

# Topics in Concurrency

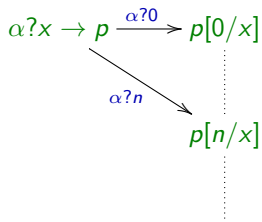
## Lecture 3

Jonathan Hayman

20 January 2013

# Towards a more basic language

- Aim: removal of variables to reveal symmetry of input and output
- Transitions for value-passing carry labels  $\tau$ ,  $a?n$ ,  $a!n$



- This suggests introducing **prefix**  $\alpha?n.p$  (as well as  $\alpha!n.p$ ) and view  $\alpha?x \rightarrow p$  as a sum  $\sum_n \alpha?n.p[n/x]$   $\nwarrow$  **infinite sum**
- View  $\alpha?n$  and  $\alpha!n$  as **complementary** actions
- Synchronization can only occur on complementary actions

# Pure CCS

- Actions:  $a, b, c, \dots$
- Complementary actions:  $\bar{a}, \bar{b}, \bar{c}, \dots$
- Internal action:  $\tau$
- Notational convention:  $\bar{\bar{a}} = a$
- Processes:

$p ::= \lambda.p$	prefix	$\lambda$ ranges over $\tau, a, \bar{a}$ for any action $a$
$\sum_{i \in I} p_i$	sum	$I$ is an indexing set
$p_0 \parallel p_1$	parallel	
$p \setminus L$	restriction	$L$ a set of actions
$p[f]$	relabelling	$f$ a function on actions
$P$	process identifier	

- Process definitions:

$$P \stackrel{\text{def}}{=} p$$

# Transition rules for pure CCS

- Nil process no rules
- Guarded processes

$$\lambda.p \xrightarrow{\lambda} p$$

- Sum

$$\frac{p_j \xrightarrow{\lambda} p' \quad j \in I}{\sum_{i \in I} p_i \xrightarrow{\lambda} p'_0}$$

- Parallel composition

$$\frac{p_0 \xrightarrow{\lambda} p'_0}{p_0 \parallel p_1 \xrightarrow{\lambda} p'_0 \parallel p_1} \quad \frac{p_1 \xrightarrow{\lambda} p'_1}{p_0 \parallel p_1 \xrightarrow{\lambda} p_0 \parallel p'_1}$$

$$\frac{p_0 \xrightarrow{a} p'_0 \quad p_1 \xrightarrow{\bar{a}} p'_1}{p_0 \parallel p_1 \xrightarrow{\tau} p'_0 \parallel p'_1}$$

- **Restriction**

$$\frac{p \xrightarrow{\lambda} p' \quad \lambda \notin L \cup \bar{L}}{p \setminus L \xrightarrow{\lambda} p' \setminus L} \quad \text{where } \bar{L} = \{\bar{a} \mid a \in L\}$$

- **Relabelling**

$$\frac{p \xrightarrow{\lambda} p'}{p[f] \xrightarrow{f(\lambda)} p'[f]}$$

where  $f$  is a function such that  $f(\tau) = \tau$  and  $f(\bar{a}) = \overline{f(a)}$

- **Identifiers**

$$\frac{p \xrightarrow{\lambda} p' \quad P \stackrel{\text{def}}{=} p}{P \xrightarrow{\lambda} p'}$$

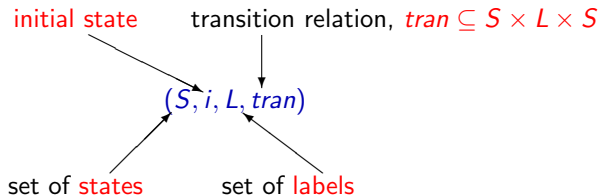
# Transition systems

- Given a CCS process  $p$ , can construct its *transition system*
- A *transition system* is:

$(S, i, L, tran)$

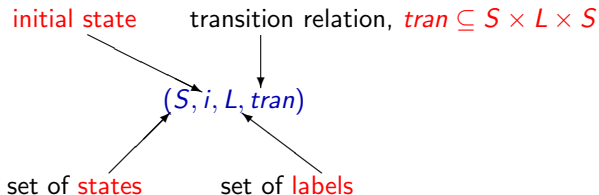
# Transition systems

- Given a CCS process  $p$ , can construct its *transition system*
- A *transition system* is:

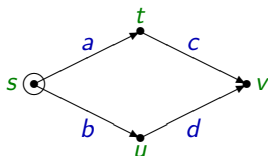


# Transition systems

- Given a CCS process  $p$ , can construct its *transition system*
- A *transition system* is:



- Graphically:



$$\begin{aligned} S &= \{s, t, u, v\} \\ i &= s \\ L &= \{a, b, c, d\} \\ tran &= \{ (s, a, t), \\ &\quad (s, b, u), \\ &\quad (t, c, v), \\ &\quad (u, d, v) \} \end{aligned}$$



# Transition systems from CCS

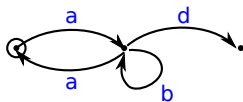
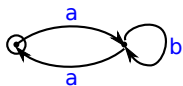
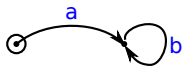
- The transition system for a CCS process  $p$  has reachable terms as states and transitions as generated by the operational semantics
- Example:  $a.\bar{b}.\mathbf{nil} + \bar{b}.a.\mathbf{nil}$
- Example:  $a \parallel b$  (note: dropped  $\mathbf{nil}$ )
- Example:  $a \parallel \bar{a}$
- Example:  $(a \parallel \bar{a}) \setminus a$
- Example:  $(a \parallel \bar{b})[f]$  where  $f(a) = w$  and  $f(b) = w$
- Example:  $a[f] \parallel \bar{b}[f]$  where  $f(a) = w$  and  $f(b) = w$
- Example:  $P$  where

$$P \stackrel{\text{def}}{=} a.Q$$

$$Q \stackrel{\text{def}}{=} b.Q + a.P$$

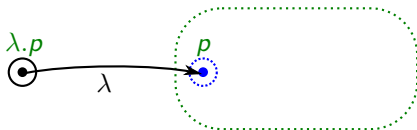
# Realising transition systems

Give pure CCS terms for:



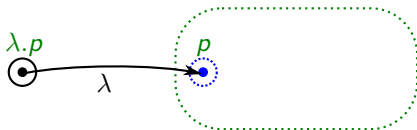
# CCS operations on transition systems

- $\lambda.p$ :

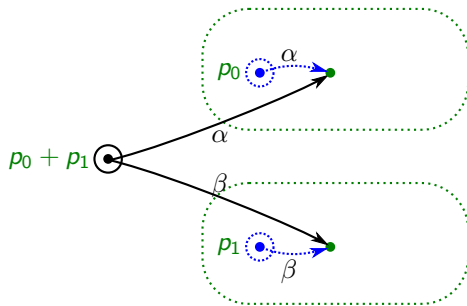


# CCS operations on transition systems

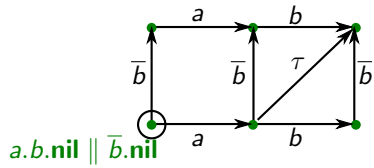
- $\lambda.p$ :



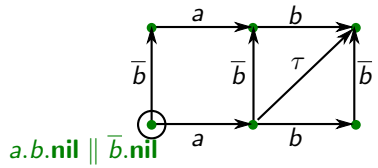
- $p_0 + p_1$ :



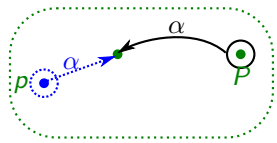
- $a.b \parallel \bar{b}$ :



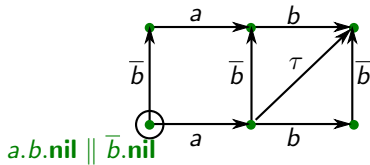
- $a.b \parallel \bar{b}$ :



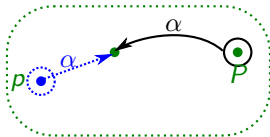
- $P$  where  $P \stackrel{\text{def}}{=} p$ :



- $a.b \parallel \bar{b}$ :



- $P$  where  $P \stackrel{\text{def}}{=} p$ :



$p \setminus L, p[f]: \dots$

A denotational semantics!

# From value-passing to pure

A translation giving a pure CCS process  $\hat{p}$  from a value-passing CCS closed term  $p$

$p$	$\hat{p}$
<b>nil</b>	<b>nil</b>



# From value-passing to pure

A translation giving a pure CCS process  $\hat{p}$  from a value-passing CCS closed term  $p$

$p$	$\hat{p}$
<b>nil</b>	<b>nil</b>
$(\tau \rightarrow p)$	$(\tau.\hat{p})$

# From value-passing to pure

A translation giving a pure CCS process  $\hat{p}$  from a value-passing CCS closed term  $p$

$p$	$\hat{p}$
<b>nil</b>	<b>nil</b>
$(\tau \rightarrow p)$	$(\tau.\hat{p})$
$(\alpha!a \rightarrow p)$	$\overline{\alpha m}.\hat{p}$ where $a$ evaluates to $m$

# From value-passing to pure

A translation giving a pure CCS process  $\hat{p}$  from a value-passing CCS closed term  $p$

$p$	$\hat{p}$
<b>nil</b>	<b>nil</b>
$(\tau \rightarrow p)$	$(\tau.\hat{p})$
$(\alpha!a \rightarrow p)$	$\overline{\alpha m}.\hat{p}$ where $a$ evaluates to $m$
$(\alpha?x \rightarrow p)$	$\sum_{m \in \mathbf{Num}} \alpha m.p[\widehat{m/x}]$

# From value-passing to pure

A translation giving a pure CCS process  $\hat{p}$  from a value-passing CCS closed term  $p$

$p$	$\hat{p}$
<b>nil</b>	<b>nil</b>
$(\tau \rightarrow p)$	$(\tau.\hat{p})$
$(\alpha!a \rightarrow p)$	$\overline{\alpha m}.\hat{p}$ where $a$ evaluates to $m$
$(\alpha?x \rightarrow p)$	$\sum_{m \in \text{Num}} \alpha m.\widehat{p[m/x]}$
$(b \rightarrow p)$	$\hat{p}$ if $b$ evaluates to true <b>nil</b> if $b$ evaluates to false

# From value-passing to pure

A translation giving a pure CCS process  $\hat{p}$  from a value-passing CCS closed term  $p$

$p$	$\hat{p}$
<b>nil</b>	<b>nil</b>
$(\tau \rightarrow p)$	$(\tau.\hat{p})$
$(\alpha!a \rightarrow p)$	$\overline{\alpha m}.\hat{p}$ where $a$ evaluates to $m$
$(\alpha?x \rightarrow p)$	$\sum_{m \in \mathbf{Num}} \alpha m.p[\widehat{m/x}]$
$(b \rightarrow p)$	$\hat{p}$ if $b$ evaluates to true <b>nil</b> if $b$ evaluates to false
$p_0 + p_1$	$\hat{p}_0 + \hat{p}_1$

# From value-passing to pure

A translation giving a pure CCS process  $\hat{p}$  from a value-passing CCS closed term  $p$

$p$	$\hat{p}$
<b>nil</b>	<b>nil</b>
$(\tau \rightarrow p)$	$(\tau.\hat{p})$
$(\alpha!a \rightarrow p)$	$\overline{\alpha m}.\hat{p}$ where $a$ evaluates to $m$
$(\alpha?x \rightarrow p)$	$\sum_{m \in \text{Num}} \alpha m.p[\widehat{m/x}]$
$(b \rightarrow p)$	$\hat{p}$ if $b$ evaluates to true <b>nil</b> if $b$ evaluates to false
$p_0 + p_1$	$\hat{p}_0 + \hat{p}_1$
$p_0 \parallel p_1$	$\hat{p}_0 \parallel \hat{p}_1$

# From value-passing to pure

A translation giving a pure CCS process  $\hat{p}$  from a value-passing CCS closed term  $p$

$p$	$\hat{p}$
<b>nil</b>	<b>nil</b>
$(\tau \rightarrow p)$	$(\tau.\hat{p})$
$(\alpha!a \rightarrow p)$	$\overline{\alpha m}.\hat{p}$ where $a$ evaluates to $m$
$(\alpha?x \rightarrow p)$	$\sum_{m \in \mathbf{Num}} \alpha m.p[m/x]$
$(b \rightarrow p)$	$\hat{p}$ if $b$ evaluates to true <b>nil</b> if $b$ evaluates to false
$p_0 + p_1$	$\hat{p}_0 + \hat{p}_1$
$p_0 \parallel p_1$	$\hat{p}_0 \parallel \hat{p}_1$
$p \setminus L$	$\hat{p} \setminus \{\alpha m \mid \alpha \in L \ \& \ m \in \mathbf{Num}\}$

# From value-passing to pure

A translation giving a pure CCS process  $\hat{p}$  from a value-passing CCS closed term  $p$

$p$	$\hat{p}$
<b>nil</b>	<b>nil</b>
$(\tau \rightarrow p)$	$(\tau.\hat{p})$
$(\alpha!a \rightarrow p)$	$\overline{\alpha m}.\hat{p}$ where $a$ evaluates to $m$
$(\alpha?x \rightarrow p)$	$\sum_{m \in \mathbf{Num}} \alpha m.p[\widehat{m/x}]$
$(b \rightarrow p)$	$\hat{p}$ if $b$ evaluates to true <b>nil</b> if $b$ evaluates to false
$p_0 + p_1$	$\hat{p}_0 + \hat{p}_1$
$p_0 \parallel p_1$	$\hat{p}_0 \parallel \hat{p}_1$
$p \setminus L$	$\hat{p} \setminus \{\alpha m \mid \alpha \in L \ \& \ m \in \mathbf{Num}\}$
$P(a_1, \dots, a_k)$	$P_{m_1, \dots, m_k}$ where $a_i$ evaluates to $m_i$

For every definition  $P(x_1, \dots, x_k)$ , we have a collection of definitions  $P_{m_1, \dots, m_k}$  indexed by  $m_1, \dots, m_k \in \mathbf{Num}$ .



# Examples

Let  $translate(p) = \hat{p}$ :

$$\begin{aligned} & translate(\alpha?x \rightarrow (x \leq 5 \rightarrow (\beta!x \rightarrow \mathbf{nil}))) \\ = & \sum_{n \in \mathbf{Num}} \alpha n. translate(x \leq 5 \rightarrow (\beta!x \rightarrow \mathbf{nil})) \end{aligned}$$

# Examples

Let  $translate(p) = \hat{p}$ :

$$\begin{aligned} & translate(\alpha?x \rightarrow (x \leq 5 \rightarrow (\beta!x \rightarrow \mathbf{nil}))) \\ = & \sum_{n \in \mathbf{Num}} \alpha n. translate(x \leq 5 \rightarrow (\beta!x \rightarrow \mathbf{nil})) \end{aligned}$$

$$\begin{aligned} & translate(n \leq 5 \rightarrow (\beta!n \rightarrow \mathbf{nil})) \\ = & \begin{cases} \overline{\beta n}. \mathbf{nil} & \text{if } n \leq 5 \\ \mathbf{nil} & \text{if } n > 5 \end{cases} \end{aligned}$$

## Theorem

$$p \xrightarrow{\lambda} p' \text{ iff } \widehat{p} \xrightarrow{\widehat{\lambda}} \widehat{p}'$$

# Recursion: an alternative

- Instead of a process

$$P \text{ where } P \stackrel{\text{def}}{=} p$$

we can use

$$\text{rec}(P = p)$$

with rule

$$\frac{p[\text{rec}(P = p)/P] \xrightarrow{\lambda} p'}{\text{rec}(P = p) \xrightarrow{\lambda} p'}$$

- Example:  $\text{rec}(P = a.\mathbf{nil} + b.P)$

# Recursion: an alternative

- Instead of a process

$P$  where  $P \stackrel{\text{def}}{=} p$  and  $Q = q$

we can use the notation

$\text{rec}_1(P = p, Q = q)$

and for  $Q$  we can use

$\text{rec}_2(P = p, Q = q)$

# Recursion: an alternative

- Instead of a process

$$P \text{ where } P \stackrel{\text{def}}{=} p \text{ and } Q = q$$

we can use the notation

$$\text{rec}_1(P = p, Q = q)$$

and for  $Q$  we can use

$$\text{rec}_2(P = p, Q = q)$$

- Generally, instead of  $P_j$  where  $P_i = p_i$  is a collection of definitions indexed by  $i \in I$ , can use

$$\text{rec}_j(P_i = p_i)_{i \in I}$$

which is also written

$$\text{rec}_j(\vec{P} = \vec{p})$$

# Proofs of correctness

- By satisfying formulas in a logic
- By satisfying an equivalence