Components of I/O automaton: (A)

a) sig(A)
b) states(A)
c) start(A)
d) trans(A). \( \Pi \)  
(e) tasks(A)

\[ (s, \Pi, s') \]

\( \Pi \) is said to be enabled in a state s if \( \Pi \) a transition to s'.

Page 1 of 9
Task C is said to be enabled in a state $s$ if some action of C is enabled in $s$.

Execution of an I/O Automaton

Finite sequence of alternating states and actions

$s_0, \pi_1, s_1, \pi_2, s_2, \ldots \pi_n, s_n \rightarrow \text{exec}(A)$

$(s_k, \pi_{k+1}, s_{k+1})$ as a transition

Or it could be an infinite sequence of the above kind.
\[ \Rightarrow \alpha \cdot \alpha' \]

\[ \alpha' \cdot \alpha' \rightarrow \text{an execution} \]

\[ \Rightarrow \text{Trace of an execution } \alpha \text{ of } A \]

as the subsequence of \( \alpha \)

consisting of all external actions

set of \( \text{traces}(A) \rightarrow \text{traces}(A) \)
\textbf{ex.1:} Message alphabet \( M \{1, 2\} \)

\[ \lambda \rightarrow \text{empty queue.} \]

An execution would be

\[ \begin{aligned}
\lambda, & \quad \text{send}(1)_{i,j}, [1], \quad \text{receive}(1)_{i,j}, [\lambda], \\
& \quad \text{send}(2)_{i,j}, [2], \quad \text{receive}(2)_{i,j}, [\lambda] \\
\rightarrow & \quad \lambda, \quad \text{send}(1)_{i,j}, [1], \quad \text{receive}(1)_{i,j}, [\lambda], \quad \text{send}(2)_{i,j}, [\lambda], \\
\rightarrow & \quad \lambda, \quad \text{send}(1)_{i,j}, [1], \quad \text{send}(1)_{i,j}, [1, 2], \quad \text{send}(1)_{i,j}. \\
\end{aligned} \]
Operations on an I/O automaton.

Composition is an action $\Pi$

Restrictions. A countable collection $\{S_i\}_{i \in I}$ of sign.

are said to be compatible if $\forall i, j \in I$ $(i \neq j)$.

all of the following hold

1) $\text{int}(S_i) \cap \text{acts}(S_j) = \emptyset$

2) $\text{out}(S_i) \cap \text{out}(S_j) = \emptyset$

3) no action is contained in infinitely many sets $\text{acts}(S_i)$. 

Composition of $S = \prod_{i \in I} S_i$ of a countable collection of compatible signatures $\{S_i\}_{i \in I}$

1) $\text{out}(S) = \bigcup_{i \in I} \text{out}(S_i)$

2) $\text{init}(S) = \bigcup_{i \in I} \text{init}(S_i)$

3) $\text{in}(S) = \bigcup_{i \in I} \text{in}(S_i) - \bigcup_{i \in I} \text{out}(S_i)$

Composition of the Automaton

- $\text{sig}(A) = \prod_{i \in I} \text{sig}(A_i)$ (as defined above)
- $\text{States}(A) = \prod_{i \in I} \text{States}(A_i)$ (Cartesian product space)
- $\text{Start}(A) = \prod_{i \in I} \text{Start}(A_i)$
\text{trans}(A) = \text{set of } (s_i, \Pi, s_i') \text{ such that }
\forall i \in \mathbb{E}, \text{ if } \Pi \in \text{acts}(A_i) \text{ then }
(s_i, \Pi, s_i') \in \text{trans}(A) \text{ otherwise }

s_i = s_i'.

\text{tasks}(A) = \bigcup_{i \in \mathbb{I}} \text{tasks}(A_i).

A \times B \times C \quad \text{A has } \Pi \text{ as an o/p action}
\& \ B, C \text{ have } \Pi \text{ as i/p actions.
Transitions

1) init(V)_i
2) send(V)_i;j
3) receive(V)_i;j
4) decide(V)_i;j

P1: [2, num] → [2, 1]
   decide: +
   → 2, 0

P2: [num, 1] → [2, 1]

\( n = 2 \)

init(?)_1, init(1)_2, send(2)_1, 2
receive(2)_1, 2, send(1)_2, 1
receive(1)_2, 1, init(2)_2, 1
   decide(5)_1, decide(2)_2
\[ \text{Defn:} \]
\[ \alpha | A_i \Rightarrow \text{given an exec. } \alpha = s_0, t_i, s_i, \ldots \]
\[ \Rightarrow \text{is the sequence got by deleting each } \]
\[ \text{pair } T_0, S_r \text{ s.t. } T_r \notin \text{acts}(A_i). \]
\[ \beta | A_i \]

**Th 1**