Theoretical Foundations of the UML

Lecture 18: Statecharts Semantics

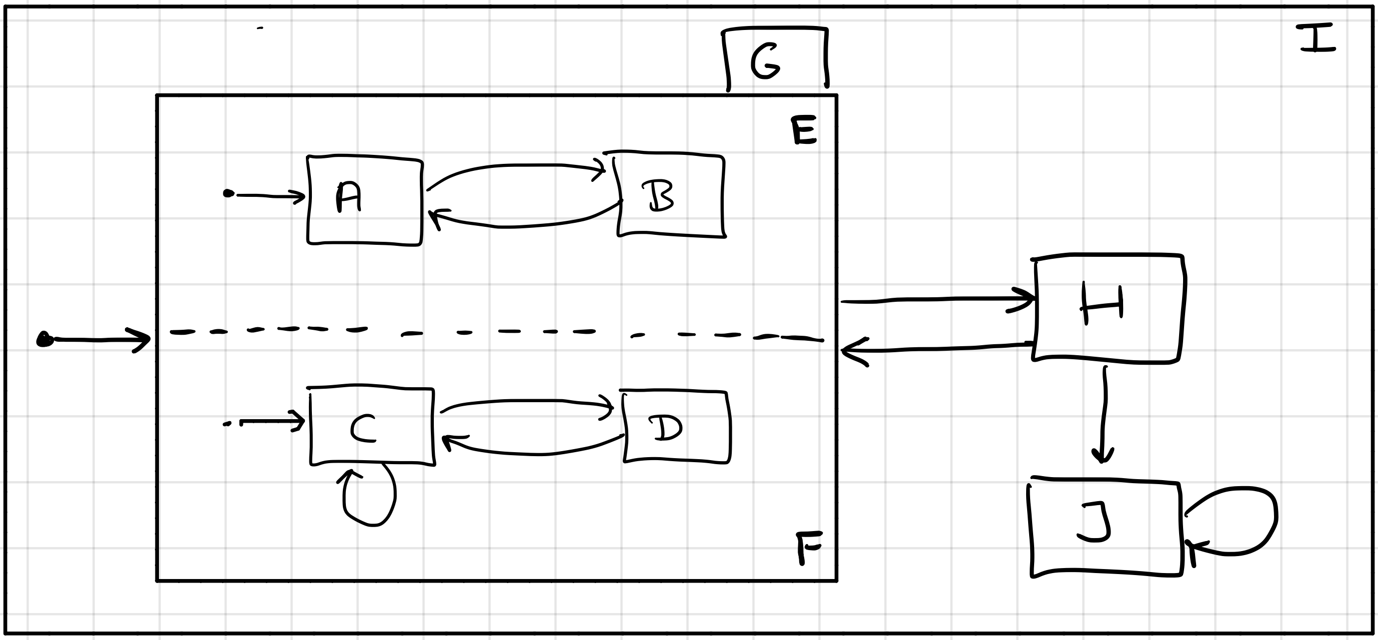
Joost-Pieter Katoen

Lehrstuhl für Informatik 2
Software Modeling and Verification Group

`moves.rwth-aachen.de/teaching/ss-20/fuml/`

June 23, 2020

Outline



I = root node

N = set of nodes $= \{A, B, C, \dots, I, J\}$

children : $N \to 2^N$ e.g. children $(H) = \emptyset$

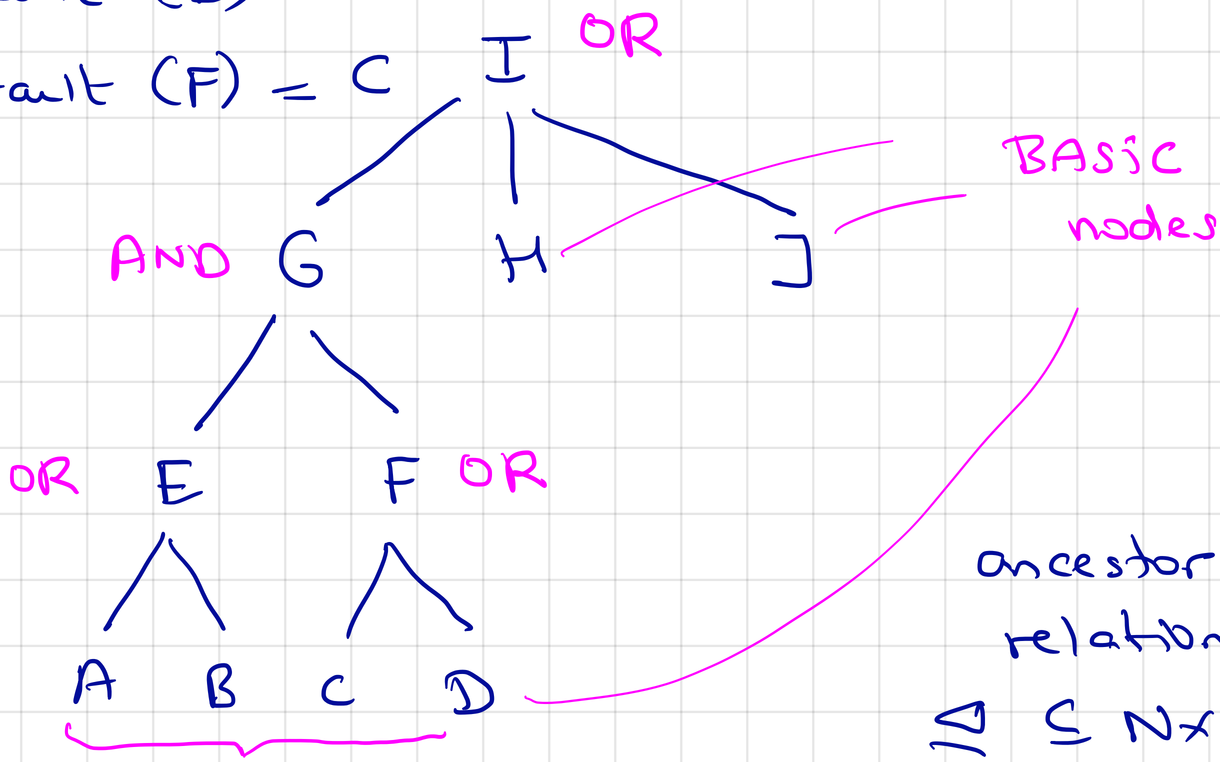
children $(I) = \{G, H, J\}$

children $(G) = \{E, F\}$

default $(I) = G$

default $(E) = A$

default $(F) = C$



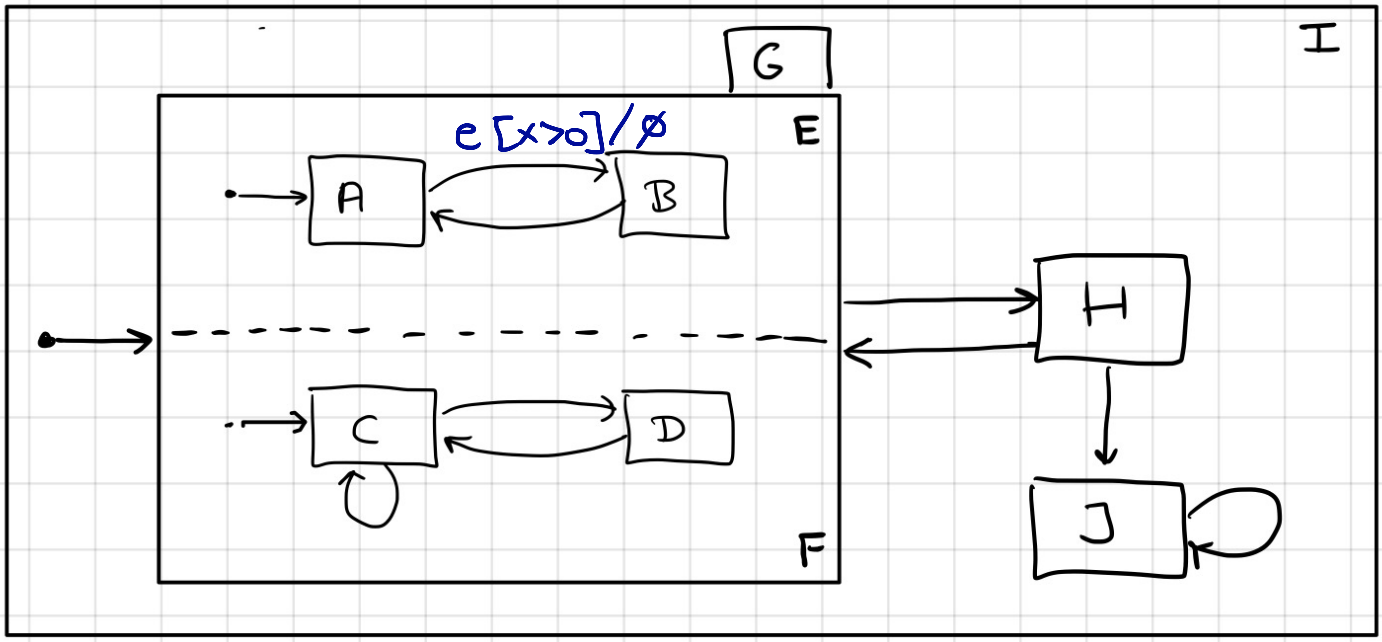
ancestor relation

$\trianglelefteq \subseteq N \times N$

$C \trianglelefteq F \qquad F \trianglelefteq G$

$C \trianglelefteq G$

$H \not\trianglelefteq D$



Edges: $\underset{\subseteq N}{X} \xrightarrow{e[g]/A} \underset{\subseteq N}{Y}$ hyperedge

event e

$\{A\} \xrightarrow{e[x>0]/\emptyset} \{B\}$

guard g

$\{G\} \xrightarrow{e[x=2]/\dots} \{H\}$

A actions

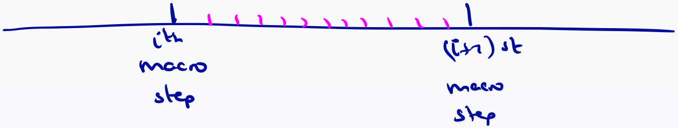
Overview

Towards a Statechart semantics

- Formal semantics: map $(SC_1, \ldots, SC_k)$ onto a single Mealy machine

Towards a Statechart semantics

- Formal semantics: map $(SC_1, \ldots, SC_k)$ onto a single Mealy machine
- This is done using a step semantics distinguishing macro and micro steps
- Macro steps are "observable" and are subdivided into a finite number of micro steps that cannot be prolonged



Towards a Statechart semantics

- Formal semantics: map $(SC_1, \ldots, SC_k)$ onto a single Mealy machine
- This is done using a step semantics distinguishing macro and micro steps

send j.e   send j+1, e'

- Macro steps are "observable" and are subdivided into a finite number of micro steps that cannot be prolonged

n   n+1   n+2

- In a macro step, a maximal set of edges is performed

assignments / send j.e

- Events generated in macro step n are only available in macro step $n+1$
  - If such event is not "consumed" in step $n+1$, it dies, and is not available in step $n+2$, $n+3$, …

Assumptions [Eshuis & Wieringa, 2000]

- Input to a macro step is a set of events (and not a queue)

  the order of event generation is ignored, i.e., if e and e' are generated in macro step i, the order in which they are generated is irrelevant in step $i{+}1$

unordered

Assumptions [Eshuis & Wieringa, 2000]

- Input to a macro step is a set of events (and not a queue)

  the order of event generation is ignored, i.e., if e and e' are generated in macro step i, the order in which they are generated is irrelevant in step $i{+}1$

- A macro step reacts to all available events

  events can only be used in macro step immediately following their generation

- Instantaneous edges and actions

- Unlimited concurrency

  there is no limit on the number of events that can be consumed in a macro step

- Perfect communication, i.e., messages are not lost

What does a single StateChart mean?

$\underbrace{(SC_1, \ldots, SC_k)}_{\text{several statecharts}} \longmapsto$ single Mealy machine

Intuitive semantics as a transition system:

- State = $\underbrace{\text{a set of nodes}}_{\text{in the statechart}}$ ("current control") + $\underbrace{\text{the values of variables}}$

currently "active" statechart nodes

Intuitive semantics as a transition system:

- State = a set of nodes ("current control") + the values of variables

  ↓ in the statechart

- Edge is enabled if guard holds in current state

$g = x > 0$ – needs to hold in current state

$X \xrightarrow{e\,[g]/A} Y$

enabled

needs to be available

What does a single StateChart mean?

Intuitive semantics as a transition system:

- State = a set of nodes ("current control") + the values of variables
- Edge is enabled if guard holds in current state
- Executing edge $X \xrightarrow{e[g]/A} Y$ = perform actions A, consume event e
  - leave source nodes X and switch to target nodes Y

  $\Rightarrow$ events are unordered, and considered as a set
- Principle: execute as many edges at once (without conflict)

  $\Rightarrow$ the total execution of such maximal set is a macro step

Overview

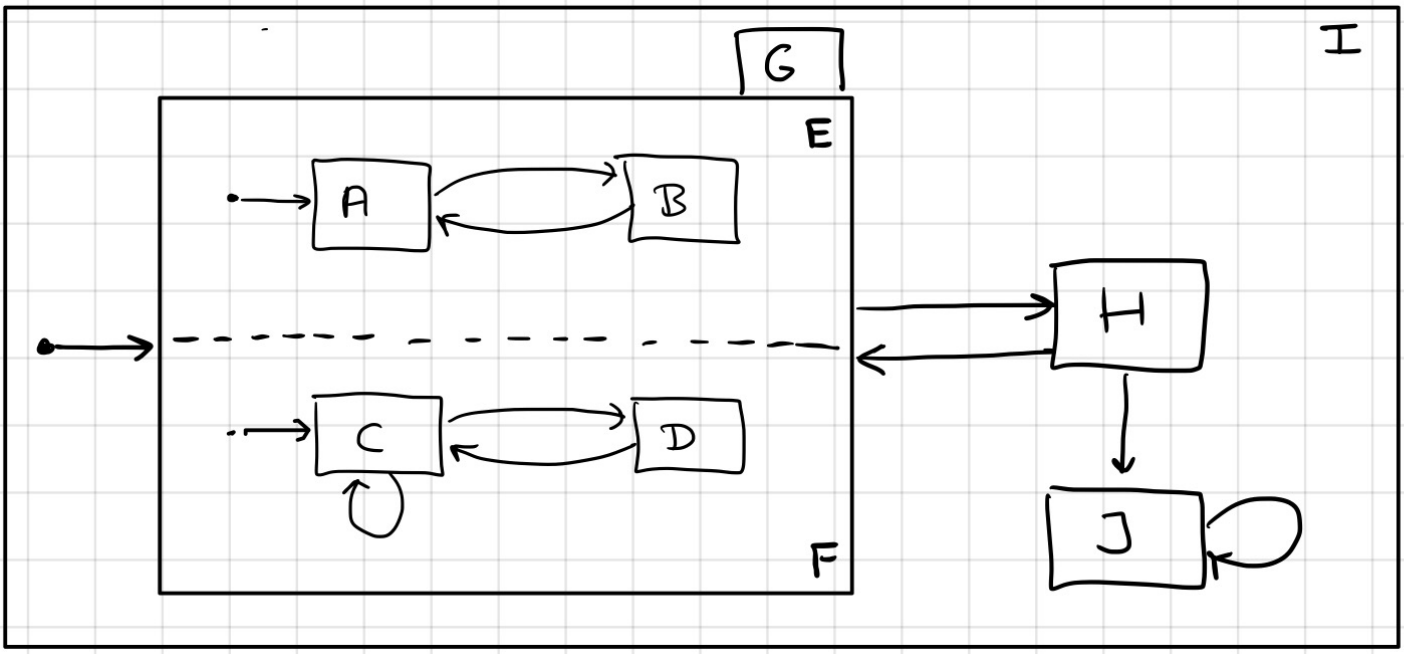
States and configurations

Definition (Configuration)

A configuration of $SC = (N, E, Edges)$ is a set $C \subseteq N$ of nodes satisfying:

- $\text{root} \in C$
- $x \in C$ and $type(x) = \text{OR}$ implies $|children(x) \cap C| = 1$
- $x \in C$ and $type(x) = \text{AND}$ implies $children(x) \subseteq C$

Let $Conf$ denote the set of configurations of SC.



Example configurations

$$C_1 = \{ I, H \}$$

$$C_2 = \{ I, G, E, F, \underline{A}, D \}$$

$$C_3 = \{ I, G, E, F, B, C \} \quad \text{etc.}$$

States $= \quad (C_1, \emptyset, \{x = 0, y = 0\})$

$$(C_2, \{e, e'\}, \{x = 1, y = 0\})$$

etc.

Enabledness: $\underline{\underline{\{A\}}} \xrightarrow{e\,[true]\,/\,\emptyset} \{B\}$

is enabled in $(\underline{\underline{C_2}}, \{e\}, V)$

States and configurations

Definition (Configuration)

A configuration of $SC = (N, E, Edges)$ is a set $C \subseteq N$ of nodes satisfying:

- root $\in C$
- $x \in C$ and $type(x) = \text{OR}$ implies $|children(x) \cap C| = 1$
- $x \in C$ and $type(x) = \text{AND}$ implies $children(x) \subseteq C$

Let $Conf$ denote the set of configurations of SC.

Definition (State)

State of $SC = (N, E, Edges)$ is a triple $\underline{(C, I, V)}$ where

- C is a configuration of SC
- $I \subseteq V$ is the set of events to be processed ("available" events)
- V is a valuation of the variables. e.g $x=3$, $y=17$, $c=$"A"

Overview

how to define the transition relation between states?

$$C_1, I_1, V_1 \longrightarrow C_2, I_2, V_2$$

Enabling of an edge

$x>0 \wedge y<10$

g is a global Boolean statement, covering the entire statecharts

Definition (Enabledness)

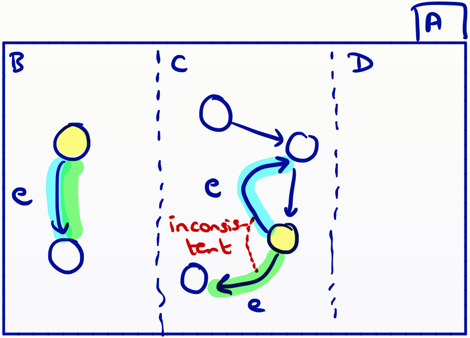
Edge $X \xrightarrow{e[g]/A} Y$ is enabled in state (C, I, V) whenever: $sc_1, \ldots, sc_n$

- $X \subseteq C$, i.e. all source nodes are in configuration C
- $(\underbrace{(C_1, \ldots, C_n)}_{\text{configurations}}, \underbrace{(V_1, \ldots, V_n)}_{\text{variable valuations}}) \models g$, i.e., guard g is satisfied in all statecharts
- either $e \neq \perp$ implies $e \in I$, or $e = \perp$

Let $En(C, I, V)$ denote the set of enabled edges in state (C, I, V).

Macro steps

- On receiving an input e, several edges in SC may become enabled
- Then, a maximal and consistent set of enabled edges is taken



Macro steps

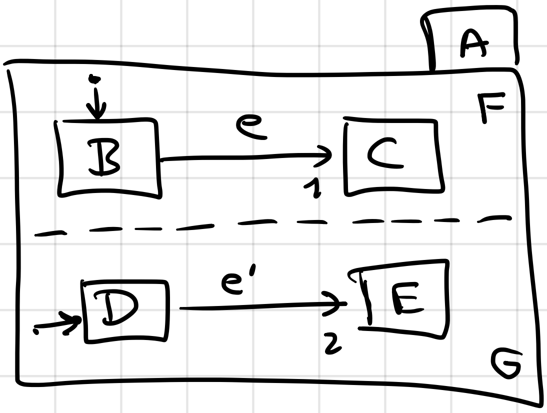
- On receiving an input e, several edges in SC may become enabled ✓
- Then, a maximal and consistent set of enabled edges is taken
- If there are several such sets, choose one nondeterministically
- Edges in concurrent components can be taken simultaneously
- But edges in other components cannot; they are inconsistent
- To resolve nondeterminism (partly), priorities are used

Overview

Consistency: examples

To define consistency formally, we need some auxiliary concepts

1



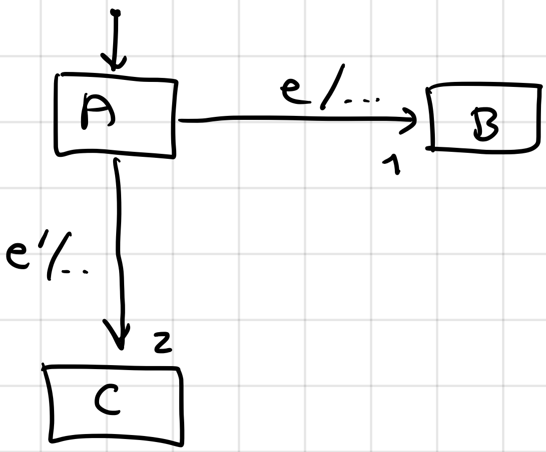
pair of edges (1,2)

intuition: (1,2) are consistent as they can happen in parellel

in state = $(\{A, F, G, B, D\}, \{e, e', \dots\}, V)$

both edges 1, 2 are enabled, and as they are consistent, they can both be taken.

2

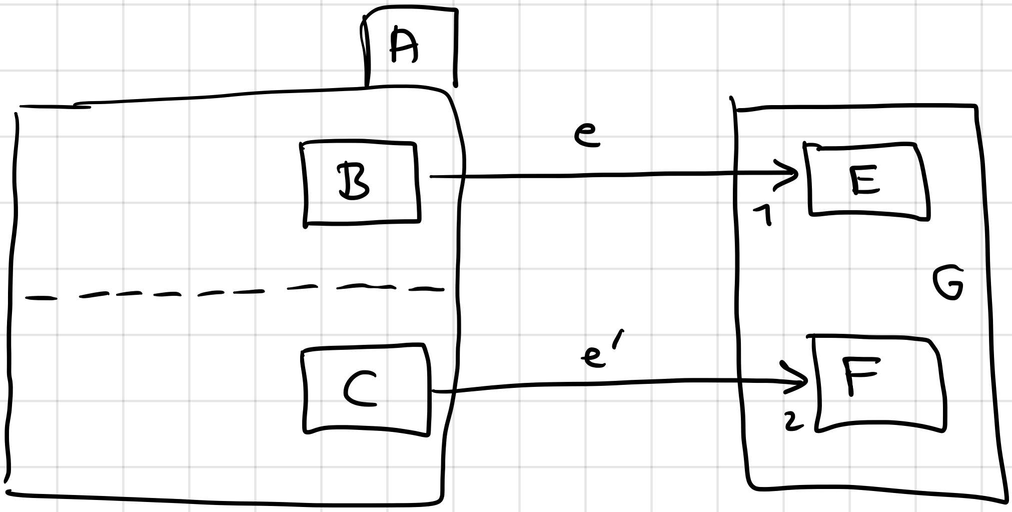


(1,2) is <u>not</u> consistent since from A only one edge can be taken

in state $(\{A, \dots\}, \{e, e', \dots\}, V)$

both 1, 2 are enabled, but only one of them can be taken

3



Edges (1,2) are inconsistent, as the targets E and F are both part of OR-node G.

Definition (Least common ancestor)

For $X \subseteq N$, the least common ancestor, denoted $lca(X)$, is the node $y \in N$ such that:

$$\underline{(\forall x \in X.\, x \trianglelefteq y)} \quad \text{and} \quad \forall z \in N.\, (\underline{\forall x \in X.\, x \trianglelefteq z}) \text{ implies } \underline{y \trianglelefteq z}.$$

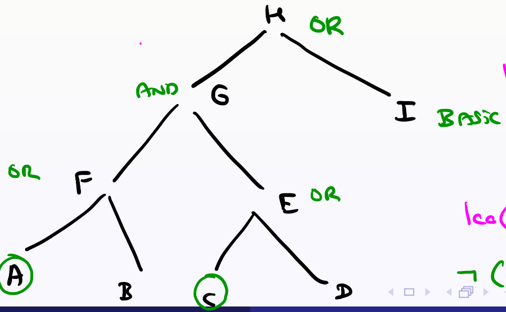
$lca\{A,C\} = G$

$type(G) = AND$

thus $A \perp C$

$F \perp E$

BASIC $\longrightarrow$



$X = \{C, D\}$

$lca(X) = E$

$X' = \{A, C, E\}$

$lca(X') = G$

$\neg(C \perp D)$

Least common ancestor

Definition (Least common ancestor)

For $X \subseteq N$, the least common ancestor, denoted $lca(X)$, is the node $y \in N$ such that:

$$(\forall x \in X.\, x \trianglelefteq y) \quad \text{and} \quad \forall z \in N.\, (\forall x \in X.\, x \trianglelefteq z) \text{ implies } y \trianglelefteq z.$$

Intuition

Node y is an ancestor of any node in X (first clause), and is a descendant of any node which is an ancestor of any node in X (second clause).

Orthogonality of nodes

Definition (Orthogonality of nodes)

Nodes $x, y \in N$ are orthogonal, denoted $x \perp y$, if

$$\neg(x \trianglelefteq y) \quad \text{and} \quad \neg(y \trianglelefteq x) \quad \text{and} \quad \mathit{type}(\mathit{lca}(\{\, x, y \,\})) = \text{AND}.$$

Orthogonality of nodes

Definition (Orthogonality of nodes)

Nodes $x, y \in N$ are orthogonal, denoted $x \perp y$, if

$$\neg(x \trianglelefteq y) \quad \text{and} \quad \neg(y \trianglelefteq x) \quad \text{and} \quad \mathit{type}(\mathit{lca}(\{\, x, y \,\})) = \text{AND}.$$

Orthogonality captures the notion of independence. Orthogonal nodes can execute enabled edges independently, and thus concurrently.

Scope

Definition (Scope of edge)

The scope of edge $X \overset{\cdots}{\longrightarrow} Y$ is the most nested OR-node that is an ancestor of both X and Y.

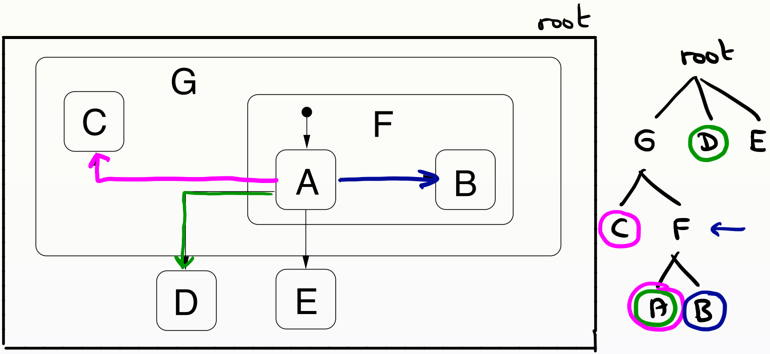
stated differently, A is not left by taking $X \overset{\cdots}{\longrightarrow} Y$.

A

Intuition

The scope of edge $X \overset{\cdots}{\longrightarrow} Y$ is the most nested OR-node that is unaffected by executing the edge $X \overset{\cdots}{\longrightarrow} Y$.

Scope: example



$scope(A \to D) = \text{root}$ and $scope(A \to C) = G$ and $scope(A \to B) = F$

Consistency: formal definition

Definition (Consistency)

1. Edges $ed, ed' \in Edges$ are consistent if:

$$ed = ed' \quad \text{or} \quad scope(ed) \perp scope(ed').$$

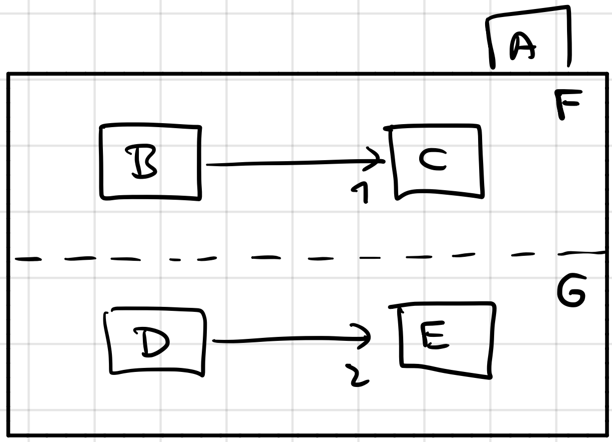
2. $T \subseteq Edges$ is consistent if all edges in T are pairwise consistent. $Cons(T)$ is the set of edges that are consistent with all edges in $T \subseteq Edges$

$$Cons(T) = \{ed \in Edges \mid \forall ed' \in T : ed \text{ is consistent with } ed'\}$$

Example

On the black board.

1



1 & 2 cons, as

$scope(1) = F$

$scope(2) = G$

$F \perp G$

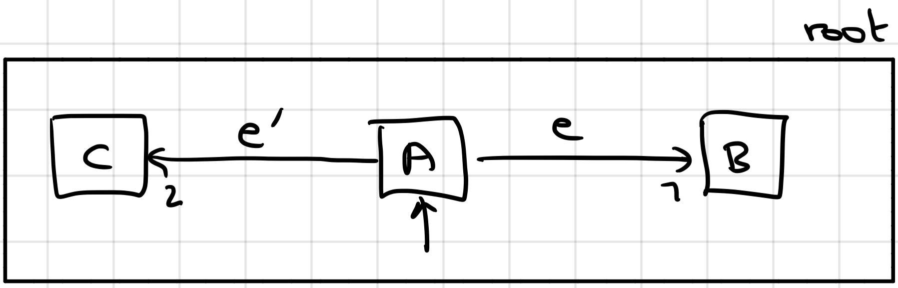
$lca(\{F, G\}) = A$

$type(A) = AND$

So, edges 1 and 2 are

<u>consistent</u>

2



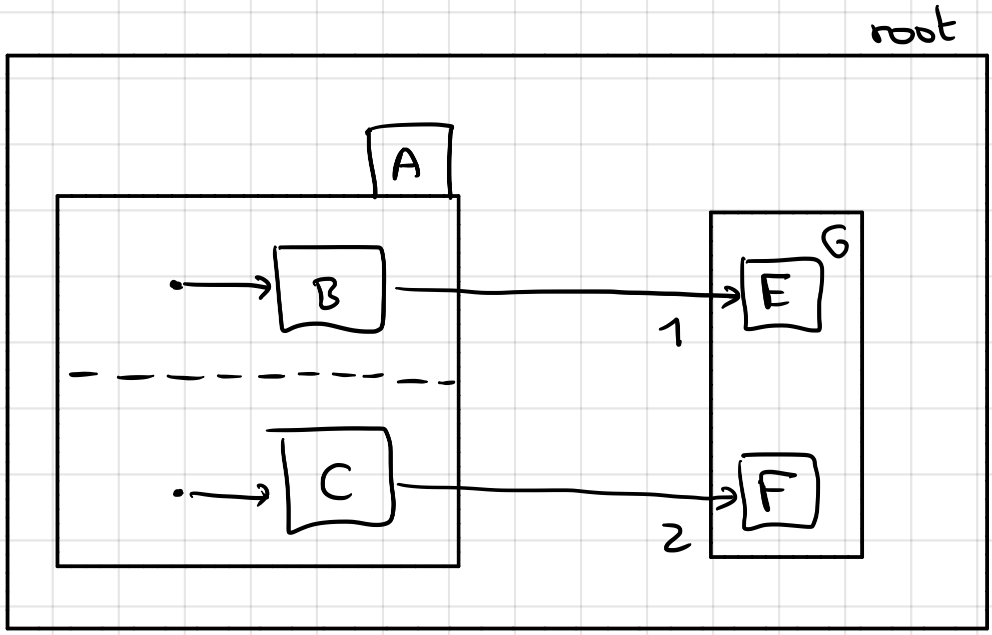
1, 2 is <u>not</u> consistent

$scope(1) = root = scope(2)$

$root \not\trianglelefteq root$

since $root \trianglelefteq root$

3



1, 2 is <u>not</u> consistent

$scope(1) = root$

$= scope(2)$

$root \not\trianglelefteq root$

What is now a macro step?

states $= (C, I, V)$

$$\big\downarrow T$$

(C', I', V')

A macro step is a set T of edges such that:

- all edges in step T are enabled

What is now a macro step?

A macro step is a set T of edges such that:

- all edges in step T are enabled
- all edges in T are pairwise consistent, that is:
  - they are identical or
  - scopes are (descendants of) different children of the same AND-node
- enabled edge ed is not in step T implies

  there exists $ed' \in T$ such that ed is inconsistent with ed', and the priority of ed' is not smaller than ed
- step T is maximal (wrt. set inclusion)

Overview

Priorities

Priorities restrict (but do not abandon) nondeterminism between multiple enabled edges.

Priorities

Priorities restrict (but do not abandon) nondeterminism between multiple enabled edges.

Definition (Priority relation)

The priority relation $\preceq \, \subseteq Edges \times Edges$ is a partial order defined for $ed, ed' \in Edges$ by:

$$\underline{ed \preceq ed'} \quad \text{if} \quad \underline{scope(ed')} \trianglelefteq \underline{scope(ed)}$$

So, ed' has priority over ed if its scope is a descendant of ed's scope.

Priorities

Priorities restrict (but do not abandon) nondeterminism between multiple enabled edges.

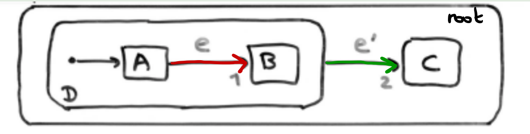
Definition (Priority relation)

The priority relation $\preceq \subseteq Edges \times Edges$ is a partial order defined for $ed, ed' \in Edges$ by:

$$ed \preceq ed' \quad \text{if} \quad scope(ed') \trianglelefteq scope(ed)$$

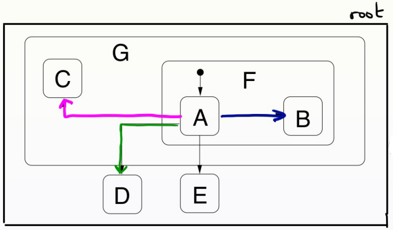
So, ed' has priority over ed if its scope is a descendant of ed's scope.

Example:



$2 \preceq 1$ since $scope(1) = D \trianglelefteq scope(2) = \text{root}$.

Priority: examples



$A \to D \qquad A \to C$

$scope(A \to D) = root$

$scope(A \to C) = G$

$G \trianglelefteq root$, so

$A \to D \preceq A \to C$

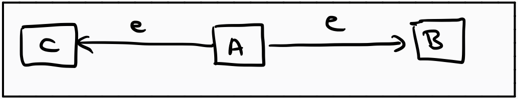
$scope(A \to B) = F$

$F \trianglelefteq G$

$A \to C \preceq A \to B$

Nondeterminism

Priorities rule out some nondeterminism, but not necessarily all.



What is now a macro step?

A macro step is a set T of edges such that:

- all edges in step T are enabled

$$(C, I, V) \xrightarrow{T \subseteq Edges} (C', I', V')$$

What is now a macro step?

A macro step is a set T of edges such that:

- all edges in step T are enabled
- all edges in T are pairwise consistent
  - they are identical or
  - scopes are (descendants of) different children of the same AND-node
- step T is maximal (wrt. set inclusion)
  - T cannot be extended with any enabled, consistent edge
- priorities: enabled edge ed is not in step T implies

  $\exists ed' \in T.\ (\underline{ed \text{ is inconsistent with } ed'} \wedge \underline{\neg(ed' \preceq ed)})$

A macro step — formally

A macro step is a set T of edges such that:

A macro step — formally

A macro step is a set T of edges such that:

- enabledness: $T \subseteq En(C, I, V)$

$$(C, I, V) \overset{T}{\rightsquigarrow} (C', I', V')$$

A macro step — formally

A macro step is a set T of edges such that:

- enabledness: $T \subseteq En(C, I, V)$
- consistency: $T \subseteq Cons(T)$
- maximality: $En(C, I, V) \cap Cons(T) \subseteq T$
- priority: $\forall ed \in En(C, I, V) - T$ we have $(\exists ed' \in T.\ (ed$ is inconsistent with $ed' \wedge \neg(ed' \preceq ed)))$ ← (*)

$T = f(T)$

Note:

The first three points yield: $T = En(C, I, V) \cap Cons(T)$. ← (*)

Computing the set T of macro steps in state (C, I, V)

function $\mathit{nextStep}(C, I, V)$

$T := \varnothing$

while $T \subset \mathit{En}(C, I, V) \cap \mathit{Cons}(T)$

do let $ed \in \mathit{High}\,(\underbrace{(\mathit{En}(C, I, V) \cap \mathit{Cons}(T))} - T)$;

$\quad T := T \cup \{ed\}$

not yet in T

od

return T.

where $\mathit{High}(T) \;=\; \{ed \in T \mid \neg(\exists ed' \in T.\, ed \preceq ed')\}$

Correctness

Theorem:

For any state (C, I, V), $nextStep(C, I, V)$ is a macro step.

Proof.

The proof goes in two steps:

1. We prove enabledness, consistency, and maximality by applying some standard results from fixed point theory, in particular Tarski's-Kleene fixpoint theorem;
2. Then we consider priority and use some monotonicity argument.

$\square$

Step execution

$(C, I, V) \xrightarrow{T} (C', I', V')$

What happens in performing a step?

For a single statechart, executing a step results in performing the actions of all the edges in the step, and changing "control" to the target nodes of these edges.

Interference

Actions in statechart SC_j may influence the sets of events of other statecharts, e.g., SC_i with $i \neq j$ if action *send i.e* is performed by SC_j in a step.

Thus:

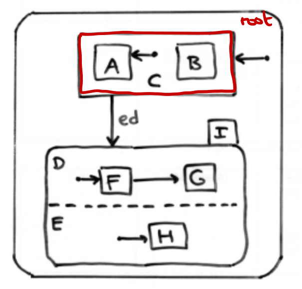
Execution of steps is considered on the system $(SC_1, \ldots, SC_n)$.

Default completion

Definition (Default completion)

The default completion C' of some set C of nodes is the canonical (unique) superset of C such that C' is a configuration. If C' contains an OR-node x and $children(x) \cap C = \varnothing$ implies $default(x) \in C'$.

Example:



AND

1. Default completion of

   $C_1 = \{\text{root}, I\}$ is $C' = C_1 \cup \{D, E, F, H\}$

2. Default completion of

   $C_2 = \{\text{root}, C\}$ is $C' = C_2 \cup \{A\}$.

Step execution by a single statechart

- Let C_j be the current configuration of statechart SC_j
- Let $T_j \subseteq \ Edges_j$ be a step for SC_j
- The next state (C'_j, I'_j, V'_j) of statechart SC_j is given by:
  1. C'_j is the default completion of

$$\bigcup_{X \xrightarrow{e[g]/A} Y \in T_j} \left(Y \ \cup \ \{x \in C_j \mid \forall X \rightarrow Y \in T_j . \underbrace{\neg(x \trianglelefteq scope(X \rightarrow Y))}_{\text{nodes that are unaffected by taking edge } X \rightarrow Y} \} \right)$$

Step execution by a single statechart

- Let C_j be the current configuration of statechart SC_j
- Let $T_j \subseteq \mathit{Edges}_j$ be a step for SC_j
- The next state (C'_j, I'_j, V'_j) of statechart SC_j is given by:
  1. C'_j is the default completion of

$$\bigcup_{X \xrightarrow{e[g]/A} Y \in T_j} Y \;\cup\; \{x \in C_j \mid \forall X \to Y \in T_j.\, \neg(x \trianglelefteq \mathit{scope}(X \to Y))\}$$

  2. $I'_j = \underbrace{\bigcup_{k=1}^{n}}_{\text{all } SC_k\text{'s}} \{e \mid \exists X \xrightarrow{e[g]/A} Y \in T_k.\, \mathit{send}\ j.e \in A\}$

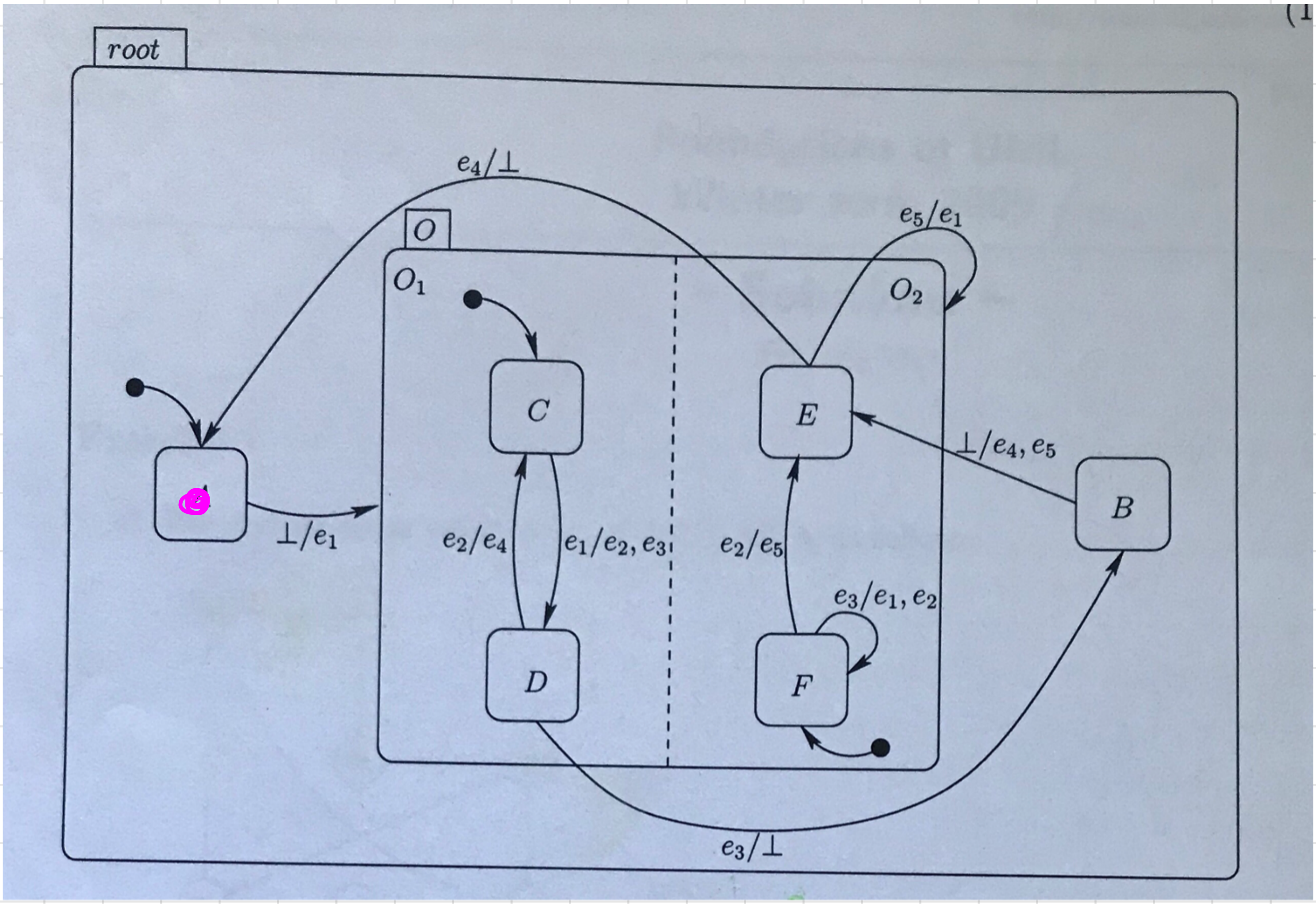
set of events available for the next macro steps

Step execution by a single statechart

- Let C_j be the current configuration of statechart SC_j
- Let $T_j \subseteq \mathit{Edges}_j$ be a step for SC_j
- The next state (C'_j, I'_j, V'_j) of statechart SC_j is given by:
  1. C'_j is the default completion of

$$\bigcup_{X \xrightarrow{e[g]/A} Y \in T_j} Y \;\cup\; \underline{\{x \in C_j \mid \forall X \rightarrow Y \in T_j.\, \neg(x \trianglelefteq \mathit{scope}(X \rightarrow Y))\}}$$

  2. $I'_j = \bigcup_{k=1}^{n} \{e \mid \exists X \xrightarrow{e[g]/A} Y \in T_k.\, \mathit{send}\ j.e \in A\}$
  3. $V'_j(v) = \begin{cases} V_j(v) & \text{if } \forall X \xrightarrow{e[g]/A} Y \in T_j.\, v := \ldots \notin A \\ \underline{\mathit{val}(\text{expr})} & \text{if } \exists X \xrightarrow{e[g]/A} Y \in T_j.\, v := \text{expr} \in A \end{cases}$



Variables are omitted $(C, I, V) \longrightarrow (C, I)$, where C = conf and $I \subseteq E$

default completion of root

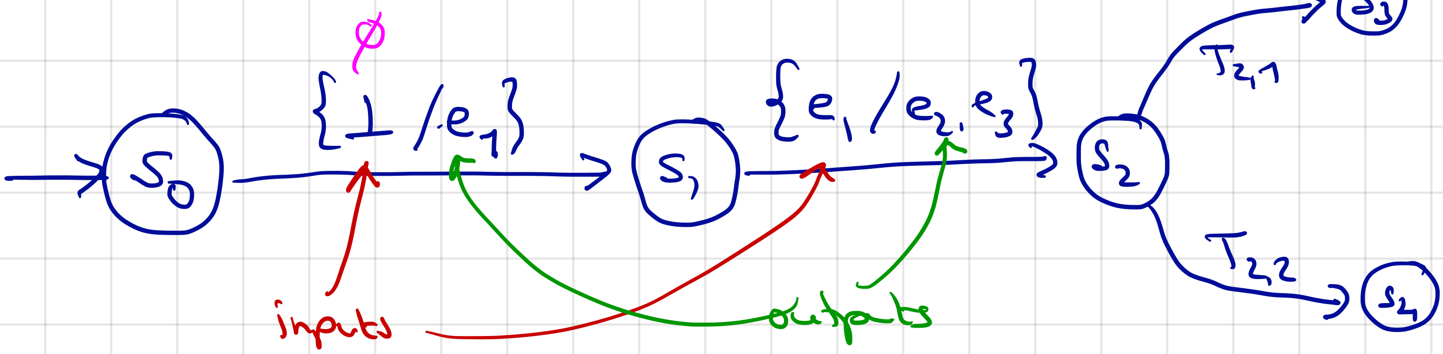
$n = 1$

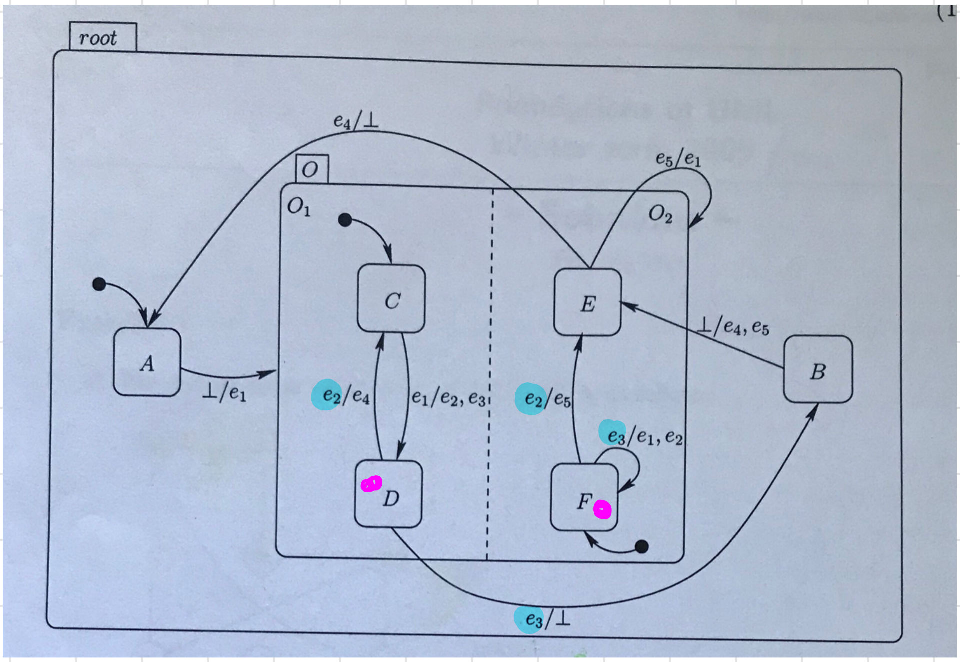
$$s_0 = (\{root, A\}, \emptyset)$$

$$En(s_0) = \{ A \xrightarrow{\perp/e_1} O \} = Step(s_0) = T_0$$

$$s_1 = (\{O, root, O_1, O_2, C, F\}, \{e_1\})$$

$$scope(A \to O) = root$$





$$s_1 = \left(\{ root, O, O_1, O_2, C, F \}, \{ \ \} \right)$$

$$step(s_1) = \{ C \xrightarrow{e_1/e_2, e_3} \underline{D} \}$$

$$s_2 = \left(\{ D, root, O, O_1, O_2, F \}, \{ e_2, e_3 \} \right)$$

$$En(s_2) = \{ \underbrace{D \to C, D \to B}_{\text{inconsistent}}, \underbrace{F \to F, F \to E}_{\text{inconsistent}} \}$$

$$step(s_2) = \{ \{ D \to C, F \to F \}, \{ D \to C, F \to E \} \}$$

the edge $D \to B$ has a lower priority than $D \to C$

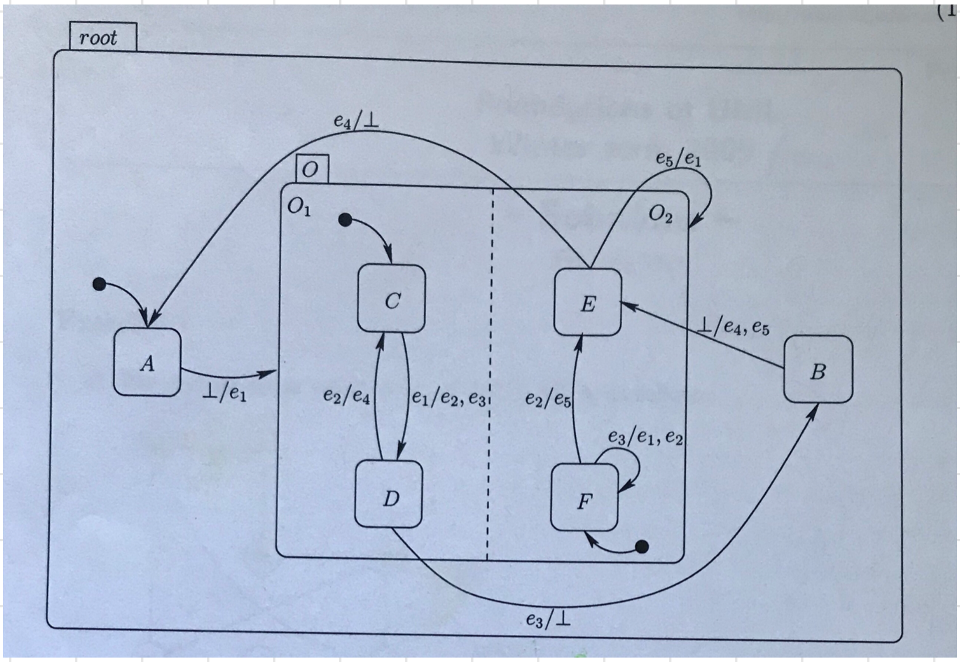
as $scope(D \to B) = root$, $scope(D \to C) = O_1$

$O_1 \trianglelefteq root$, thus $D \to B \preceq D \to C$

so $step(s_2) = \{ T_{2,1}, T_{2,2} \}$

Take $T_{2,1}$ in state s_2 yields the state

$$s_3 = \left(\{ C, F, root, O, O_1, O_2 \}, \{ e_1, e_2, e_4 \} \right)$$

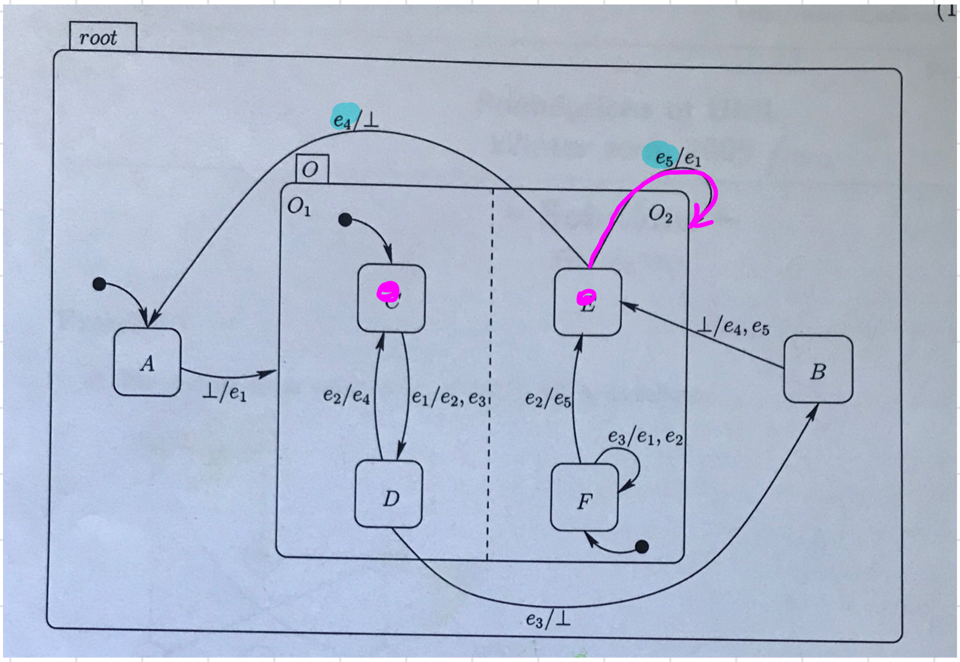


Take $T_{2,2}$ in state s_2:

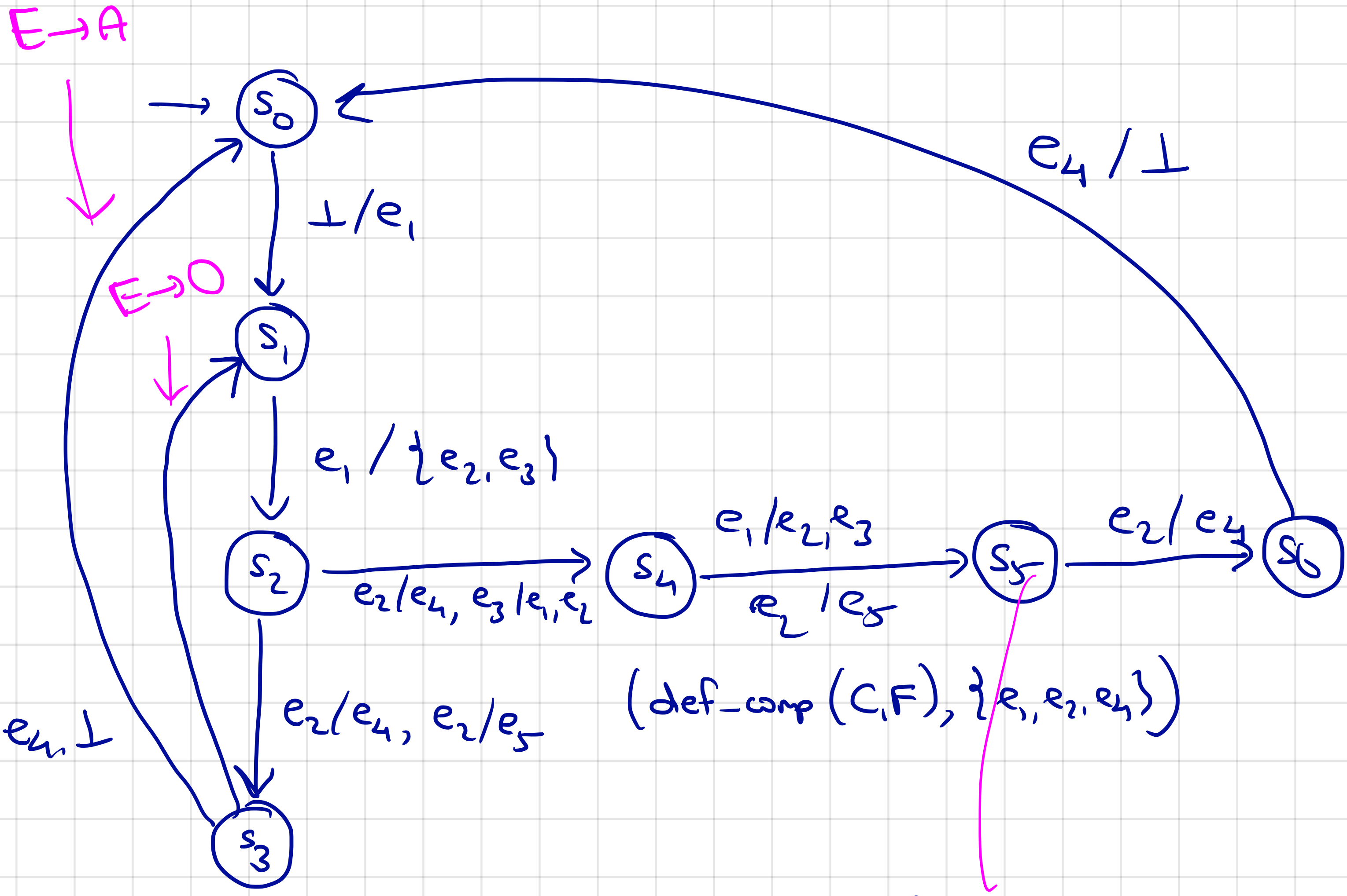
$\hookrightarrow = \{(D \to C, F \to E)\}$

yields the state

$$(\{C, E, root, O, O_1, O_2\}, \{e_4, e_5\}) = S_4$$



Semantics of this statechart is



(def-compl (C,E), $\{e_4, e_5\}$)

(def-comp (D,E), $\{e_1, e_3, e_5\}$)

$s_6 =$ (def-compl (C,E), $\{e_4\}$)

Mealy machines [Mealy, 1953]

Definition (Mealy machine)

A Mealy machine $\mathcal{A} = (Q, q_0, \Sigma, \Gamma, \delta, \omega)$ with:

- Q is a finite set of states with initial state $q_0 \in Q$
- Σ is the input alphabet
- Γ is the output alphabet
- $\delta : Q \times \Sigma \rightarrow Q$ is the deterministic (input) transition function, and
- $\omega : Q \times \Sigma \rightarrow \Gamma$ is the output function

Mealy machines [Mealy, 1953]

Definition (Mealy machine)

A Mealy machine $\mathcal{A} = (Q, q_0, \Sigma, \Gamma, \delta, \omega)$ with:

- Q is a finite set of states with initial state $q_0 \in Q$
- Σ is the input alphabet
- Γ is the output alphabet
- $\delta : Q \times \Sigma \to Q$ is the deterministic (input) transition function, and
- $\omega : Q \times \Sigma \to \Gamma$ is the output function

Intuition

A Mealy machine (or: finite-state transducer) is a finite-state automaton that produces output on a transition, based on current input and state.

Mealy machines [Mealy, 1953]

Definition (Mealy machine)

A Mealy machine $\mathcal{A} = (Q, q_0, \Sigma, \Gamma, \delta, \omega)$ with:

- Q is a finite set of states with initial state $q_0 \in Q$
- Σ is the input alphabet
- Γ is the output alphabet
- $\delta : Q \times \Sigma \to Q$ is the deterministic (input) transition function, and
- $\omega : Q \times \Sigma \to \Gamma$ is the output function

Intuition

A Mealy machine (or: finite-state transducer) is a finite-state automaton that produces output on a transition, based on current input and state.

Moore machines

In a Moore machine $\omega : Q \to \Gamma$, output is purely state-based.

From statecharts to a Mealy machine (1)

$n=1$ $\quad q = (C, I, V)$

States

A state q is a tuple of the (local) states of SC_1 through SC_n.

Input and output events

Any input is a set of events, and any output is a set of events.

Next-state function δ

Defines the effect of executing a step.

Output function ω

Defines all events sent to some SC outside the system $(SC_1, \ldots, SC_n)$.

From statecharts to a Mealy machine (2)

States

A state q is a tuple of the (local) states of SC_1 through SC_k.

Formally:

- $Q = \prod_{k=1}^{n} (Conf_k \times 2^{E_k} \times Val_k)$ is the set of states
  - where $Conf_k$ is the set of configurations of SC_k,
  - E_k is the set of the events of SC_k,
  - and Val_k is the set of variable valuations of SC_k

From statecharts to a Mealy machine (2)

States

A state q is a tuple of the (local) states of SC_1 through SC_k.

Formally:

- $Q = \prod_{k=1}^{n} (\mathit{Conf}_k \times 2^{E_k} \times \mathit{Val}_k)$ is the set of states
  - where Conf_k is the set of configurations of SC_k,
  - E_k is the set of the events of SC_k,
  - and Val_k is the set of variable valuations of SC_k
- $q_0 = \prod_{k=1}^{n} (C_{0,k}, \varnothing, \mathit{Val}_{0,k})$ is the initial state
  - where $C_{0,k}$ is the default completion of the set $\{\text{root}\}$
  - the initial set of events is empty
  - $\mathit{Val}_{0,k}$ is the initial variable valuation of SC_k

From statecharts to a Mealy machine (3)

Input and output events

Any input is a set of events, and any output is a set of events.

Formally,

- Input alphabet: $\Sigma = 2^E - \{ \varnothing \}$
  - where $E = \bigcup_{k=1}^{n} E_k$ is the set of events in all statecharts
- Output alphabet: $\Gamma = 2^{E'}$
  - with $E' = \underbrace{\left\{ send\ j.e \in \bigcup\limits_{k=1}^{n} SC_k \mid j \notin \{1, \ldots, n\} \right\}}_{\text{all outputs that cannot be consumed}}$

From statecharts to a Mealy machine (4)

Next-state function δ

Defines the effect of executing a step.

Formally,

- $(s'_1, \ldots, s'_n) \in \delta((s_1, \ldots, s_n), E)$ where
  - $s''_i = (C'_i, I''_i, V'_i)$ is the next state after executing some $T_i = \mathit{nextStep}(C_i, I_i, V_i)$
  - and $s'_i = (C'_i, I''_i \cup (E \cap E_i), V'_i)$

From statecharts to a Mealy machine (5)

Output function ω

Defines all events sent to some SC outside the system $(SC_1, \ldots, SC_n)$.

Formally,

- $\omega((s_1, \ldots, s_n), E) =$
$$\left\{ \text{send } j.e \mid j \notin \{1, \ldots, n\} \wedge \exists i. \exists X \xrightarrow{e[g]/\text{send } j.e} Y \in nextStep(C_i, I_i, V_i) \right\}$$