Software Design, Modelling and Analysis in UML

Lecture 17: Hierarchical State Machines II

2014-01-20

Prof. Dr. Andreas Podelski, Dr. Bernd Westphal
Albert-Ludwigs-Universität Freiburg, Germany

Contents & Goals

Last Lecture:
- State Machines and OCL
- Hierarchical State Machines Syntax
- Initial and Final State

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:
  - Composite State Semantics
  - The Rest
Composite States
(formalisation follows [Damm et al., 2003])

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

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

\[ \text{write} \]

\[ \text{resigned} \]
Composite States

and instead of

write

Recall: Syntax

translates to

\[
\begin{align*}
\{(top, st), (s, st), (s_1, st) (s_1', st), (s_2, st) (s_2', st), (s_3, st) (s_3', st)\}, \\
\{top \mapsto \{s\}, s \mapsto \{\{s_1, s_1'\}, \{s_2, s_2'\}, \{s_3, s_3'\}\}, s_1 \mapsto \emptyset, s_1' \mapsto \emptyset, \ldots\}, \\
\text{region} \\
\rightarrow, \psi, \text{annot}\end{align*}
\]
Composite States: Blessing or Curse?

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

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

- what may happen on \( E \)?
- what may happen on \( E, F \)?
- can \( E, G \) kill the object?
- ...

\[8/44\]
State Configuration

- The type of \( st \) is from now on a set of states, i.e. \( st : 2^S \)
- A set \( S_1 \subseteq S \) is called (legal) state configurations if and only if
  - \( \text{top} \in S_1 \), and
  - for each state \( s \in S_1 \), for each non-empty region \( \emptyset \neq R \in \text{region}(s) \), exactly one (non pseudo-state) child of \( s \) (from \( R \)) is in \( S_1 \), i.e.
    \[
    |\{ s_0 \in R \mid \text{kind}(s_0) \in \{ \text{st, fin} \} \} \cap S_1 | = 1.
    \]
**Syntax: Fork/Join**

- For brevity, we always consider transitions with (possibly) multiple sources and targets, i.e.
  \[ \psi : (\rightarrow) \rightarrow (2^S \setminus \emptyset) \times (2^S \setminus \emptyset) \]

- For instance,

  \[
  \begin{array}{c}
  s_1 \quad s_2 \quad s_3 \\
  \text{tr[gd]/act} \\
  s_4 \quad s_5 \quad s_6
  \end{array}
  \]

  translates to

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

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

---

**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,
- \( s' \leq s \) and \( s'' \leq s \) implies \( s' \leq s'' \) or \( s'' \leq s' \).
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,
- \( s' \leq s \) and \( s'' \leq s \) implies \( s' \leq s'' \) or \( s'' \leq s' \).

Least Common Ancestor and Ting

- The least common ancestor is the function \( \text{lca} : 2^S \setminus \{\emptyset\} \rightarrow S \) such that
  - The states in \( S_1 \) are (transitive) children of \( \text{lca}(S_1) \), i.e.
    \[
    \text{lca}(S_1) \leq s, \quad \text{for all } s \in S_1 \subseteq S,
    \]
  - \( \text{lca}(S_1) \) 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: \( \text{top} \)).
Least Common Ancestor and Ting

- 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. \( s_1 \not\preceq s_2 \) and \( s_2 \not\preceq s_1 \), and
  - they “live” in different regions of an AND-state, i.e.
    \[
    \exists s, \text{region}(s) = \{S_1, \ldots, S_n\} \ni \exists 1 \leq i \neq j \leq n : s_1 \in \text{child}^*(S_i) \land s_2 \in \text{child}^*(S_j),
    \]

A set of states \( S_1 \subseteq S \) is called **consistent**, denoted by \( \downarrow S_1 \), if and only if for each \( s, s' \in S_1 \),
- \( s \leq s' \), or
- \( s' \leq s \), or
- \( s \perp s' \).
Legal Transitions

A hierarchical state-machine \((S, kind, region, \rightarrow, \psi, annot)\) is called well-formed if and only if for all transitions \(t \in \rightarrow\),

(i) source and destination are consistent, i.e. \(\downarrow source(t)\) and \(\downarrow target(t)\).

(ii) source (and destination) states are pairwise orthogonal, i.e.

\[
\forall s, s' \in source(t) (\in target(t)), s \perp s'.
\]

(iii) the top state is neither source nor destination, i.e.

\[
\text{top} \notin source(t) \cup target(t).
\]

• Recall: final states are not sources of transitions.

Example:
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
  \[ \text{source}(t) \cup \text{target}(t). \]
- 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 \( u \) if and only if
  - \( T \) is consistent,
  - \( T \) is maximal wrt. priority,
  - all transitions in \( T \) share the same trigger,
  - all guards are satisfied by \( \sigma(u) \), and
  - for all \( t \in T \), the source states are active, i.e.
    \[ \text{source}(t) \subseteq \sigma(u)(\text{st}) (\subseteq S). \]
Transitions in Hierarchical State-Machines

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

- Then $(\sigma, \varepsilon) \xrightarrow{\langle \text{cons, Snd} \rangle} (\sigma', \varepsilon')$ 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 transformer of all affected states, highest depth first,
  - the transformer of $t$,
  - the entry transformer of all affected states, lowest depth first.

\[ \Rightarrow \text{adjust (2.), (3.), (5.) accordingly.} \]

Entry/Do/Exit Actions, Internal Transitions
Entry/Do/Exit Actions

- 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). $E_1, \ldots, E_n \in T$, 'entry', 'do', 'exit' are reserved names!
- Recall: each action’s supposed to have a transformer. Here: $t_{act_{E_1}}$, $t_{act_{E_2}}$, ...
- Taking the transition above then amounts to applying
  \[ t_{act_{E_2}} \circ t_{act} \circ t_{act_{E_1}} \]
  instead of only
  \[ t_{act} \]
  \[ \rightsquigarrow \] adjust (2.), (3.) accordingly.

Internal Transitions

- 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: Entry/Exit/Internal as Abbreviations

- ... as abbreviation 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 (see later).

Abbreviation may avoid confusion in context of hierarchical states (see later).

Do Actions

- Intuition: 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...?
References


