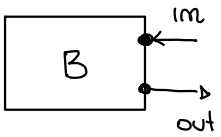
LCD (09/03)

* VALUE PASSING CCS

pure (synchronisation) CCS

Example

buffer



input $m \in \mathbb{N}$

output $m+1 \in \mathbb{N}$

$B \overset{def}{=} in(x).\ B'(x)$

$B'(x) \overset{def}{=} \overline{out}(x+1).\ B$

behaviour

input

$$\frac{}{a(x).\ P \xrightarrow{a(m)} P\{m/x\}} \qquad m \in \mathbb{N}$$

↑ process P where each free occurrence of x is replaced by m

output

$$\frac{}{\bar{a}(e).\ P \xrightarrow{\bar{a}(m)} P}$$

if e evaluates to m

silent

$$\frac{}{\tau.\ P \xrightarrow{\tau} P}$$

constants

$$\frac{P\{m_1/x_1, \ldots, m_n/x_n\} \xrightarrow{\alpha} P'}{A(e_1, \ldots, e_n) \xrightarrow{\alpha} P'}$$

if e_i evaluates to m_i

$A(x_1, \ldots, x_n) \overset{def}{=} P$

* How do I evaluate expressions with variables?

$x + 2 \longrightarrow$

I only execute processes where all variables are bound : closed processes

```
fact (m) :
  if m=0 then
     return 1
  else
     return m * fact(m-1)
```

fact (m+3)

fact (42)

We should define

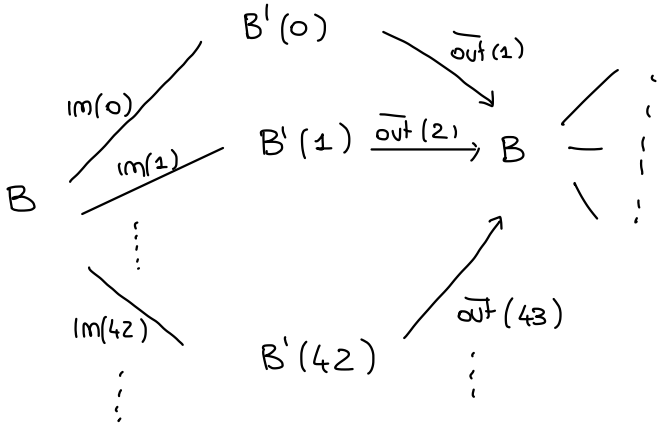
→ free / bound variables

→ closed processes evaluates, by each rule, to close processes ...

Example

$B = in(x).\ B'(x)$

$B'(x) = \overline{out}(x+1).\ B$



* conditionals

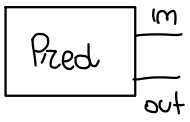
if b then P else Q

$$\frac{P \xrightarrow{\alpha} P'}{\text{if } b \text{ then } P \text{ else } Q \xrightarrow{\alpha} P'}$$

if b evaluates to true

$$\frac{Q \xrightarrow{\alpha} Q'}{\text{if } b \text{ then } P \text{ else } Q \xrightarrow{\alpha} Q'}$$ if b evaluates to false

Exercise: buffer



input $m \in \mathbb{N}$

output $m-1$ if $m>0$

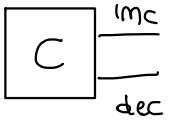
0 otherwise

$Pred \stackrel{def}{=} in(x).\ P(x)$

$P(x) \stackrel{def}{=}$ if $x>0$ then $\overline{out}(x-1).\ Pred$

else $\overline{out}(0).\ Pred$

Exercise: counter



$C(x) \stackrel{def}{=} inc(x).\ C(x+1) +$

if $x>0$ then $dec.\ C(x-1)$

else $C(0)$

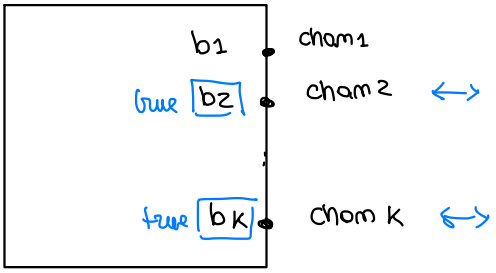
$c(x) \stackrel{def}{=} inc(x).\ C(x+1) +$

if $x>0$ then $dec.\ C(x-1)$

else 0

NOTE: The rule is not

$$\frac{}{\text{if } b \text{ then } P \text{ else } Q \longrightarrow P}$$ if b is true



* CCS Value Passing

→ variables for values x, y, z Var

→ channels a, b, c $\mathcal{A}$

→ process constants $K \in \mathcal{K}$ $K(x_1, \_, x_n)$ ↑ arity

→ expressions

→ arithmetic expressions

$e ::= n \mid e + e \mid e \times e \mid \ldots$

→ boolean expressions

$b ::= e = e \mid e \leq e \mid \neg b \mid b \wedge b \mid \ldots$

→ Syntax

$P, Q ::= K(e_1, \ldots, e_n) \mid a(x).P \mid \bar{a}(e).P \mid \tau.P \mid$

$K(x_1, \ldots, x_n) \stackrel{\text{def}}{=} P$ if b then P $\mid \sum_{i \in I} P_i$

$P \mid Q \mid P[f] \mid P \setminus L$

if b then P else Q

$\stackrel{?}{=}$

if b then P + if $\neg b$ then Q

|

$$\frac{}{\tau.P \xrightarrow{\tau} P} \qquad \frac{}{a(x).P \xrightarrow{a(m)} P\{^m/_x\}}\ m \in \mathbb{N} \qquad \frac{}{\bar{a}(e).P \xrightarrow{\bar{a}(m)} P}\ e \text{ eval to } m$$

$$\frac{P\{^{m_1}/_{x_1}, \dots, ^{m_h}/_{x_h}\} \xrightarrow{\alpha} P'}{K(e_1, \dots, e_h) \xrightarrow{\alpha} P'} \quad \text{if } K(x_1, \dots, x_h) \stackrel{def}{=} P$$

$$\frac{P \xrightarrow{\alpha} P'}{\text{if } b \text{ then } P \xrightarrow{\alpha} P'} \quad \text{if } b \text{ evaluates to true}$$

$$\frac{P \xrightarrow{\bar{a}(m)} P' \qquad Q \xrightarrow{a(m)} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'} \qquad \frac{}{a(x).P \xrightarrow{a(x)} P}$$

$$\frac{P \xrightarrow{\bar{a}(m)} P' \qquad Q \xrightarrow{a(x)} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'\{^m/_x\}} \quad ?$$

POSSIBLE, MORE COMPLEX THAN THIS

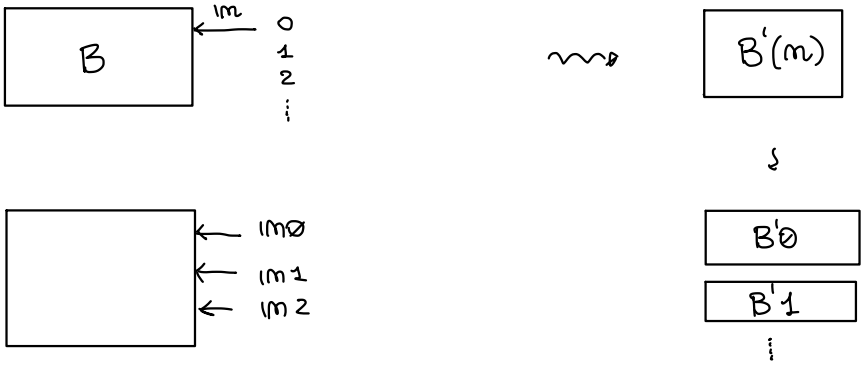
$$\underbrace{(a(x).\ \bar{a}(x).0) \mid \bar{b}(x).0}_{Q}$$

* ENCODING CCS with VALUE PASSING into PURE CCS

$[\![\ ]\!] : \text{CCS-VP} \longrightarrow \text{CCS}$

$\mathcal{A}$ channels

K constants $K(x_1, x_n)$



in CCS

$$\mathcal{A}' = \{ a_m \mid a \in \mathcal{A},\ m \in \mathbb{N} \}$$

$$K' = \{ K_{m_1,\dots,m_h} \mid K \in K \text{ with arity } h,\ m_1,\dots,m_h \in \mathbb{N} \}$$

$$[\![a(x).P]\!] = \sum_{m \in \mathbb{N}} a_m . [\![P\{^m/_x\}]\!]$$

$$[\![\bar{a}(e).P]\!] = \overline{a_m} . [\![P]\!] \qquad e \text{ evaluates to } m$$

$$[\![\tau.P]\!] = \tau . [\![P]\!]$$

$$[\![\sum_{i \in I} P_i]\!] = \sum_{i \in I} [\![P_i]\!]$$

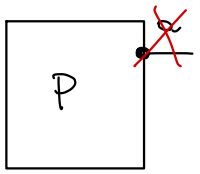
$$[\![P \mid Q]\!] = [\![P]\!] \mid [\![Q]\!]$$

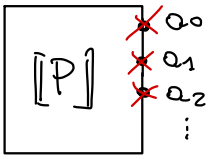
$$[\![P \setminus L]\!] = [\![P]\!] \setminus \{ a_m \mid a \in L,\ m \in \mathbb{N} \}$$

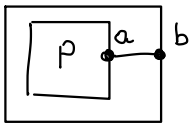
$$[\![P]\!][f] = [\![P]\!][f']$$

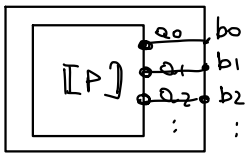
$$\begin{cases} f'(a_m) = f(a)_m \\ f'(\tau) = \tau \end{cases}$$

$P \setminus a$







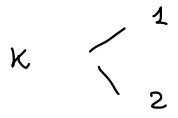


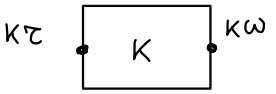
$$[\![\text{if } b \text{ then } P]\!] = \begin{cases} [\![P]\!] & \text{if } b \text{ evaluates to true} \\ 0 & \text{otherwise} \end{cases}$$

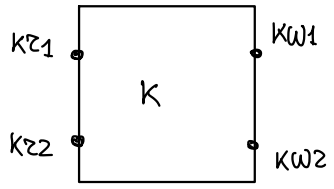
$$[\![K(e_1,\dots,e_h)]\!] = K_{m_1,\dots,m_h} \qquad \text{where } e_i \text{ evaluates to } m_i$$

$$K_{m_1 \dots m_h} \overset{\text{def}}{=} [\![P\{^{m_1}/_{x_1} \dots {}^{m_h}/_{x_h}\}]\!] \qquad \text{where } K \overset{\text{def}}{=} P$$

Peterson (Post Pessom)







$$K(x) \overset{def}{=} \overline{Kr}(x).\ K(x) + Kw(y).\ K(y)$$

$$K1 \overset{def}{=} \overline{Kr1}.\ K1 + \sum_{m \in \{1,2\}} Kwm.\ Km$$

$$\overset{def}{=} \overline{Kr1}.\ K1 + Kw1.\ K1 + Kw2.\ K2$$

$$K2 = \text{-----}$$

MINI PROJECT : Write a compiler CCS-VP into CCS

$x : [2..4]$ (finite interval types)

$x : \{a, b, c, \ldots\}$ enumeration

Theorem : Let $[\![\ ]\!] : \text{CCS-VP} \to \text{CCS}$ defined above.

Then for all (closed) programs in CCS-VP P

(i) if $P \xrightarrow{\alpha} P'$ then $[\![P]\!] \xrightarrow{\hat{\alpha}} [\![P']\!]$

(ii) if $[\![P]\!] \xrightarrow{\hat{\alpha}} Q$ then $P \xrightarrow{\alpha} P'$ and $[\![P']\!] = Q$

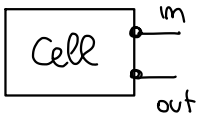
where

$$\hat{\alpha} = \begin{cases} a_m & \text{if } \alpha = a(m) \\ \overline{a_m} & \text{if } \alpha = \bar{a}(m) \\ \tau & \text{otherwise } (\alpha = \tau) \end{cases}$$

proof

(exam exercise)

EXERCISE :



$Cell = in(x) . C(x)$

$C(x) = \overline{out}(x) . Cell$

Using Cell as a component

(1) Unordered two-place buffer

$B_2 = in(x) . B_1(x)$

$B_1(x) = in(y) . B_0(x,y) + \overline{out}(x) . B_2$

$B_0(x,y) = \overline{out}(x) . B_1(y) + \overline{out}(y) . B_1(x)$

$B_2 \overset{?}{\approx}$

(2) FIFO two-place buffer

$F_2 = in(x) . F_1(x)$

$F_1(x) = in(y) . F_0(x,y) + \overline{out}(x) . F_2$

$F_0(x,y) = \overline{out}(x) . F_1(y)$