Software Design, Modelling and Analysis in UML

Lecture 15: Hierarchical State Machines I

2016-01-14

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

Albert-Ludwigs-Universität Freiburg, Germany

Contents & Goals

Last Lecture:

- step, RTC-step, divergence
- initial state, UML model semantics (so far)
- create, destroy actions

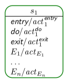
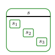
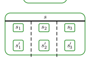
This Lecture:

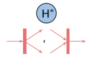
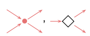
- **Educational Objectives:** Capabilities for following tasks/questions.
  - What is simple state, OR-state, AND-state?
  - What is a legal state configuration?
  - What is a legal transition?
  - How is enabledness of transitions defined for hierarchical state machines?
- **Content:**
  - Legal state configurations
  - Legal transitions
  - Rules (i) to (v) for hierarchical state machines

Hierarchical State-Machines

The Full Story

UML distinguishes the following **kinds of states**:

	example
simple state	
final state	◉
composite state	
OR	
AND	

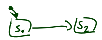
	example
pseudo-state	
initial	●
(shallow) history	H
deep history	H*
fork/join	
junction, choice	
entry point	○
exit point	⊗
terminate	×
submachine state	$S : s$

Representing All Kinds of States

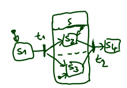
- **Until now**: init state

$$(S, s_0, \rightarrow), \quad s_0 \in S, \rightarrow\ \subseteq S \times (\mathscr{E} \cup \{\_\}) \times Expr_{\mathscr{S}} \times Act_{\mathscr{S}} \times S$$

set of states, edges; source state, trigger, guard, action, dest state



$\rightsquigarrow (\{s_1, s_2\}, s_1, \{(s_1, \_, true, skip, s_2)\})$



NEW: $(\{s_1, s_2, s_3, s_4, s, top\}, \{t_1, t_2\},$
$\{t_1 \mapsto (\{s_1\}, \{s_2, s_3\}), \ldots\}$
$\{t_1 \mapsto (\_, true, skip), \ldots\})$

Representing All Kinds of States

- **Until now**:

$$(S, s_0, \rightarrow), \quad s_0 \in S, \rightarrow\ \subseteq S \times (\mathscr{E} \cup \{\_\}) \times Expr_{\mathscr{S}} \times Act_{\mathscr{S}} \times S$$

- **From now on: (hierarchical) state machines**

$$(S, kind, region, \rightarrow, \psi, annot)$$

where

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

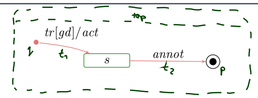
$(S, kind, region, \to, \psi, annot)$

	example	$\in S$	$kind$	$region$
simple state	s	s	st	∅
final state	q	q	fin	∅
composite state				
OR	s: s_1, s_2, s_3 ,	s	st	{{s1, s2, s3}}
AND	s: s_1, s_2, s_3, s_1', s_2', s_3'	s	st	{{s1, s1'}, {s2, s2'}, {s3, s3'}}
submachine state	(later)			
pseudo-state	p, …	p	init	∅

$\underbrace{(s, kind(s))}$ for short

region

From UML to Hierarchical State Machine: By Example



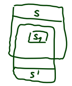
… denotes $(S, kind, region, \to, \psi, annot) =$

$$(\underbrace{\{(q, init), (s, st), (p, fin), (top, st)\}}_{S, kind},$$
$$\underbrace{\{q \mapsto \emptyset, p \mapsto \emptyset, s \mapsto \emptyset, top \mapsto \{\{q, s, p\}\}\}}_{region},$$
$$\underbrace{\{t_1, t_2\}}_{\to}, \underbrace{\{t_1 \mapsto (\{q\}, \{s\}), t_2 \mapsto (\{s\}, \{p\})\}}_{\psi},$$
$$\underbrace{\{t_1 \mapsto (tr, gd, act), t_2 \mapsto annot\}}_{annot})$$

Well-Formedness: Regions

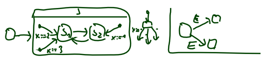
	$\in S$	$kind$	$region \subseteq 2^S, S_i \subseteq S$	$child \subseteq S$
final state	s	*fin*	$\emptyset$	$\emptyset$
pseudo-state	s	*init*, …	$\emptyset$	$\emptyset$
simple state	s	*st*	$\emptyset$	$\emptyset$
composite state	s	*st*	$\{S_1, \dots, S_n\}, n \geq 1$	$S_1 \cup \dots \cup S_n$
implicit top state	top	*st*	$\{S_1\}$	S_1

- Final and pseudo states **must not comprise** regions.
- States $s \in S$ with $kind(s) = st$ **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.

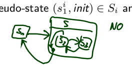


- **Note**: The region function induces a **child** function.
- **Note**: Diagramming tools (like Rhapsody) can ensure well-formedness.

Well-Formedness Continued



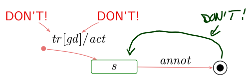
- Each non-empty region has **exactly one** initial pseudo-state and at least one transition from there to a state of the region, i.e.
  - for each $s \in S$ with $region(s) = \{S_1, \dots, S_n\}$,
  - for each $1 \leq i \leq n$, there exists exactly one initial pseudo-state $(s_1^i, init) \in S_i$ and at least one transition $t \in \to$ with s_1^i as source,



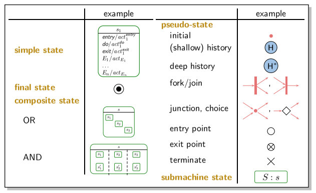
- Initial pseudo-states are not targets of transitions.

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. $t \in \to$ from s with $kind(s) = st$ implies $annot(t) = (\_, true, act)$.
- Final states are not sources of transitions.



Plan

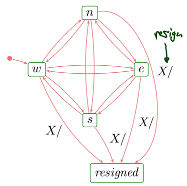


- Composite states.
- Initial pseudostate, final state.
- Entry/do/exit actions, internal transitions.
- History and other pseudostates, the rest.

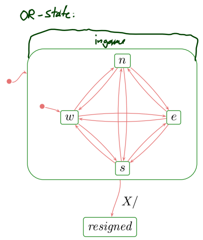
Composite States

Composite States

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

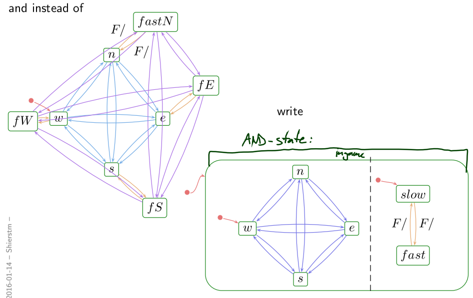


write

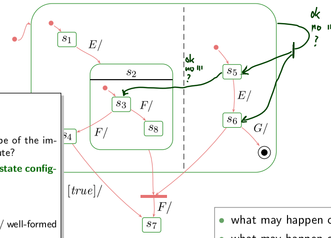


Composite States

and instead of



Composite States: Blessing or Curse?



Plan:

States:

- what is the type of the implicit st attribute?
- what are **legal state configurations**?

Transitions:

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

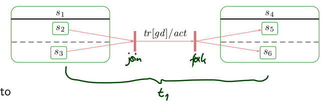
- what may happen on E?
- what may happen on $\underline{E}, F$?
- can $\underline{E}, G$ kill the object?
- ...

Syntax: Fork/Join

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

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

- For instance,



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

State Configuration

- The type of (implicit attribute) st is from now on **a set of** states, i.e. $\mathscr{D}(S_{\mathcal{M}_C}) = 2^S$
- 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 $\emptyset \neq R \in region(s)$, exactly one (non pseudo-state) child of s is in S_1, i.e.

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

$\{n, fast\}$

$\{top, ingame, n, fast\}$

n fast

ingame

top

$\{n, fast, w\}$

not OK

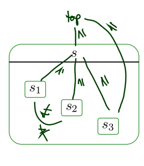
$\{n\}$

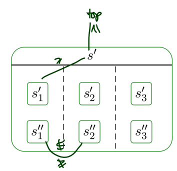
not OK

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 $s'' \leq s'$.



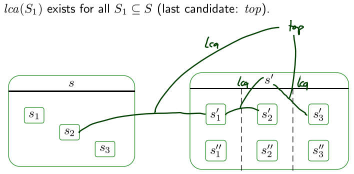


Least Common Ancestor

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

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

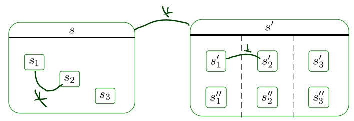
  - $lca(S_1)$ is ~~minimal~~ maximal, i.e. if $\hat{s} \leq s$ for all $s \in S_1$, then $\hat{s} \leq lca(S_1)$
- **Note**: $lca(S_1)$ exists for all $S_1 \subseteq S$ (last candidate: top).



Orthogonal States

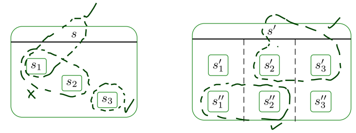
- Two states $s_1, s_2 \in S$ are called **orthogonal**, denoted $s_1 \perp s_2$, if and only if
  - they are unordered, i.e. $s_1 \not\leq s_2$ and $s_2 \not\leq s_1$, and
  - they live in different regions of an AND-state, i.e.

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



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
  - $s \perp s'$.



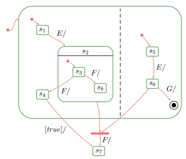
Legal Transitions

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

- source and destination are consistent, i.e. $\downarrow source(t)$ and $\downarrow target(t)$,
- source (and destination) states are pairwise orthogonal, i.e.
  - forall $s, s' \in source(t)$ $(\in target(t))$, $s \perp s'$,
- 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.

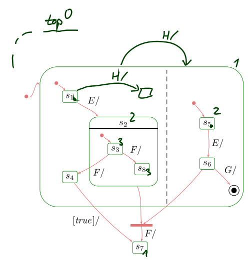
Example:



The Depth of States

- $depth(top) = 0$,
- $depth(s') = depth(s) + 1$, for all $s' \in child(s)$

Example:



Enabledness in Hierarchical State-Machines

- The **scope** ("set of possibly affected states") of a transition t is the **least common region** (!) of

$$source(t) \cup target(t).$$

- 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.

$$prio(t) := \max\{depth(s) \mid s \in source(t)\}$$

- A set of transitions $T \subseteq \to$ 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), in their scope
  - all transitions in T share the same trigger,
  - for all $t \in T$, the source states are active, i.e.

$$source(t) \subseteq \sigma(u)(st) \; (\subseteq S).$$

as before

  - all guards are satisfied by $\sigma(u)$, $\boxed{\text{and}}$

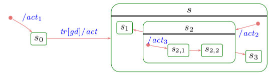
Transitions in Hierarchical State-Machines

- Let T be a set of transitions enabled in u.
- Then $(\sigma, \varepsilon) \xrightarrow[u]{(cons, Snd)} (\sigma', \varepsilon')$ 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**,
  - σ', ε', $cons$, and 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.

$\rightsquigarrow$ 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 ($\rightarrow$ 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.

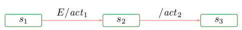
Final States

- If $(\sigma, \varepsilon) \xrightarrow[u]{(cons, Snd)} (\sigma', \varepsilon')$
  and all **simple states** in $st \in \sigma(u)(st)$ are **final**, i.e. $kind(s) = fin$, then
  - stay **unstable** if there is a common parent of the simple states in $\sigma(u)(st)$ which is source of a transition without trigger and satisfied guard,
  - otherwise **kill** u.

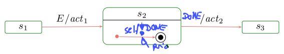
$\rightsquigarrow$ adjust Rules (i), (ii), (iii), and (v) accordingly.

Observation: u never "survives" reaching a state (s, fin) with $s \in child(top)$.

Observation:



vs.



References

References

OMG (2011a). Unified modeling language: Infrastructure, version 2.4.1. Technical Report formal/2011-08-05.

OMG (2011b). Unified modeling language: Superstructure, version 2.4.1. Technical Report formal/2011-08-06.