

## Components of I/O automaton: (A)

a) sig(A)

b) states(A)

c) start(A)

d) trans(A).

$\pi$   
(s,  $\pi$ , s')

e) tasks(A)

→  $\pi$  is said to  
be enabled in a state 's'  
if  $\exists$  a transition to s'  
(s,  $\pi$ , s')

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 & actions

$s_0, \pi_1, s_1, \pi_2, s_2, \dots, \pi_r, s_r \rightarrow \text{execs}(A)$

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

or it could be an infinite sequence of the above kind

→  $\alpha \cdot \alpha'$

$\alpha \cdot \alpha' \rightarrow$  an execution

→ Trace of an execution  $\alpha$  of  $A$   
 as the subsequence of  $\alpha$   
 consisting of all external actions

set of traces  $(A) \rightarrow$  traces $(A)$

ex!: Message alph  $M \{1, 2\}$   
 $\lambda \rightarrow$  empty queue.

An execution would be

$[\lambda], \text{send}(1)_{i,j}, [1], \text{receive}(1)_{i,j}, [\lambda]$   
 $\text{send}(2)_{i,j}, [2], \text{receive}(2)_{i,j}, [\lambda]$

$\rightarrow [\lambda], \text{send}(1)_{i,j}, [1], \text{receive}(1)_{i,j}, [\lambda], \text{send}(2)_{i,j}, [2]$   
 $\rightarrow [\lambda], \text{send}(1)_{i,j}, [1], \text{send}(1)_{i,j}, [1,1], \text{send}(1)_{i,j}, [1,1,1].$

## Operations on an I/O automaton.

Composition  $\div$  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  $\{S_i\}_{i \in I}$

$$1) \text{ out}(s) = \bigcup_{i \in I} \text{out}(s_i)$$

$$2) \text{ int}(s) = \bigcup_{i \in I} \text{int}(s_i)$$

$$3) \text{ in}(s) = \bigcup_{i \in I} \text{in}(s_i) - \bigcup_{i \in I} \text{out}(s_i)$$

Composition of the Automaton

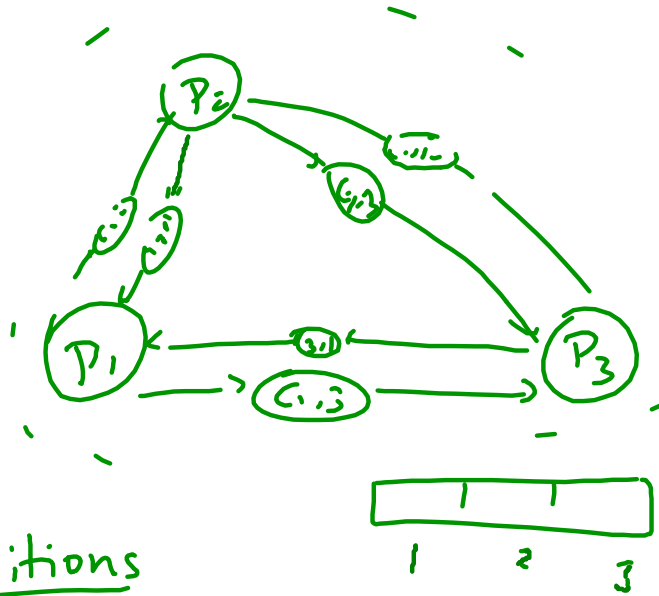
- $\text{sig}(A) = \prod_{i \in I} \text{sig}(A_i)$  (as defined above)
- $\text{States}(A) = \prod_{i \in I} \text{States}(A_i)$  (Cartesian product space)
- $\text{Start}(A) = \prod_{i \in I} \text{start}(A_i)$

$\text{trans}(A) \rightarrow$  set of triples  $(s_i, \pi, s_i')$  such that  
 $\forall i \in I$ . if  $\pi \in \text{acts}(A_i)$  then  
 $(s_i, \pi, s_i') \in \text{trans}(A)$  otherwise  
 $s_i = s_i'$ .

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

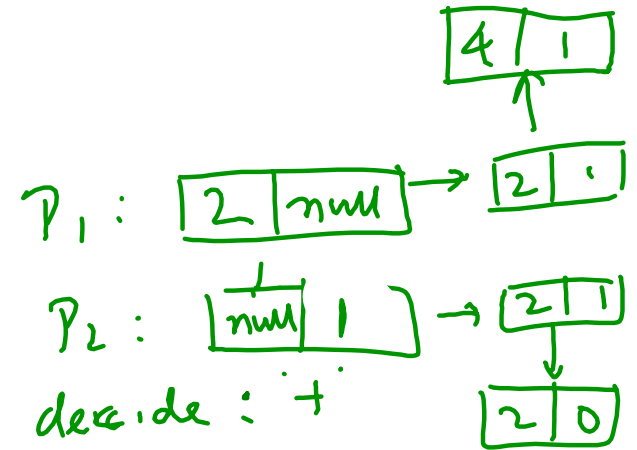
$A \times B \times C$       $A$  has  $\pi$  as an o/p action  
 &  $B, C$  have  $\pi$  as i/p actions.

(etc)



Transitions

- 1)  $init(v)_i$
- 2)  $send(v)_{i,j}$
- 3)  $receive(v)_{i,j}$
- 4)  $decide(v)_{i,j}$



n=2

$init(2)_1, init(1)_2, send(2)_{1,2}$   
 $receive(2)_{1,2}, send(1)_{2,1}$   
 $receive(1)_{2,1}, init(2)_1, init(0)_2$   
 $decide(5)_1, decide(2)_2$

Def'n :

$\alpha | A_i \rightarrow$  given an exec.  $\alpha = S_0, T_1, S_1, \dots$

$\searrow$  is the sequence got by deleting each pair  $\pi_r, S_r$  s.t.  $\pi_r \notin \text{acts}(A_i)$ .

$\beta | A_i$

Th 1