Calculus of Communicating Systems (CCS, Milner ’80)

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

do: Which kind of communication?

- ether:
  1. send is always possible (unbounded)
  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 \rightarrow write
  2. receive \rightarrow read
  3. order: no guarantee

idea: no distinction between
      → active entities → agents
      → passive → medium
**Slogan**: everything is a process

**Processes**: communicating via synchronous interactions (handshake)

**Structure**

Example: Computer scientist

coffee → no publications

Behaviour → CCS program

- Structure
- Interaction

**Syntax of CCS programs**

* Imachin (mil)
  
  O     dead back

* Action prefixing

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

  caim . O
  coffee . O
  caim . coffee . O
In general given an action (input/output channel) \( \alpha \)
\[ \Rightarrow P \]

* Process constant

\[ \text{Break} \overset{df}{=} \text{coin} \cdot \text{coffee, 0} \]
\[ \text{Clock} \overset{df}{=} \text{tick} \cdot \text{Clack} \]
\[ \text{CM} \overset{df}{=} \text{coin} \cdot \text{coffee} \cdot \text{CM} \]

* Non deterministic Choice

given processes \( P \) and \( Q \)
\[ P + Q \]
\[ \text{CTM} \overset{df}{=} \text{coin} \cdot (\text{coffee} \cdot \text{CTM} + \text{tea} \cdot \text{CM} ) \]
\[ \text{CTM'} \overset{df}{=} \text{coin} \cdot \text{coffee} \cdot \text{CTM'} + \text{coin} \cdot \text{tea} \cdot \text{CM'} \]

different behaviour!

Exercise: Broken Clock

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

\[ \text{Clock} = \text{tick} \cdot \text{Clack} \]
\[ \text{BC} = \text{tick} \cdot (\text{BC} + 0) \]
\[ \text{BC} + 0 = \text{BC} \]

\[ \text{BC} = \text{tick} \cdot \text{BC} + \text{tick} \cdot 0 \]

\[ \text{OK} \]
Exercise: \( CM = \text{coin} . \overline{\text{coffee}} . CM \)

failing machine:
- can input a coin without providing coffee
- at any time it can fail emitting signal \( \overline{\text{fail}} \)

\[
BCM = \text{coin} . BCM + \text{coin} . \overline{\text{coffee}} . BCM + \text{coin} . \overline{\text{fail}} . O
\]
\[
+ \text{coin} . \overline{\text{coffee}} . \overline{\text{fail}} . O + \overline{\text{fail}} . O
\]

or

\[
BCM = \overline{\text{fail}} . O + \text{coin} (BCM + \overline{\text{coffee}} . BCM)
\]

**Parallel Composition**

\[
CM = \text{coin} . \overline{\text{coffee}} . CM
\]

\[
CS = \overline{\text{pub}} . \text{coin} . \text{coffee} . CS
\]

\[
CM \parallel CS
\]

**Restriction**

\[
(\text{CM} \parallel \text{CS}) \setminus \{ \text{coin}, \text{coffee} \}
\]
\* **Re labeling**

\[\begin{align*}
    \text{CHOC} & \equiv \text{com. choc. CHOC} \\
    \text{CHIPS} & \equiv \text{com. chips. CHIPS} \\
    \vdots \\
    \text{VM} & \equiv \text{com. item. VM} \\
    \text{CHOC} & = \text{VM} \left[ \begin{array}{c} \text{choc} \\ \text{item} \end{array} \right] \\
    \text{CHIPS} & = \text{VM} \left[ \begin{array}{c} \text{chips} \\ \text{item} \end{array} \right]
\end{align*}\]

\* **Behaviour**

Processes will perform

\[\rightarrow \text{ state transitions}\]

\[\rightarrow \text{ determined by communications}\]

\[\begin{align*}
    \text{CS} & \equiv \text{pub. CS1} \\
    \text{CS1} & \equiv \text{com. CS2} \\
    \text{CS2} & \equiv \text{coffee. CS}
\end{align*}\]

\[\begin{align*}
    \text{CM} & \equiv \text{com. CM1} \\
    \text{CM1} & \equiv \text{coffee. CM}
\end{align*}\]

\[\begin{align*}
    \text{CS} \xrightarrow{\text{pub.}} \text{CS1} \xrightarrow{\text{com.}} \text{CS2} \\
    \text{CM} \xrightarrow{\text{com.}} \text{CM1}
\end{align*}\]

\[\begin{align*}
    \text{CM} & \mid \text{CS} \\
    \downarrow \text{pub.} \\
    \text{CM} & \mid \text{CS1} \xrightarrow{\tau} \text{CM1} \mid \text{CS2}
\end{align*}\]
We need to define rigorously:

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