LCD (24/02)

Calculus of Communicating Systems (Milner, '80)

CCS

pb: Which kind of communication ?

* ether

① send is always possible (unless full) (unbounded)

② receive is possible (if there are messages)

③ no order guarantee

* buffer (bounded)

①, ②

③ order of messages is maintained

* shared memory

① sending → writing to memory

② receiving → reading from memory

③ no order guarantee

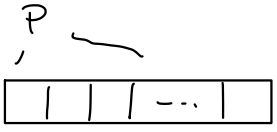
⋮

Idea: no distinction between

→ active entities (agents)

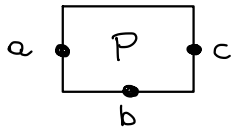
→ passive " (medium)

EVERYTHING IS A PROCESS



PROCESS : interacting through synchronous message passing

STRUCTURE :

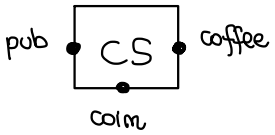


- name
- interface (port, channel)

BEHAVIOUR : CCS program

- structure
- interactions

Example : Computer scientist



Syntax of CCS

* Inaction (nil)

0 deadlock

* Action prefixing

given a channel coin, coffee

$\overline{coin}.0$

$coffee.0$

$\overline{coin}.coffee.0$

$\overline{pub}.\overline{coin}.coffee.0$

in general, given a channel c and procss P

$c.P$ $\overline{c}.P$

* Process Constants

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

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

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

$CS \overset{def}{=} \overline{pub}.\ \overline{coin}.\ coffee.\ CS$

* Non deterministic Choice

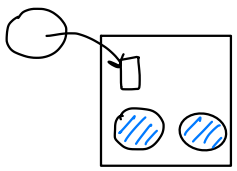
given processes P, Q one construct $P + Q$

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

$\sim$?

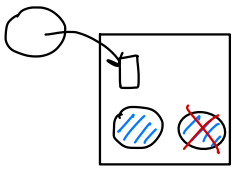
$CTM_1 \overset{def}{=} coin.\ (\ \overline{coffee} + \overline{tea}\ ).\ CTM$ NOT IN THE SYNTAX • is not sequential composition

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



CTM

$\not\sim$



CTM'

same sequences of interactions (same traces)

EXERCISE : Broken Clock

it emits a tick and then it can stop after any number of ticks

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

$BC \overset{def}{=} tick.\ (BC + 0)$ $\qquad BC + 0 \sim BC$

$BC \overset{def}{=} tick.\ BC + tick.\ 0$

EXERCISE : Failing coffee machine

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

→ can get a coin and provide nothing

→ at any moment, can fail emitting a $\overline{fail}$ message

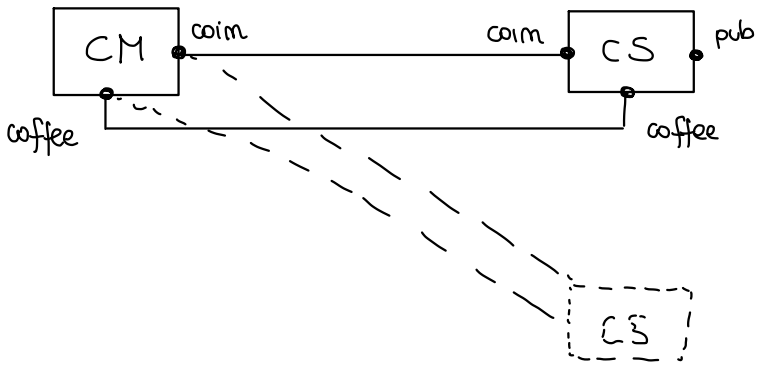
$BCM \stackrel{def}{=} coin.(\overline{coffee}.BCM + \overline{fail}.0 + BCM) + \overline{fail}.0$

$BCM \stackrel{def}{=} coin.\overline{coffee}.BCM + coin.BCM + \overline{fail}.0$??

* Parallel Composition

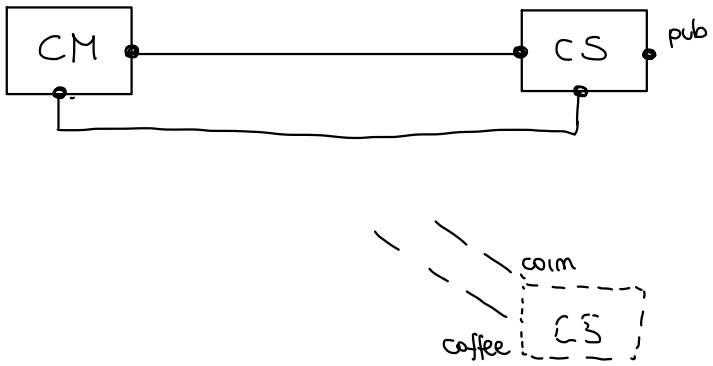
given two processes P, Q you can consider $P \mid Q$

$CM \stackrel{def}{=} coin.\overline{coffee}.CM$ $\quad CS = \overline{pub}.\overline{coin}.coffee.CS$



* RESTRICTION

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



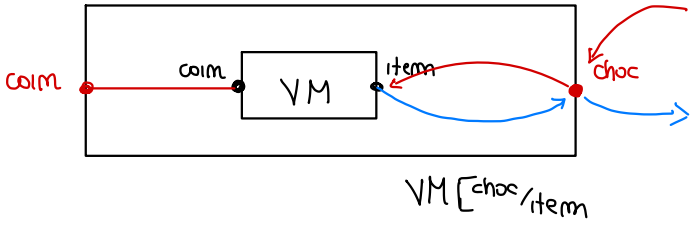
* Relabelling

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

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

$\vdots$

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



$VM[choc/item]$

$CHOC = VM[choc/item]$

$CHIPS = VM[chips/item]$

* BEHAVIOUR

focus on interactions

→ state transitions

→ triggered by communications (send / receive)

$$P \xrightarrow{com} P'$$

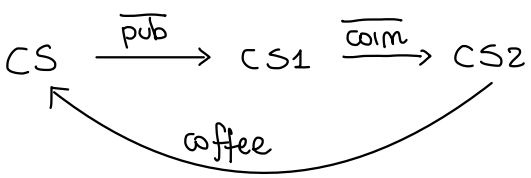
$CS = \overline{pub}.\ CS1$

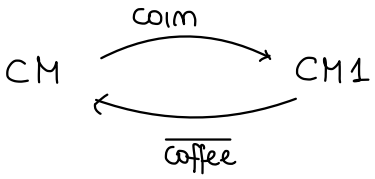
$CS1 = \overline{coin}.\ CS2$

$CS2 = coffee.\ CS$

$CM = coin.\ CM1$

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





$CS \mid CM$

$\downarrow \overline{pub}$

$CS1 \mid CM \xrightarrow{\tau} CS2 \mid CM1$

coin ~~$\overline{coin}$~~

↑ invisible action / silent action

System "Office"

$CS = \overline{pub}.CS1$

$CS1 = \overline{coin}.CS2$

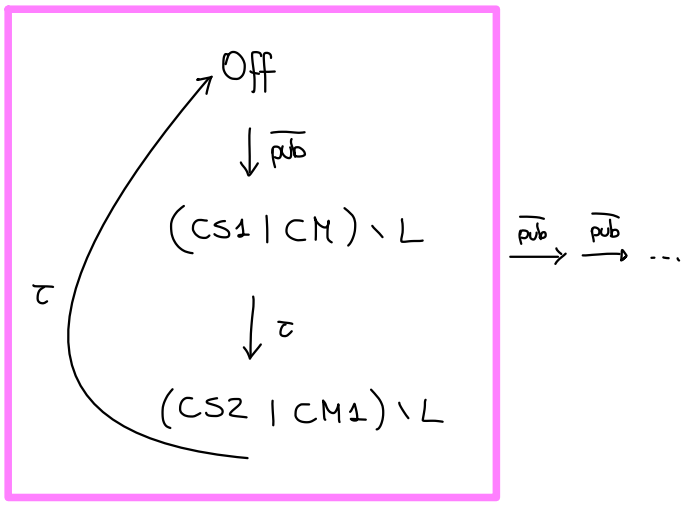
$CS2 = coffee.CS$

$CM = coin.CM1$

$CM1 = \overline{coffee}.CM$

$Off = (CS \mid CM) \setminus L$

$L = \{coin, coffee\}$



$Spec = \overline{pub}.Spec$

$\downarrow \overline{pub}$

$\downarrow \overline{pub}$

$Spec \sim Off$

↑ implementation

We need

→ syntax

→ operational behaviour

→ program equivalence

→ verification algorithms and tools