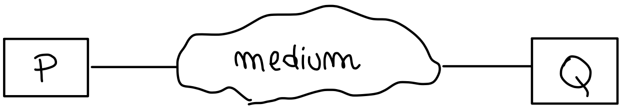
LCD (04/02/2024)

Calculus of Communicating Systems (CCS, Milner '80)

Idea: set of processes { - executing in parallel
- interaction / communication

pb: Which kind of communication ?



- ether :

  1. send is always possible (~~unbounded~~ bounded)
  2. receive is possible if a message is available (destructive)
  3. no order guarantee

- buffer

  1, 2 as before (bounded ?)

  3. order of messages is preserved

- shared memory

  1. send ⇝ write
  2. receive ⇝ read
  3. order : no guarantee

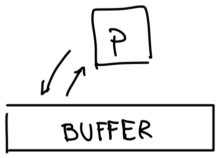
  ⋮

Idea: no distinction between

→ active entities → agents

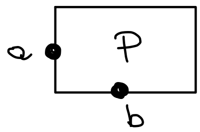
→ passive " → medium

slogan: everything is a process



PROCESSES: communicating via synchronous interactions
(hand shake)

STRUCTURE

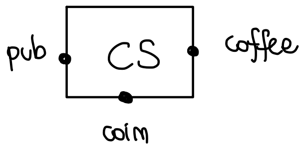


name

interface (ports, channels)

Example: Computer scientist

coffee ⇝ publications



Behaviour → CCS program

- structure
- interaction

Syntax of CCS programs

* Inaction (nil)

0 deadlock

* Action prefixing

given a channel (coin, coffee ....)

$\overline{coin}.0$

$coffee.0$

$\overline{coin}.coffee.0$

in general given an <u>action</u> (input / output channel) α

$\alpha . P$

* <u>Process constant</u>

$Break \overset{def}{=} \overline{coin} . coffee . 0$

$Clock \overset{def}{=} \overline{tick} . Clock$

$CM \overset{def}{=} coin . \overline{coffee} . CM$

* <u>Non Deterministic Choice</u>

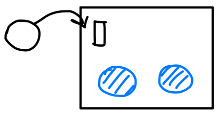
given processes P and Q $\qquad P + Q$

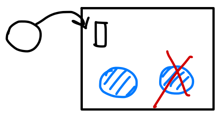
$CTM \overset{def}{=} coin . (\overline{coffee} . CTM + \overline{tea} . CTM)$

$CTM' \overset{def}{=} coin . \overline{coffee} . CTM' + coin . \overline{tea} . CTM'$

¿ equivalent ?

different behaviour !


CTM


CTM'

<u>Exercise</u> : Broken Clock

it surely emits one tick , then it can stop at any time

$Clock = \overline{tick} . Clock$

$BC = \overline{tick} . (BC + 0)$ $\qquad BC + 0 = BC$

NO

$BC = \overline{tick} . BC + \overline{tick} . 0$

OK

Exercise : $CM = coin . \overline{coffee} . CM$

failing machine

→ can input a coin without providing coffee

→ at any time it can fail emitting signal $\overline{fail}$

$$BCM = coin . BCM + coin . \overline{coffee} . BCM + \cancel{coin . \overline{fail}} . 0$$
$$+ \cancel{coin . \overline{coffee} . \overline{fail}} . 0 + \overline{fail} . 0$$

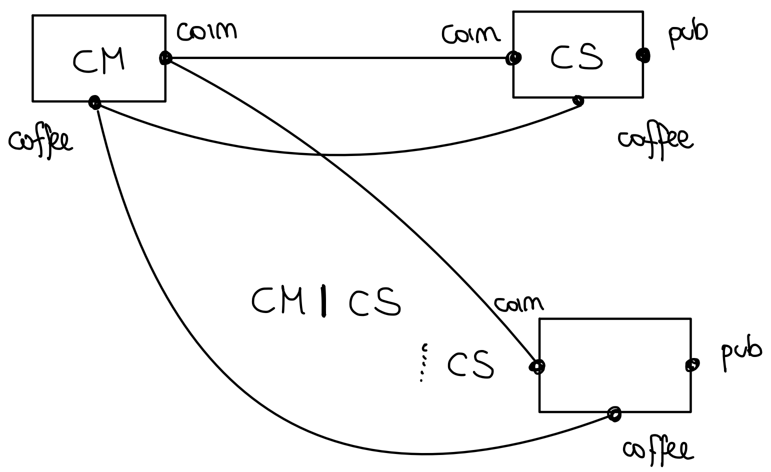
or

equivalent?

$$BCM = \overline{fail} . 0 + coin (BCM + \overline{coffee} . BCM)$$

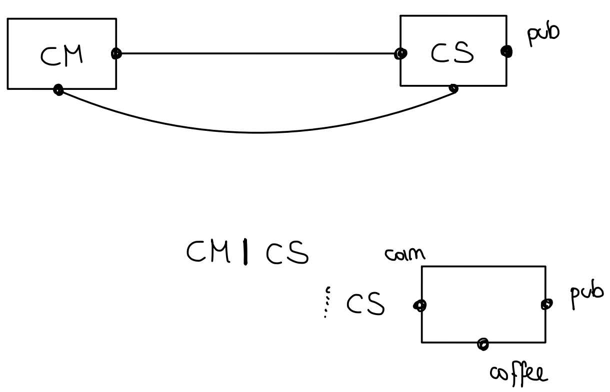
* Parallel Composition

$CM = coin . \overline{coffee} . CM$ $\qquad$ $CS = \overline{pub} . \overline{coin} . coffee . CS$



* Restriction

$(CM \mid CS) \setminus \{ coin , coffee \}$



* Relabelling

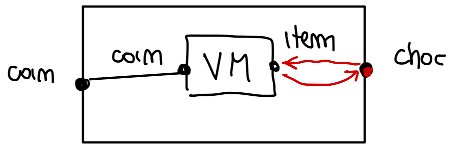
$$CHOC \overset{def}{=} coin.\overline{choc}.CHOC$$

$$CHIPS \overset{def}{=} coin.\overline{chips}.CHIPS$$

$\vdots$

$$VM \overset{def}{=} coin.\overline{item}.VM$$

$$CHOC = VM\left[{}^{choc}/_{item}\right]$$



$$CHIPS = VM\left[{}^{chips}/_{item}\right]$$

* Behaviour

processes will perform

→ state transitions

→ determined by communications

$$CS = \overline{pub}.CS1$$
$$CS1 = \overline{coin}.CS2$$
$$CS2 = coffee.CS$$

$$CM = coin.CM1$$
$$CM1 = \overline{coffee}.CM$$

$$\begin{tikzcd} CS \arrow[r, "\overline{pub}"] & CS1 \arrow[r, "\overline{coin}"] & CS2 \arrow[ll, bend left, "coffee"] \end{tikzcd}$$

$$\begin{tikzcd} CM \arrow[r, "coin"] & CM1 \arrow[l, bend left, "\overline{coffee}"] \end{tikzcd}$$

$$CM \mid CS$$
$$\downarrow \overline{pub}$$
$$CM \mid CS1 \xrightarrow{\tau} CM1 \mid CS2$$

invisible / silent action

$coin \mid \overline{coin}$?

$$CS = \overline{pub}.\,CS1$$
$$CS1 = \overline{coin}.\,CS2$$
$$CS2 = coffee.\,CS$$

$$CM = coin.\,CM1$$
$$CM1 = \overline{coffee}.\,CM$$

$$\begin{tikzcd} CS \arrow[r, "\overline{pub}"] & CS1 \arrow[r, "\overline{coin}"] & CS2 \arrow[ll, bend left, "coffee"] \end{tikzcd}$$

$$\begin{tikzcd} CM \arrow[r, "coin"] & CM1 \arrow[l, bend left, "\overline{coffee}"] \end{tikzcd}$$

$$Office = (CS \mid CM) \setminus \underbrace{\{coin, coffee\}}_{L}$$

$$\boxed{\begin{tikzcd} Office \arrow[d, "\overline{pub}"] \\ (CS1 \mid CM) \setminus L \arrow[d, "\tau"] \\ (CS2 \mid CM) \setminus L \arrow[uu, bend left, "\tau"] \end{tikzcd}}$$

$$\xrightarrow{\overline{pub}} \; \xrightarrow{\overline{pub}}$$

$$Spec = \overline{pub}.\,Spec$$

$$Spec \sim Off$$

specification

implementation

We need to define rigorously:

- syntax
- operational behaviour
- program equivalence
- verification algorithms and tools