Software Design, Modelling and Analysis in UML
Lecture 15: Hierarchical State Machines II

2017-01-10

Prof. Dr. Andreas Podelski, Dr. Bernd Westphal

Albert-Ludwigs-Universität Freiburg, Germany

Content

- Hierarchical State Machines
  - Recall:
    - Abstract Syntax: States
    - (Legal) System Configurations
  - Abstract Syntax: Transitions
    - orthogonal states,
    - legal transitions
  - Enabledness of Fork/Join Transitions
    - least common ancestor,
    - scope,
    - priority and depth,
    - maximality
  - Transitions (or steps) of Hierarchical State Machines
Recall

Blessing or Curse...?

Plan:
States / Syntax:
- What is the abstract syntax of a diagram?
States / Semantics:
- what is the type of the implicit st attribute?
- what are legal system configurations?
Transitions / Syntax:
- what are legal / well-formed transitions?
Transitions / Semantics:
- when is a legal transition enabled?
- which effects do transitions have?

For example: From $s_1$, $s_5$,...
- what may happen on $E$?
- what may happen on $E$, $F$?
- can $E$, $G$ kill the object?
- ...

For example: From $s_2$, $s_3$, $s_4$, $s_5$...
Representing All Kinds of States

- So far:

\[(S, s_0, \rightarrow), \quad s_0 \in S, \quad \rightarrow \subseteq S \times (\emptyset \cup \{\_\}) \times \text{Expr}_\mathcal{F} \times \text{Act}_\mathcal{F} \times S\]

- 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
- \(\text{kind}: S \rightarrow \{\text{st, init, fin, shist, dhist, fork, join, junc, choi, ent, exi, term}\}\)
  - is a function which labels states with their kind.
- \(\text{region}: S \rightarrow 2^2\) is a function which characterises the regions of a state.
- \(\mathcal{H}\) is a set of transitions.
- \(\psi: (\mathcal{H}) \rightarrow 2^2 \times 2^2\) is an incidence function, and
- \(\text{annot}: (\mathcal{H}) \rightarrow (\emptyset \cup \{\_\}) \times \text{Expr}_\mathcal{F} \times \text{Act}_\mathcal{F}\)
  - provides an annotation for each transition.

\(s_0\) is then redundant – replaced by proper state (!) of kind 'init'.

From UML to Hierarchical State Machine: By Example

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

- \(S = \{\text{top, } s_1, s, s_2\}\)
- \(\text{kind} = \{\text{top} \rightarrow \text{st}, s_1 \rightarrow \text{init, } s \rightarrow \text{st, } s_2 \rightarrow \text{fin}\}\)
- \(\text{or } (S, \text{kind}) = \{(\text{top, st}), (s_1, \text{init}), (s, \text{st}), (s_2, \text{fin})\}\)
- \(\text{region} = \{\text{top} \rightarrow \{\{s_1, s, s_2\}\}, s_1 \rightarrow \emptyset, s \rightarrow \emptyset, s_2 \rightarrow \emptyset\}\)
- \(\rightarrow, \psi, \text{annot}; \text{in a minute.}\)
Recall

Plan:
States / Syntax:
- What is the abstract syntax of a diagram?
States / Semantics:
- what is the type of the implicit \( st \) attribute?
- what are legal system configurations?
Transitions / Syntax:
- what are legal / well-formed transitions?
Transitions / Semantics:
- when is a legal transition enabled?
- which effects do transitions have?

For example: From \( s_1, s_5 \),
- what may happen on \( E, F \)?
- can \( E, G \) kill the object?
- ... 

Semantics: State Configuration

- The type of (implicit attribute) \( st \) is from now on a set of states, i.e. \( \mathcal{D}(St_{sc}) = 2^S \)
- A set \( S_1 \subseteq S \) is called (legal) state configuration if and only if
  - \( top \in S_1 \), and
  - for each region \( R \) of a state in \( S_1 \), exactly one (non pseudo-state) element of \( R \) is in \( S_1 \), i.e.
  \[ \forall s \in S_1 \forall R \in \text{region}(s) \bullet |\{ s' \in R \mid \text{kind}(s') \in \{ \text{st, fin} \} \} \cap S_1 | = 1. \]
- Examples:
  \[ s_1 = \{ s_3, s_4 \} \times \]
  \[ s_2 = \{ s_6 \} \times \]
  \[ s_3 = \{ s_7, s_8, s_5 \} \times \]
  \[ s_4 = \{ s_9, s_{10}, s_{11} \} \times \]
  \[ s_5 = \{ s_{12}, s_{13}, s_{14}, s_{15} \} \times \]
  \[ s_6 = \{ s_{16}, s_{17}, s_{18}, s_{19} \} \times \]
  \[ s_7 = \{ s_{20}, s_{21}, s_{22}, s_{23} \} \times \]
  \[ s_8 = \{ s_{24}, s_{25}, s_{26}, s_{27} \} \times \]
  \[ s_9 = \{ s_{28}, s_{29}, s_{30}, s_{31} \} \times \]
  \[ s_{10} = \{ s_{32}, s_{33}, s_{34}, s_{35} \} \times \]
  \[ s_{11} = \{ s_{36}, s_{37}, s_{38}, s_{39} \} \times \]
Recall

Plan:
States / Syntax:
- What is the abstract syntax of a diagram?
States / Semantics:
- what is the type of the implicit st attribute?
- what are legal system configurations?
Transitions / Syntax:
- what are legal / well-formed transitions?
Transitions / Semantics:
- when is a legal transition enabled?
- which effects do transitions have?

For example: From $s_1$, $s_5$,
- what may happen on $E$?
- what may happen on $F$?
- can $E$, $G$ kill the object?
- ...

Blessing or Curse...?

$\sqrt{1}$
$X: \{1\}$

$\sqrt{1}$
$X: \{1\}$
Recall

Plan:

States / Syntax:
- What is the abstract syntax of a diagram?

States / Semantics:
- what is the type of the implicit st attribute?
- what are legal system configurations?

Transitions / Syntax:
- what are legal / well-formed transitions?

Transitions / Semantics:
- when is a legal transition enabled?
- which effects do transitions have?

Transitions 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,

  \[
  \begin{array}{c}
  s_1 \\
  s_2 \\
  s_3 \\
  s_4 \\
  s_5 \\
  s_6 \\
  \end{array}
  \]

  translates to

  \[
  (S, \text{kind}, \text{region}, \{t_1\}, \{t_1 \mapsto (\{s_2, s_3\}, \{s_5, s_6\})\}, \{t_1 \mapsto (\text{tr}, \text{gd}, \text{act})\})
  \]

- Naming convention: \( \psi(t) = (\text{source}(t), \text{target}(t)) \).
Orthogonal States

- Two states $s_1, s_2 \in S$ are called orthogonal, denoted $s_1 \perp s_2$, if and only if
  - they "live" in different regions of one AND-state, i.e.

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

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) states are pairwise orthogonal, i.e.
  - $\forall s,s' \in \text{source}(t) (\in \text{target}(t)) \cdot s \perp s'$,

- the top state is neither source nor destination, i.e.
  - $\text{top} \notin \text{source}(t) \cup \text{target}(t)$.

Recall: final states are not sources of transitions.

Example:
Plan

<table>
<thead>
<tr>
<th>simple state</th>
<th>example</th>
</tr>
</thead>
<tbody>
<tr>
<td>final state</td>
<td>example</td>
</tr>
<tr>
<td>composite state</td>
<td>example</td>
</tr>
<tr>
<td>OR</td>
<td></td>
</tr>
<tr>
<td>AND</td>
<td></td>
</tr>
</tbody>
</table>

- Transitions involving non-pseudo states.
- Initial pseudostate, final state.
- Entry/do/exit actions, internal transitions.
- History and other pseudostates, the rest.

Scope

- 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 disjoint.
A Partial Order on States

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,

\[
\begin{align*}
\text{OR } & \quad s \geq s' \quad \text{if } s \in \text{child}(s') \\
\text{OR } & \quad s \leq s' \quad \text{implies } s' \leq s'' \text{ or } s'' \leq s'.
\end{align*}
\]

A Partial Order on States

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,

\[
\begin{align*}
\text{OR } & \quad s \geq s' \quad \text{if } s \in \text{child}(s') \\
\text{OR } & \quad s \leq s' \quad \text{implies } s' \leq s'' \text{ or } s'' \leq s'.
\end{align*}
\]
Least Common Ancestor

- The **least common ancestor** is the function $lca : 2^S \rightarrow S$ such that
  
  1. The states in $S_1$ are (transitive) children of $lca(S_1)$, i.e.
     
     $$lca(S_1) \leq s, \text{ for all } s \in S_1 \subseteq S,$$
  
  2. $lca(S_1)$ is maximal, i.e. if $\hat{s} \leq s$ for all $s \in S_1$ then $\hat{s} \leq lca(S_1)$
  
  - **Note:** $lca(S_1)$ exists for all $S_1 \subseteq S$ (last candidate: top).

Scope

- 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 disjoint.
Priority and Depth

- The **priority** of transition \( t \) is the depth of its innermost source state, i.e.

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

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

**Example:**

![Diagram](image)

Enabledness in Hierarchical State-Machines

- A set of transitions \( T \subseteq \rightarrow \) is **enabled** for an object \( u \) in \( (\sigma, \epsilon) \) if and only if
  - \( T \) is consistent,
  - for all \( t \in T \), the source states are active, i.e.
    \[
    \text{source}(t) \subseteq \sigma(u)(st) \subseteq S.
    \]
  - all transitions in \( T \) have the same trigger \( tr \) and
    - \( tr = \_ \) and \( u \) is unstable, or
    - \( tr = E \) and there is an \( E \) ready for \( u \) in \( \epsilon \),
  - the guards of all transitions in \( T \) are satisfied in \( \sigma \) wrt. \( u \), and

A set \( T \) of enabled transitions is called **maximal** wrt.

- **extension** if and only if there is no transition \( t' \notin T \) such that \( T \cup \{t'\} \) is enabled.
- **priority** if and only if for each \( t \in T \), there is no \( t' \in \rightarrow \) such that
  - \( \text{prio}(t') > \text{prio}(t) \),
  - \( (T \setminus \{t\}) \cup \{t'\} \) is enabled, and
  - \( st' \geq st \) for some \( st' \in \text{source}(t') \) and \( st \in \text{source}(t) \).
Transitions in Hierarchical State-Machines

• Let $T$ be a maximal (extension and priority) set of transitions enabled for $u$ in $(\sigma, \varepsilon)$.

• Then $(\sigma, \varepsilon) \xrightarrow{(\text{cons, Snd})} (\sigma', \varepsilon')$ 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 (i), (ii), (iii), (v) accordingly.

(For state machines with only simple states, and no trigger, guard, or action on transitions originating at initial states: Same behaviour as before.)

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 $\text{region}(s) = \{S_1, \ldots, S_n\}$,
  • 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.
An Intuition for “Or-States”

- In a sense, composite states are about
  - abbreviation,
  - structuring, and
  - avoiding redundancy.

- Idea: instead of

An Intuition for “And-States”

and instead of
References