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

The Full Story

UML distinguishes the following kinds of states:

<table>
<thead>
<tr>
<th>Kind of State</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Simple State</strong></td>
<td>![Simple State Diagram]</td>
</tr>
<tr>
<td><strong>Final State</strong></td>
<td>![Final State Diagram]</td>
</tr>
<tr>
<td><strong>Composite State</strong></td>
<td>![Composite State Diagram]</td>
</tr>
<tr>
<td><strong>OR</strong></td>
<td>![OR Diagram]</td>
</tr>
<tr>
<td><strong>AND</strong></td>
<td>![AND Diagram]</td>
</tr>
<tr>
<td><strong>Pseudo-State</strong></td>
<td>![Pseudo-State Diagram]</td>
</tr>
<tr>
<td><strong>Submachine State</strong></td>
<td>![Submachine State Diagram]</td>
</tr>
</tbody>
</table>
Representing All Kinds of States

- Until now: $(S, s_0, \to)$, $s_0 \in S, \to \subseteq S \times (\mathcal{E} \cup \{\bot\}) \times \text{Expr}_\mathcal{A} \times \text{Act}_\mathcal{A} \times S$

- From now on: (hierarchical) state machines $(S, \text{kind}, \text{region}, \to, \psi, \text{annot})$

where

- $S \supseteq \{\text{top}\}$ is a finite set of states (as before).
- $\text{kind} : S \to \{\text{st, init, fin, shist, dhist, fork, join, junc, choi, ent, exi, term}\}$ is a function which labels states with their kind, (new)
- $\text{region} : S \to 2^{2^S}$ is a function which characterises the regions of a state, (new)
- $\to$ is a set of transitions, (changed)
- $\psi : (\to)^{-1} \times 2^S \times 2^S$ is an incidence function, and (new)
- $\text{annot} : (\to)^{-1} \to (\mathcal{E} \cup \{\bot\}) \times \text{Expr}_\mathcal{A} \times \text{Act}_\mathcal{A}$ 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

#### Table

<table>
<thead>
<tr>
<th>Type</th>
<th>Example</th>
<th>( S, \text{kind}, \text{region}, \rightarrow, \psi, \text{annot} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>simple state</td>
<td><img src="image" alt="Simple State" /></td>
<td>( s \in S, s_k, \emptyset )</td>
</tr>
<tr>
<td>final state</td>
<td><img src="image" alt="Final State" /></td>
<td>( s \in S, s_f, \emptyset )</td>
</tr>
<tr>
<td>composite state</td>
<td><img src="image" alt="Composite State" /></td>
<td>( s \in S, s_{OR}, {s_1, s_2, s_3} )</td>
</tr>
<tr>
<td></td>
<td><img src="image" alt="Composite State" /></td>
<td>( s \in S, s_{AND}, {s_1, s_2, s_3}, {s_2, s_3} )</td>
</tr>
<tr>
<td>submachine state</td>
<td>(later)</td>
<td>( s \in S, s_{sub}, \emptyset )</td>
</tr>
<tr>
<td>pseudo-state</td>
<td><img src="image" alt="Pseudo-State" /></td>
<td>( s \in S, s_{ps}, \emptyset )</td>
</tr>
</tbody>
</table>

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

---

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

... denotes \( (S, \text{kind}, \text{region}, \rightarrow, \psi, \text{annot}) = \)

\[
\left\{ (s, \text{kind}), (s, \text{region}), (s, \rightarrow), (s, \psi), (s, \text{annot}) \right\}
\]

\[
\left\{ \{s_1, s_2, s_3\}, \{s_2, s_3\}, \{s_2, s_3\}, \{s_2, s_3\}, \{s_2, s_3\} \right\}
\]

\[
\left\{ \{s_1, s_2, s_3\}, \{s_2, s_3\}, \{s_2, s_3\}, \{s_2, s_3\}, \{s_2, s_3\} \right\}
\]

\[
\left\{ \{s_1, s_2, s_3\}, \{s_2, s_3\}, \{s_2, s_3\}, \{s_2, s_3\}, \{s_2, s_3\} \right\}
\]
Well-Formedness: Regions

- 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.

Well-Formedness Continued

- 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 $(s_i^i, \text{init}) \in S_i$ and at least one transition $t \in \rightarrow$ with $s_i^i$ 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. $t \in \rightarrow$ 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

- **OR-state**:

  \[ n \rightarrow w \rightarrow e \rightarrow X/ \rightarrow resigned \]

- **AND-state**:

  \[ n \rightarrow w \rightarrow e \rightarrow F/ \rightarrow fastN \]

  \[ n \rightarrow F/ \rightarrow fast \]

  \[ n \rightarrow F/ \rightarrow slow \]

  \[ n \rightarrow F/ \rightarrow fast \]

and instead of

- **write**

  \[ n \rightarrow w \rightarrow e \rightarrow X/ \rightarrow resigned \]

- **write**

  \[ n \rightarrow w \rightarrow e \rightarrow F/ \rightarrow fast \]

  \[ n \rightarrow F/ \rightarrow fast \]

  \[ n \rightarrow F/ \rightarrow slow \]

  \[ n \rightarrow F/ \rightarrow fast \]
Composite States: Blessing or Curse?

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

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

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,

\[
\begin{align*}
|s_1| & \rightarrow |s_2| \rightarrow |s_3| \rightarrow |s_4| \\
tr[gd]/act & \\
\end{align*}
\]

translates to

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

- 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.
  \[ \mathcal{P}(S_{MC}) = 2^S \]
- A set \( S_1 \subseteq S \) is called (legal) state configurations if and only if
  - \( top \in S_1 \), 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.
    \[ \left| \{s \in R \mid \text{kind}(s) \in \{st, fin\} \} \right| = 1. \]

A Partial Order on States

The substate- (or child-) relation induces a partial order on states:
- \( top \leq s \), 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 \( s'' \leq s' \).

\[
\begin{align*}
\{ \text{n, not, o} \} & \quad \text{not OK} \\
\{ \text{n} \} & \quad \text{not OK}
\end{align*}
\]

\[
\begin{align*}
\{ \text{n, top} \} & \quad \text{not OK} \\
\{ \text{top, ingame, n, top} \} & \quad \text{top}
\end{align*}
\]
Least Common Ancestor

- The least common ancestor is the function $\text{lea} : 2^S \to S$ such that
  - The states in $S_1$ are (transitive) children of $\text{lea}(S_1)$, i.e.
    $$\text{lea}(S_1) \leq s,$$
    for all $s \in S_1 \subseteq S$,
  - $\text{lea}(S_1)$ is minimal, i.e. if $\hat{s} \leq s$ for all $s \in S_1$, then $\hat{s} \leq \text{lea}(S_1)$
  - Note: $\text{lea}(S_1)$ exists for all $S_1 \subseteq S$ (last candidate: 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.
    $$\exists s, \text{region}(s) = \{S_1, \ldots, S_n\}, 1 \leq i \neq j \leq n : s_1 \in \text{child}(S_i) \land s_2 \in \text{child}(S_j),$$
Consistent State Sets

- 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
  - \( s \perp s' \).

Legal Transitions

A hierarchical state-machine \((S, \text{kind}, \text{region}, \to, \psi, \text{annot})\) is called **well-formed** if and only if for all transitions \( t \in \to \),
- 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.
  - for all \( s, s' \in \text{source}(t) \) (\( \in \text{target}(t) \)), \( s \perp s' \),
- the top state is neither source nor destination, i.e.
  - \( \text{top} \notin \text{source}(t) \cup \text{source}(t) \).

**Recall**: final states are not sources of transitions.

**Example**: final states are not sources of transitions.
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 \( T \) 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.
    \[
    \text{source}(t) \subseteq \sigma(u)(\text{st}) (\subseteq S).
    \]
  - all guards are satisfied by \( \sigma(u) \).
Transitions in Hierarchical State-Machines

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

- Then $(\sigma, \varepsilon) \xrightarrow{(\text{cons, Snd})}{_u} (\sigma', \varepsilon')$ 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.

\[\Rightarrow\text{ adjust Rules (ii), (iii), (v) accordingly.}\]

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**

**Observation:** *u* never "survives" reaching a state (*s, fin*) with *s* ∈ child(*top*).
References
