s$</td>
<td>init</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td><strong>simple state</strong></td>
<td>$s$</td>
<td>st</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td><strong>composite state</strong></td>
<td>$s$</td>
<td>st</td>
<td>${S_1, \ldots, S_n}, n \geq 1$</td>
<td>$S_1 \cup \cdots \cup S_n$</td>
</tr>
<tr>
<td><strong>implicit top state</strong></td>
<td>$top$</td>
<td>st</td>
<td>${S_1}$</td>
<td>$S_1$</td>
</tr>
</tbody>
</table>

- Final and pseudo states **must not comprise** regions.
- States $s \in S$ with $\text{kind}(s) = \text{st}$ **may comprise** regions.

Naming conventions can be defined based on regions:

- No region: simple state.
- One region: OR-state.
- Two or more regions: AND-state.

- Each state (except for $top$) **must** lie in exactly one region.

**Note:** The region function induces a child function.

**Note:** Diagramming tools (like Rhapsody) can ensure well-formedness.
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 \( \text{region}(s) = \{ S_1, \ldots, S_n \} \),
- for each \( 1 \leq i \leq n \), there exists exactly one initial pseudo-state \((s_i^1, \text{init}) \in S_i\) and at least one transition \( t \in \rightarrow \) with \( s_i^1 \) 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. \( t \in \rightarrow \) from \( s \) with \( \text{kind}(s) = \text{st} \) implies \( \text{annot}(t) = (\_, \text{true}, \text{act}) \).
- Final states are not sources of transitions.
- Composite states.
- Initial pseudostate, final state.
- Entry/do/exit actions, internal transitions.
- History and other pseudostates, the rest.
Composite States
In a sense, composite states are about abbreviation, structuring, and avoiding redundancy.

**Idea**: in Tron, for the Player’s Statemachine, instead of
and instead of

![Diagram of Composite States]

write

**AND-state:**

- **slow**
- **F/**
- **fast**
- **F/**
**Plan:**

**States:**
- what is the type of the implicit $st$ attribute?
- what are legal state configurations?

**Transitions:**
- what are legal / well-formed transitions?
- when is a legal transition enabled?
- which effects do transitions have?

- what may happen on $E$?
- what may happen on $E, F$?
- can $E, G$ kill the object?
- ...

---

**Composite States: Blessing or Curse?**

![Diagram of composite states with nodes $s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8$ and transitions $E$, $F$, $G$.]
Syntax: Fork/Join

- For simplicity, we consider transitions with (possibly) multiple sources and targets, i.e.

\[ \psi : (\rightarrow) \rightarrow (2^S \setminus \emptyset) \times (2^S \setminus \emptyset) \]

- For instance,

\[
\begin{align*}
\psi(t_1) &= (source(t_1), target(t_1)) \\
(S, \text{kind}, \text{region}, \{t_1\}, \{t_1 \mapsto (\{s_2, s_3\}, \{s_5, s_6\})\}, \{t_1 \mapsto (tr, gd, act)\}) \\
&\rightarrow \psi \\
\text{annot}
\end{align*}
\]

- Naming convention: \( \psi(t) = (source(t), target(t)) \).
State Configuration

- The type of (implicit attribute) \( st \) is from now on a set of states, i.e.
  \[ \mathcal{D}(S_{MC}) = 2^S \]

- A set \( S_1 \subseteq S \) is called (legal) state configurations if and only if
  - \( \text{top} \in S_1 \), and
  - with each state \( s \in S_1 \) that has a non-empty region \( \emptyset \neq R \in \text{region}(s) \),
    exactly one (non pseudo-state) child of \( s \) is in \( S_1 \), i.e.
    \[ |\{ s \in R \mid \text{kind}(s) \in \{ st, fin \}\} \cap S_1| = 1. \]

\[ \{ n, fin \} \]
\[ \text{not ok} \]

\[ \{ n \} \]
\[ \text{not ok} \]
The substate- (or child-) relation induces a partial order on states:

- \( \text{top} \leq s \), for all \( s \in S \),
- \( s \leq s' \), for all \( s' \in \text{child}(s) \),
- transitive, reflexive, antisymmetric,
- \( s' \leq s \) and \( s'' \leq s \) implies \( s' \leq s'' \) or \( s'' \leq s' \).
The least common ancestor is the function \( \text{lca} : 2^S \to S \) such that

- The states in \( S_1 \) are (transitive) children of \( \text{lca}(S_1) \), i.e.

\[
\text{lca}(S_1) \leq s, \text{ for all } s \in S_1 \subseteq S,
\]

- \( \text{lca}(S_1) \) is minimal, i.e. if \( \hat{s} \leq s \) for all \( s \in S_1 \), then \( \hat{s} \leq \text{lca}(S_1) \)

- Note: \( \text{lca}(S_1) \) exists for all \( S_1 \subseteq S \) (last candidate: \( \text{top} \)).
Orthogonal States

- Two states \( s_1, s_2 \in S \) are called **orthogonal**, denoted \( s_1 \perp s_2 \), if and only if
  - they are unordered, i.e. \( s_1 \not\leq s_2 \) and \( s_2 \not\leq s_1 \), and
  - they live in different regions of an AND-state, i.e.

\[
\exists s, \text{region}(s) = \{S_1, \ldots, S_n\}, 1 \leq i \neq j \leq n : s_1 \in \text{child}(S_i) \land s_2 \in \text{child}(S_j),
\]

\[
\begin{align*}
&\quad s_1 \\
&\quad s_2 \\
&\quad s_3 \\
\end{align*}
\]

\[
\begin{align*}
&\quad s_1' \\
&\quad s_2' \\
&\quad s_3' \\
\end{align*}
\]

\[
\begin{align*}
&\quad s_1'' \\
&\quad s_2'' \\
&\quad s_3'' \\
\end{align*}
\]
A set of states $S_1 \subseteq S$ is called **consistent**, denoted by $\downarrow S_1$, if and only if for each $s, s' \in S_1$,

- $s \leq s'$,
- $s' \leq s$, or
- $s \perp s'$.
A hierarchical state-machine \((S, \text{kind}, \text{region}, \rightarrow, \psi, \text{annot})\) is called \textbf{well-formed} if and only if for all transitions \(t \in \rightarrow,\)

- source and destination are consistent, i.e. \(\downarrow \text{source}(t)\) and \(\downarrow \text{target}(t),\)
- source (and destination) states are pairwise orthogonal, i.e.
  - \(\text{forall } s, s' \in \text{source}(t) (\in \text{target}(t)), s \perp s',\)
- the top state is neither source nor destination, i.e.
  - \(\text{top} \notin \text{source}(t) \cup \text{source}(t).\)

\textbf{Recall}: final states are not sources of transitions.

\textbf{Example}: 

![Diagram showing a well-formed hierarchical state-machine with states and transitions labeled with conditions and regions.](image)
The Depth of States

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

Example:
Enabledness in Hierarchical State-Machines

• The **scope** ("set of possibly affected states") of a transition $t$ is the **least common region** (!) of

$$source(t) \cup target(t).$$

• 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.

$$prio(t) := \max\{\text{depth}(s) \mid s \in source(t)\}$$

• A set of transitions $T \subseteq \rightarrow$ is **enabled** in an object $u$ if and only if

  • $T$ is consistent,

  • $T$ is maximal wrt. priority (all transitions in $T$ have the same, highest priority),

  • all transitions in $T$ share the same trigger,

  • for all $t \in T$, the source states are active, i.e.

  $$source(t) \subseteq \sigma(u)(st) \subseteq S,$$

  • all guards are satisfied by $\sigma(u)$. 


• Let $T$ be a set of transitions enabled in $u$.

• Then $\left(\sigma, \varepsilon\right) \xrightarrow{\left(\text{cons, Snd}\right)} \left(\sigma', \varepsilon'\right)$ if

$\sigma'(u)(st)$ consists of the target states of $T$,

i.e. for simple states the simple states themselves, for composite states the initial states,

$\sigma'$, $\varepsilon'$, $\text{cons}$, and $\text{Snd}$ are the effect of firing each transition $t \in T$ one by one, in any order, i.e. for each $t \in T$,

• the exit action transformer ($\rightarrow$ later) of all affected states, highest depth first,
• the transformer of $t$,
• the entry action transformer ($\rightarrow$ later) of all affected states, lowest depth first.

$\leadsto$ adjust Rules (ii), (iii), (v) accordingly.
Initial and Final States
**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 (→ later).

**Recall:** For simplicity, we assume exactly one initiation transitions — could be more, choose non-deterministically.

**Special case:** the region of *top*.
- If class $C$ has a state-machine, then “create-$C$ transformer” is the concatenation of
  - the transformer of the “constructor” of $C$ (here not introduced explicitly) and
  - a transformer corresponding to one initiation transition of the top region.
Final States

- If \((\sigma, \varepsilon) \xrightarrow{\text{cons, Snd}} (\sigma', \varepsilon')\)
  and all simple states in \(st \in \sigma(u)(st)\) are \textbf{final}, i.e. \(\text{kind}(s) = \text{fin}\), then
  - stay \textbf{unstable} if there is a common parent of the simple states in \(\sigma(u)(st)\)
    which is source of a transition without trigger and satisfied guard,
  - otherwise \textbf{kill} \(u\).

\(\sim\) adjust Rules (i), (ii), (iii), and (v) accordingly.

\textbf{Observation}: \(u\) never “survives” reaching a state \((s, \text{fin})\) with \(s \in \text{child}(\text{top})\).

\textbf{Observation}:

\begin{align*}
  &s_1 & E/\text{act}_1 & s_2 & /\text{act}_2 & s_3
\end{align*}

\(\text{vs.}\)

\begin{align*}
  &s_1 & E/\text{act}_1 & s_2 & \text{DONE/act}_2 & s_3
\end{align*}
References
