Software Design, Modelling and Analysis in UML

Lecture 14: Hierarchical State Machines II

2012-01-17

Prof. Dr. Andreas Podelski, **Dr. Bernd Westphal**

Albert-Ludwigs-Universität Freiburg, Germany

Contents & Goals

Last Lecture:

- Putting It All Together: ODs define initial system states
- Hierarchical State Machines: kind, region
- Initial pseudostate, final state

This Lecture:

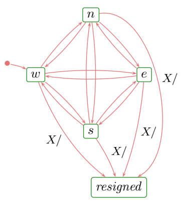
- **Educational Objectives:** Capabilities for following tasks/questions.
  - What does this **hierarchical** State Machine mean? What **may happen** if I inject this event?
  - What is: AND-State, OR-State, pseudo-state, entry/exit/do, final state, ...
- **Content:**
  - Composite states
  - Legal state configuration
  - Lca, depth, ...
  - Exit/Entry, internal transitions
  - History and others

Composite States

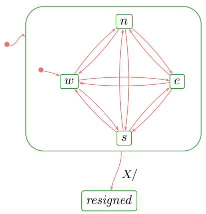
(formalisation follows [Damm et al., 2003])

Composite States

- In a sense, composite states are about **abbreviation**, **structuring**, and **avoiding redundancy**.
- Idea: in Tron, for the Player's Statemachine, instead of

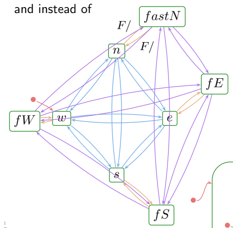


write

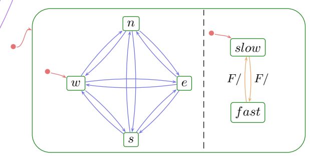


Composite States

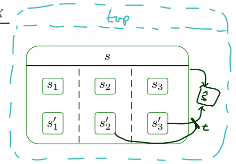
and instead of



write



Recall: Syntax



translates to

$$(\underbrace{\{(top, st), (s, st), (s_1, st), (s_1', st), (s_2, st), (s_2', st), (s_3, st), (s_3', st)\}}_{S, kind},$$

$$\underbrace{\{top \mapsto \{\{s\}, \hat{s}\}, s \mapsto \{\{s_1, s_1'\}, \{s_2, s_2'\}, \{s_3, s_3'\}\}, s_1 \mapsto \emptyset, s_1' \mapsto \emptyset, \dots\}}_{region},$$

: (→) → $\mathcal{E} \cup \{\_\} \times Expr \times Act$

$$\to, \psi, annot)$$

t   ψ : (→) → $(2^S \times 2^S)$

Syntax: Fork/Join

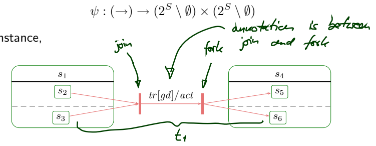
SPECIAL CASE: $\boxed{s} \xrightarrow{\text{annot}} \boxed{s'}$ (t_2)
maps to: t_2, $t_2 \mapsto (\{s\}, \{s'\})$, $t_2 \mapsto$ annot

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

$$\psi : (\to) \to (2^S \setminus \emptyset) \times (2^S \setminus \emptyset)$$

- For instance,

join — annotation is between join and fork — fork

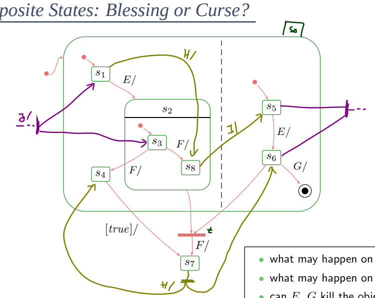


translates to

$$(S, kind, region, \underbrace{\{t_1\}}_{\to}, \underbrace{\{t_1 \mapsto (\{s_2, s_3\}, \{s_5, s_6\})\}}_{\psi}, \underbrace{\{t_1 \mapsto (tr, gd, act)\}}_{annot})$$

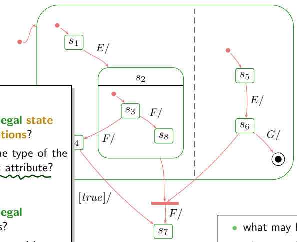
- Naming convention: $\psi(t) = (source(t), target(t))$.

Composite States: Blessing or Curse?



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

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?

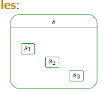
- what may happen on E?
- what may happen on E, F?
- can E, G kill the object?
- ...

States: st, (Legal) State Configurations

- 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
  - $top \in S_1$, and
  - for ~~with~~ each state $s \in S_1$ for each ~~that has a~~ non-empty region $\emptyset \neq R \in region(s)$, exactly one (non pseudo-state) child of s from R is in S_1, i.e.

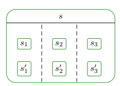
$$|\{s \in R \mid kind(s) \in \{st, fin\}\} \cap S_1| = 1.$$

- **Examples**:



$S_4 = \{top, s, s_2\}$ is LEGAL (or $\{s_2\}$)

$S_1 = \{s\}$ NOT LEGAL. top missing
$S_2 = \{s, top\}$ NOT LEGAL. no child of s in
region(top) $= \{s\}$
region(s) $= \{s_1, s_2, s_3\}$
$S_3 = \{s, top, s_1, s_3\}$ NOT LEGAL: not exactly one



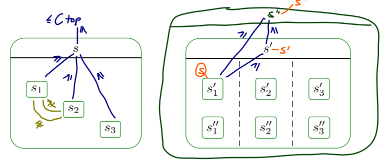
child(s) $= \{s_1, s'_1, \ldots, s'_3\}$

$S_1 = \{top, s_1, s_2, s\}$ NOT LEGAL, need s_3 or s'_3
$S_2 = \{top, s, s_1\}$ NOT LEGAL
$S_3 = \{top, s, s_1, s'_2, s_3\}$ is LEGAL
(or $\{s_1, s'_2, s_3\}$ for short)

Towards Transitions: 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 $s' \in child(s)$,
- transitive, reflexive, antisymmetric,
- $s' \leq s$ and $s'' \leq s$ implies $s' \leq s''$ or $s'' \leq s'$.



Least Common Ancestor and Ting

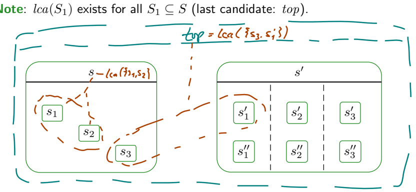
ups, misleading name; better: closest, greatest, innermost

- The **least common ancestor** is the function $lca : 2^S \setminus \emptyset \to S$ such that
  - The states in S_1 are (transitive) children of $lca(S_1)$, i.e.

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

CLAIM: $\forall S_1 \subseteq S \bullet top \in S_1 \implies lca(S_1) = top$

  - $lca(S_1)$ is minimal, 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).

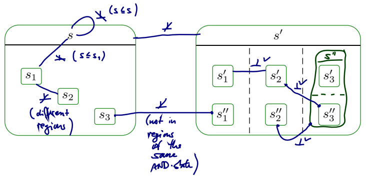


Least Common Ancestor and Ting

- 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. $s_1 \not\leq s_2$ and $s_2 \not\leq s_1$, and
  - they "live" in different regions of an AND-state, i.e.

recursive application of child

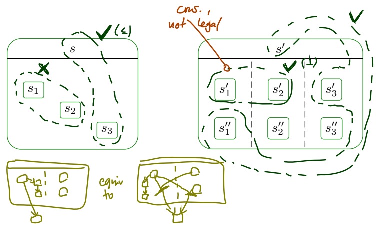
$$\exists\, s, region(s) = \{S_1, \dots, S_n\}, 1 \leq i \neq j \leq n : s_1 \in child^*(S_i) \wedge s_2 \in child^*(S_j),$$



Least Common Ancestor and Ting

- 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
  - $s \perp s'$.

CLAIM: $\forall S_1 \subseteq S$ • S_1 is legal state config. $\Rightarrow S_1$ is consistent



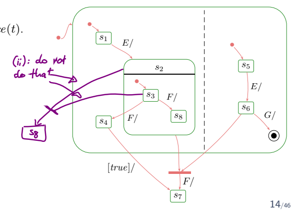
Legal Transitions

A hiearchical 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 unordered, 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.
  - $top \notin source(t) \cup source(t)$.
- Recall: final states are not sources of transitions.

Example:

CLAIM: (ii) ⇒ (i)



References

References

[Crane and Dingel, 2007] Crane, M. L. and Dingel, J. (2007). UML vs. classical vs. rhapsody statecharts: not all models are created equal. *Software and Systems Modeling*, 6(4):415–435.

[Damm et al., 2003] Damm, W., Josko, B., Votintseva, A., and Pnueli, A. (2003). A formal semantics for a UML kernel language 1.2. IST/33522/WP 1.1/D1.1.2-Part1, Version 1.2.

[Fecher and Schönborn, 2007] Fecher, H. and Schönborn, J. (2007). UML 2.0 state machines: Complete formal semantics via core state machines. In Brim, L., Haverkort, B. R., Leucker, M., and van de Pol, J., editors, *FMICS/PDMC*, volume 4346 of *LNCS*, pages 244–260. Springer.

[Harel and Gery, 1997] Harel, D. and Gery, E. (1997). Executable object modeling with statecharts. *IEEE Computer*, 30(7):31–42.

[Harel and Kugler, 2004] Harel, D. and Kugler, H. (2004). The rhapsody semantics of statecharts. In Ehrig, H., Damm, W., Große-Rhode, M., Reif, W., Schnieder, E., and Westkämper, E., editors, *Integration of Software Specification Techniques for Applications in Engineering*, number 3147 in LNCS, pages 325–354. Springer-Verlag.

[OMG, 2007] OMG (2007). Unified modeling language: Superstructure, version 2.1.2. Technical Report formal/07-11-02.