Modeling & Verification

CCS (Calculus of Communicating Systems)

Max Tschaikowski (tschaikowski@cs.aau.dk)
Slides courtesy of Giorgio Bacci

in the last Lecture

- Reactive Systems
- Labelled Transition Systems
- Process Algebra (atoms + operations)
- CCS (informally)

in this Lecture

- CCS - the basic principles & motivations
- Examples
- CCS - formal definition (syntax & semantics)

CCS - Motivations

The Calculus of Communicating Systems (CCS) is a prototypical example of process algebra to reason about communication and interactions between reactive systems.

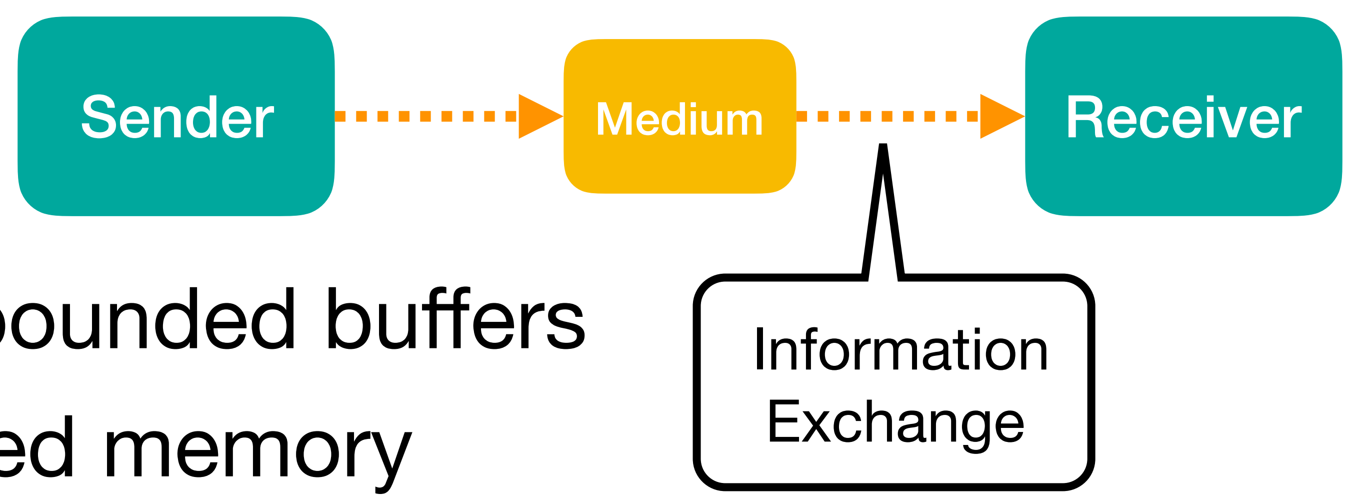
A Calculus of Communicating Systems,
Robin Milner.
Lecture Notes in Computer Science,
Volume 92, 1980. Springer-Verlag.

Robin Milner
(Turing Award winner)

The Key Issue

How to describe communication or interaction between processes running at the same time?

The standard view

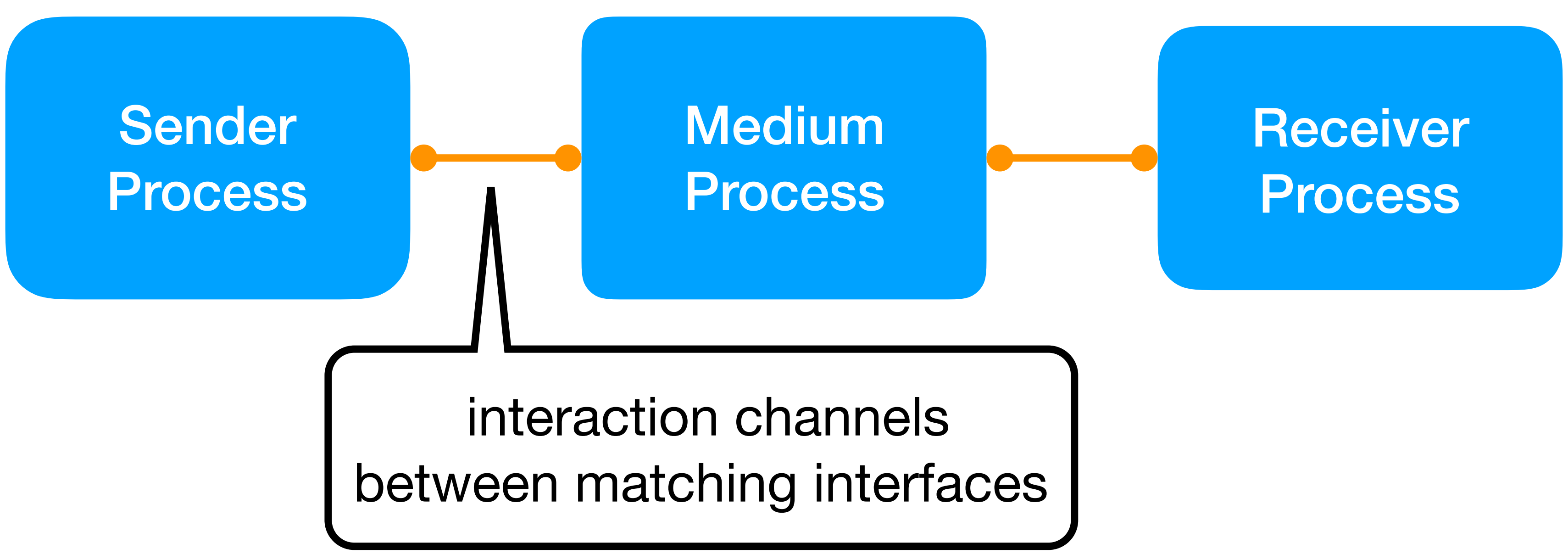


- (un)bounded buffers
- shared memory
- tuple spaces (used in Linda-like languages)
- others ...

Communicating Processes

The key new idea to describe communication
in a genuine general way is to make
no distinction between active/passive components

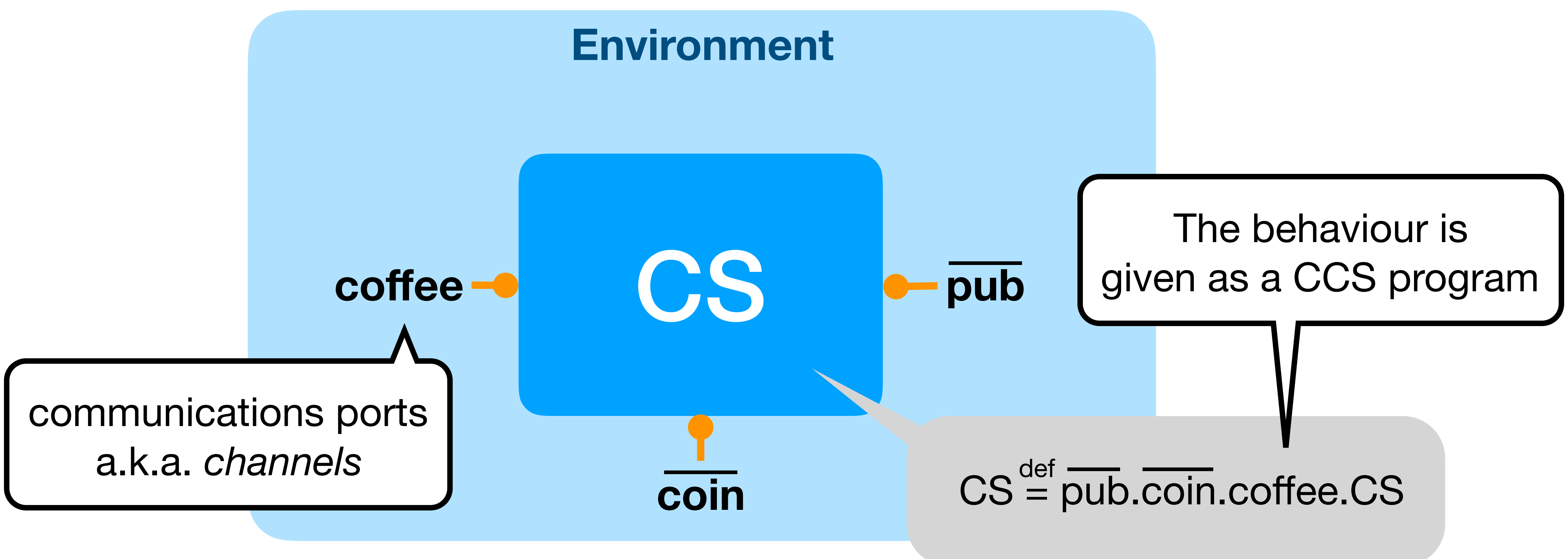
Everything is a process!



The "shape" of a Process

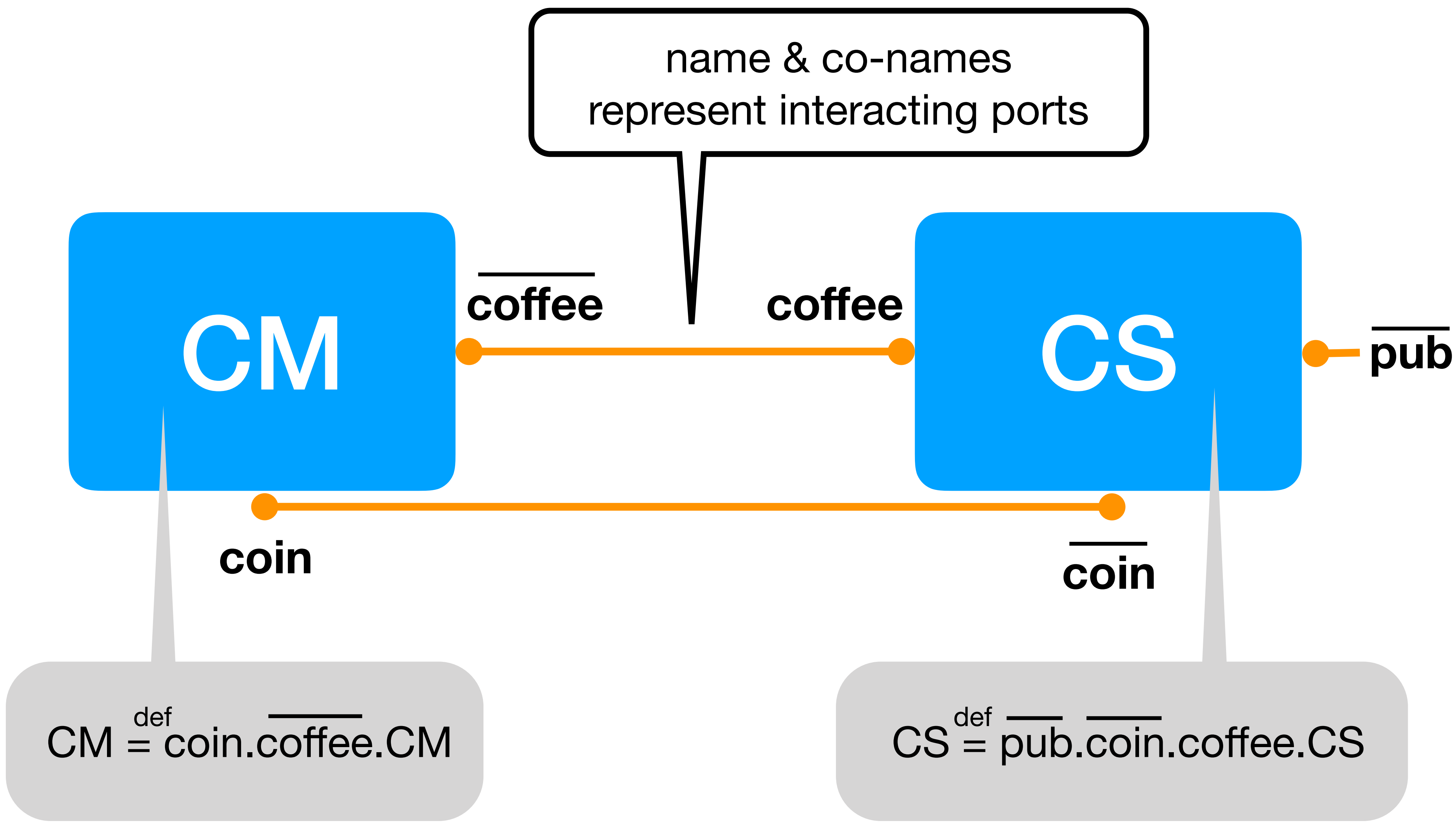
Each process is characterised by a *name*, an *interface*, and a *behaviour*

"*A computer scientist is a machine for turning coffee into publications*"(*)



(*) "*A mathematician is a machine for turning coffee into theorems*"
Misattributed to Paul Erdös - actually by Alfréd Rényi.

Interaction via interfaces

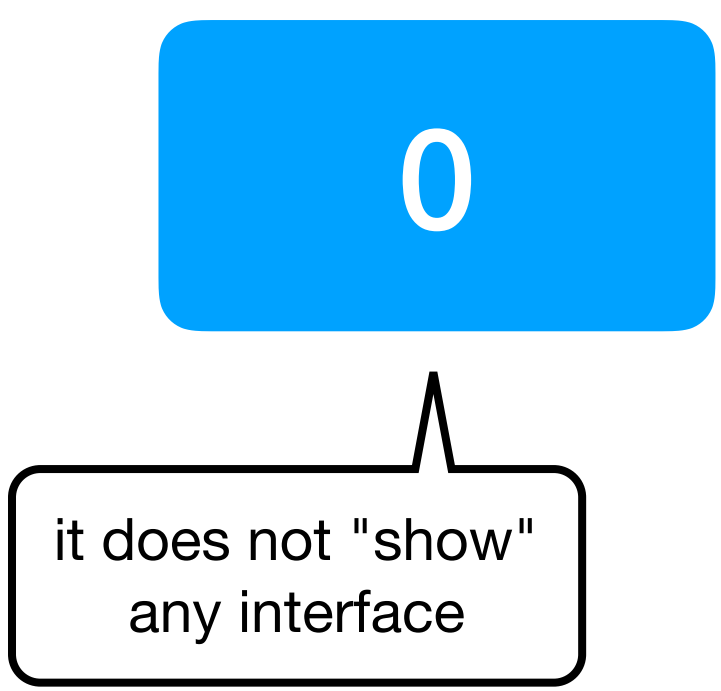


The Algebra of CCS Processes

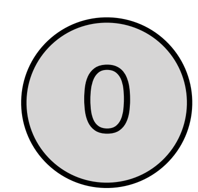
The Nil Process

The Nil process is the most basic process, used to represent inactivity (deadlock behaviour)

Interface



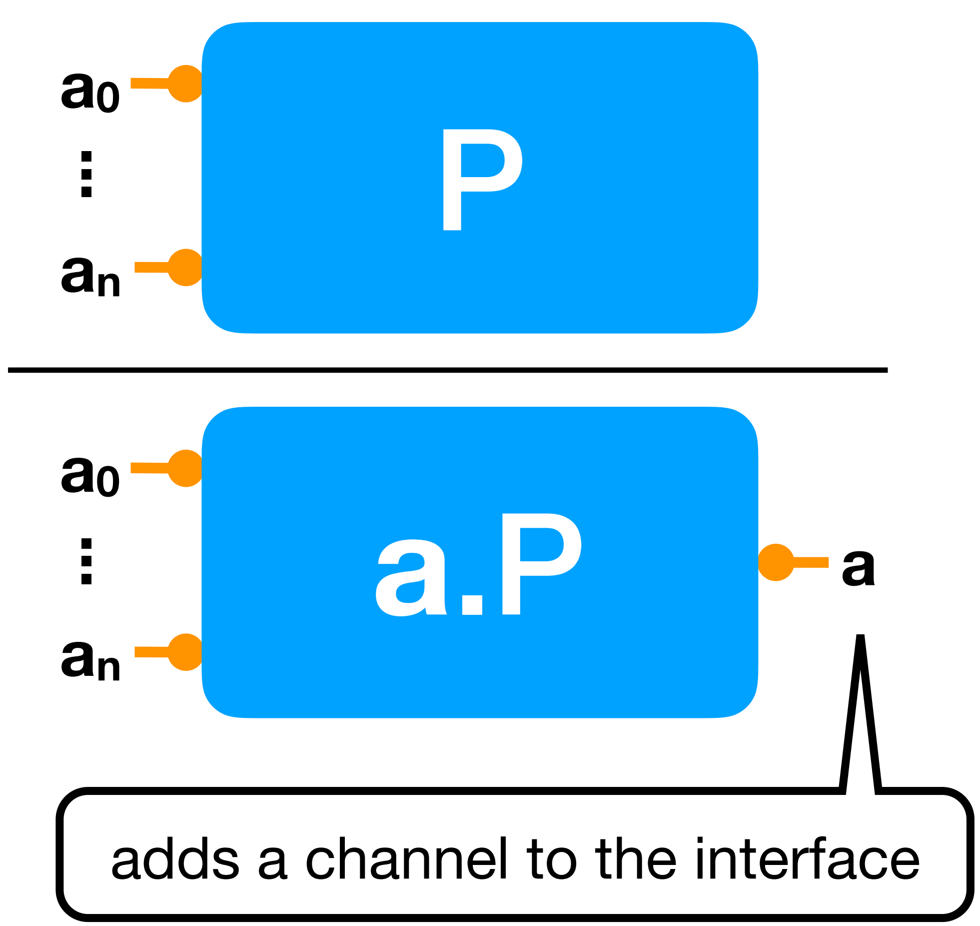
Behaviour (as an LTS)



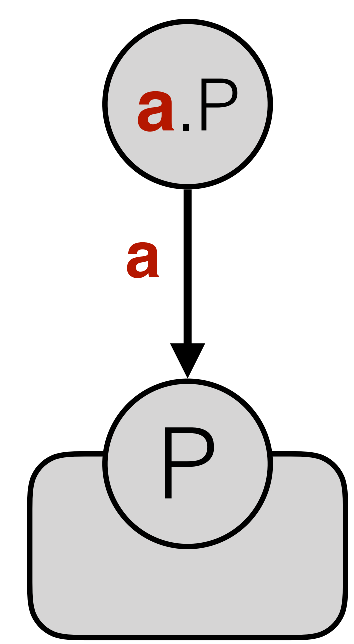
Action Prefixing

Prefixing is the operation taking a process P to **a**.P that does the action **a** and behaves like P thereafter

Interface



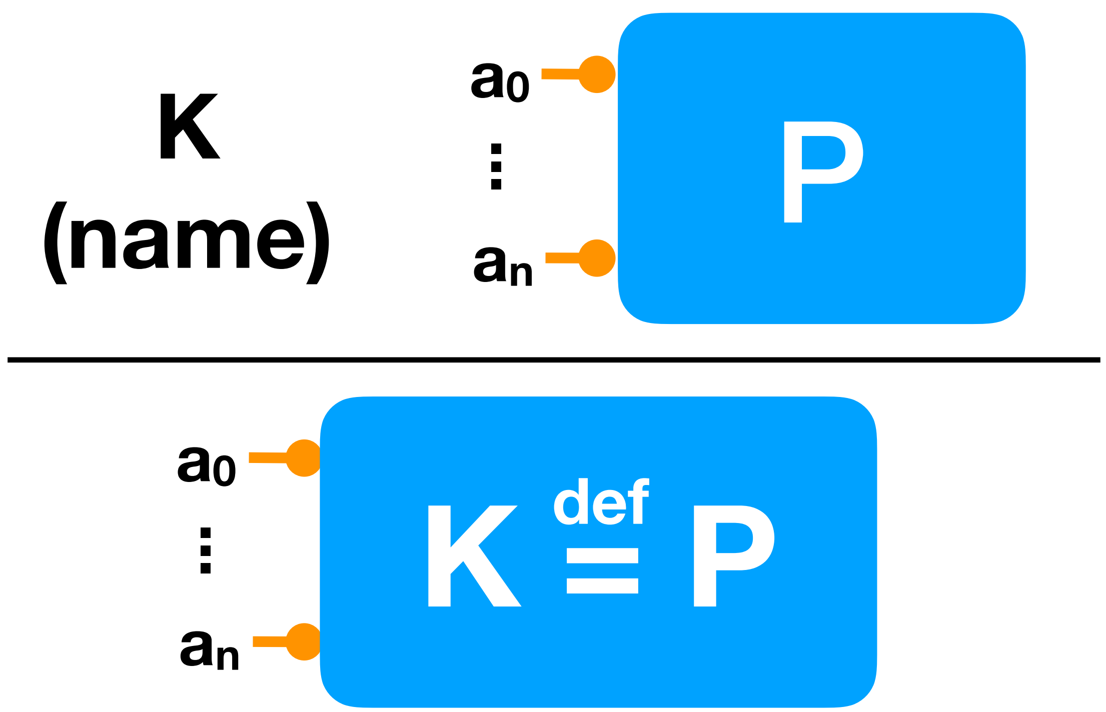
Behaviour (as an LTS)



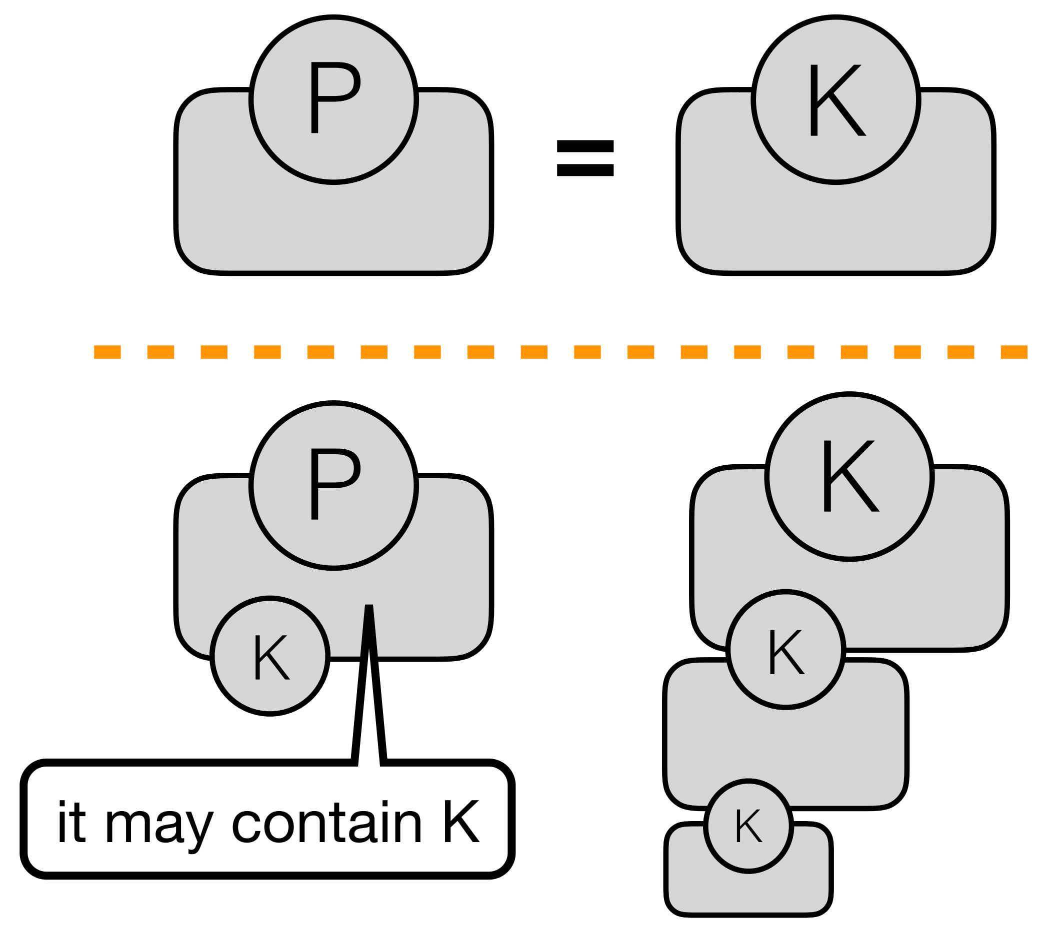
Names & Definitions

Having names for processes allows us to give definitions to process behaviours (possibly recursive)

Interface



Behaviour (as an LTS)



Recursion (Example)

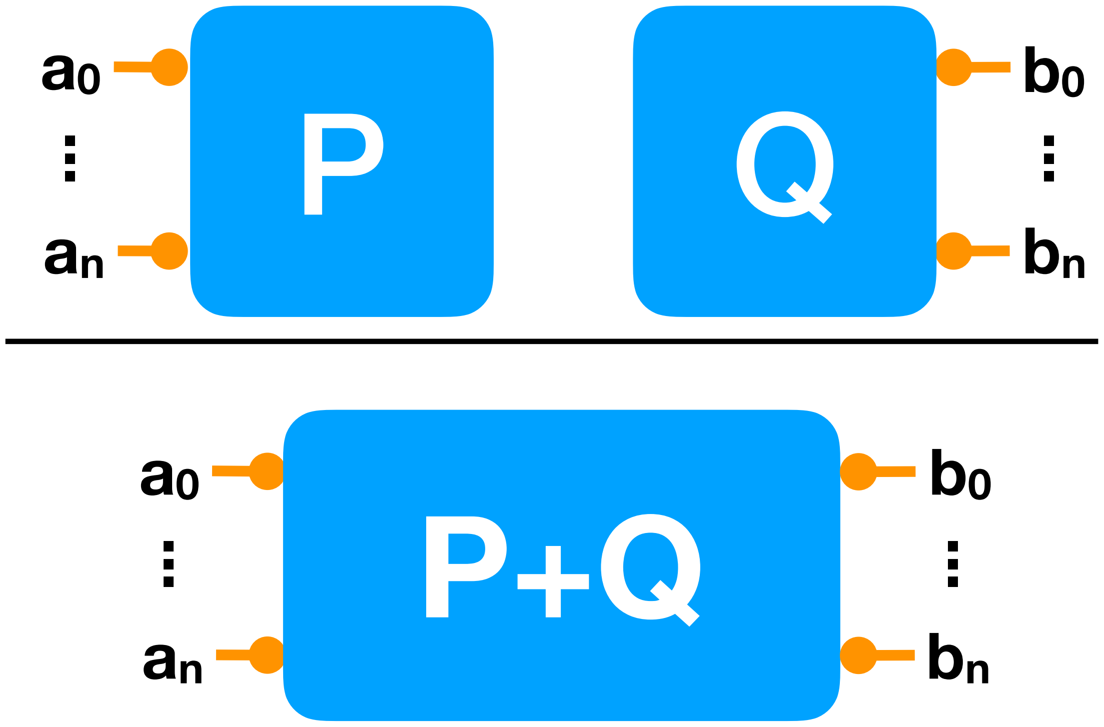
$$
\begin{aligned}
\mathrm{Clock} &\stackrel{\mathrm{def}}{=} \mathbf{tick}.\mathrm{Clock} \\
&= \mathbf{tick}.\mathbf{tick}.\mathrm{Clock} \\
&= \mathbf{tick}.\mathbf{tick}.\mathbf{tick}.\mathrm{Clock} \\
&= \mathbf{tick}.\mathbf{tick}.\mathbf{tick}.\mathbf{tick}.\mathrm{Clock} \\
&\ \vdots \\
&= \underbrace{\mathbf{tick}.\mathbf{tick}.\ \ldots\ \mathbf{tick}.\mathbf{tick}}_{\textbf{n-times}}.\mathrm{Clock}
\end{aligned}
$$

for each positive integer n

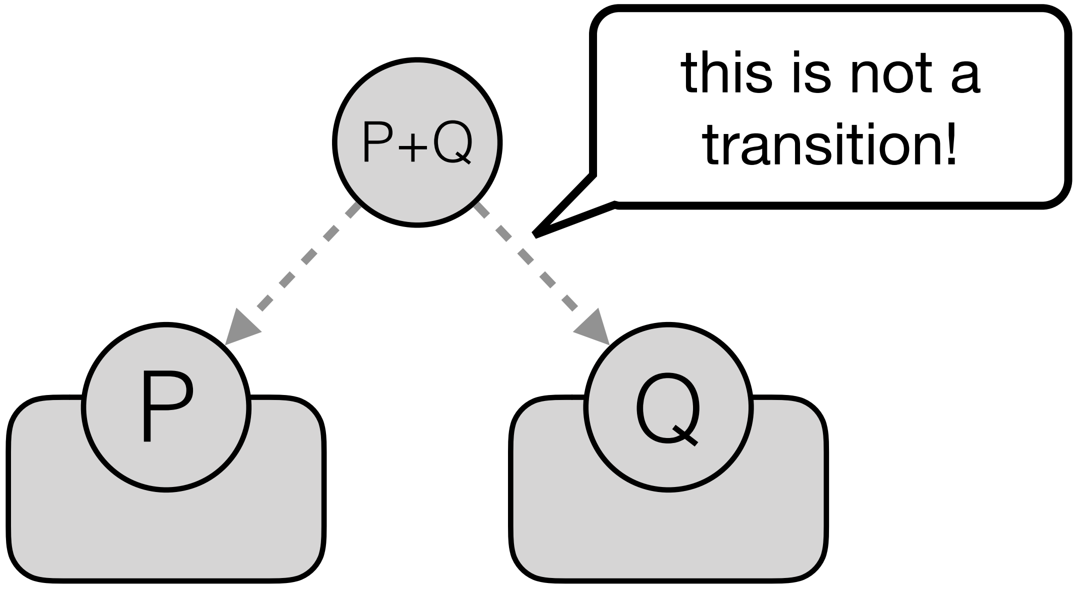
Choice

The choice operator acts as non-deterministic combinator.
The process P+Q has the capabilities of both P and Q.

Interface

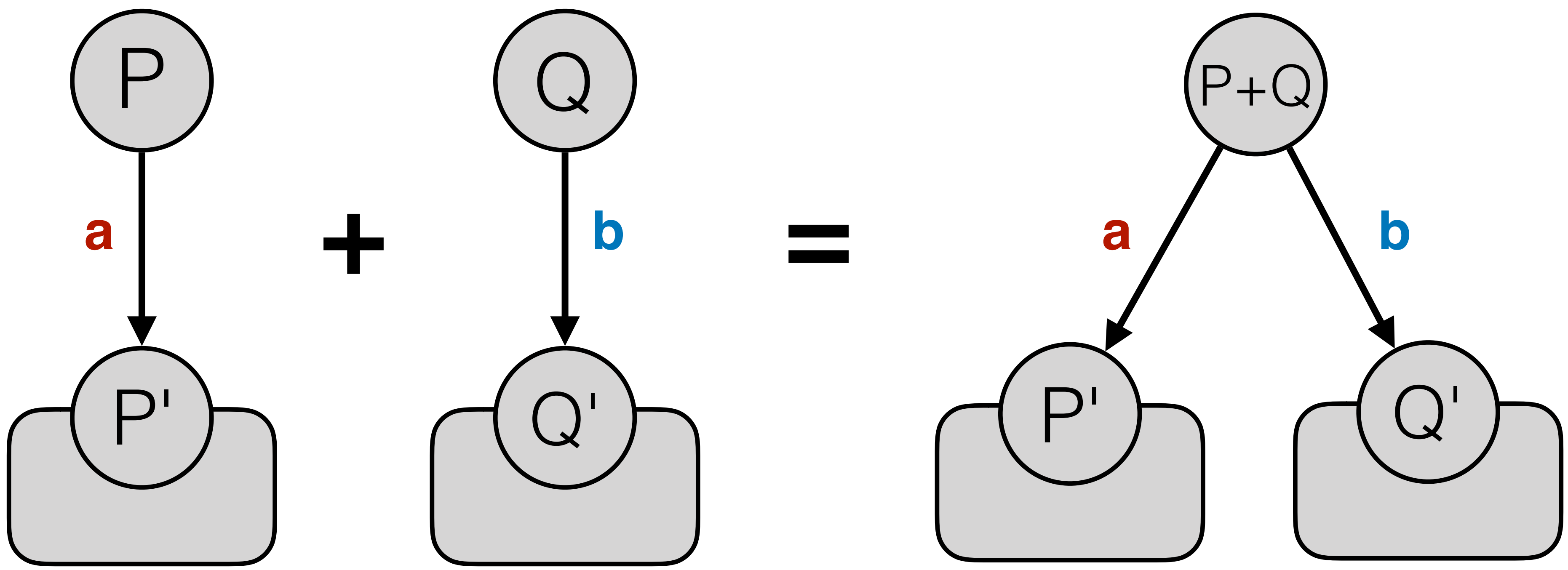


Behaviour (only the intuition)



Choice

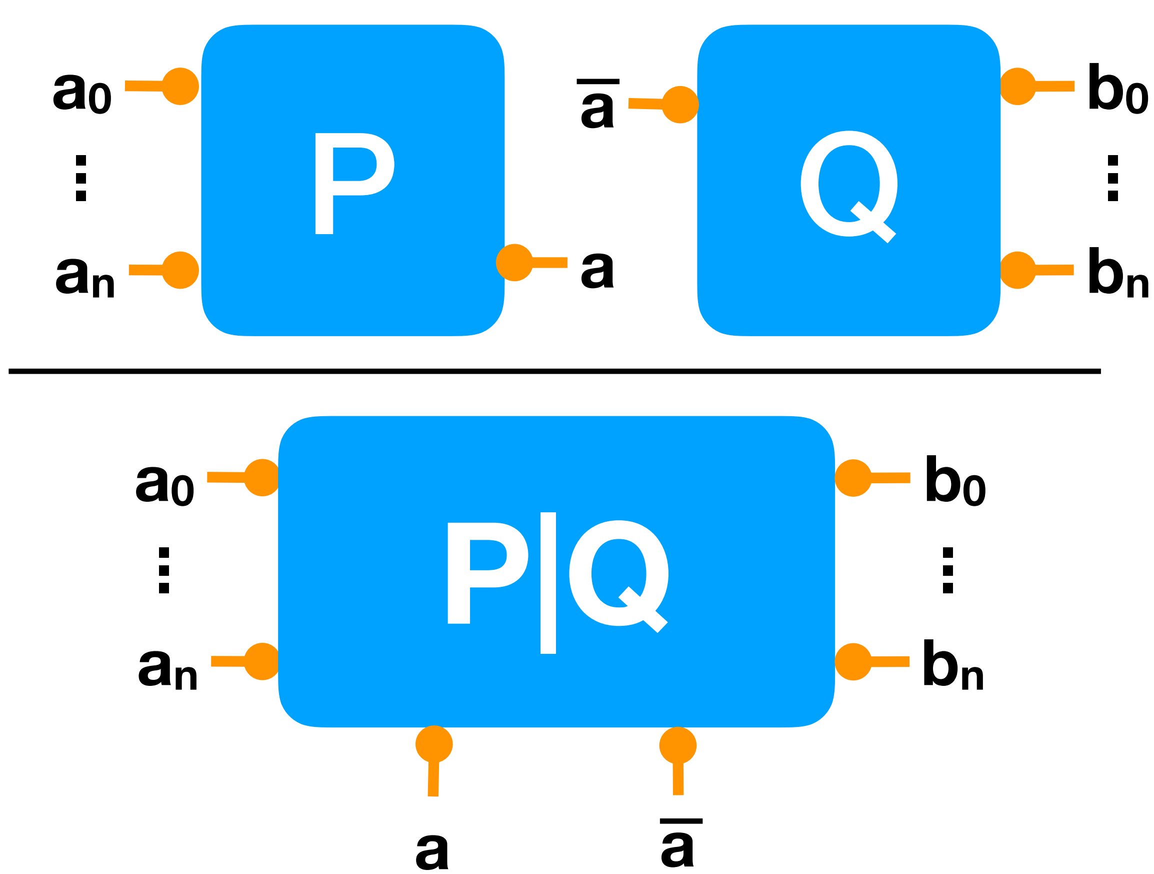
Behaviour (as an LTS - example)



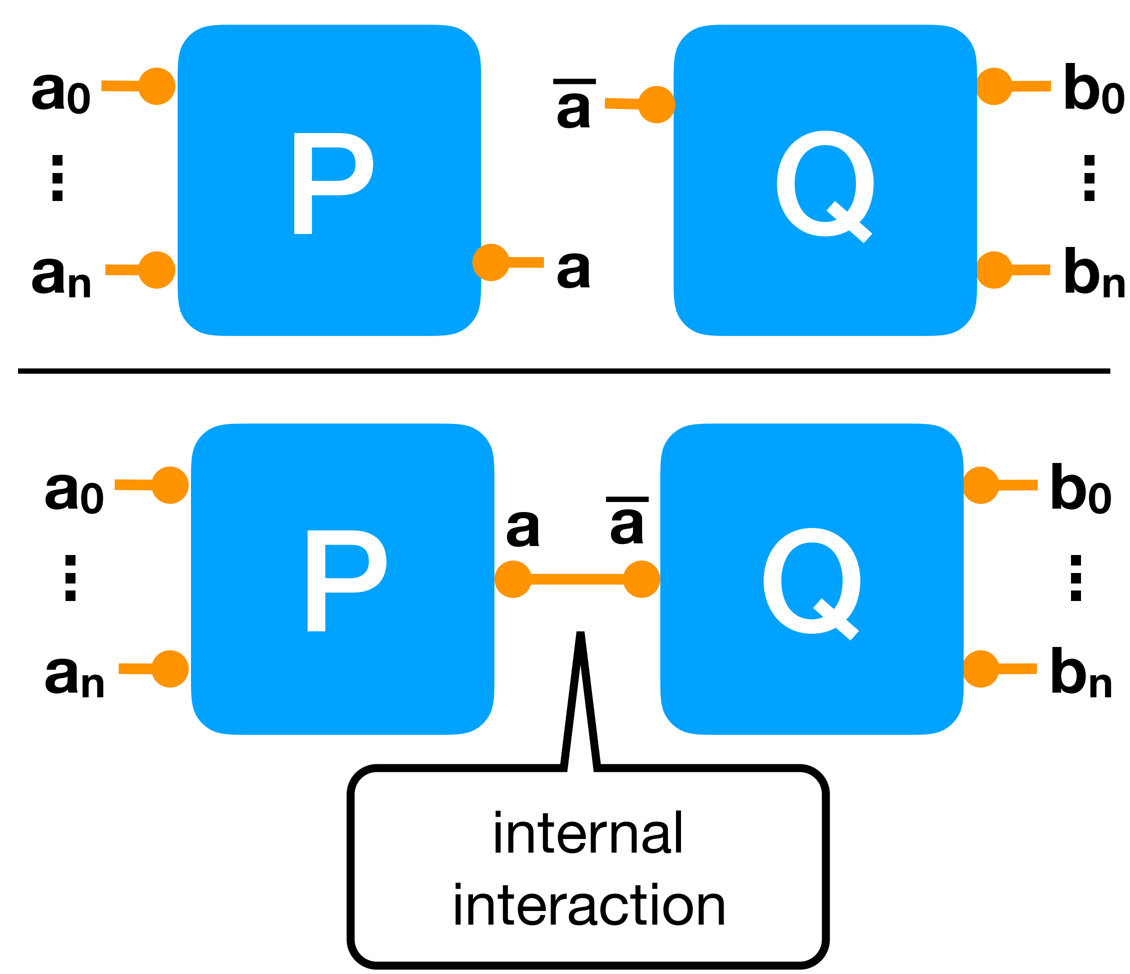
Parallel composition

The parallel composition offers the possibility of making the processes P and Q interacting with each other through synchronous communication

Interface

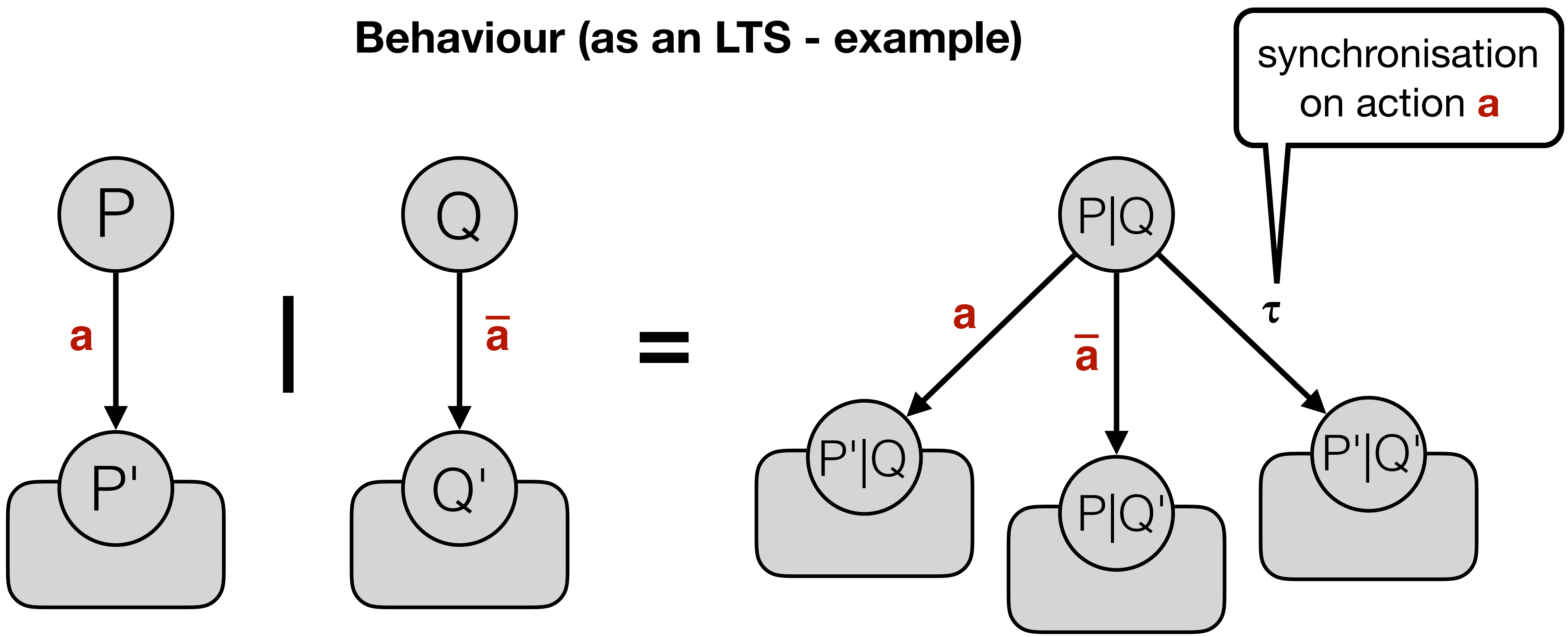


Interfaces (interaction view)



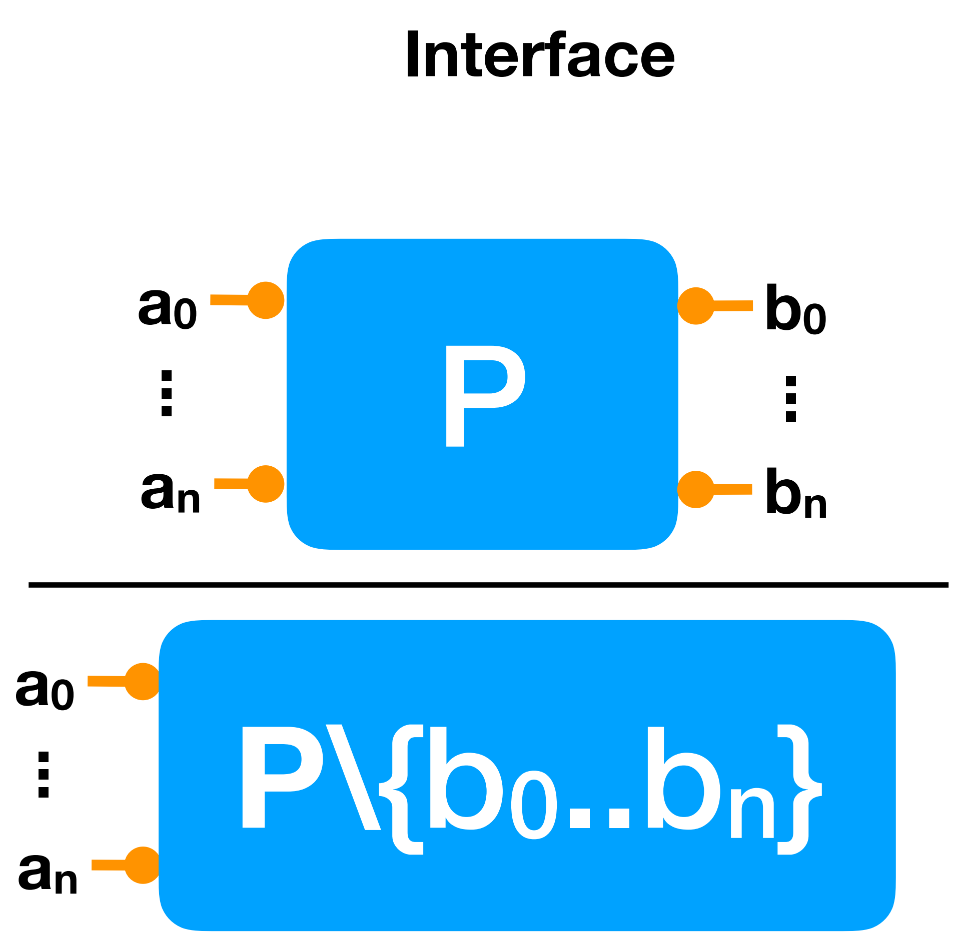
Parallel Composition

Behaviour (as an LTS - example)



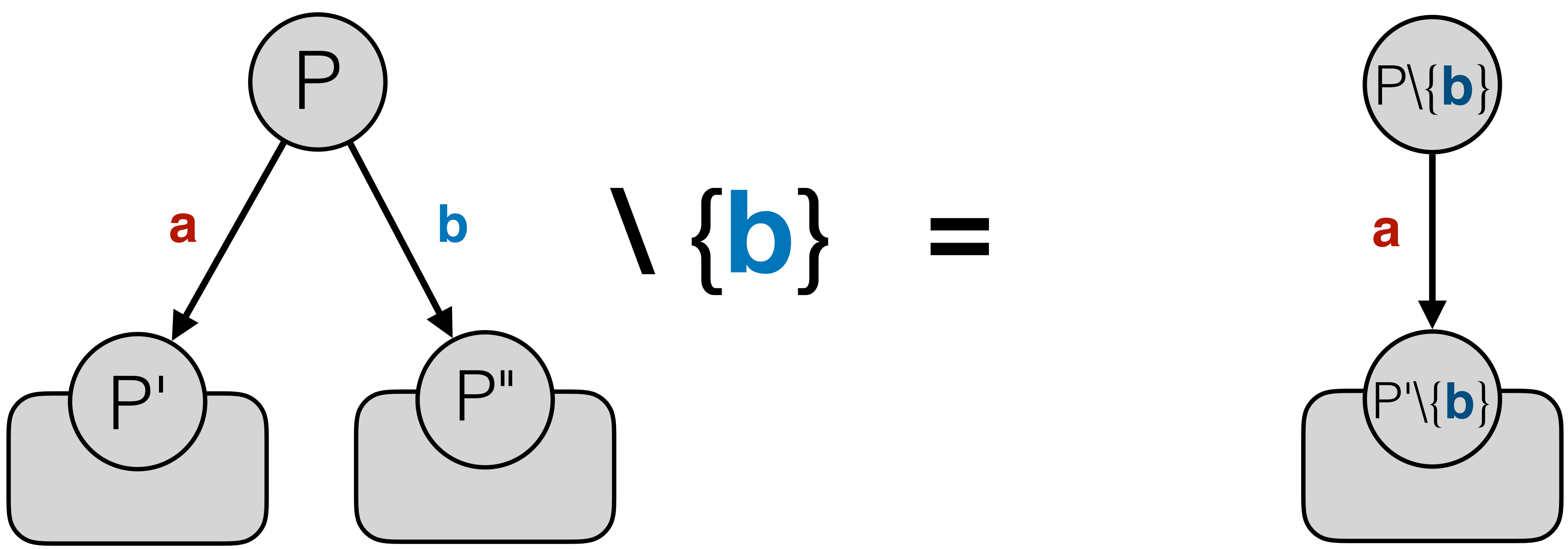
Restriction

The restriction operator allows one to restrict the visibility of specific communication ports (public vs private)



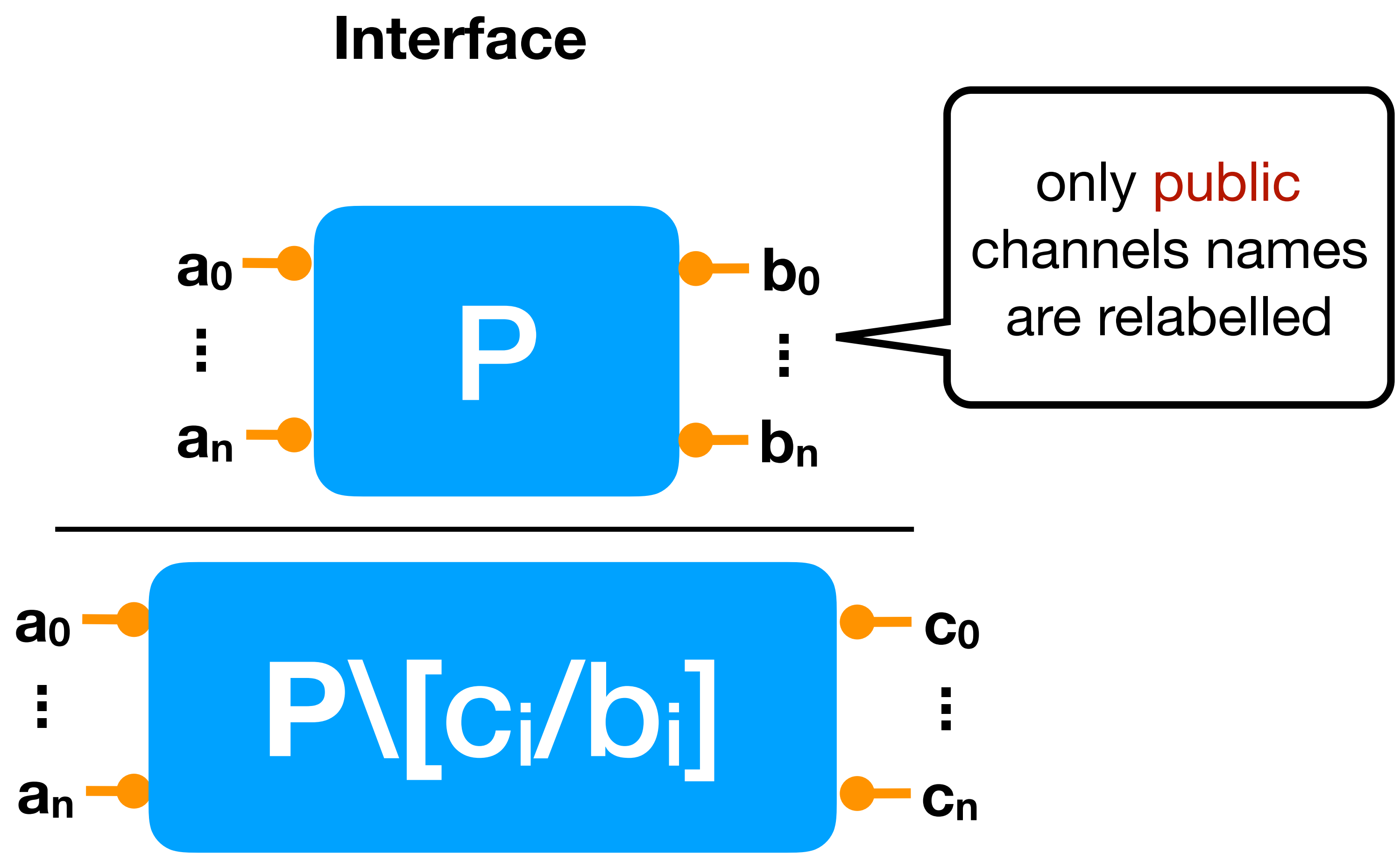
Restriction

Behaviour (as an LTS - example)



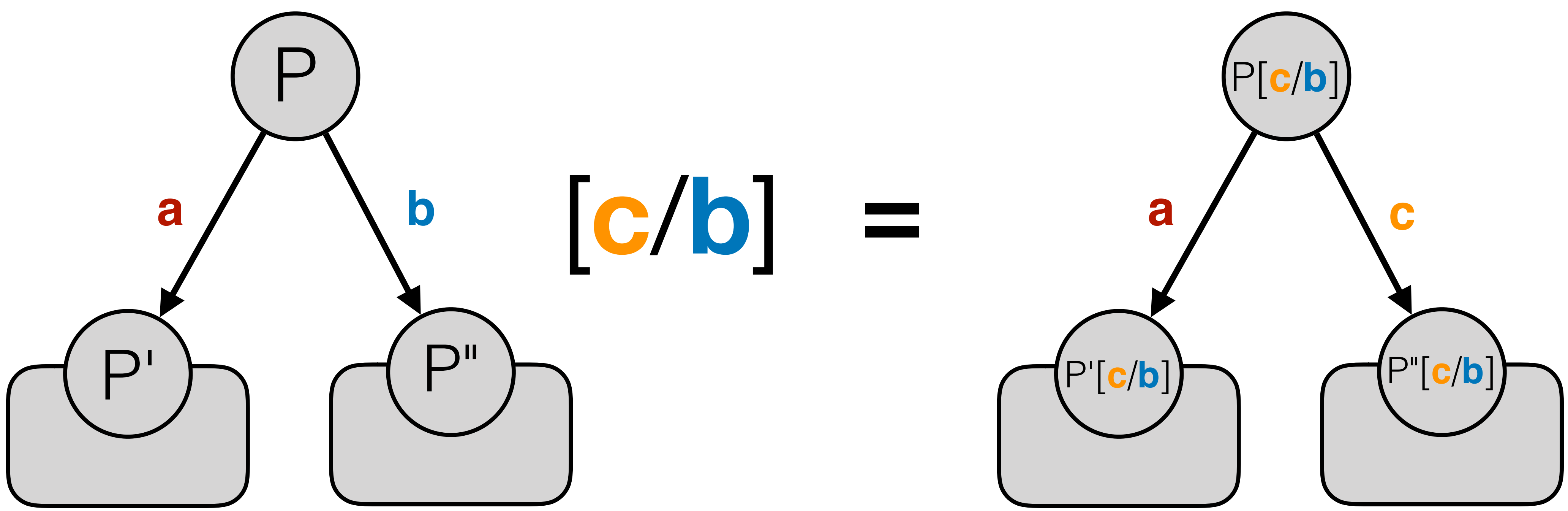
Relabelling

Relabelling gives the programmer an easy way to define "generic" process procedures



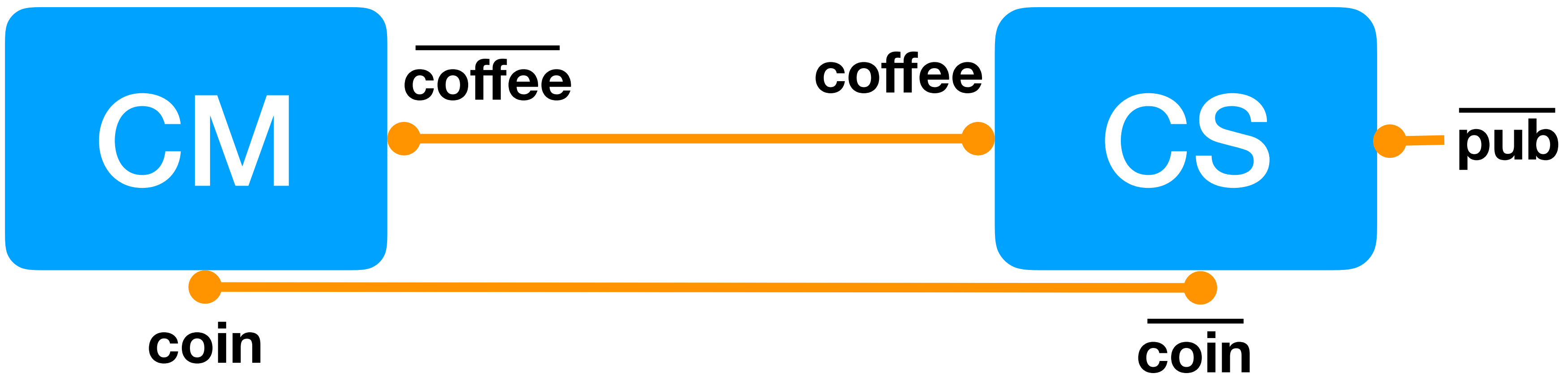
Relabelling

Behaviour (as an LTS - example)



break?

Example 1

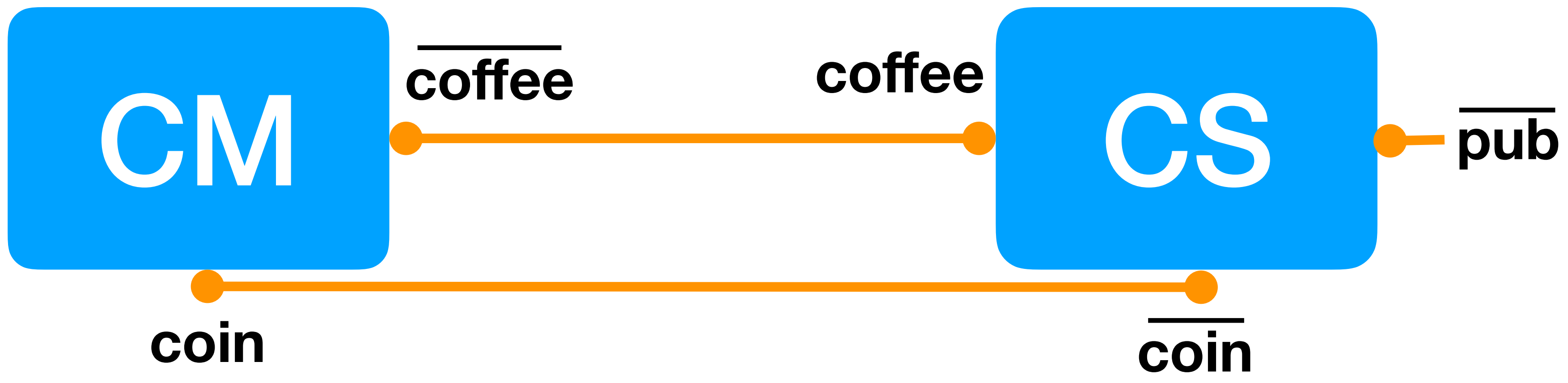


CM | CS

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

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

Example 1

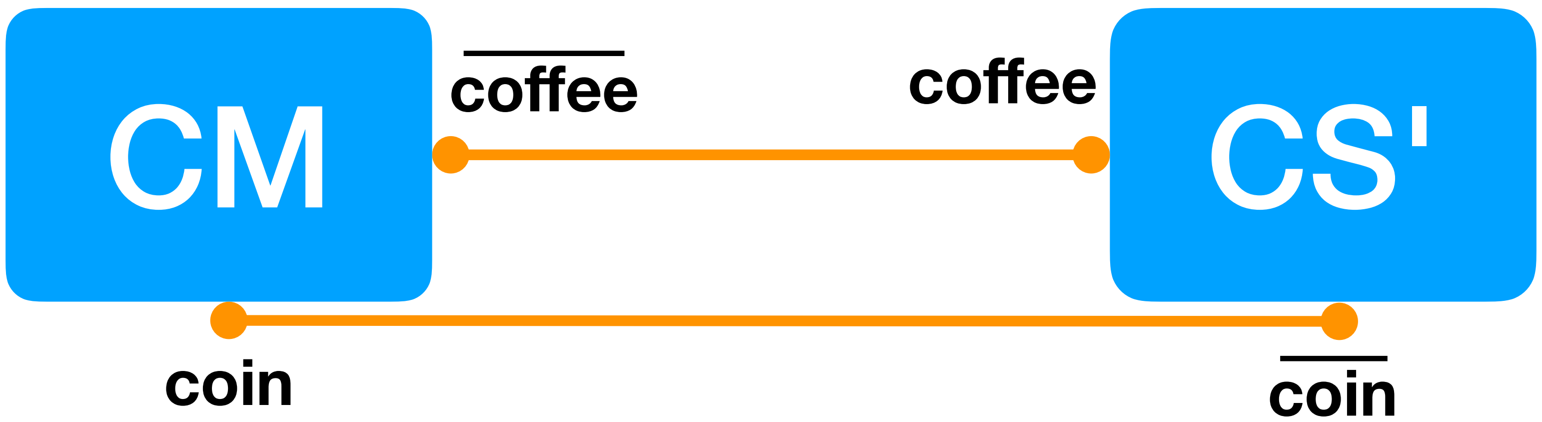


$$(\text{coin}.\overline{\text{coffee}}.\text{CM}) \mid (\overline{\text{pub}}.\overline{\text{coin}}.\text{coffee}.\text{CS})$$

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

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

Example 1



$$(\text{coin}.\overline{\text{coffee}}.\text{CM}) \mid (\overline{\text{pub}}.\overline{\text{coin}}.\text{coffee}.\text{CS})$$

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

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

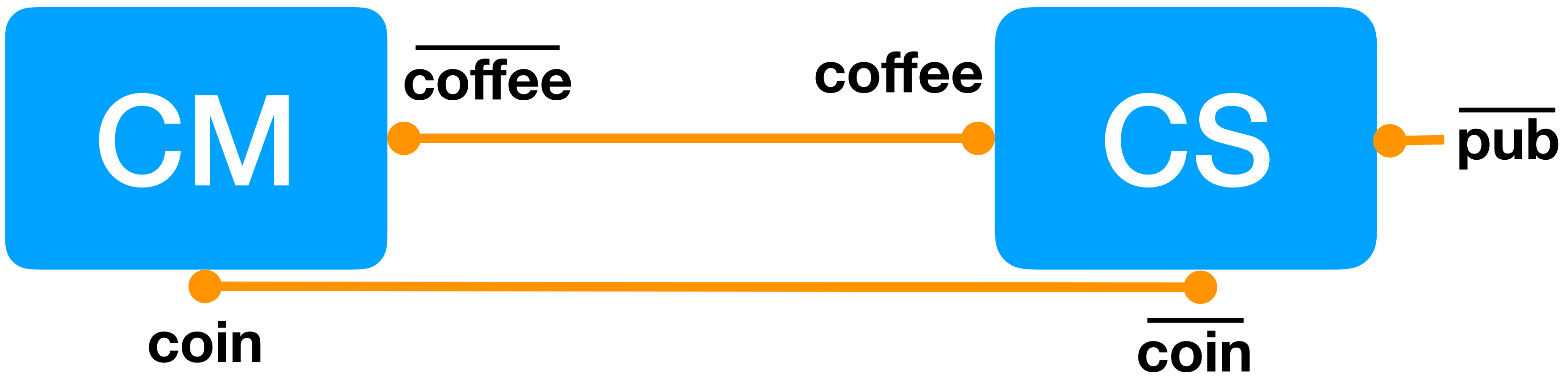
Example 1

$$(\text{coin}.\overline{\text{coffee}}.\text{CM}) \mid (\overline{\text{pub}}.\overline{\text{coin}}.\text{coffee}.\text{CS})$$

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

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

Example 1

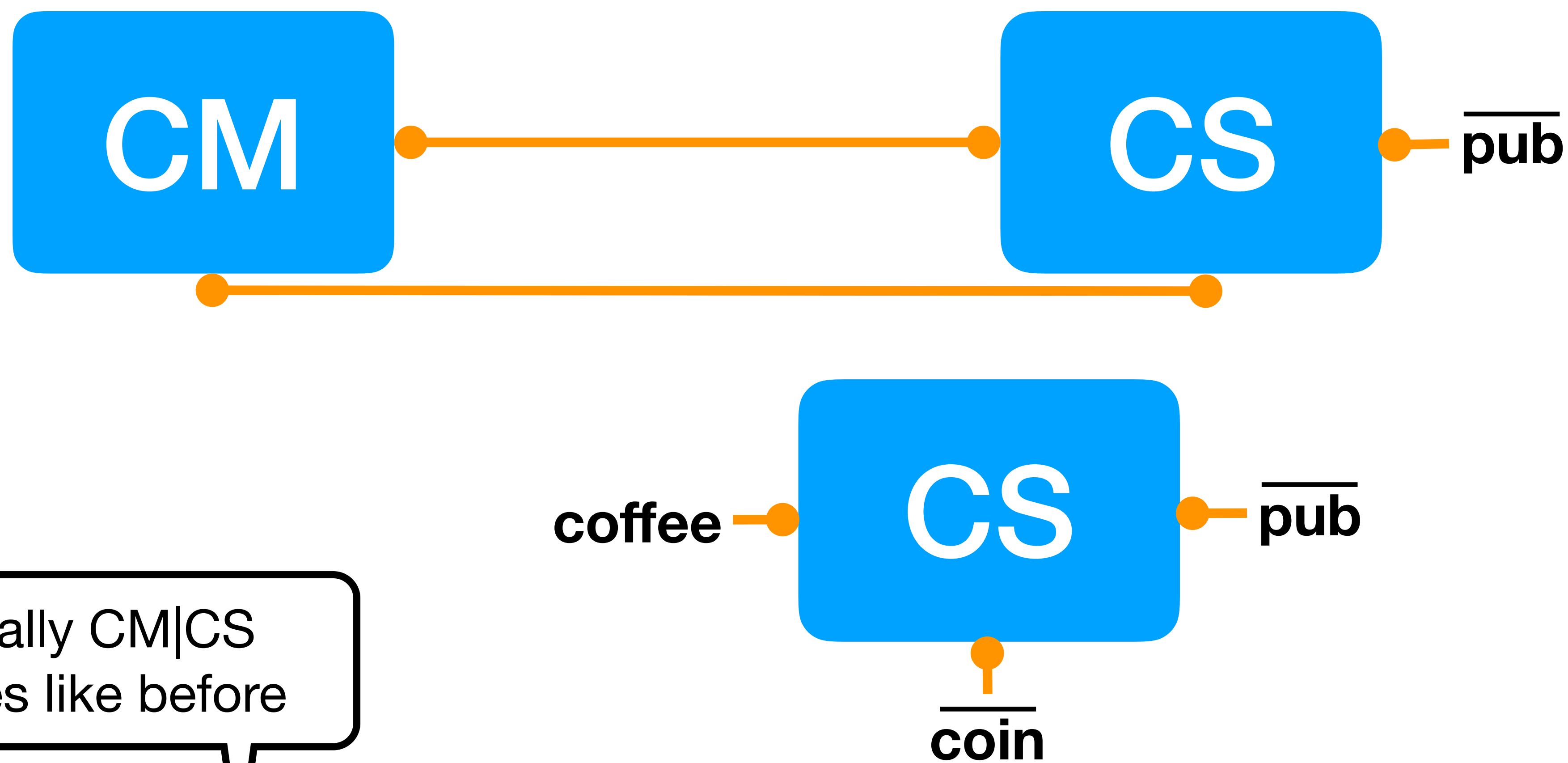


$$(\text{coin}.\overline{\text{coffee}}.\text{CM}) \mid (\overline{\text{pub}}.\overline{\text{coin}}.\text{coffee}.\text{CS})$$

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

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

Example 2



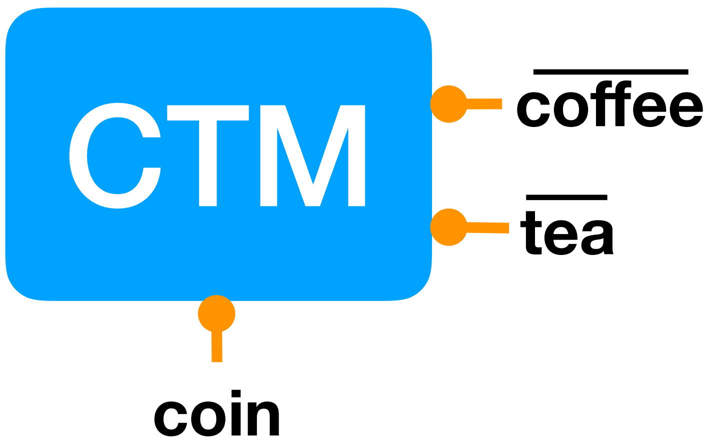
internally CM|CS
behaves like before

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

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

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

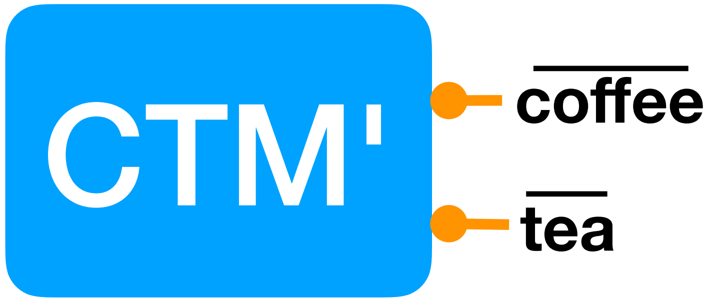
Example 3



$$\text{coin}.(\overline{\text{coffee}}.\text{CTM} + \overline{\text{tea}}.\text{CTM})$$

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

Example 3



$$\text{coin}.(\overline{\text{coffee}}.\text{CTM} + \overline{\text{tea}}.\text{CTM})$$

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

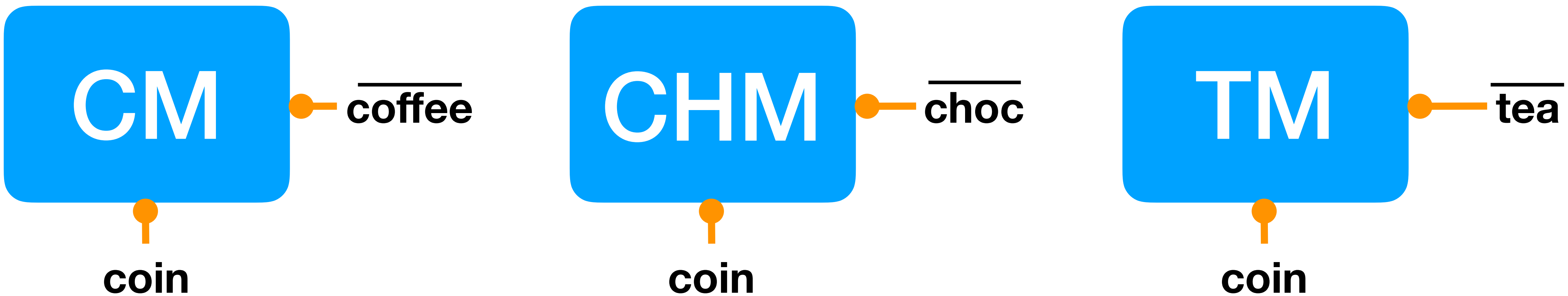
Example 3

only one behaviour is chosen!

$$\text{coin}.(\overline{\text{coffee}}.\text{CTM} + \overline{\text{tea}}.\text{CTM})$$

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

Example 4



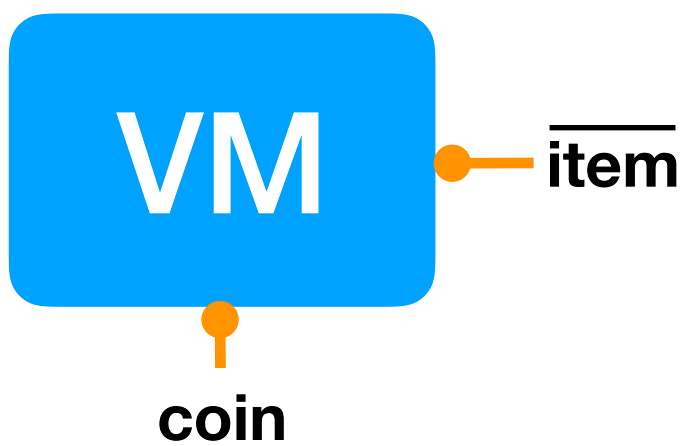
$$CM \stackrel{\text{def}}{=} coin.\overline{coffee}.CM$$

$$CHM \stackrel{\text{def}}{=} coin.\overline{choc}.CHM$$

$$TM \stackrel{\text{def}}{=} coin.\overline{tea}.TM$$

they have almost the same behaviour

Example 4



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

we can use renaming to generalise behaviours

$$CM \overset{def}{=} coin.\overline{coffee}.CM = VM[coffee/item]$$

$$CHM \overset{def}{=} coin.\overline{choc}.CHM = VM[choc/item]$$

$$TM \overset{def}{=} coin.\overline{tea}.TM = VM[tea/item]$$

formally!

CCS

Syntax & Semantics

Labels vs Actions

Let $\mathbb{A}$ be a set of *(channel) names*

Definition

Let $\bar{\mathbb{A}} = \{\bar{a} \mid a \in A\}$ be the set of *co-names*.

We let

- $\mathbb{L} = \mathbb{A} \cup \bar{\mathbb{A}}$ be the set of *labels*, and
- $Act = \mathbb{L} \cup \{\tau\}$ the set of *actions*,
  where τ is the *internal* (or *silent*) action

By convention, $\bar{\bar{a}} = a$.

A function $f : Act \to Act$ is a *relabelling function* if

$$f(\tau) = \tau \quad \text{and} \quad f(\bar{a}) = \overline{f(a)}$$

CCS Expressions

syntax

$P := K$	constant ($K \in \mathbb{K}$)
$\alpha.P$	prefixing ($\alpha \in Act$)
$\sum_{i \in I} P_i$	summation
$P \mid Q$	parallel composition
$P \setminus L$	restriction ($L \subseteq \mathbb{A}$)
$P[f]$	relabelling ($f : Act \to Act$)

Conventions

The set of all CCS expressions is denoted by $\mathcal{P}$

Notation

$P_1+P_2 = \sum_{i\in\{1,2\}} P_i$ $\quad$ $0 = \sum_{i\in\emptyset} P_i$

Precedence of operators

1. restriction & relabelling (tightest binding)
2. action prefixing
3. parallel composition
4. summation

CCS Programs

Definition

A *CCS program* is a set of *defining equations* of the form

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

where $K \in \mathbb{K}$ is a process constant and $P \in \mathcal{P}$ is a CCS process expression.

We assume that each constant process $K \in \mathbb{K}$ has a **unique** associated defining equation (note: recursion is allowed!)

Semantics of CCS

The behaviour of CCS processes, intuitively, is clear. However, intuition alone can lead us to wrong conclusions and, most importantly, cannot be fed into computers!

Definition

The formal semantics of processes is given in the form of the following LTS

$$(\text{Proc}, \text{Act}, \{\xrightarrow{\alpha} \mid \alpha \in \text{Act}\})$$

- set of *states* Proc = $\mathcal{P}$
- set of *actions* Act = $\mathbb{L} \cup \{\tau\}$
- and *transition relations* are given by SOS rules

SOS Rules for CCS(*)

$$(ACT)\quad \frac{}{\alpha.P \xrightarrow{\alpha} P}$$

$$(SUM_j)\quad \frac{P_j \xrightarrow{\alpha} P_j'}{\sum_{i\in I} P_i \xrightarrow{\alpha} P_j'} \quad \textit{where } j\in I$$

$$(COM1)\quad \frac{P \xrightarrow{\alpha} P'}{P|Q \xrightarrow{\alpha} P'|Q}$$

$$(COM2)\quad \frac{Q \xrightarrow{\alpha} Q'}{P|Q \xrightarrow{\alpha} P|Q'}$$

$$(COM3)\quad \frac{P \xrightarrow{a} P' \qquad Q \xrightarrow{\bar{a}} Q'}{P|Q \xrightarrow{\tau} P'|Q'}$$

$$(RES)\quad \frac{P \xrightarrow{\alpha} P'}{P\backslash L \xrightarrow{\alpha} P'\backslash L} \quad \textit{where } \alpha,\bar{\alpha}\notin L$$

$$(REL)\quad \frac{P \xrightarrow{\alpha} P'}{P[f] \xrightarrow{f(\alpha)} P'[f]}$$

$$(CON)\quad \frac{P \xrightarrow{\alpha} P'}{K \xrightarrow{\alpha} P'} \quad \textit{where } K \overset{def}{=} P$$

(*) We assume that $a\in\mathbb{A}$ is an arbitrary label and $\alpha\in Act$ an arbitrary action.

Deriving transitions

Let A = a.A. Then

$$((A \mid a.0) \mid b.0)[c/a] \xrightarrow{c} ((A \mid a.0) \mid b.0)[c/a]$$

Deriving transitions

Let A = a.A. Then

$$(REL)\ \frac{P \xrightarrow{\alpha} P'}{P[f] \xrightarrow{f(\alpha)} P'[f]}$$

$$(REL)\ \frac{(\,(A \mid a.0) \mid b.0\,) \xrightarrow{a} (\,(A \mid a.0) \mid b.0\,)}{(\,(A \mid a.0) \mid b.0\,)[c/a] \xrightarrow{c} (\,(A \mid a.0) \mid b.0\,)[c/a]}$$

Deriving transitions

Let A = a.A. Then

$$(COM1)\ \frac{P \xrightarrow{\alpha} P'}{P|Q \xrightarrow{\alpha} P'|Q}$$

$$(REL)\ \dfrac{(COM1)\ \dfrac{A \mid a.0 \xrightarrow{a} A \mid a.0}{(\ (A \mid a.0) \mid b.0\ ) \xrightarrow{a} (\ (A \mid a.0) \mid b.0\ )}}{(\ (A \mid a.0) \mid b.0\ )[c/a] \xrightarrow{c} (\ (A \mid a.0) \mid b.0\ )[c/a]}$$

Deriving transitions

Let A = a.A. Then

$$\text{(COM1)}\ \frac{P \xrightarrow{\alpha} P'}{P|Q \xrightarrow{\alpha} P'|Q}$$

$$\text{(REL)}\ \dfrac{\text{(COM1)}\ \dfrac{\text{(COM1)}\ \dfrac{A \xrightarrow{a} A}{A \mid a.0 \xrightarrow{a} A \mid a.0}}{(\,(A \mid a.0) \mid b.0\,) \xrightarrow{a} (\,(A \mid a.0) \mid b.0\,)}}{(\,(A \mid a.0) \mid b.0\,)[c/a] \xrightarrow{c} (\,(A \mid a.0) \mid b.0\,)[c/a]}$$

Deriving transitions

Let A = a.A. Then

$$\text{(CON)}\ \frac{P \xrightarrow{\alpha} P'}{K \xrightarrow{\alpha} P'}\ \textit{where } K \stackrel{def}{=} P$$

$$\text{(REL)}\ \dfrac{\text{(COM1)}\ \dfrac{\text{(COM1)}\ \dfrac{\text{(CON)}\ \dfrac{a.A \xrightarrow{a} A}{A \xrightarrow{a} A}\ A \stackrel{def}{=} a.A}{A \mid a.0 \xrightarrow{a} A \mid a.0}}{(\,(A \mid a.0) \mid b.0\,) \xrightarrow{a} (\,(A \mid a.0) \mid b.0\,)}}{(\,(A \mid a.0) \mid b.0\,)[c/a] \xrightarrow{c} (\,(A \mid a.0) \mid b.0\,)[c/a]}$$

Deriving transitions

Let A = a.A. Then

$$\textit{(ACT)}\ \frac{}{\alpha.P \xrightarrow{\alpha} P}$$

$$\textit{(REL)}\ \dfrac{\textit{(COM1)}\ \dfrac{\textit{(COM1)}\ \dfrac{\textit{(CON)}\ \dfrac{\textit{(ACT)}\ \dfrac{}{a.A \xrightarrow{a} A}}{A \xrightarrow{a} A}\quad A \overset{def}{=} a.A}{A \mid a.0 \xrightarrow{a} A \mid a.0}}{(\,(A \mid a.0) \mid b.0\,) \xrightarrow{a} (\,(A \mid a.0) \mid b.0\,)}}{(\,(A \mid a.0) \mid b.0\,)[c/a] \xrightarrow{c} (\,(A \mid a.0) \mid b.0\,)[c/a]}$$

The LTS of Processes