Software Design, Modelling and Analysis in UML

Lecture 15: Hierarchical State Machines II

2017-01-10

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

Albert-Ludwigs-Universität Freiburg, Germany

Content

- Hierarchical State Machines
  - Recall:
    - Abstract Syntax: States
    - (Legal) System Configurations
  - Abstract Syntax: Transitions
    - orthogonal states,
    - legal transitions
  - Enabledness of Fork/Join Transitions
    - least common ancestor,
    - scope,
    - priority and depth,
    - maximality
  - Transitions (or steps)
    of Hierarchical State Machines

Recall

Blessing or Curse. . . ?

Plan:

States / Syntax:

- What is the abstract syntax of a diagram?

States / Semantics:

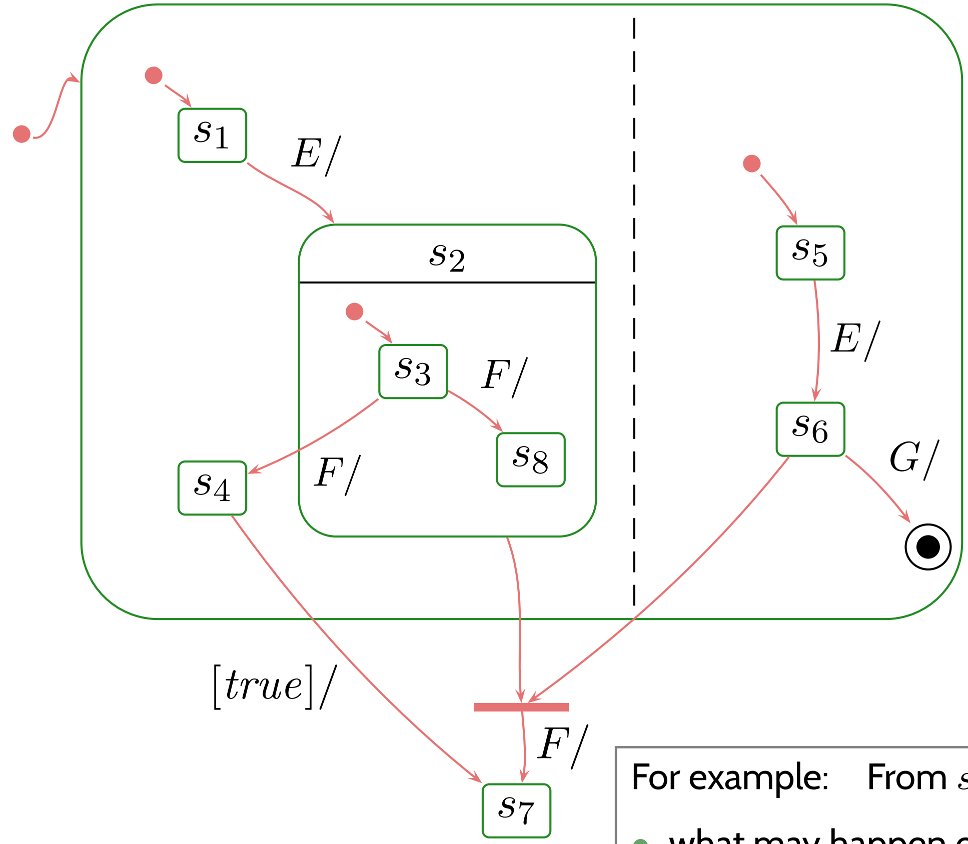
- what is the type of the implicit st attribute?
- what are **legal system configurations**?

Transitions / Syntax:

- what are **legal** / well-formed transitions?

Transitions / Semantics:

- when is a legal transition enabled?
- which effects do transitions have?



For example: From s_1, s_5,

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

Representing All Kinds of States

- **So far**:

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

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

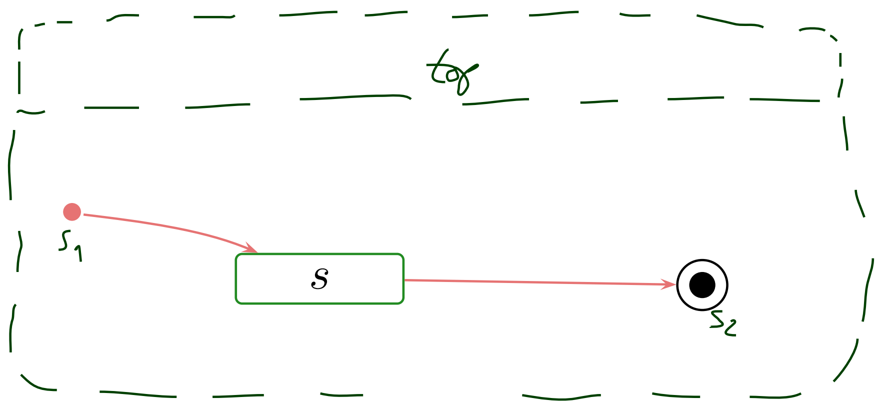
$$(S, kind, region, \rightarrow, \psi, annot)$$

where

- $S \supseteq \{top\}$ is a finite set of states **(new: top)**,
- $kind : S \rightarrow \{$*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



... denotes $(S, kind, region, \to, \psi, annot)$ with

- $S = \{top, s_1, s, s_2\}$
- $kind = \{top \mapsto \textit{st}, s_1 \mapsto \textit{init}, s \mapsto \textit{st}, s_2 \mapsto \textit{fin}\}$
- or $(S, kind) = \{(top, \textit{st}), (s_1, \textit{init}), (s, \textit{st}), (s_2, \textit{fin})\}$
- $region = \{top \mapsto \{\{s_1, s, s_2\}\}, s_1 \mapsto \emptyset \quad , s \mapsto \emptyset \quad , s_2 \mapsto \emptyset \quad \}$
- $\to, \psi, annot$: in a minute.

– 14 – 2016-12-22 – Shiersyn –

– 15 – 2017-01-10 – main –

Recall

Plan:

States / Syntax:

- What is the abstract syntax of a diagram? ✓

States / Semantics:

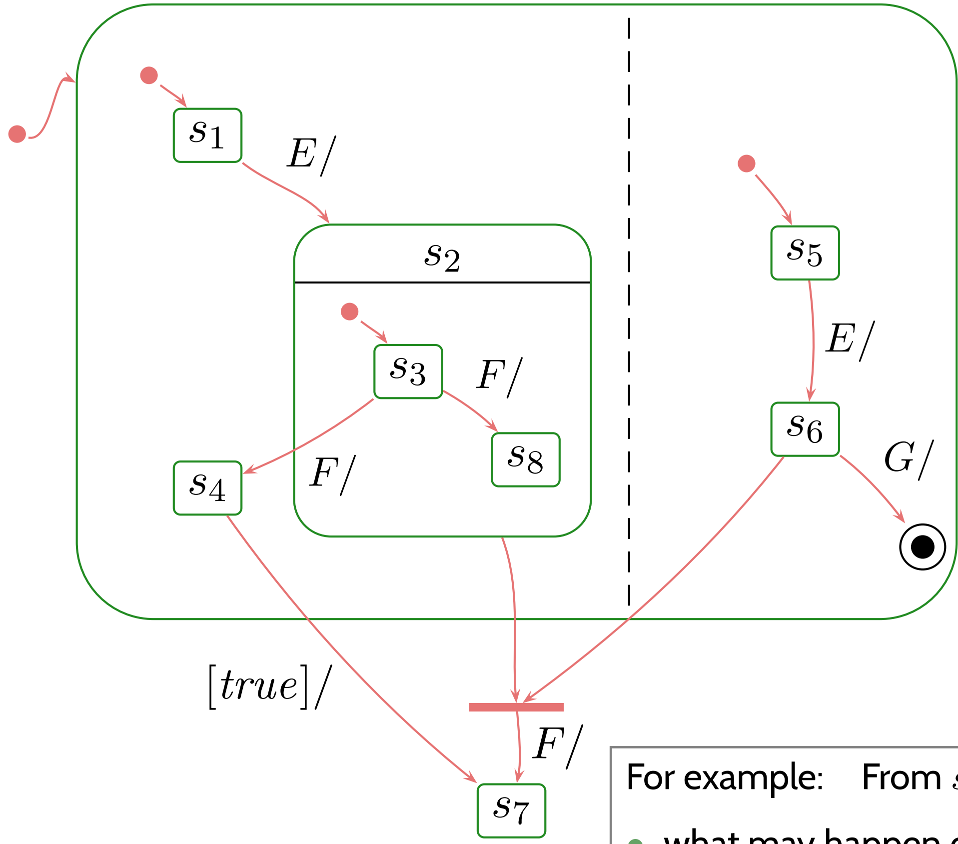
- what is the type of the implicit st attribute?
- what are **legal system configurations**?

Transitions / Syntax:

- what are **legal** / well-formed transitions?

Transitions / Semantics:

- when is a legal transition enabled?
- which effects do transitions have?



For example: From s_1, s_5,

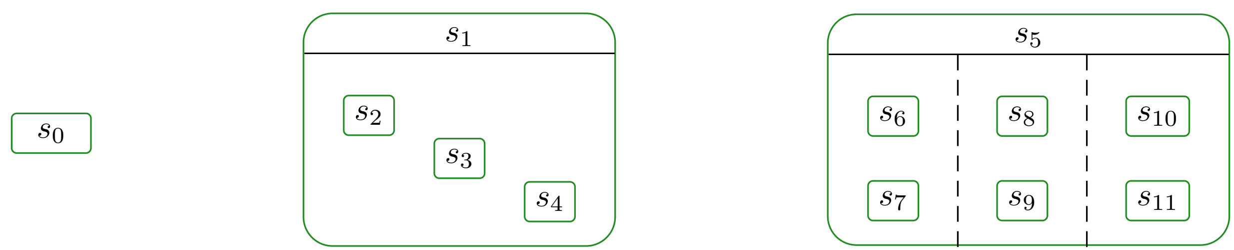
- what may happen on E?
- what may happen on $\underline{E}, F$?
- can $\underline{E}, G$ kill the object?
- ...

Semantics: State Configuration

- The type of (implicit attribute) st is from now on **a set of** states, i.e. $\mathscr{D}(S_{M_C}) = 2^S$
- A set $S_1 \subseteq S$ is called (**legal**) **state configuration** if and only if
  - $top \in S_1$, and
  - for each region R of a state in S_1,
    exactly one (non pseudo-state) element of R is in S_1, i.e.

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

- **Examples**:



$S_1 = \{s_0\}$ ✗
$S_2 = \{s_0, top\}$ ✓

$S_3 = \{s_1, top\}$ ✗
$S_4 = \{s_1, top, s_3, s_4\}$ ✗
$S_5 = \{s_1, top, s_4\}$ ✓

$S_6 = \{s_5, top, s_6, s_9\}$ ✗
$S_7 = \{s_5, top, s_6, s_9, s_{10}\}$ ✓
$S_8 = \{top, s_0, s_1, s_2\}$ ✗

Recall

Plan:

States / Syntax:

- What is the abstract syntax of a diagram? ✓

States / Semantics:

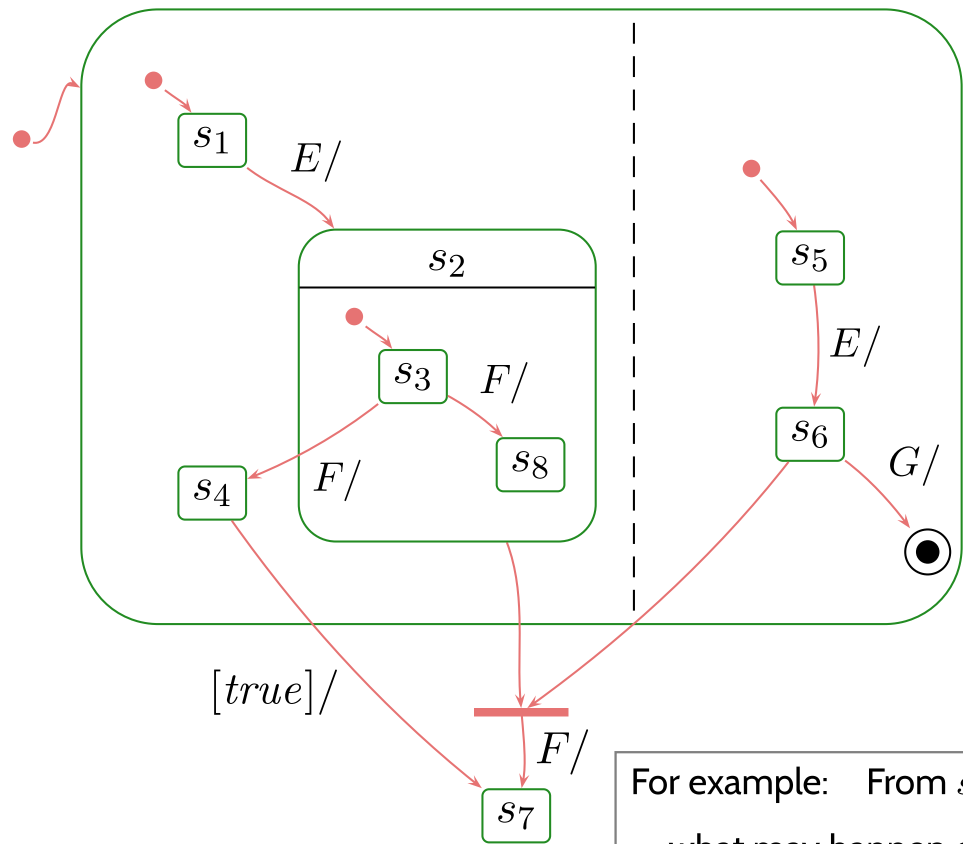
- what is the type of the implicit st attribute?
- what are **legal system configurations**? ✓

Transitions / Syntax:

- what are **legal** / well-formed transitions?

Transitions / Semantics:

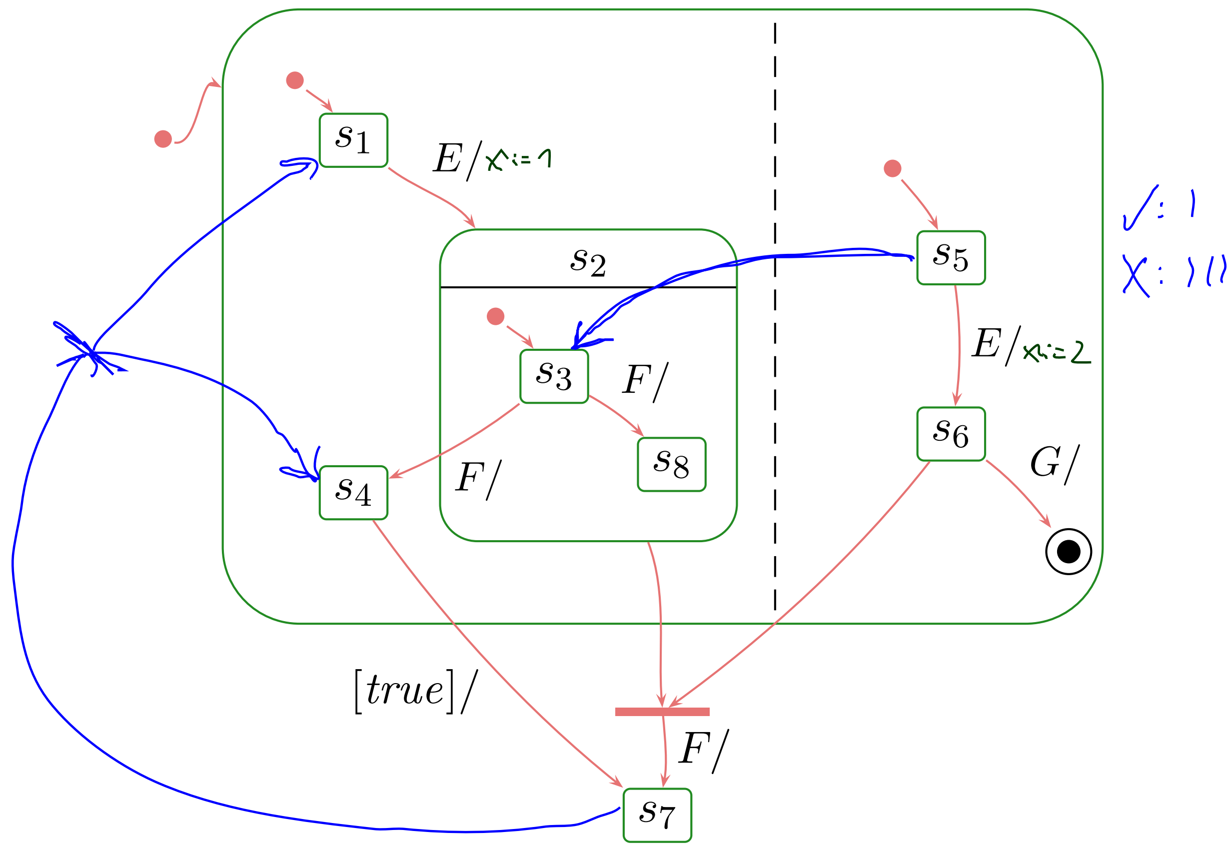
- when is a legal transition enabled?
- which effects do transitions have?



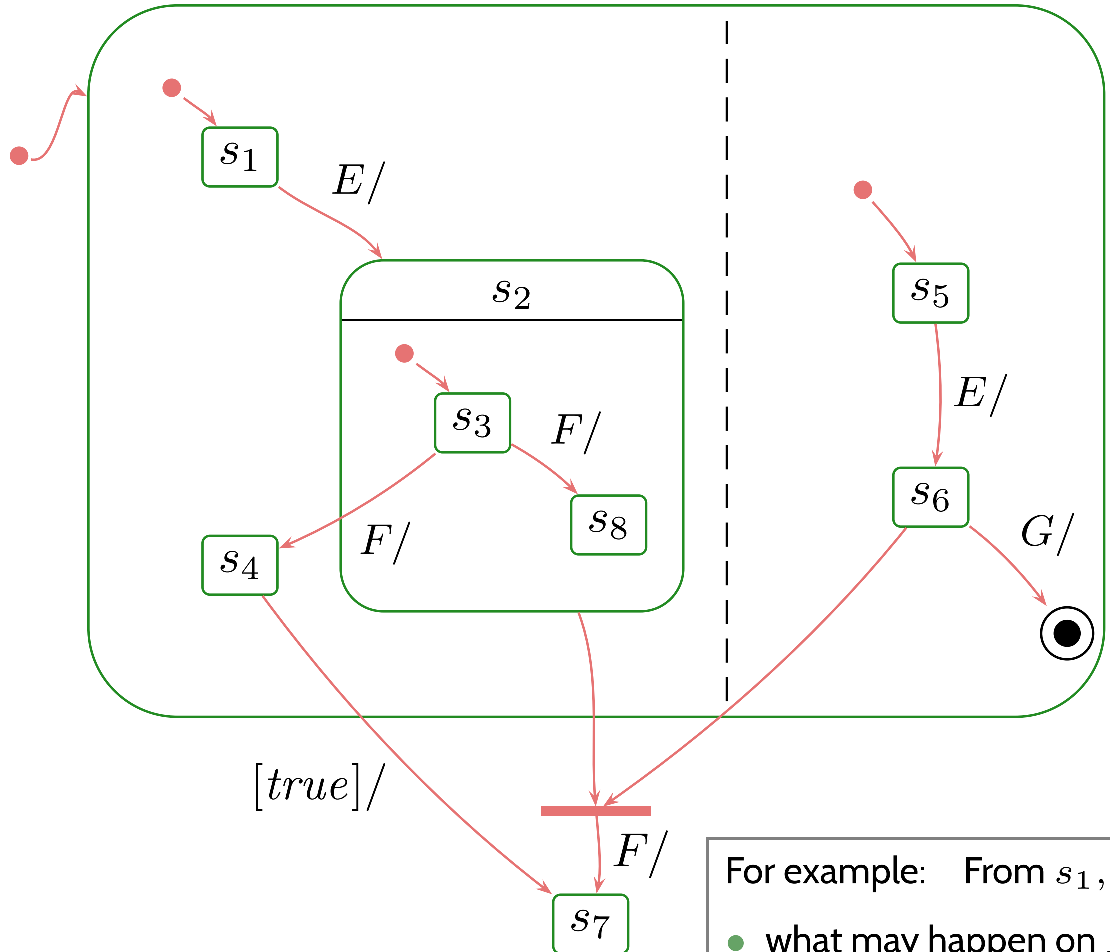
For example: From s_1, s_5,

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

Blessing or Curse. . . ?



Recall



Plan:

States / Syntax:

- What is the abstract syntax of a diagram?

States / Semantics:

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

Transitions / Syntax:

- what are **legal** / well-formed transitions?

Transitions / Semantics:

- when is a legal transition enabled?
- which effects do transitions have?

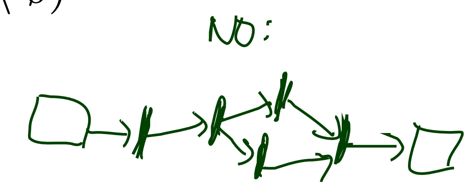
For example: From s_1, s_5,

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

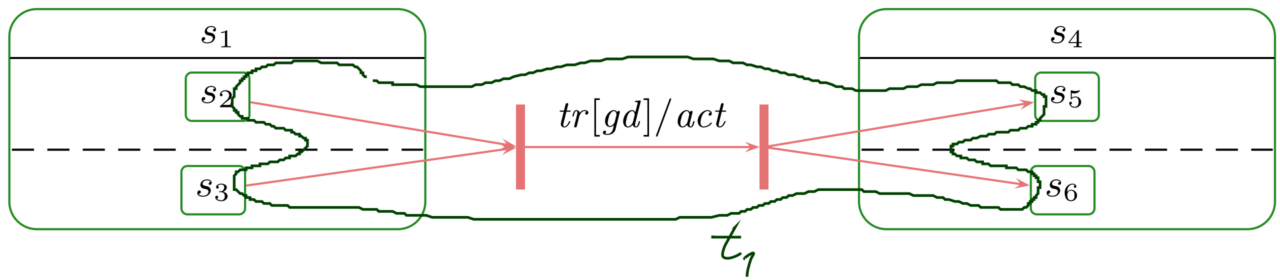
Transitions Syntax: Fork/Join

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

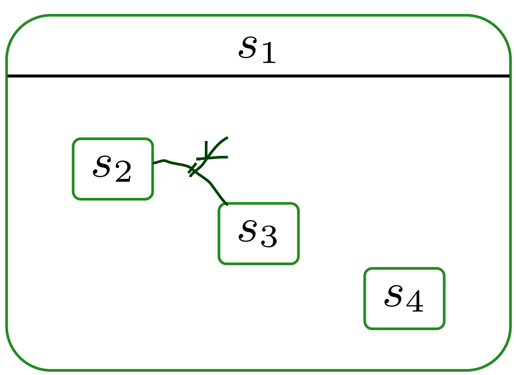
Orthogonal States

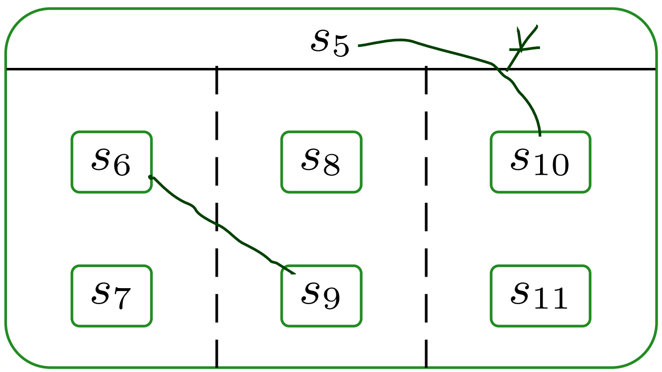
- Two states $s_1, s_2 \in S$ are called **orthogonal**, denoted $s_1 \perp s_2$, if and only if
  - they "live" in different regions of **one** AND-state, i.e.

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

$$\mathit{region}(s_5) = \{ \underbrace{\{s_6, s_7\}}_{S_i}, \underbrace{\{s_8, s_9\}}_{S_j}, \{s_{10}, s_{11}\} \}$$

$$s_6 \in \mathit{child}^*(S_i), \quad s_9 \in \mathit{child}^*(S_j)$$





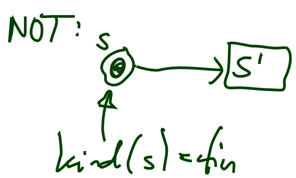
Legal Transitions

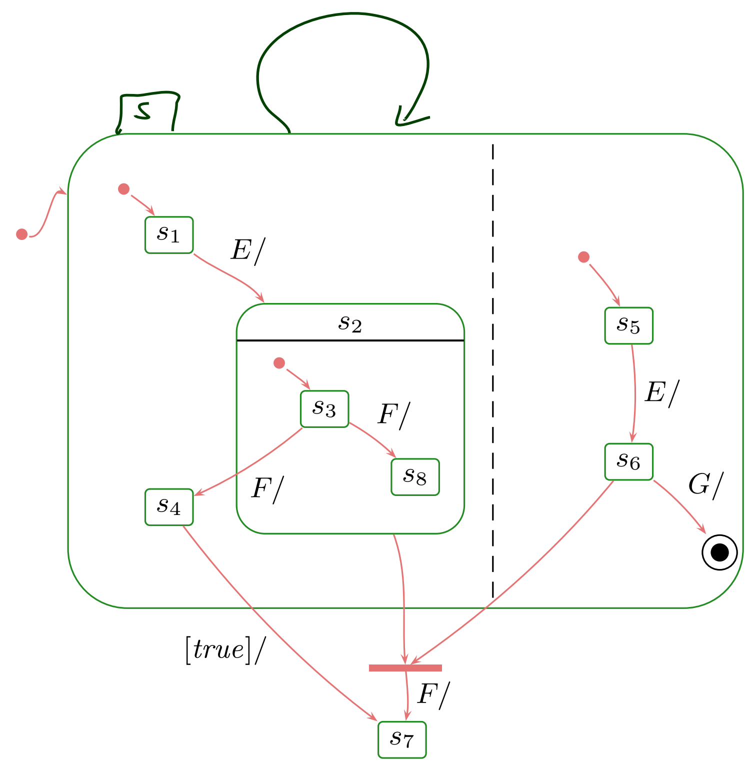
A hierarchical state-machine $(S, kind, region, \rightarrow, \psi, annot)$
is called **well-formed** if and only if for all transitions $t \in \rightarrow$,

- source (and destination) states are pairwise orthogonal, i.e.
  - $\forall\, s \neq s' \in source(t)\ (\in target(t)) \bullet s \perp s'$,
- the top state is neither source nor destination, i.e.
  - $top \notin source(t) \cup \cancel{soured}(t)$.
    target

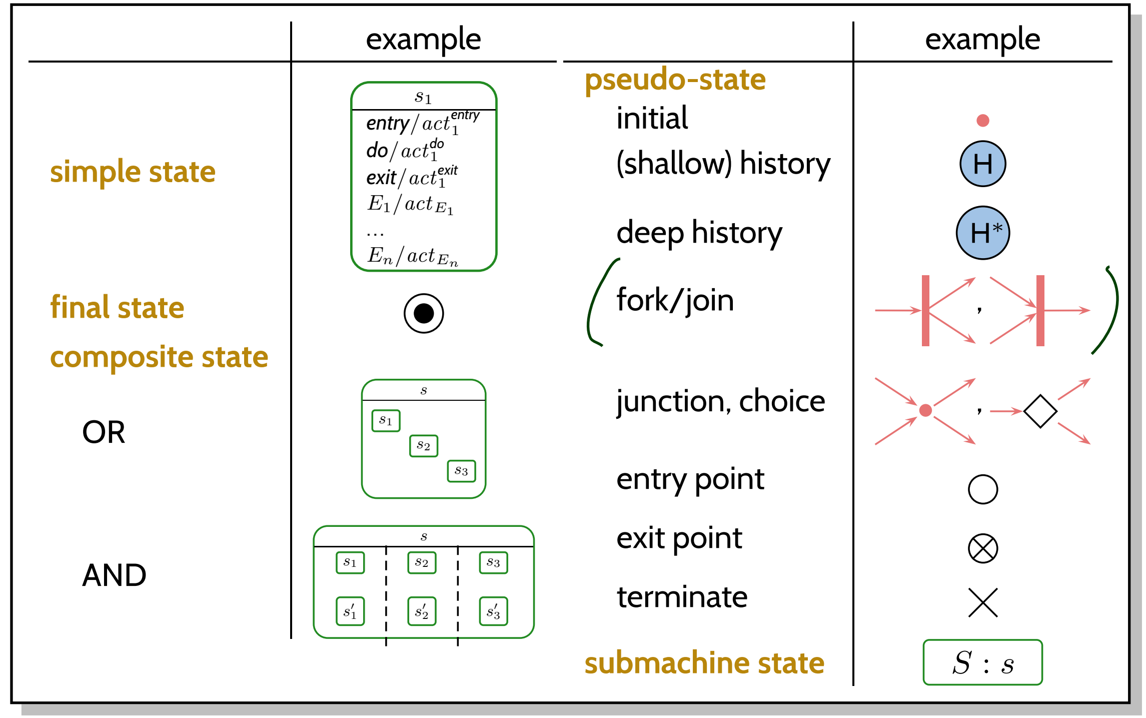
Recall: final states are not sources of transitions.

Example:

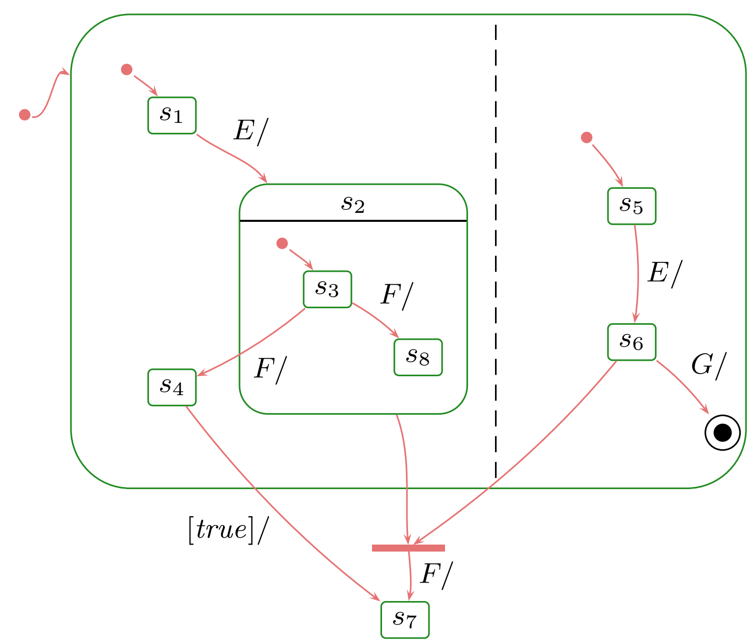




Plan



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



Scope

- 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 disjoint.

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 $s'' \leq s'$.

$s' \geq s \leq s''$

OR $s \geq s'$ iff $s \in child^*(s')$

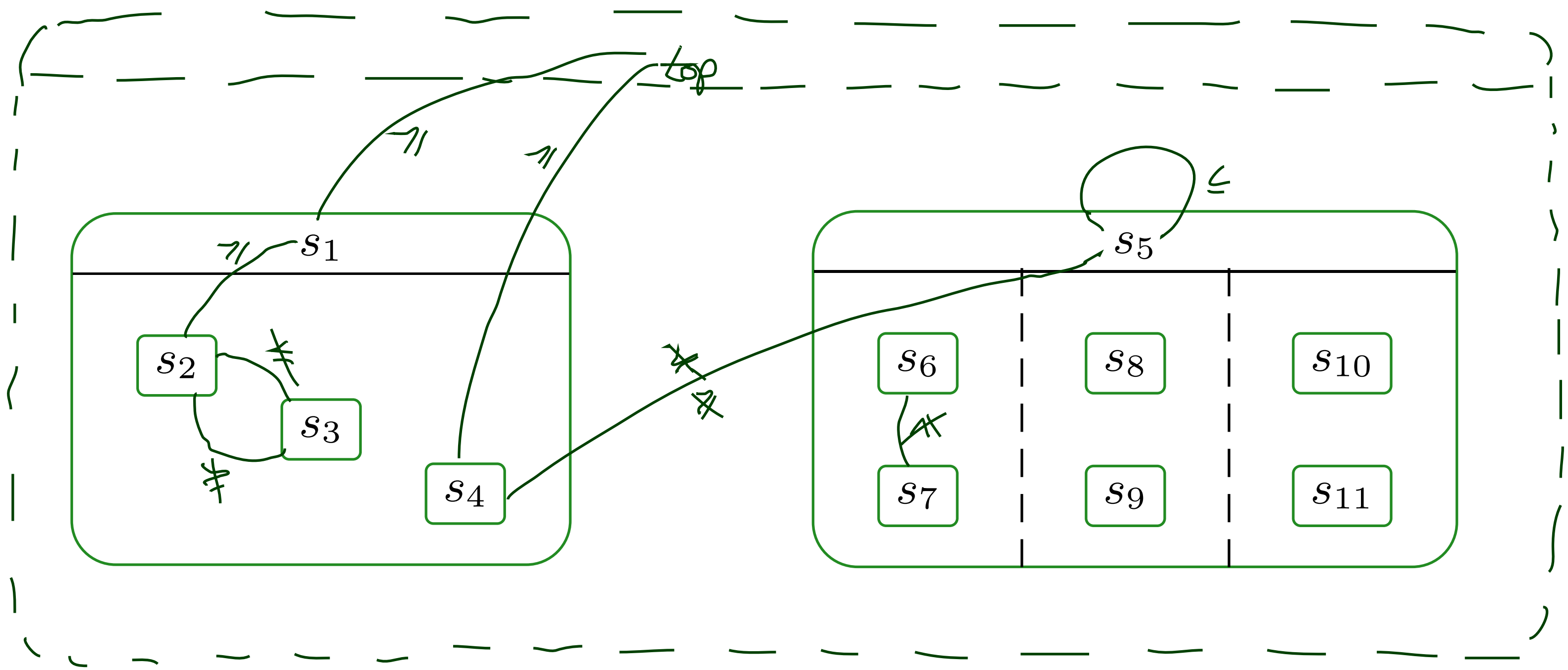
A Partial Order on States

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 $s'' \leq s'$.

OR $s \geq s'$ iff $s \in child^*(s')$

$s' \geq s \leq s''$



Least Common Ancestor

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

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

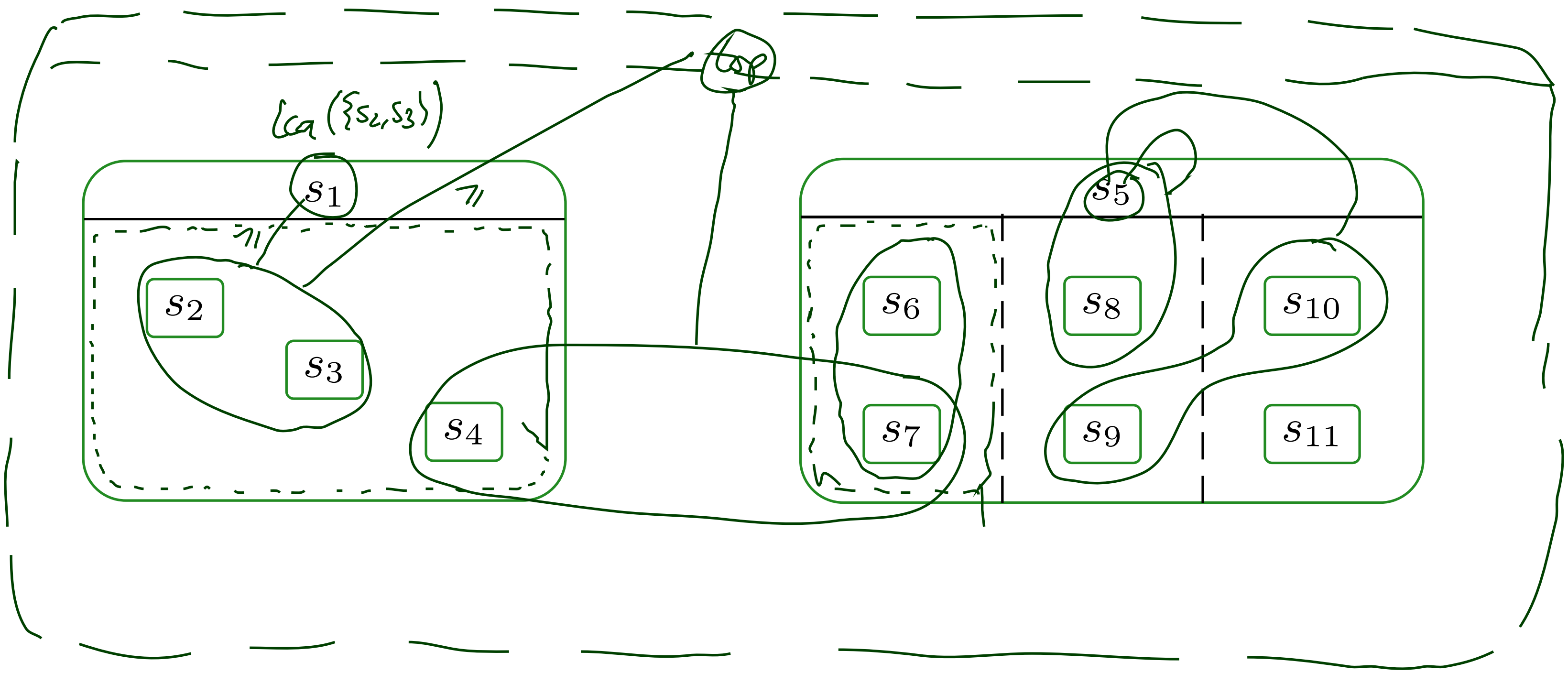
(1)

  - (2) $lca(S_1)$ is maximal, i.e. if $\hat{s} \leq s$ for all $s \in S_1$ then $\hat{s} \leq lca(S_1)$

$(\forall s \in S_1 \bullet \hat{s} \leq s)$

$\Rightarrow \hat{s} \leq lca(S_1)$

- **Note**: $lca(S_1)$ exists for all $S_1 \subseteq S$ (last candidate: top).



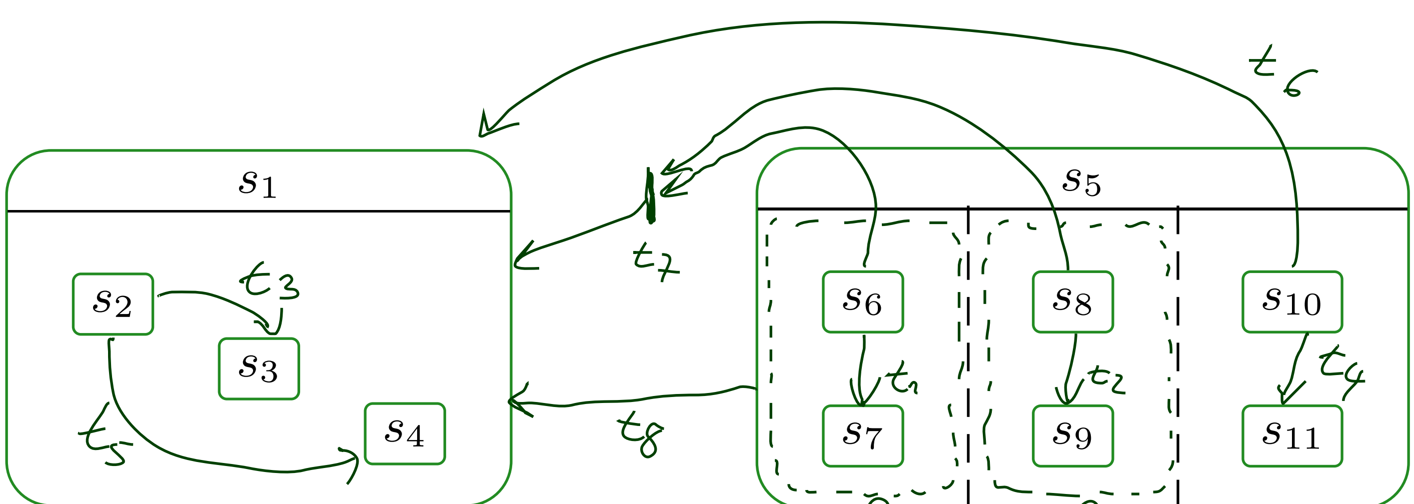
Scope

- 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 disjoint.

t₁ not cons. t₆



$R_1 = scope(t_1) \cap R_2 = scope(t_2) = \emptyset$

t_3 cons. t_4

t_3 not cons. t_5

Priority and Depth

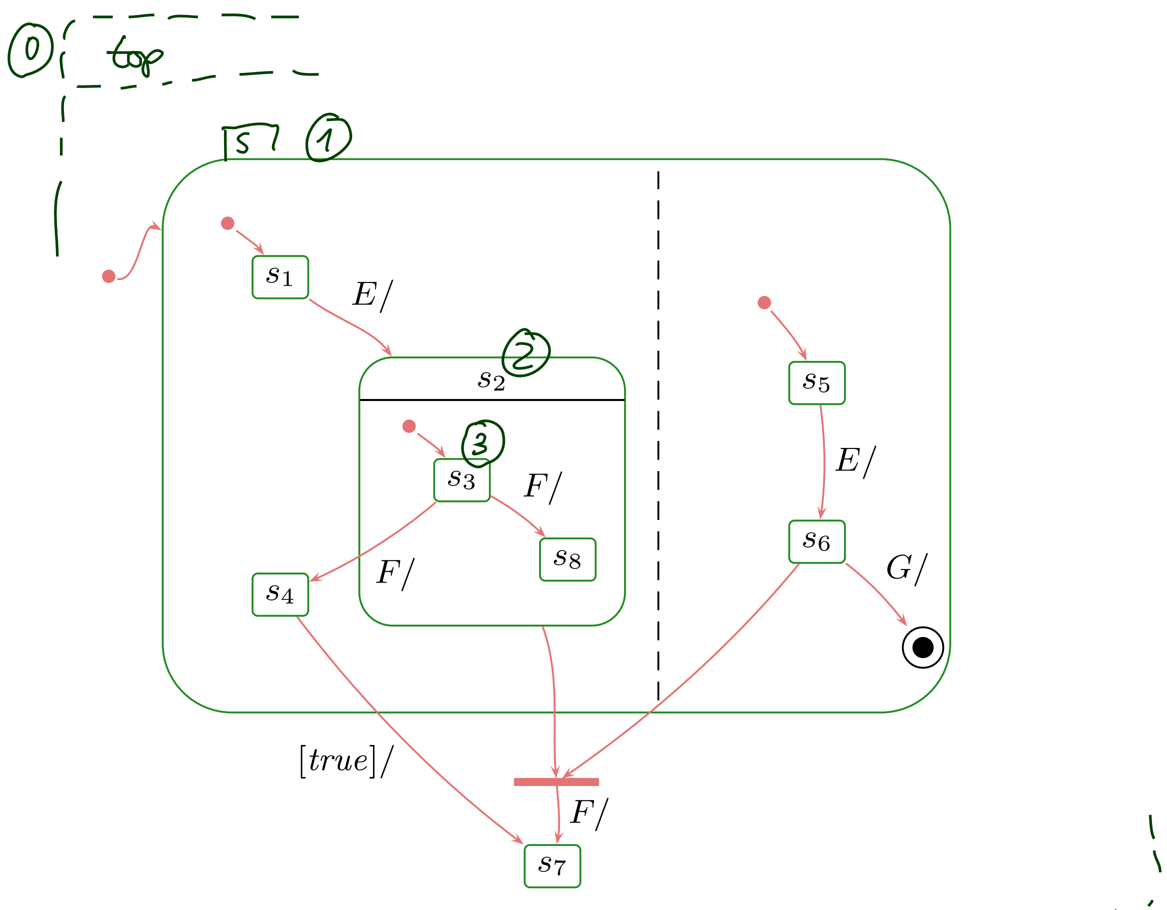
- The **priority** of transition t is the depth of its innermost source state, i.e.

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

  where

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

Example:



Enabledness in Hierarchical State-Machines

- A set of transitions $T \subseteq \rightarrow$ is **enabled** for an object u in (σ, ε) if and only if
  - T is **consistent**,
  - for all $t \in T$, the **source states are active**, i.e.

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

  - all transitions in T **have the same trigger** tr and
    - $tr = \_$ and u is **unstable**, or
    - $tr = E$ and there is an E ready for u in ε,
  - the guards of all transitions in T are satisfied in $\tilde{\sigma}$ wrt. u, and

A set T of **enabled transitions** is called **maximal** wrt.

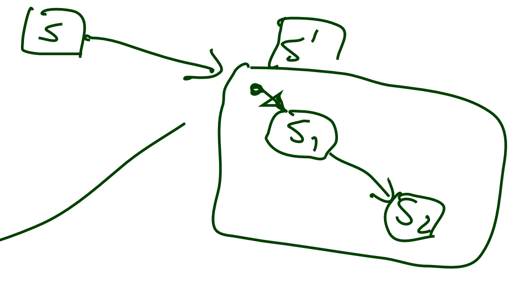
- **extension** if and only if there is no transition $t' \notin T$ such that $T \cup \{t'\}$ is enabled.
- **priority** if and only if for each $t \in T$, there is no $t' \in \rightarrow$ such that
  - $prio(t') > prio(t)$,
  - $(T \setminus \{t\}) \cup \{t'\}$ is enabled, and
  - $st' \geq st$ for some $st' \in source(t')$ and $st \in source(t)$.

Transitions in Hierarchical State-Machines

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

  - 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 (i), (ii), (iii), (v) accordingly.

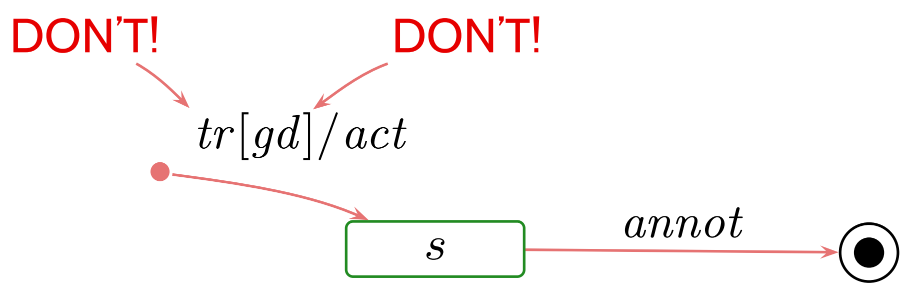
(For state machines with only simple states, and no trigger, guard, or action on transitions originating at initial states: **Same behaviour as before**.)

Additional Well-Formedness Constraints

- 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 $(s_1^i, \mathit{init}) \in S_i$ and at least one transition $t \in \rightarrow$ with s_1^i as source,
- Initial pseudo-states are not targets of transitions.

For simplicity:

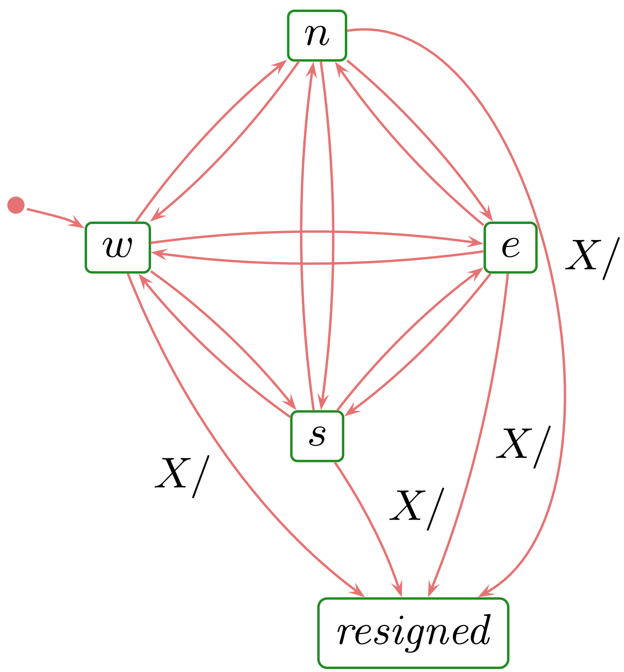
- The target of a transition with initial pseudo-state source in S_i is (also) in S_i.
- Transitions from initial pseudo-states have no trigger or guard, i.e. $t \in \rightarrow$ from s with $kind(s) = \mathit{st}$ implies $annot(t) = (\_, \mathit{true}, act)$.
- Final states are not sources of transitions.



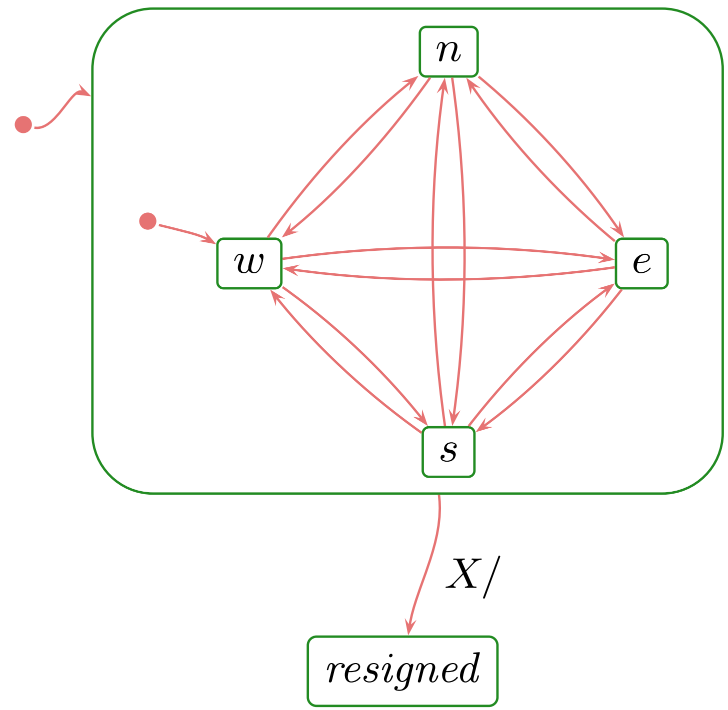
An Intuition for “Or-States”

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

- **Idea**: instead of

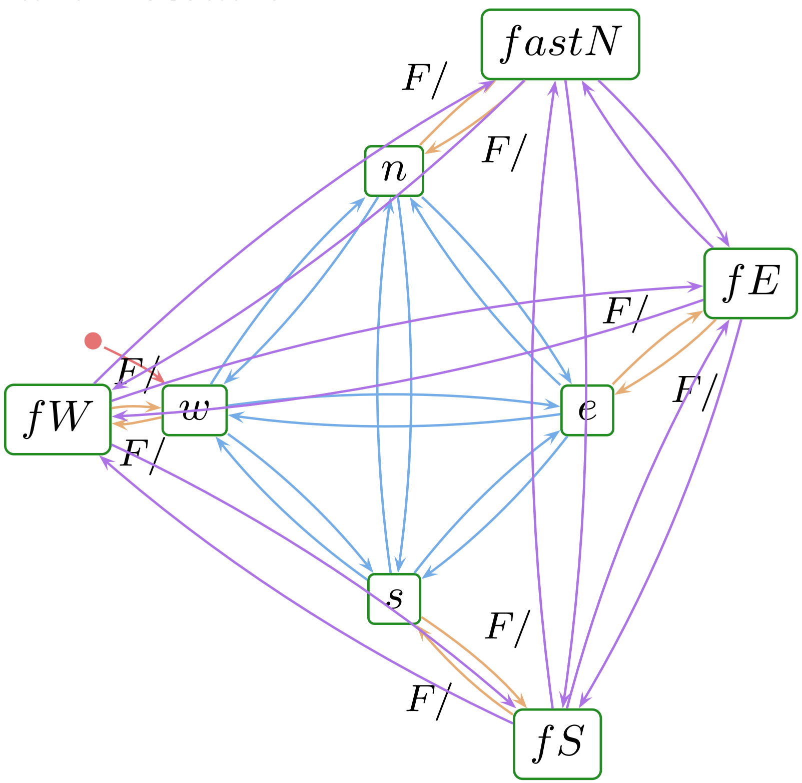


write

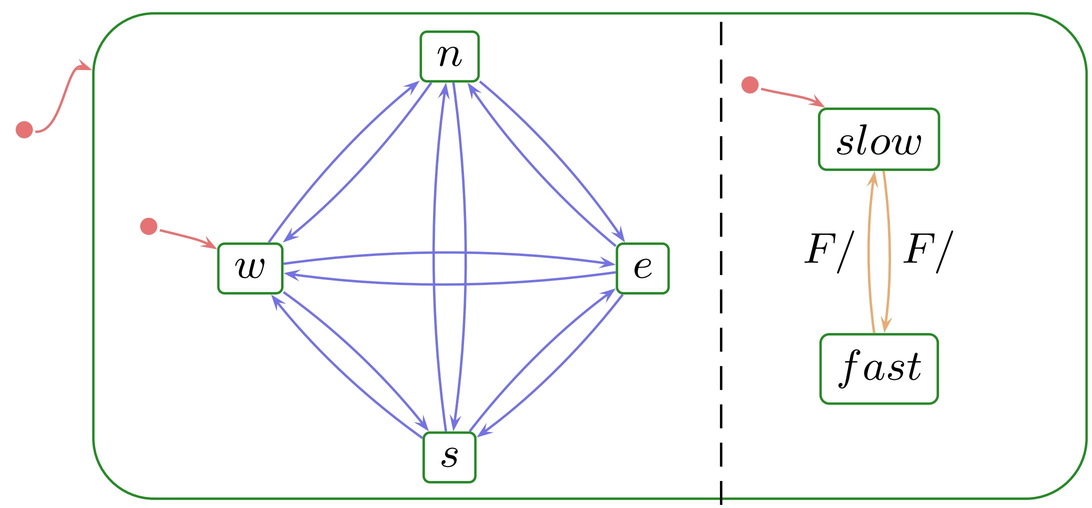


An Intuition for “And-States”

and instead of



write



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.