**Contents & Goals**

**Last Lecture:**
- 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, ...?

**Composite States**

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

Idea: in Tron, for the Player's State Machine, instead of writing

we resigned

write

and instead of fast write

slow fast
For brevity, we always consider transition sets with (possibly) multiple sources and targets, i.e., \( \psi : (\rightarrow) \rightarrow (2S \cup \emptyset) \times (2S \cup \emptyset) \).

For instance, \( s_1 s_2 s_3 s_4 s_5 s_6 \) translates to:

\[
(\{s_2, s_3\}, \{s_5, s_6\}) \psi (\{t_1\} \rightarrow \{t_1 \rightarrow (\{s_2, s_3\}, \{s_5, s_6\})\} \rightarrow \annot) \]

Naming convention:

\( \psi (t) = (\text{source}(t), \text{target}(t)) \).

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?
Recall: final states are

\[ \text{source} \cup \text{target} \]

... 

(i) source and destination are consistent, i.e., they “live” in different regions of an AND-state, i.e., ...
In general, with each state \( s \subseteq \tau \), a set of transitions enabled is called \( \mathcal{G}_d(s) \) (when \( \mathcal{G}_d(s) \neq \emptyset \)), and another set of transitions \( \mathcal{G}_e(s) \) if and only if their scopes are orthogonal (i.e., states in scopes are orthogonal). The Depth of States is defined as the maximum transition depth. Legal Transitions are those that are enabled and consistent.

The Depth of States is defined as the maximum transition depth. Legal Transitions are those that are enabled and consistent.

---

Transitions in Hierarchical State-Machines

---

Example:

- The Depth of States

---

Example:

- The Depth of States
Now, what is it exactly while the do-action is executing?

- "An object is either idle or doing a run-to-completion step."

Recall the overall UML State Machine philosophy:

- If the state is left before termination, the do-action is stopped.
- Otherwise, if the state is terminated, then the state is considered completed.
- After entering a state, start its do-action.

Intuition:

- The concept of history and other pseudo-states:
  - Some code generators assume that internal transitions have priority!
  - Abbreviation may avoid confusion in the context of hierarchical states (see later).
  - If internal actions should have priority, that is: Entry/Internal/Exit don't add expressive power to Core State Machines.
- 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.
- According to (2), the internal transitions can be embedded into an OR-state.
- Adjust the transitions accordingly.
- The state is neither left nor entered, so: no exit, no entry.
Junction and Choice

- Junction ("static conditional branch")
  - \[ gd_1 / act_1 \]
  - \[ gd_2 / act_2 \]
  - "good": abbreviation
  - Unfolds to so many similar transitions with different guards, the unfolded transitions are then checked for enabledness.
  - At best, start with trigger, branch into conditions, then apply actions.

- Choice ("dynamic conditional branch")
  - "evil": may get stuck
  - Enters the transition without knowing whether there's an enabled path.
  - At best, use "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...

Hierarchical states can be "folded" for readability.

- Can even be taken from a different state-machine for re-use.

Entry and Exit Point, Submachine State, Terminate

- 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 the 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.

References