Outline

1. Intuition and Assumptions
2. States and Configurations
3. Enabledness
4. Consistency
5. Priority
I = root node
N = set of nodes = \{A, B, C, ..., I, J\}

children: N → 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

BASIC nodes

ancestor relation
\( \Delta \subseteq N \times N \)

\[ \text{default } (C) = F \text{ \&\& } F \Rightarrow G \text{ \&\& } C = G \]
Edges:

- $X \xrightarrow{e[g]} Y$  \hspace{1cm} \text{hyperedge}
- $\{A\} \xrightarrow{e[x>0]/\phi} \{B\}$
- $\{G\} \xrightarrow{e[x=2]/\psi} \{H\}$
Overview

1. Intuition and Assumptions
2. States and Configurations
3. Enabledness
4. Consistency
5. Priority
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

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

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

- 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 \(n+2, n+3, \ldots\)
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 $i+1$
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?

Intuitive semantics as a transition system:

- **State** = a set of nodes ("current control") + the values of variables in the state chart
currently "active" statechart nodes

(several statecharts) \(\rightarrow\) single Mealy machine

\((sc_1, \ldots, sc_k)\)
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

\[ g = x > 0 \]

- Needs to hold in current state

- Needs to be available

\[ e[g]/A \]

- Enabled
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 $Y$
  ⇒ events are unordered, and considered as a set

- **Principle**: execute as many edges at once (without conflict)
  ⇒ the total execution of such maximal set is a *macro step*
Overview

1. Intuition and Assumptions
2. States and Configurations
3. Enabledness
4. Consistency
5. Priority
A configuration of $SC = (N, E, Edges)$ is a set $C \subseteq N$ of nodes satisfying:

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

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

Joost-Pieter Katoen Theoretical Foundations of the UML

8/37
Example configurations

\[ C_1 = \{ I, H \} \]
\[ C_2 = \{ I, G, E, F, A, D \} \]
\[ C_3 = \{ I, G, E, F, B, C \} \text{ etc.} \]

States:\n\[ (C_1, \emptyset, \{ x=0, y=0 \}) \]
\[ (C_2, \{ e, e' \}, \{ x=1, y=0 \}) \]
\[ \text{etc.} \]

Enabledness:\n\[ \{ A \} \xrightarrow{e \text{ [true] } \emptyset} \{ B \} \]
\[ \text{is enabled in } (C_2, \{ e \}, V) = \]
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 $\text{type}(x) = \text{OR}$ implies $|\text{children}(x) \cap C| = 1$
- $x \in C$ and $\text{type}(x) = \text{AND}$ implies $\text{children}(x) \subseteq C$

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

**Definition (State)**

State of $SC = (N, E, Edges)$ is a triple $(C, I, V)$ 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. $x=3$, $y=77$, $c=\theta$
how to define the transition relation between states?
Enabling of an edge

Definition (Enabledness)

Edge $X \xrightarrow{e \mid g} Y$ is enabled in state $(C, I, V)$ whenever:

- $X \subseteq C$, i.e. all source nodes are in configuration $C$
- $(C_1, \ldots, C_n), (V_1, \ldots, V_n) \models g$, i.e., guard $g$ is satisfied in all statecharts
- either $e \neq \bot$ implies $e \in I$, or $e = \bot$

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

$g$ is a global Boolean statement, covering the entire statecharts

$X > 0 \land y < 10$

$S_1, \ldots, S_n$
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

1. Intuition and Assumptions
2. States and Configurations
3. Enabledness
4. Consistency
5. Priority
To define consistency formally, we need some auxiliary concepts.
pair of edges $\{1,2\}$

intuition: $\{1,2\}$ are consistent as they can happen in parallel

in state $= (\{A,F,G,B,D\}, \{e,e',\ldots\}, V)$
both edges $1,2$ are enabled, and as they are consistent, they can both be taken.

$(1,2)$ is not consistent since from A only one edge can be taken

in state $= (\{A,\ldots\}, \{e,e',\ldots\}, V)$
both $1,2$ are enabled, but any one of them can be taken
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:

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

$\text{lca}\{A, c\} = G$

$\text{type}\ (G) = \text{AND}$

thus $A \perp C$

$F \perp E$

Basic $\rightarrow \ A$

$X = \{c, D\}$

$\text{lca}(X) = E$

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

$\text{lca}(X') = G$

$\lnot (c \perp D)$
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:

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

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

$$
\neg (x \sqsubseteq y) \quad \text{and} \quad \neg (y \sqsubseteq x) \quad \text{and} \quad \text{type}(\text{lca}\{x, y\}) = \text{AND}.
$$
Orthogonality of nodes

**Definition (Orthogonality of nodes)**

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

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

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 $X \rightarrow Y$ is the most nested OR-node that is an ancestor of both $X$ and $Y$.

Intuition

The scope of edge $X \rightarrow Y$ is the most nested OR-node that is **unaffected** by executing the edge $X \rightarrow Y$.

*stated differently, A is not left by taking $X \rightarrow Y$. 

A
Scope: example

\[ \text{scope}(A \rightarrow D) = \text{root} \quad \text{and} \quad \text{scope}(A \rightarrow C) = G \quad \text{and} \quad \text{scope}(A \rightarrow B) = F \]
Definition (Consistency)

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

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

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

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

Example

On the black board.
1.8.2 con\textsubscript{>}, as
\[ \text{scope}(1) = F \quad \text{scope}(2) = G \]
\[ F \perp G \]
\[ \text{lca}(\{F,G\}) = A \]
\[ \text{type}(A) = \text{AND} \]

So, edges 1 and 2 are consistent.

2. 1, 2 is not consistent
\[ \text{scope}(1) = \text{root} = \text{scope}(2) \]
\[ \text{root} \neq \text{root} \]
since \text{root} \leq \text{root}

3. 1, 2 is not consistent
\[ \text{scope}(1) = \text{root} = \text{scope}(2) \]
\[ \text{root} \neq \text{root} \]
What is now a macro step?

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

- all edges in step $T$ are enabled

$$\text{states} = (c, i, v)$$

$$\downarrow T$$

$$(c', i', v')$$
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, 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

1. Intuition and Assumptions
2. States and Configurations
3. Enabledness
4. Consistency
5. Priority
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 $\preceq \subseteq \text{Edges} \times \text{Edges}$ is a partial order defined for $ed, ed' \in \text{Edges}$ by:

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

So, $ed'$ has priority over $ed$ if its scope is a descendant of $ed$’s scope.
Priorities restrict (but do not abandon) nondeterminism between multiple enabled edges.

**Definition (Priority relation)**

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

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

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

**Example:**

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

\[
\begin{align*}
A & \rightarrow D & A & \rightarrow C \\
\text{scope}(A \rightarrow D) &= \text{root} \\
\text{scope}(A \rightarrow C) &= G \\
G & \leq \text{root}, \text{ so} \\
A \rightarrow D & \leq A \rightarrow C \\
\text{scope}(A \rightarrow B) &= F \\
F & \leq G \\
A \rightarrow C & \leq A \rightarrow B
\end{align*}
\]
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**

\[
(C, \Sigma, \nu) \xrightarrow{T \subseteq \text{Edge}} (C', \Sigma', \nu')
\]
A \textbf{macro step} is a set $T$ of edges such that:

- all edges in step $T$ are \textbf{enabled}

- all edges in $T$ are \textbf{pairwise consistent}
  - they are identical or
  - scopes are (descendants of) different children of the same \textsc{AND-node}

- step $T$ is \textbf{maximal} (wrt. set inclusion)
  - $T$ cannot be extended with any enabled, consistent edge

- \textbf{priorities}: enabled edge $ed$ is not in step $T$ implies
  $$\exists ed' \in T. \ (ed \ \text{is inconsistent with} \ ed' \ \land \neg (ed' \preceq ed))$$
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:** $T \subseteq En(C, I, V)$
A macro step — formally

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

\[
\begin{align*}
\text{enabledness: } & T \subseteq \text{En}(C, I, V) \\
\text{consistency: } & T \subseteq \text{Cons}(T) \\
\text{maximality: } & \text{En}(C, I, V) \cap \text{Cons}(T) \subseteq T \\
\text{priority: } & \forall ed \in \text{En}(C, I, V) - T \text{ we have } \\
& (\exists ed' \in T. (ed \text{ is inconsistent with } ed' \land \neg (ed' \preceq ed)))
\end{align*}
\]

Note:

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

\[ T = f(T) \]

\[ (*) \]
Computing the set $T$ of macro steps in state $(C, I, V)$

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

$T := \emptyset$

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

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

od

return $T$.

where $\text{High}(T) = \{ed \in T \mid \neg(\exists ed' \in T. ed \preceq ed')\}$
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.
Step execution

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 superset of $C$ such that $C'$ is a configuration. If $C'$ contains an OR-node $x$ and $\text{children}(x) \cap C = \emptyset$ implies $\text{default}(x) \in C'$.

Example:

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

$$\bigcup \left( Y \cup \{ x \in C_j \mid \forall X \rightarrow Y \in T_j. \neg (x \leq \text{scope}(X \rightarrow Y)) \} \right)$$

- nodes that are unaffected by taking edge $X \rightarrow Y$
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

   \[
   \bigcup_{X \xrightarrow{e[g]/A} Y \in T_j} Y \cup \{x \in C_j \mid \forall X \rightarrow Y \in T_j. \neg (x \leq \text{scope}(X \rightarrow Y))\}
   \]

2. \( I'_j = \bigcup_{k=1}^{n} \{e \mid \exists X \xrightarrow{e[g]/A} Y \in T_k. \text{send } j.e \in A\} \)

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 $T_j \subseteq Edges_j$ 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

     \[
     \bigcup_{X \xrightarrow{e[g]/A} Y \in T_j} Y \cup \{ x \in C_j \mid \forall X \rightarrow Y \in T_j. \neg(x \leq \text{scope}(X \rightarrow Y)) \}\]

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

  3. $V'_j(v) = \begin{cases} 
  V_j(v) & \text{if } \forall X \xrightarrow{e[g]/A} Y \in T_j. v := \ldots \in A \\
  \text{val(expr)} & \text{if } \exists X \xrightarrow{e[g]/A} Y \in T_j. v := \text{expr} \in A 
  \end{cases}$
Variables are omitted

\[ (C, I, V) \rightarrow (C, I) \]

\[ n = 1 \]

\[ \text{conf} \in E \]

\[ s_0 = (\{ \text{root}, A \}, \emptyset) \]

\[ E_n(s_0) = \{ A \xrightarrow{\perp/e_1} O \} = \text{step}(s_0) = T_0 \]

\[ s_1 = (\{ O, \text{root}, O_1, O_2, C, F \}, \{ e_1 \}) \]

\[ \text{scope}(A \rightarrow O) = \text{root} \]

\[ \text{scope}(e_1) = \emptyset \]

\[ \text{scope}(e_2, e_3) = \text{root} \]

\[ \text{inputs} \]

\[ \text{outputs} \]
$S_1 = \left( \left\{ \text{root, } O, O_1, O_2, C, F \right\}, \{ e_1, e_2, e_3 \} \right)$

$\text{step } (S_1) = \left\{ C \xrightarrow{e_1/e_2/e_3} D \right\}$

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

$E_n(s_2) = \left\{ D \rightarrow C, D \rightarrow B, F \rightarrow F, F \rightarrow E \right\}$

Inconsistent
Inconsistent

$\text{step } (s_2) = \left\{ \left\{ D \rightarrow C, F \rightarrow F \right\}, \left\{ D \rightarrow C, F \rightarrow E \right\} \right\}$

The edge $D \rightarrow B$ has a lower priority than $D \rightarrow C$ as $\text{scope } (D \rightarrow B) = \text{root}$, $\text{scope } (D \rightarrow C) = O_1$

$O_1 \leq \text{root}$, thus $D \rightarrow B \leq D \rightarrow C$

So $\text{step } (s_2) = \left\{ T_{2,1}, T_{2,2} \right\}$

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

$s_3 = \left( \left\{ C, F, \text{root, } O, O_1, O_2 \right\}, \{ e_1, e_2, e_4 \} \right)$
Take $T_{2,2}$ in state $s_2$:

$$\downarrow = \{ (D \rightarrow C, F \rightarrow E) \}$$

yields the state

$$( \{ C, E, root, O, O_1, O_2 \}, \{ e_4, e_5 \} ) = s_4$$
Semantics of this statechart is
Definition (Mealy machine)

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

- $Q$ is a finite set of states with initial state $q_0 \in Q$
- $\Sigma$ is the input alphabet
- $\Gamma$ 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 $q_0 \in Q$
- $\Sigma$ is the input alphabet
- $\Gamma$ 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 \( q_0 \in Q \)
- \( \Sigma \) is the input alphabet
- \( \Gamma \) 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

**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 \( \omega : Q \rightarrow \Gamma \), output is purely state-based.
From statecharts to a Mealy machine (1)

<table>
<thead>
<tr>
<th>States</th>
</tr>
</thead>
<tbody>
<tr>
<td>A state $q$ is a tuple of the (local) states of $SC_1$ through $SC_n$.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Input and output events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Any input is a set of events, and any output is a set of events.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Next-state function $\delta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defines the effect of executing a step.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output function $\omega$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defines all events sent to some $SC$ outside the system ($SC_1, \ldots, SC_n$).</td>
</tr>
</tbody>
</table>
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 $SC_k$
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 $SC_k$

- $q_0 = \prod_{k=1}^{n} (C_{0,k}, \emptyset, Val_{0,k})$ is the initial state
- where $C_{0,k}$ is the default completion of the set $\{\text{root}\}$
- the initial set of events is empty
- $Val_{0,k}$ is the initial variable valuation of $SC_k$
Input and output events

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

Formally,

- **Input alphabet:** $\Sigma = 2^E - \{ \emptyset \}$
  - where $E = \bigcup_{k=1}^{n} E_k$ is the set of events in all statecharts

- **Output alphabet:** $\Gamma = 2^{E'}$
  - with $E' = \left\{ \text{send } j.e \in \bigcup_{k=1}^{n} SC_k \mid j \notin \{1, \ldots, n\} \right\}$
  - all outputs that cannot be consumed
Next-state function $\delta$

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 $T_i = \text{nextStep}(C_i, I_i, V_i)$
  - and $s'_i = (C'_i, I''_i \cup (E \cap E_i), V''_i)$
Output function $\omega$

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

Formally,

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