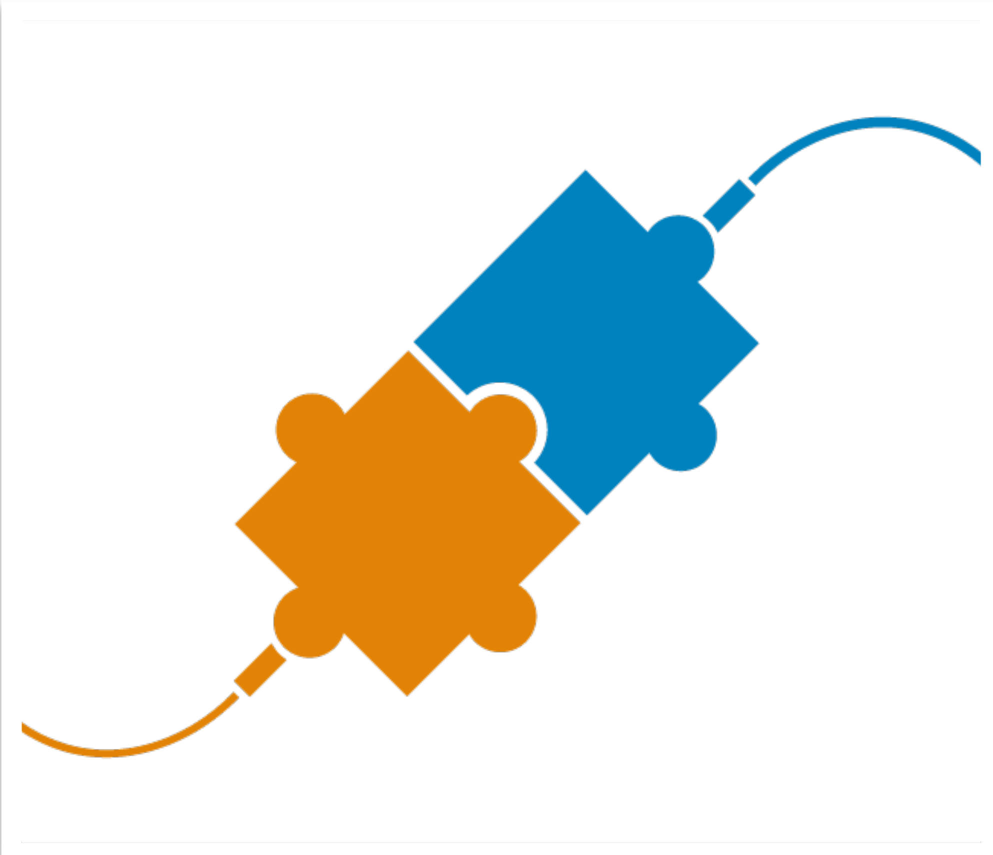


http://didawiki.di.unipi.it/doku.php/magistraleinformatica/psc/start

PSC 2023/24 (375AA, 9CFU)

Principles for Software Composition

Roberto Bruni
http://www.di.unipi.it/~bruni/

17a - CCS syntax & op. semantics

CCS
Calculus of Communicating Systems

Sequential vs concurrent

Concurrency

IMP/HOFL (sequential paradigms)
- determinacy
- any two non-terminating programs are equivalent

concurrent paradigms
- exhibit intrinsic nondeterminism to external observers
- nontermination can be a desirable feature (e.g. servers)
- not all nonterminating processes are equivalent
- interaction is a primary issue
- new notions of behaviour / equivalence are needed

CCS: basics

Process algebra

- focus on few primitive operators (essential features)
- concise syntax to construct and compose processes
- not a full-fledged programming language
- full computational power (Turing equivalent)

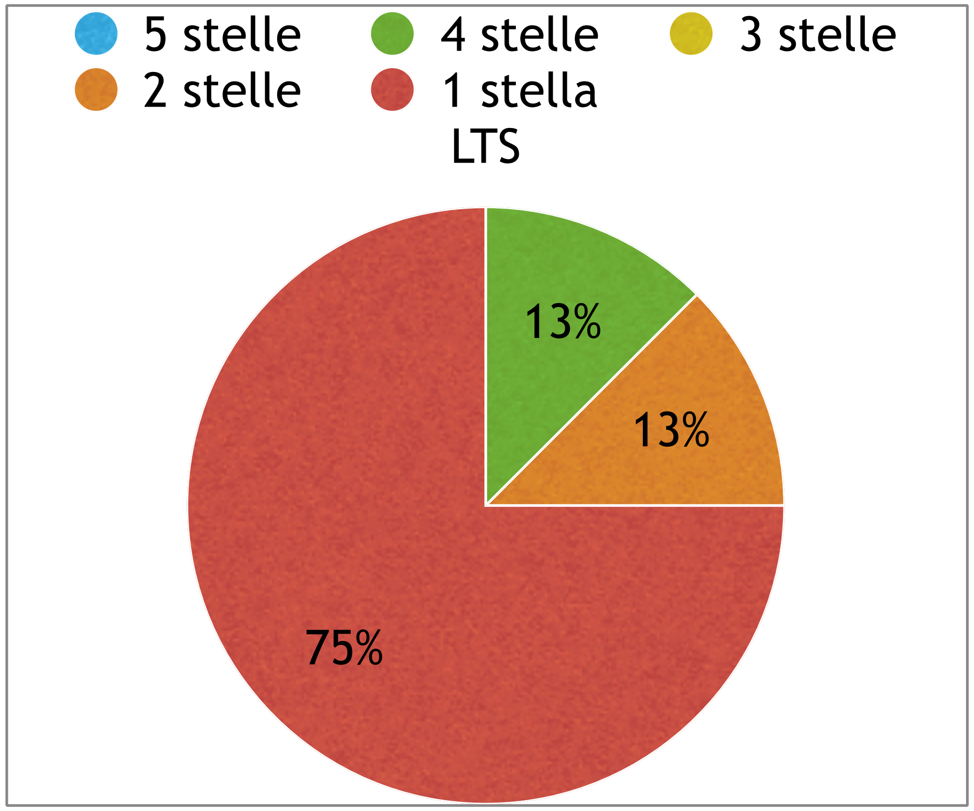
Communication

- binary, message-passing over channels

Structural Operational Semantics

- small-step style (Labelled Transition System)
- processes as states
- ongoing interactions as labels
- defined by inference rules
- defined by induction on the structure of processes

From your forms



(over 8 answers)

Labelled transitions

ongoing interaction
with the environment
(with other processes)

$$p \xrightarrow{\mu} q$$

a process
in its current state

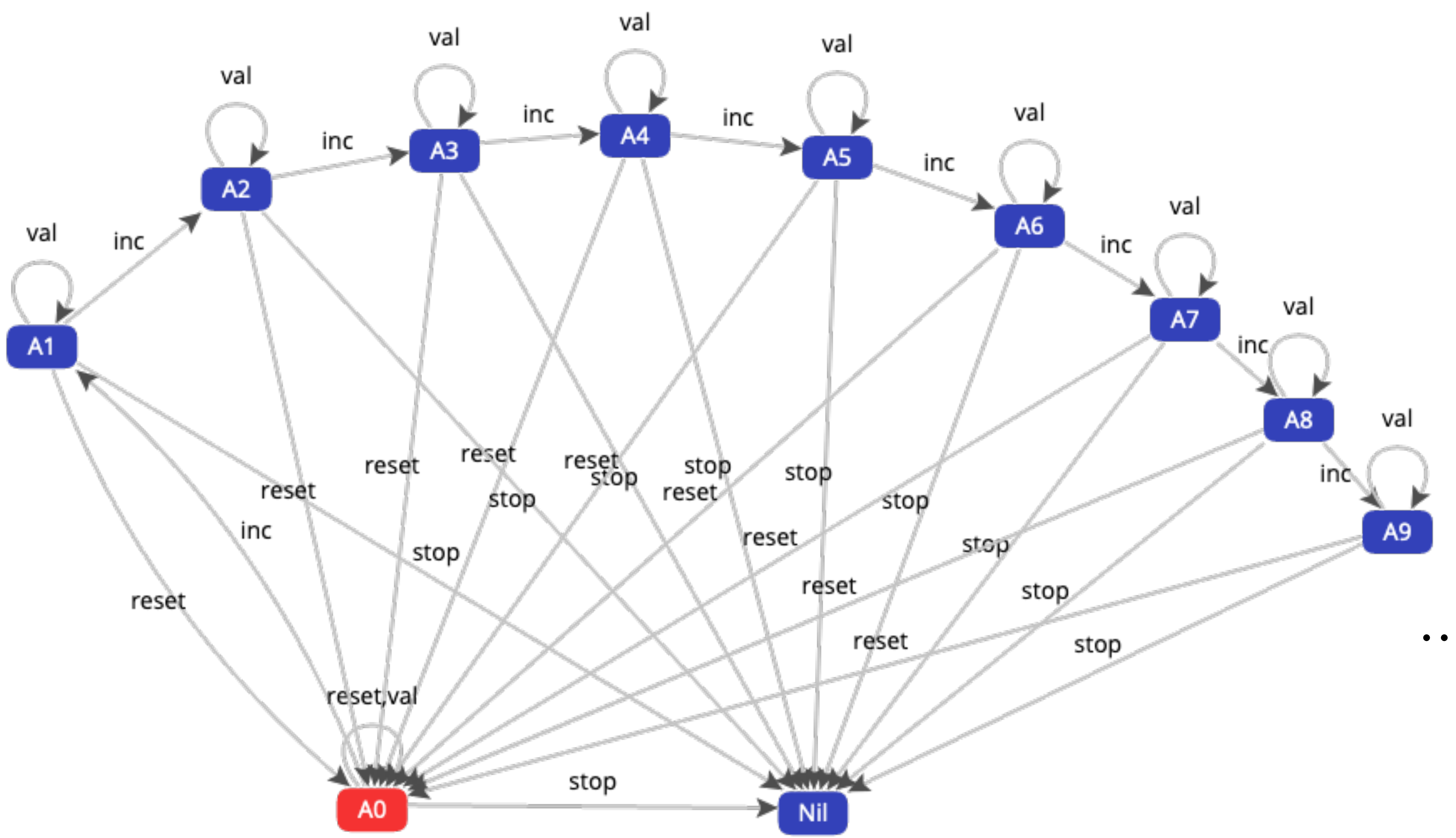
the process
state after the
interaction

number of states/transitions
can be infinite

Example: counter

$$\begin{tikzcd}
A_0 & \cdots & A_n \arrow[ll, bend left, "\mathsf{reset}"] \arrow[loop above, "\mathsf{val}"] \arrow[r, "\mathsf{inc}"] \arrow[d, "\mathsf{stop}"] & A_{n+1} & \cdots \\
 & & \mathsf{Nil} & &
\end{tikzcd}$$

LTS: Labelled Transition System



CCS: states and labels

What is a process p ?

a sequential agent
a system where many sequential agents interact

What is a label μ ?

send v on channel α

an action (e.g. an output) $\alpha!v$
a dual action (e.g. an input) $\alpha?v$

receive v on channel α

an internal action (silent action) τ
(no interaction with the environment)

concluded communication

CCS: actions & coactions

We can be even more abstract than that
without losing computational expressiveness

we disregard communicated values
(imagine there is a dedicated channel for each value)

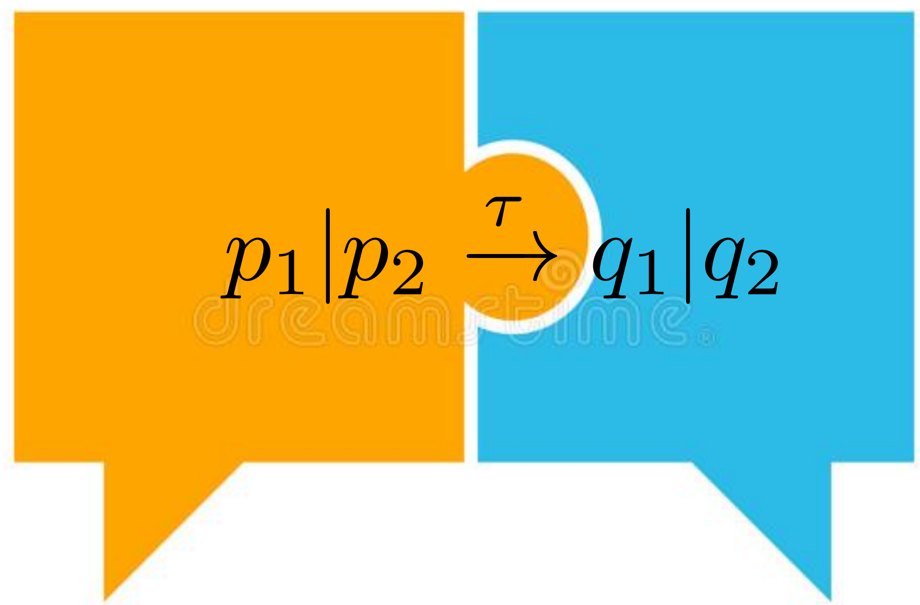
$\alpha!v$ becomes just $\overline{\alpha_v}$ or just $\overline{\alpha}$

$\alpha?v$ becomes just α_v or just α

λ denotes either α $\overline{\alpha}$

$\overline{\lambda}$ denotes its dual (assume $\overline{\overline{\alpha}} = \alpha$)

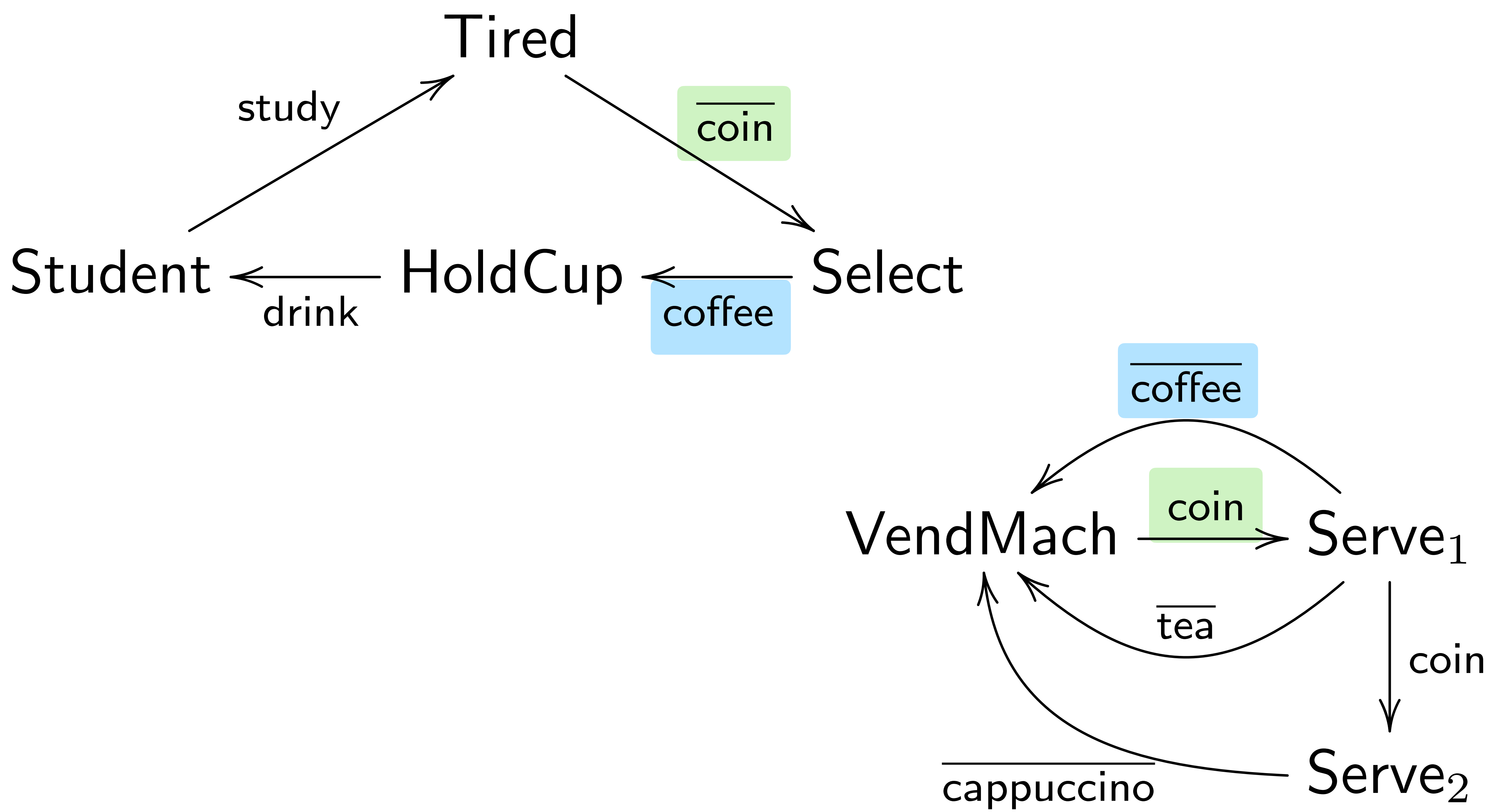
CCS: communication



$$p_1|p_2 \xrightarrow{\tau} q_1|q_2$$

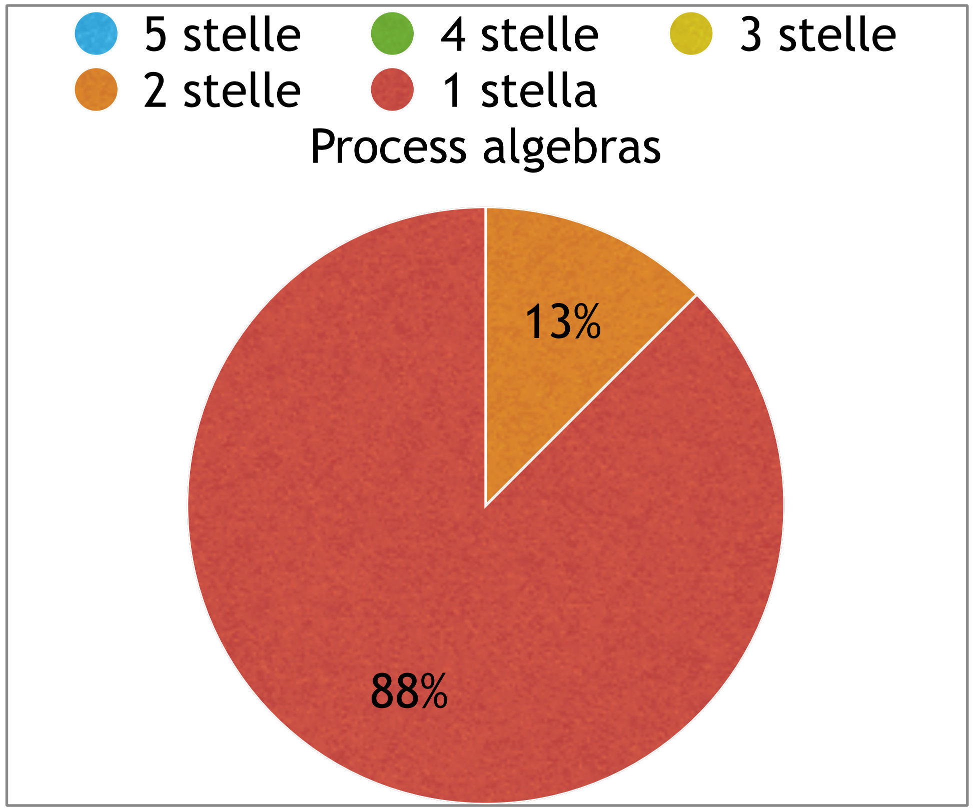
$$p_1 \xrightarrow{\lambda} q_1 \qquad\qquad p_2 \xrightarrow{\overline{\lambda}} q_2$$

Example: vending machine



CCS syntax

From your forms



(over 8 answers)

CCS: syntax

p, q	$::=$	$\mathbf{nil}$	inactive process
	$\mid$	x	process variable (for recursion)
	$\mid$	$\mu.p$	action prefix
	$\mid$	$p\backslash\alpha$	restricted channel
	$\mid$	$p[\phi]$	channel relabelling
	$\mid$	$p + q$	nondeterministic choice (sum)
	$\mid$	$p \vert q$	parallel composition
	$\mid$	$\mathbf{rec}\ x.\ p$	recursion

(operators are listed in order of precedence)

CCS: syntax

$$
\begin{array}{rcl}
p, q & ::= & \mathbf{nil} \\
& | & x \\
& | & \mu.p \\
& | & p\backslash\alpha \\
& | & p[\phi] \\
& | & p + q \\
& | & p|q \\
& | & \mathbf{rec}\ x.\ p
\end{array}
$$

$$\mathbf{rec}\ x.\ \mathsf{coffee}.x + \mathsf{tea}.\mathbf{nil} \,|\, \mathsf{water}.\mathbf{nil}$$

to be read as

$$\mathbf{rec}\ x.\ (((\mathsf{coffee}.x) + \mathsf{tea}.\mathbf{nil}) \,|\, \mathsf{water}.\mathbf{nil})$$

(operators are listed in order of precedence)

CCS: syntax

the only binder is the recursion operator

$$\mathbf{rec}\ x.\ p$$

the notion of free (process) variable is defined as usual

$$\mathsf{fv}(p)$$

a process is called *closed* if it has no free variables

the notion of capture avoiding substitution is defined as usual

$$p[{}^{q}/_{x}]$$

processes are taken up-to alpha-renaming of bound vars

$$\mathbf{rec}\ x.\ \mathsf{coin}.x = \mathbf{rec}\ y.\ \mathsf{coin}.y$$

CCS operational semantics

CCS: labels

$\mathcal{C}$ set of (input) actions, ranged by α

$\overline{\mathcal{C}}$ set of (output) co-actions, ranged by $\overline{\alpha}$

$$\mathcal{C} \cap \overline{\mathcal{C}} = \emptyset$$

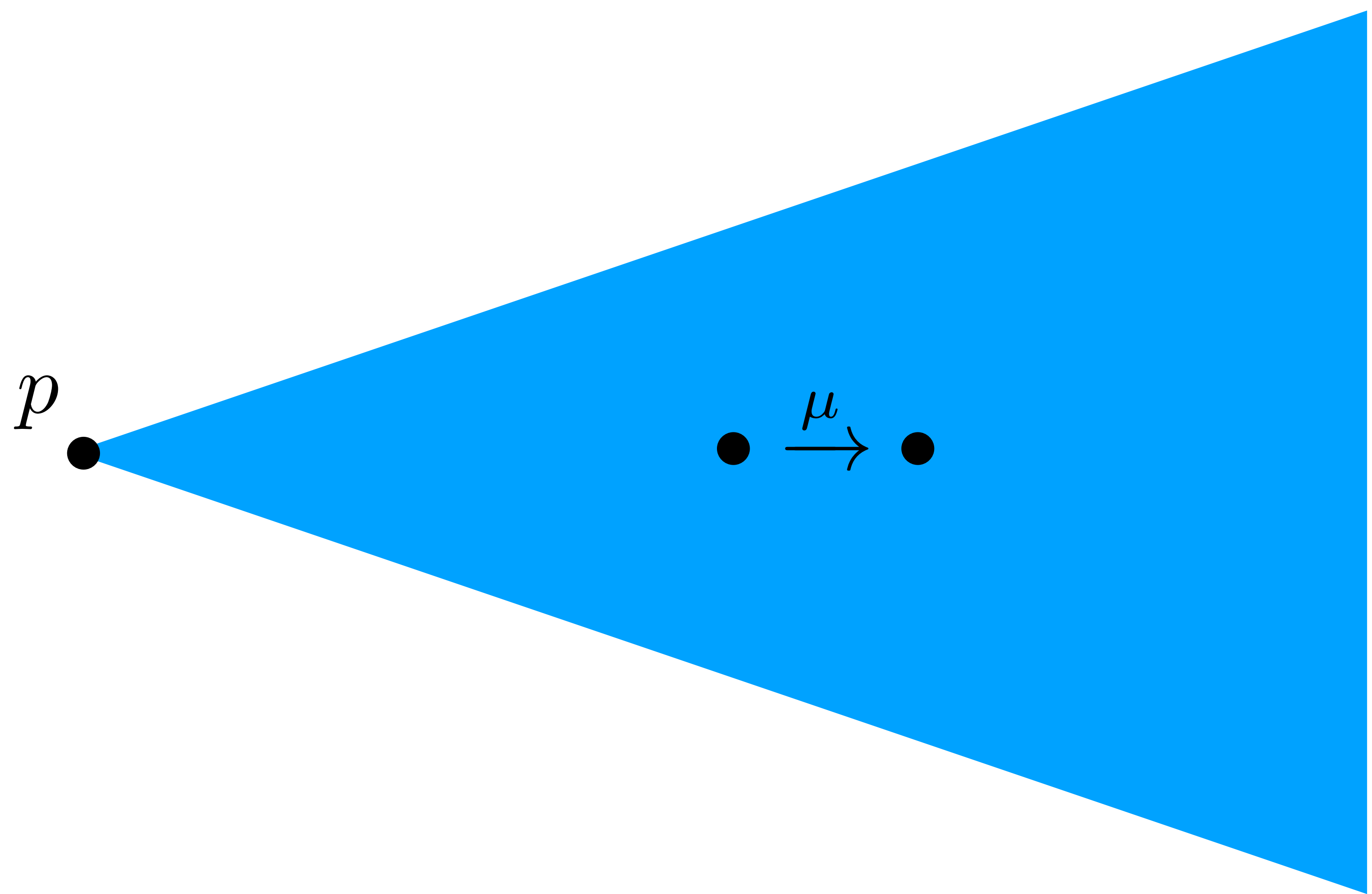
$\Lambda = \mathcal{C} \cup \overline{\mathcal{C}}$ set of observable actions, ranged by λ $\overline{\lambda}$

$\tau \notin \Lambda$ a distinguished silent action

$\mathcal{L} = \Lambda \cup \{\tau\}$ set of actions, ranged by μ

LTS of a process

the LTS of CCS is infinite (one state for each process)



starting from p , consider all reachable states:

the LTS of a process can be finite/infinite

Nil process

$$\mathbf{nil} \not\rightarrow$$

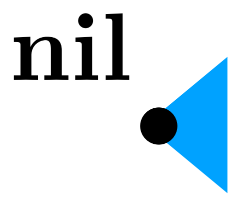
the inactive process does nothing

no interaction is possible with the environment

represents a terminated agent

no operational semantics rule associated with **nil**

LTS of a process



Action prefix

$$\text{Act)} \ \frac{}{\mu.p \xrightarrow{\mu} p}$$

an action prefixed process can perform the action
and continue as expected

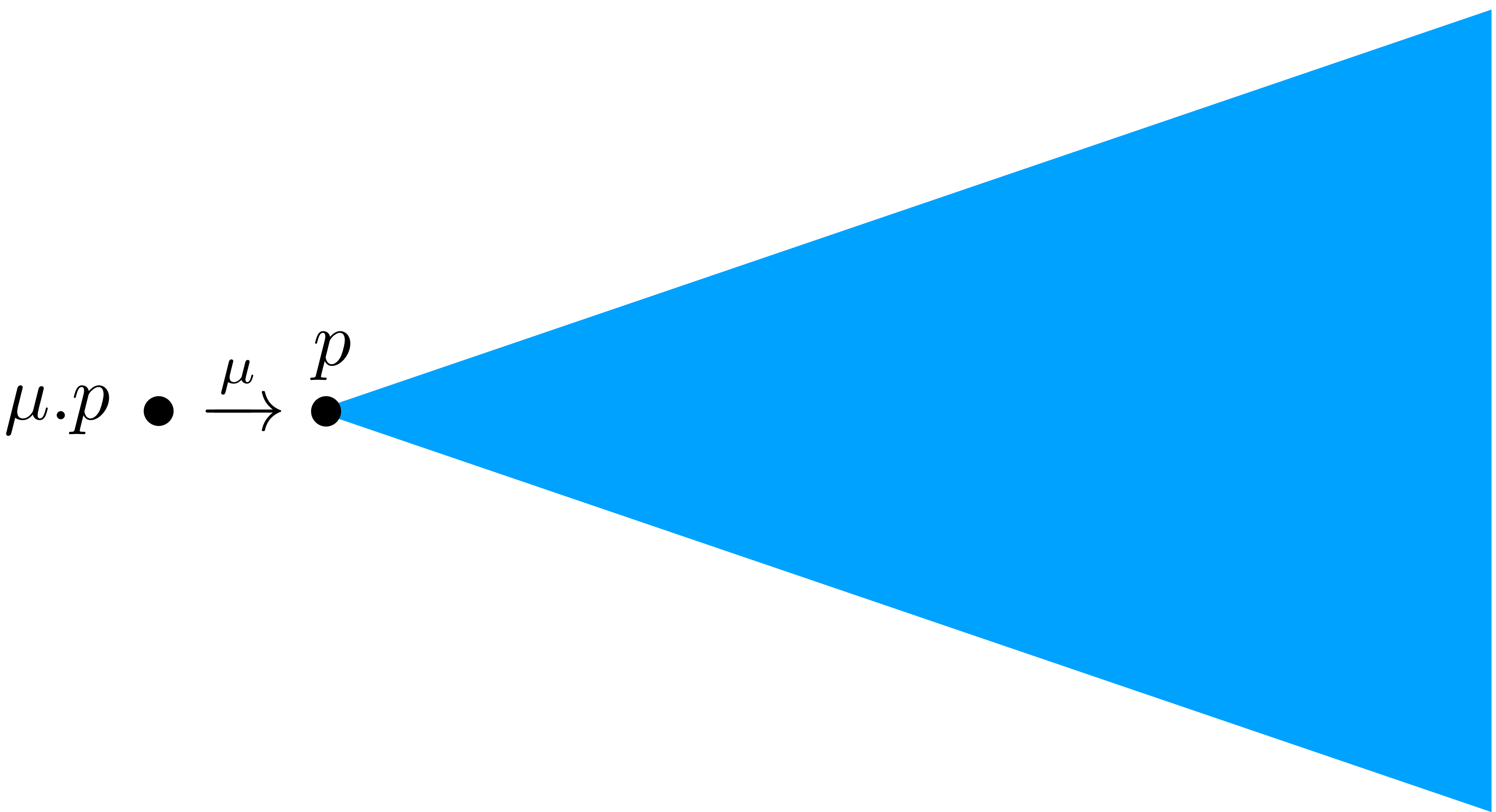
the action may involve an interaction with the environment

$$coin.\overline{coffee}.\mathbf{nil}$$

waits a coin, then gives a coffee and then it stops

$$coin.\overline{coffe}.\mathbf{nil} \xrightarrow{coin} \overline{coffe}.\mathbf{nil} \xrightarrow{\overline{coffe}} \mathbf{nil}$$

LTS of a process



Nondeterministic choice

$$\text{SumL)}\ \frac{p_1 \xrightarrow{\mu} q}{p_1 + p_2 \xrightarrow{\mu} q} \qquad \text{SumR)}\ \frac{p_2 \xrightarrow{\mu} q}{p_1 + p_2 \xrightarrow{\mu} q}$$

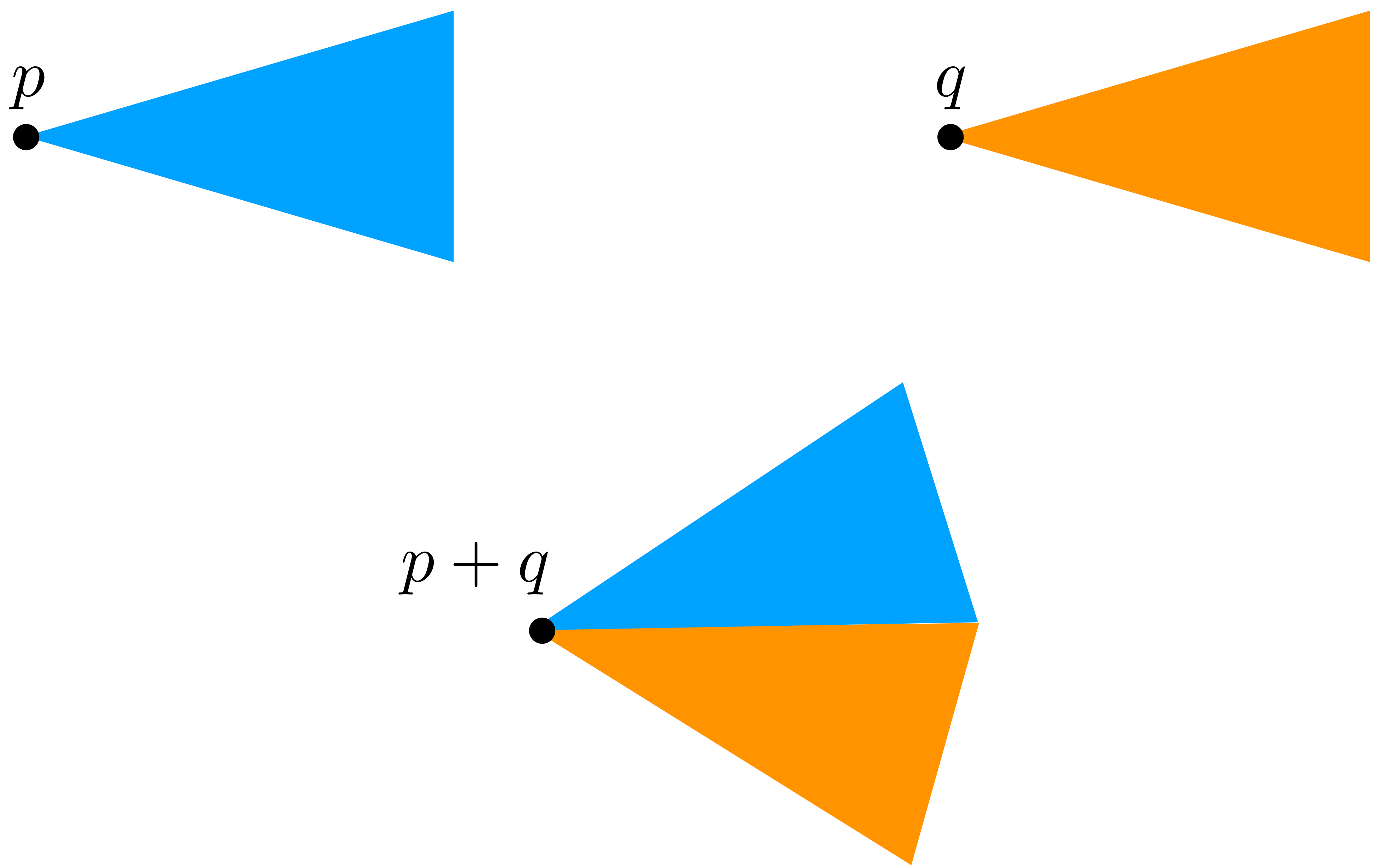
process $p_1 + p_2$ can behave either as p_1 or as p_2

$$coin.(\overline{coffee}.\mathbf{nil} + \overline{tea}.\mathbf{nil})$$

waits a coin, then gives a coffee or a tea, then it stops

$$\begin{tikzcd}
coin.(\overline{coffee}.\mathbf{nil} + \overline{tea}.\mathbf{nil}) \arrow[d, "coin"] \\
\overline{coffee}.\mathbf{nil} + \overline{tea}.\mathbf{nil} \arrow[d, bend right, "\overline{coffee}"'] \arrow[d, bend left, "\overline{tea}"] \\
\mathbf{nil}
\end{tikzcd}$$

LTS of a process



Recursion

$$\text{Rec)}\quad \frac{p[^{\mathbf{rec}\ x.\ p}/_x] \xrightarrow{\mu} q}{\mathbf{rec}\ x.\ p \xrightarrow{\mu} q}$$

like a recursive definition let $x = p$ in x

$$\mathbf{rec}\ x.\ coin.(\overline{coffee}.x + \overline{tea}.\mathbf{nil})$$

waits a coin, then gives a coffee and is ready again
or a tea and stops

$$\begin{tikzcd}
\mathbf{rec}\ x.\ coin.(\overline{coffee}.x + \overline{tea}.\mathbf{nil}) \arrow[d, "coin"] \\
\overline{coffee}.(\mathbf{rec}\ x.\ coin.(\overline{coffee}.x + \overline{tea}.\mathbf{nil})) + \overline{tea}.\mathbf{nil} \arrow[d, "\overline{tea}"] \\
\mathbf{nil}
\end{tikzcd}$$

$$\begin{tikzcd}
P \triangleq \mathbf{rec}\ x.\ coin.(\overline{coffee}.x + \overline{tea}.\mathbf{nil}) \arrow[d, bend left, "coin"] \\
\overline{coffee}.P + \overline{tea}.\mathbf{nil} \arrow[u, bend left, "\overline{coffee}"] \arrow[d, "\overline{tea}"] \\
\mathbf{nil}
\end{tikzcd}$$

Recursion via process constants

imagine some process constants A are available together with a set Δ of declarations of the form

$$A \triangleq p$$

one for each constant

$$\text{Const)}\ \frac{A \triangleq p \in \Delta \quad p \xrightarrow{\mu} q}{A \xrightarrow{\mu} q}$$

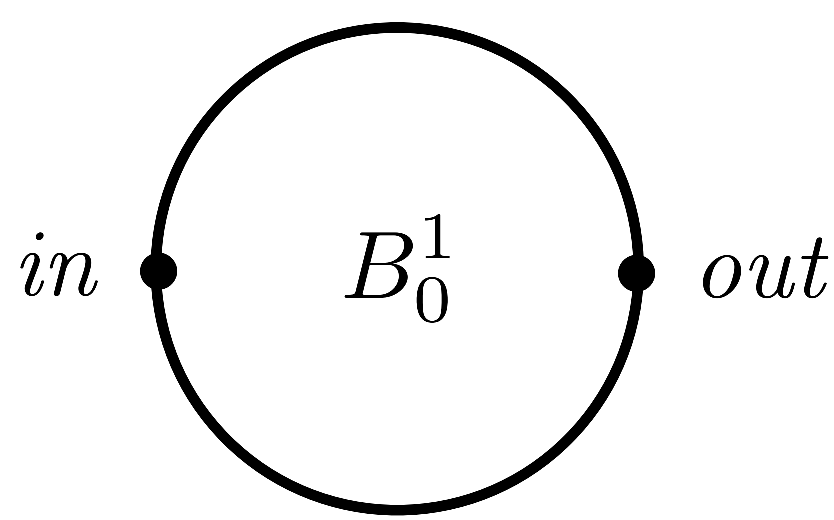
$$P \triangleq coin.(\overline{coffee}.P + \overline{tea}.\mathbf{nil})$$

CCS: capacity 1 buffer

$$\Delta = \{\ B_0^1 \triangleq in.B_1^1\ ,\ B_1^1 \triangleq \overline{out}.B_0^1\ \}$$

$$\begin{tikzcd} B_0^1 \arrow[d, bend left, "in"] \\ B_1^1 \arrow[u, bend left, "\overline{out}"] \end{tikzcd}$$

$$\mathbf{rec}\ x.\ in.\overline{out}.x$$



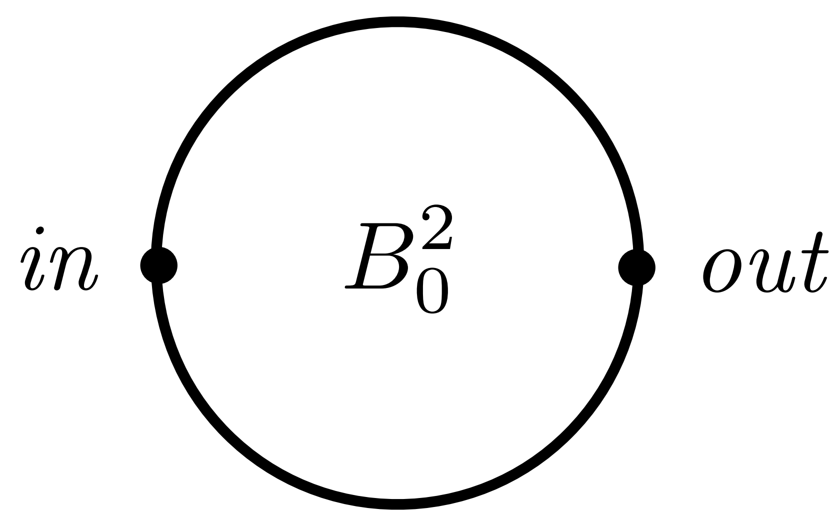
CCS: capacity 2 buffer

$$B_0^2 \triangleq in.B_1^2$$

$$B_1^2 \triangleq in.B_2^2 + \overline{out}.B_0^2$$

$$B_2^2 \triangleq \overline{out}.B_1^2$$

$$\begin{tikzcd}
B_0^2 \arrow[d, bend left, "in"] \\
B_1^2 \arrow[u, bend left, "\overline{out}"] \arrow[d, bend left, "in"] \\
B_2^2 \arrow[u, bend left, "\overline{out}"]
\end{tikzcd}$$

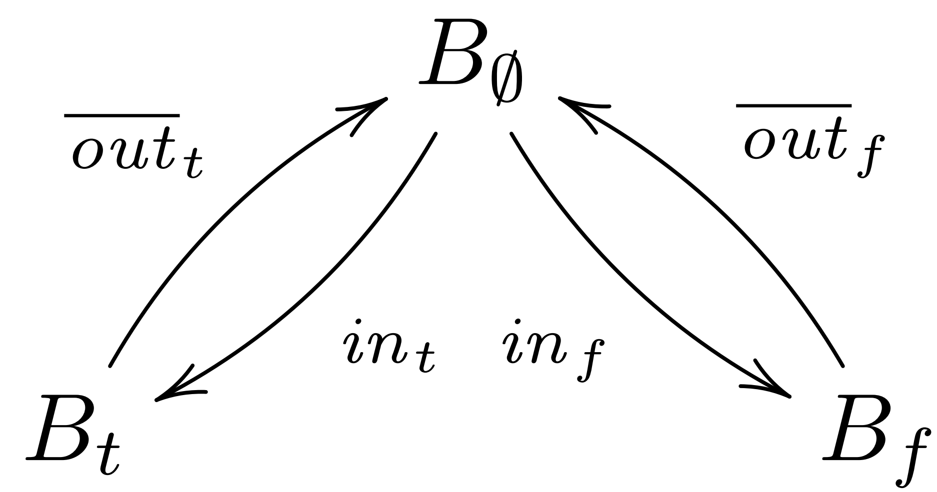


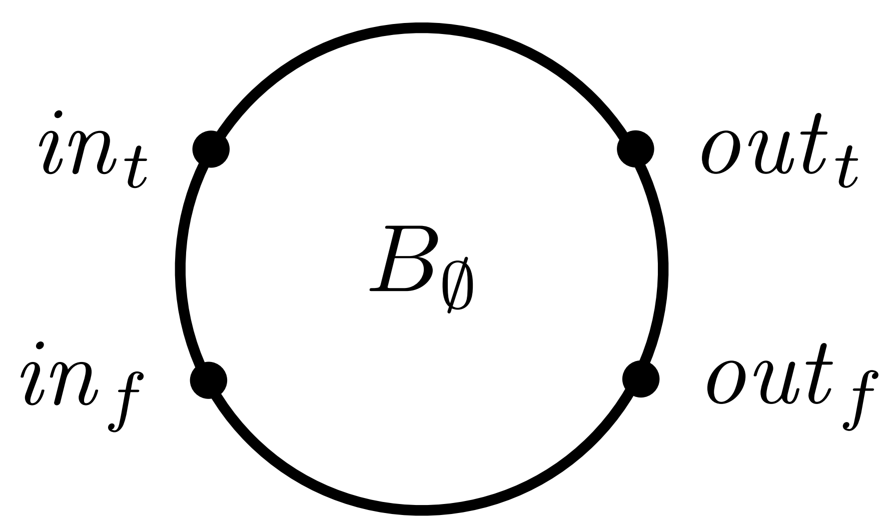
CCS: boolean buffer

$$B_\emptyset \triangleq in_t.B_t + in_f.B_f$$

$$B_t \triangleq \overline{out}_t.B_\emptyset$$

$$B_f \triangleq \overline{out}_f.B_\emptyset$$





Parallel composition

ParL) $$\frac{p_1 \xrightarrow{\mu} q_1}{p_1|p_2 \xrightarrow{\mu} q_1|p_2}$$

Com) $$\frac{p_1 \xrightarrow{\lambda} q_1 \quad p_2 \xrightarrow{\overline{\lambda}} q_2}{p_1|p_2 \xrightarrow{\tau} q_1|q_2}$$

ParR) $$\frac{p_2 \xrightarrow{\mu} q_2}{p_1|p_2 \xrightarrow{\mu} p_1|q_2}$$

processes running in parallel can interleave their actions or synchronize when dual actions are performed

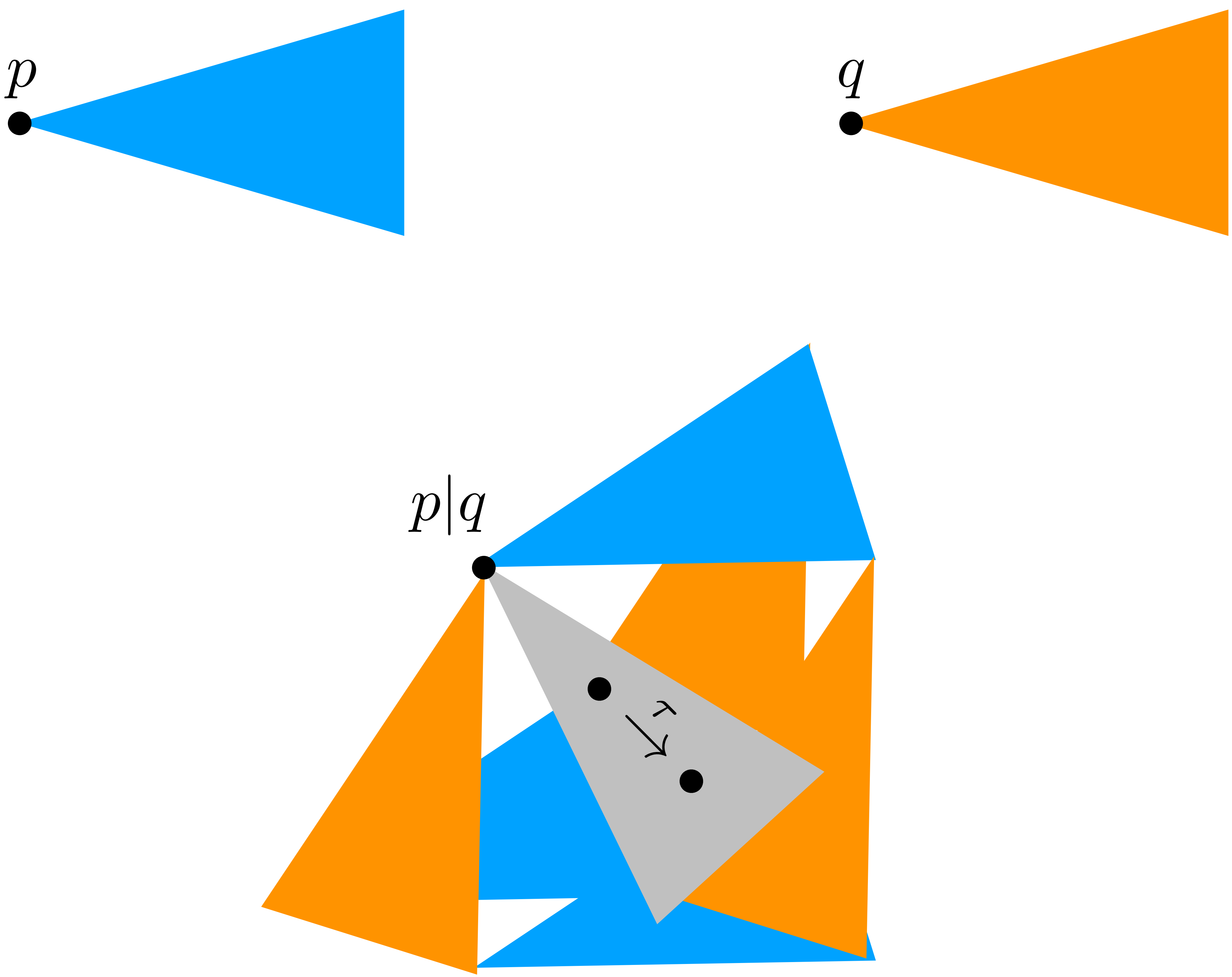
$$P \triangleq \overline{coin}.coffee.\mathbf{nil} \qquad M \triangleq coin.(\overline{coffee}.\mathbf{nil} + \overline{tea}.\mathbf{nil})$$

$$P|M \xrightarrow{\overline{coin}} coffee.\mathbf{nil}|M$$

$$P|M \xrightarrow{coin} P|(\overline{coffee}.\mathbf{nil} + \overline{tea}.\mathbf{nil})$$

$$P|M \xrightarrow{\tau} coffee.\mathbf{nil}|(\overline{coffee}.\mathbf{nil} + \overline{tea}.\mathbf{nil})$$

LTS of a process



CCS: parallel buffers

$$B^1_0|B^1_0$$

$$B^1_0 \triangleq in.B^1_1$$

$$B^1_1 \triangleq \overline{out}.B^1_0$$

CCS: parallel buffers

$$B^1_0 \triangleq in.B^1_1$$

$$B^1_1 \triangleq \overline{out}.B^1_0$$

$$\begin{tikzcd}
& B^1_0 | B^1_0 \arrow[dl, bend right, "in"'] \arrow[dr, bend left, "in"] & \\
B^1_1 | B^1_0 & & B^1_0 | B^1_1
\end{tikzcd}$$

CCS: parallel buffers

$$B^1_0 \triangleq in.B^1_1$$

$$B^1_1 \triangleq \overline{out}.B^1_0$$

$$\begin{tikzcd}
 & B^1_0|B^1_0 \arrow[dl, bend right, "in"'] \arrow[dr, bend left, "in"] & \\
B^1_1|B^1_0 \arrow[ur, bend left, "\overline{out}"] \arrow[dr, bend left, "in"] & & B^1_0|B^1_1 \\
 & B^1_1|B^1_1 &
\end{tikzcd}$$

CCS: parallel buffers

$$B_0^1 \triangleq in.B_1^1$$

$$B_1^1 \triangleq \overline{out}.B_0^1$$

$$\begin{tikzcd}
& B_0^1|B_0^1 \arrow[dl, bend right, "in"'] \arrow[dr, bend left, "in"] & \\
B_1^1|B_0^1 \arrow[ur, bend left, "\overline{out}"] \arrow[dr, "in"] & & B_0^1|B_1^1 \arrow[ul, bend right, "\overline{out}"'] \arrow[dl, "in"'] \\
& B_1^1|B_1^1 &
\end{tikzcd}$$

CCS: parallel buffers

$$B_0^1 \triangleq in.B_1^1$$

$$B_1^1 \triangleq \overline{out}.B_0^1$$

$$\begin{tikzcd}
& B_0^1|B_0^1 \arrow[dl, bend left, "in"] \arrow[dr, bend left, "in"] & \\
B_1^1|B_0^1 \arrow[ur, bend left, "\overline{out}"] \arrow[dr, bend left, "in"] & & B_0^1|B_1^1 \arrow[ul, bend left, "\overline{out}"] \arrow[dl, bend left, "in"] \\
& B_1^1|B_1^1 \arrow[ul, bend left, "\overline{out}"] \arrow[ur, bend left, "\overline{out}"] &
\end{tikzcd}$$

$$\begin{tikzcd}
B_0^2 \arrow[d, bend left, "in"] \\
B_1^2 \arrow[u, bend left, "\overline{out}"] \arrow[d, bend left, "in"] \\
B_2^2 \arrow[u, bend left, "\overline{out}"]
\end{tikzcd}$$

compare with the 2-capacity buffer

Restriction

$$\text{Res)}\ \frac{p \xrightarrow{\mu} q \quad \mu \notin \{\alpha, \overline{\alpha}\}}{p\backslash\alpha \xrightarrow{\mu} q\backslash\alpha}$$

makes the channel α private to p

no interaction on α with the environment

if p is the parallel composition of processes, then they can synchronise on α

$P \triangleq \overline{coin}.\mathit{coffee}.\mathbf{nil}$ $\qquad$ $M \triangleq coin.(\overline{\mathit{coffee}}.\mathbf{nil} + \overline{tea}.\mathbf{nil})$

$$(P|M)\backslash coin\backslash \mathit{coffee}\backslash tea \xrightarrow{\tau} (\mathit{coffee}.\mathbf{nil}|\overline{\mathit{coffee}}.\mathbf{nil} + \overline{tea}.\mathbf{nil})\backslash coin\backslash \mathit{coffee}\backslash tea$$

$$(\mathit{coffee}.\mathbf{nil}|\overline{\mathit{coffee}}.\mathbf{nil} + \overline{tea}.\mathbf{nil})\backslash coin\backslash \mathit{coffee}\backslash tea \xrightarrow{\tau} (\mathbf{nil}|\mathbf{nil})\backslash coin\backslash \mathit{coffee}\backslash tea$$

Restriction: shorthand

given $S = \{\alpha_1, ..., \alpha_n\}$ we write $p \backslash S$

instead of $p \backslash \alpha_1 \ldots \backslash \alpha_n$

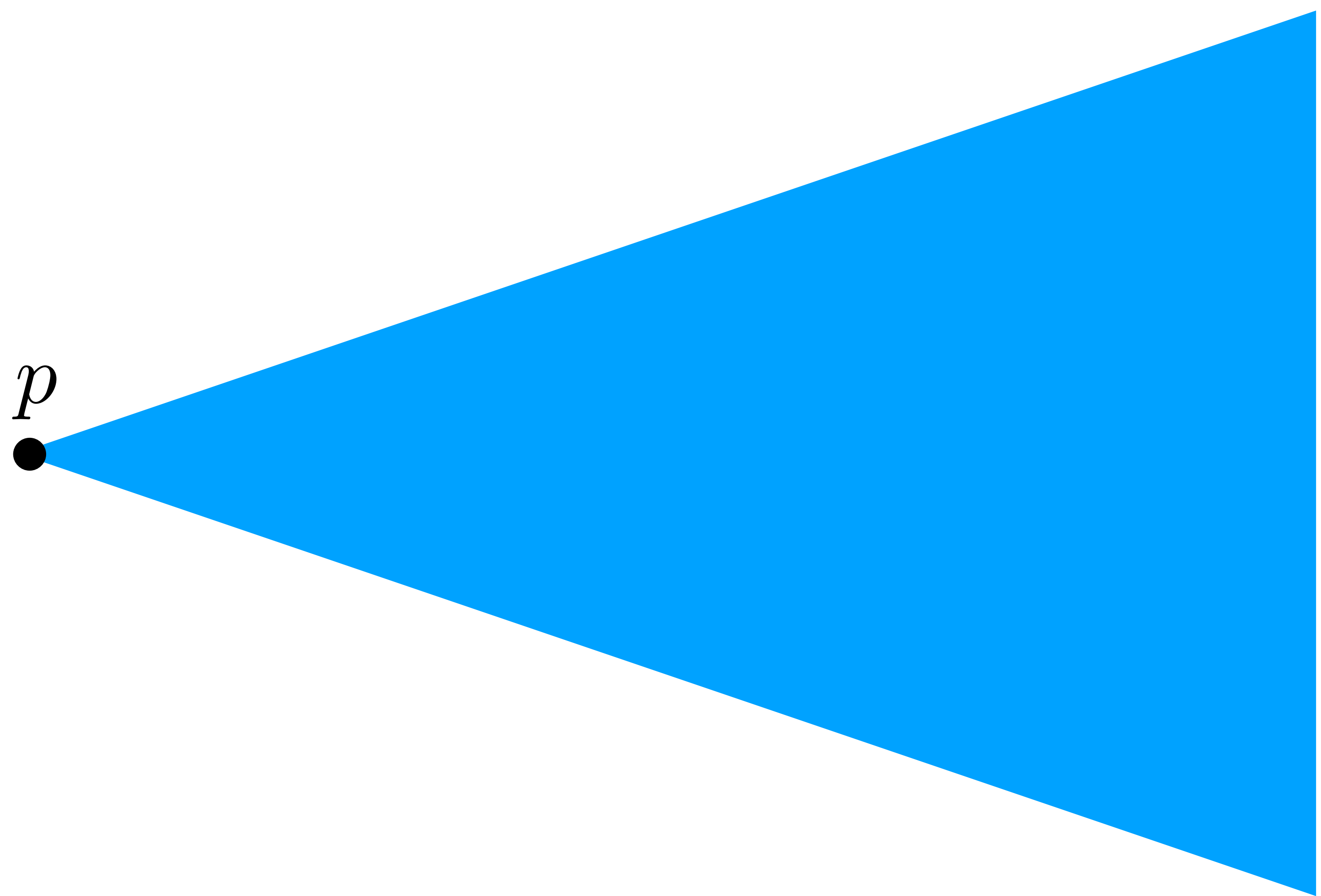
we omit trailing $\mathbf{nil}$

$$P \triangleq \overline{coin}.coffee \qquad M \triangleq coin.(\overline{coffee} + \overline{tea})$$

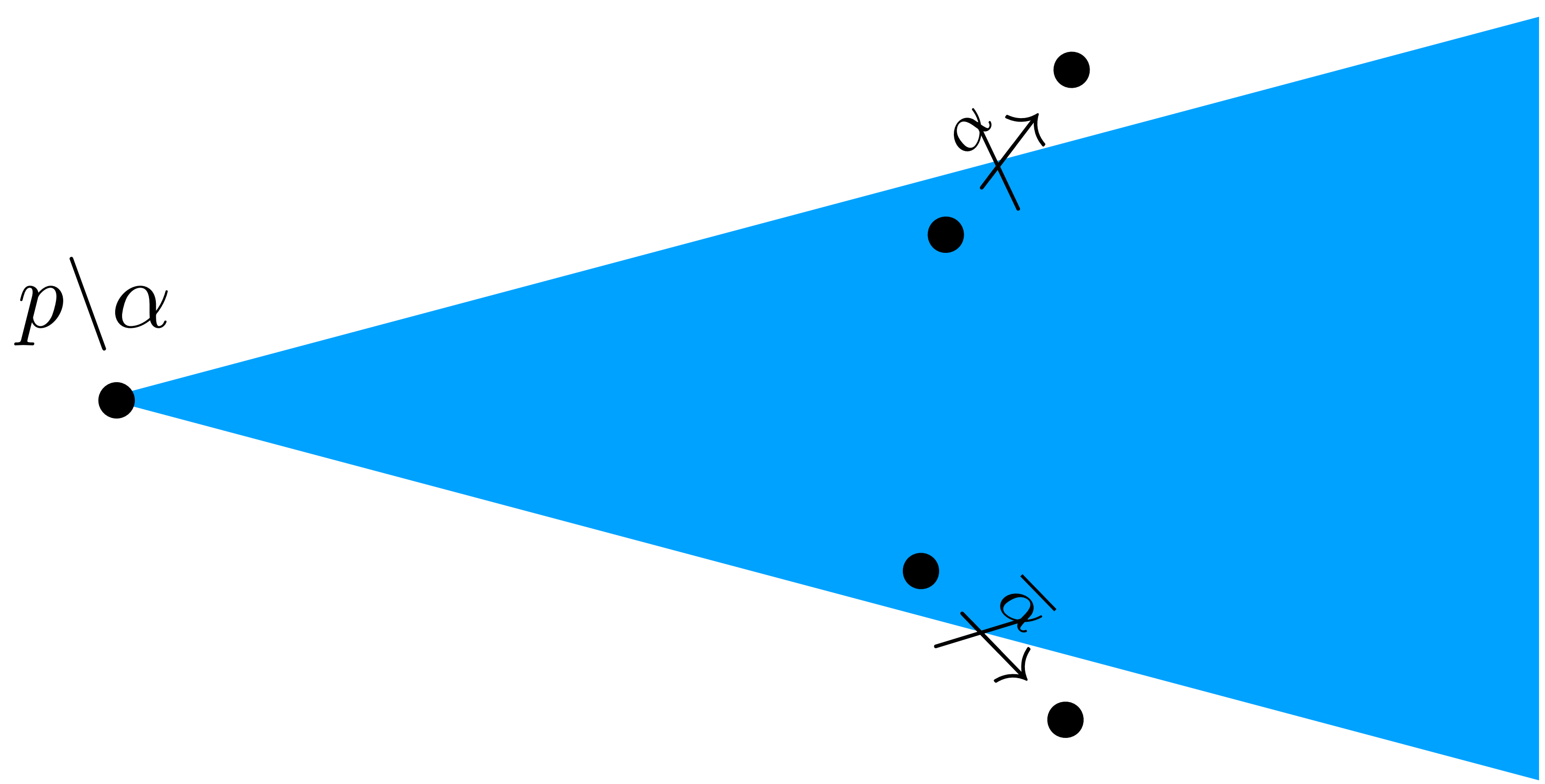
$$S \triangleq \{coin, coffee, tea\}$$

$$(P|M)\backslash S \xrightarrow{\tau} (coffee|\overline{coffee} + \overline{tea})\backslash S \xrightarrow{\tau} (\mathbf{nil}|\mathbf{nil})\backslash S$$

LTS of a process



LTS of a process



Relabelling

$$\text{Rel)}\frac{p \xrightarrow{\mu} q}{p[\phi] \xrightarrow{\phi(\mu)} q[\phi]}$$

renames the action channels according to ϕ

we assume $\phi(\tau) = \tau$ $\phi(\overline{\lambda}) = \overline{\phi(\lambda)}$

allows one to reuse processes

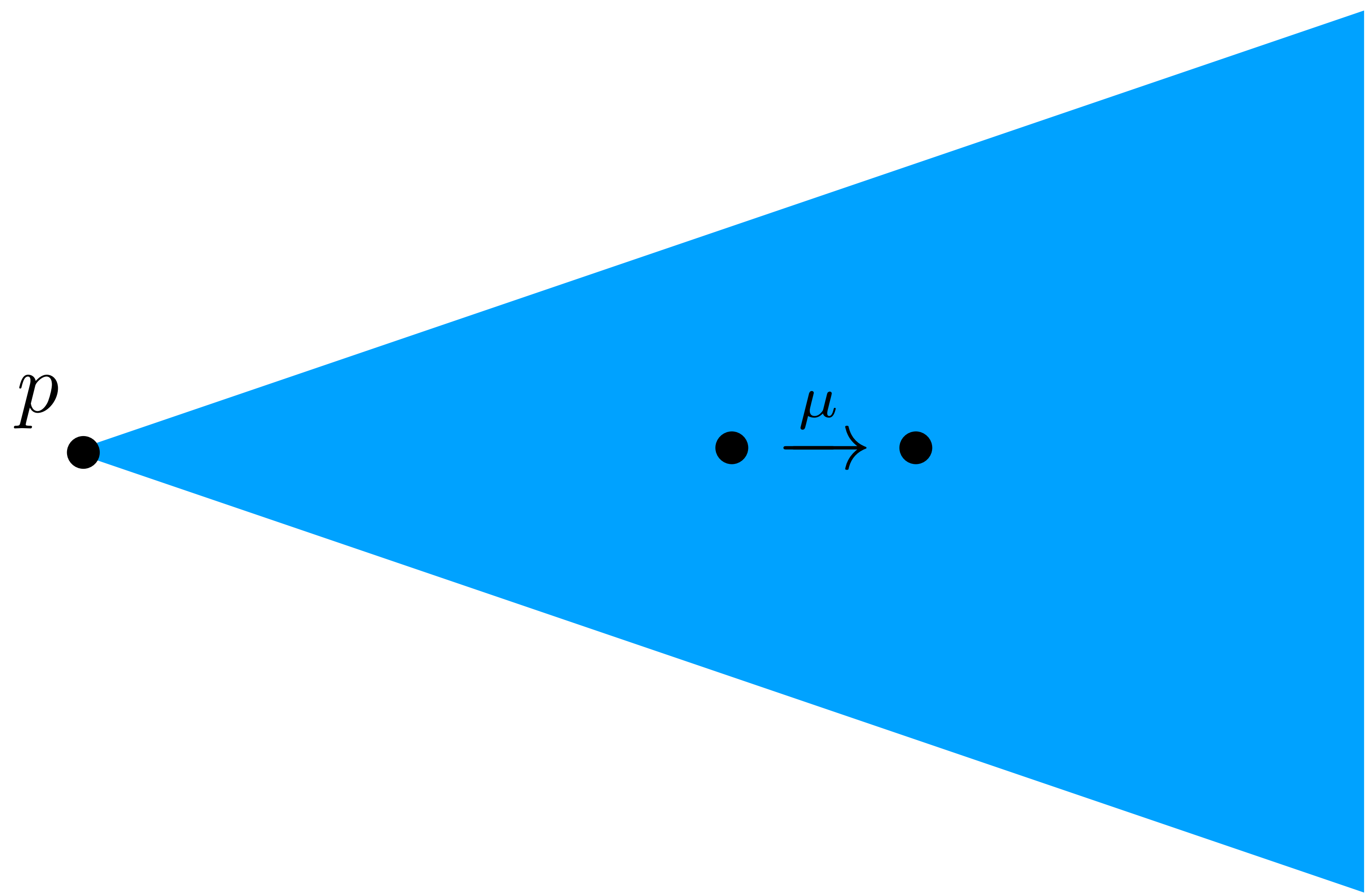
$P \triangleq \overline{coin}.coffee$

$\phi(coin) = moneta$

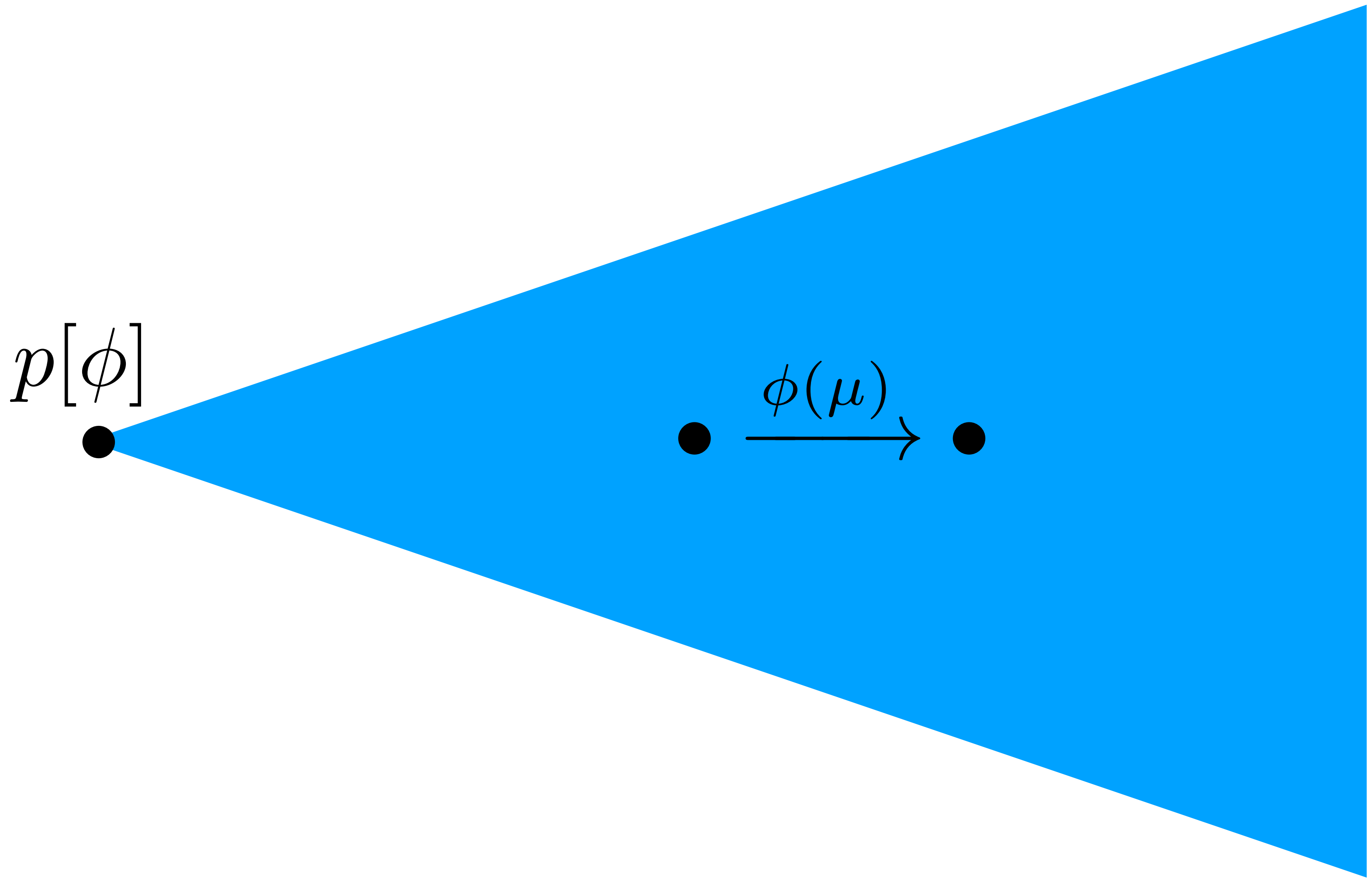
$\phi(coffee) = caff\grave{e}$

$$P[\phi] \xrightarrow{\overline{moneta}} coffee[\phi] \xrightarrow{caff\grave{e}} \mathbf{nil}[\phi]$$

LTS of a process



LTS of a process



CCS op. semantics

Act) $$\frac{}{\mu.p \xrightarrow{\mu} p}$$

Res) $$\frac{p \xrightarrow{\mu} q \quad \mu \notin \{\alpha, \overline{\alpha}\}}{p\backslash\alpha \xrightarrow{\mu} q\backslash\alpha}$$

Rel) $$\frac{p \xrightarrow{\mu} q}{p[\phi] \xrightarrow{\phi(\mu)} q[\phi]}$$

SumL) $$\frac{p_1 \xrightarrow{\mu} q}{p_1 + p_2 \xrightarrow{\mu} q}$$

SumR) $$\frac{p_2 \xrightarrow{\mu} q}{p_1 + p_2 \xrightarrow{\mu} q}$$

ParL) $$\frac{p_1 \xrightarrow{\mu} q_1}{p_1|p_2 \xrightarrow{\mu} q_1|p_2}$$

Com) $$\frac{p_1 \xrightarrow{\lambda} q_1 \quad p_2 \xrightarrow{\overline{\lambda}} q_2}{p_1|p_2 \xrightarrow{\tau} q_1|q_2}$$

ParR) $$\frac{p_2 \xrightarrow{\mu} q_2}{p_1|p_2 \xrightarrow{\mu} p_1|q_2}$$

Rec) $$\frac{p[{}^{\mathbf{rec}\ x.\ p}/_{x}] \xrightarrow{\mu} q}{\mathbf{rec}\ x.\ p \xrightarrow{\mu} q}$$

Linked buffers

$$B_0^1 \triangleq in.B_1^1 \qquad \eta(out) = c$$

$$B_1^1 \triangleq \overline{out}.B_0^1 \qquad \phi(in) = c$$

$$\begin{tikzcd} B_0^1[\phi] \arrow[d, bend left, "c"] \\ B_1^1[\phi] \arrow[u, bend left, "\overline{out}"] \end{tikzcd} \qquad \begin{tikzcd} B_0^1[\eta] \arrow[d, bend left, "in"] \\ B_1^1[\eta] \arrow[u, bend left, "\overline{c}"] \end{tikzcd}$$

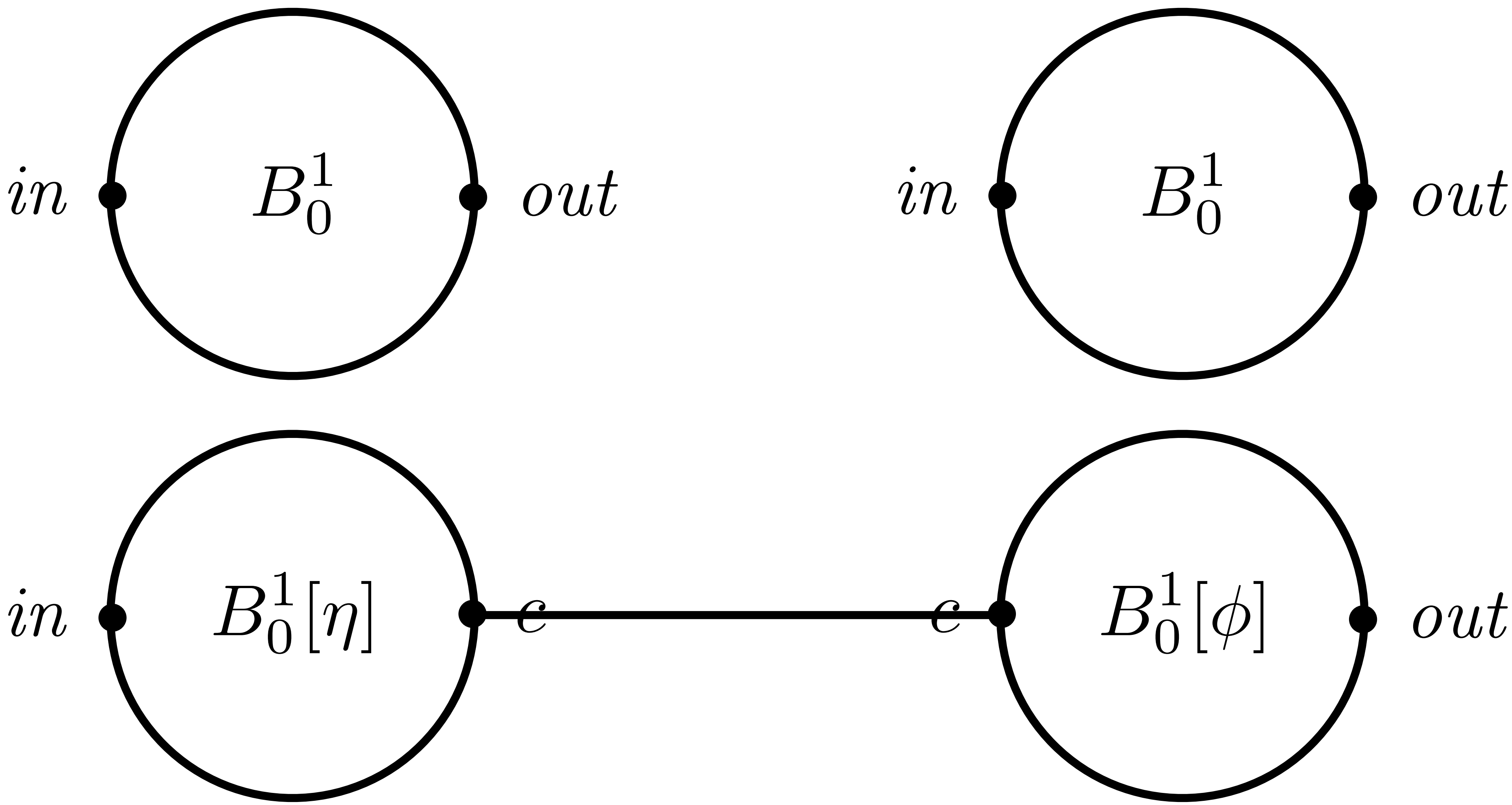
$$\begin{tikzcd} & (B_0^1[\eta] | B_0^1[\phi])\backslash c \arrow[dl, "in"'] & \\ (B_1^1[\eta] | B_0^1[\phi])\backslash c \arrow[rr, "\tau"] & & (B_0^1[\eta] | B_1^1[\phi])\backslash c \arrow[ul, "\overline{out}"'] \arrow[dl, "in"] \\ & (B_1^1[\eta] | B_1^1[\phi])\backslash c \arrow[ul, "\overline{out}"] & \end{tikzcd}$$

Linked buffers

$$B^1_0 \triangleq in.B^1_1 \qquad \eta(out) = c$$

$$B^1_1 \triangleq \overline{out}.B^1_0 \qquad \phi(in) = c$$

$$\begin{tikzcd} B^1_0[\phi] \arrow[d, bend left, "c"] \\ B^1_1[\phi] \arrow[u, bend left, "\overline{out}"] \end{tikzcd} \qquad \begin{tikzcd} B^1_0[\eta] \arrow[d, bend left, "in"] \\ B^1_1[\eta] \arrow[u, bend left, "\overline{c}"] \end{tikzcd}$$



Linked buffers

$$B_0^1 \triangleq in.B_1^1 \qquad \eta(out) = c$$

$$B_1^1 \triangleq \overline{out}.B_0^1 \qquad \phi(in) = c$$

$$p \frown q \triangleq (p[\eta] | q[\phi]) \backslash c$$

$$\begin{tikzcd}
& B_0^1 \frown B_0^1 \arrow[dl, "in"'] & \\
B_1^1 \frown B_0^1 \arrow[rr, "\tau"] & & B_0^1 \frown B_1^1 \arrow[ul, "\overline{out}"'] \arrow[dl, "in"] \\
& B_1^1 \frown B_1^1 \arrow[ul, "\overline{out}"] &
\end{tikzcd}$$

Linked boolean buffers

$$B_\emptyset \triangleq in_t.B_t + in_f.B_f \qquad \eta(out_t) = c_t \qquad \phi(in_t) = c_t$$

$$B_t \triangleq \overline{out}_t.B_\emptyset \qquad \eta(out_f) = c_f \qquad \phi(in_f) = c_f$$

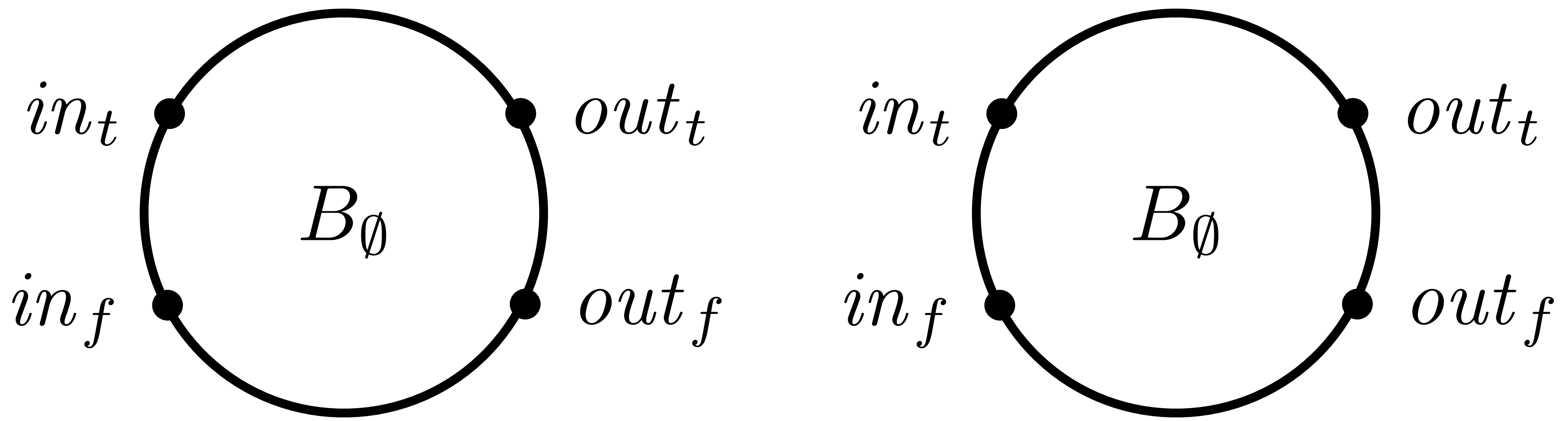
$$B_f \triangleq \overline{out}_f.B_\emptyset \qquad p \frown q \triangleq (p[\eta]|q[\phi])\backslash\{c_t, c_f\}$$

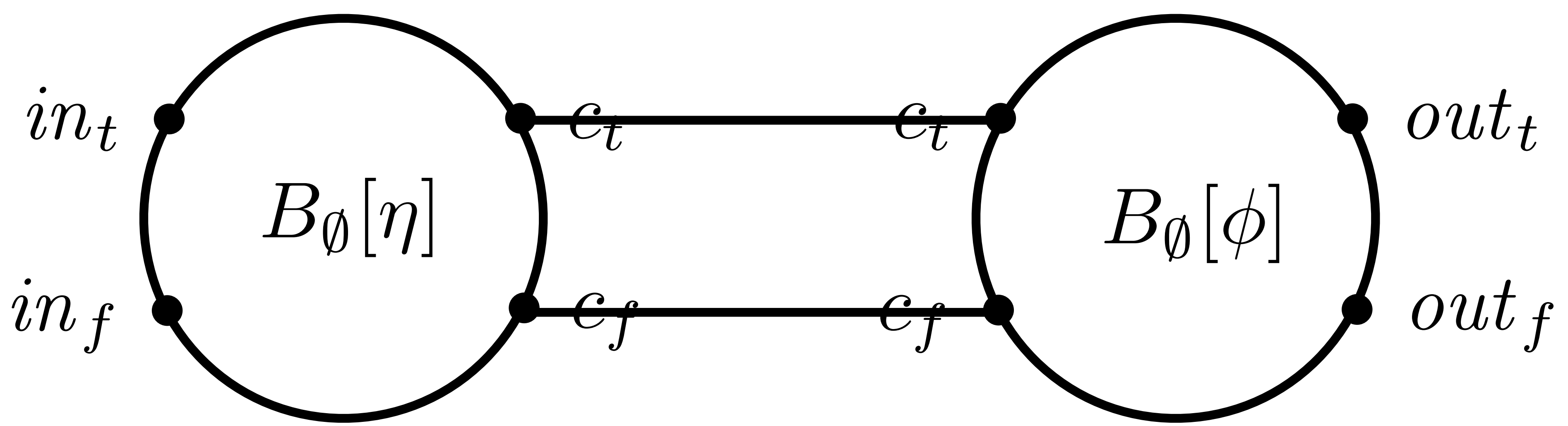
Linked boolean buffers

$$B_\emptyset \triangleq in_t.B_t + in_f.B_f$$

$$B_t \triangleq \overline{out}_t.B_\emptyset$$

$$B_f \triangleq \overline{out}_f.B_\emptyset$$



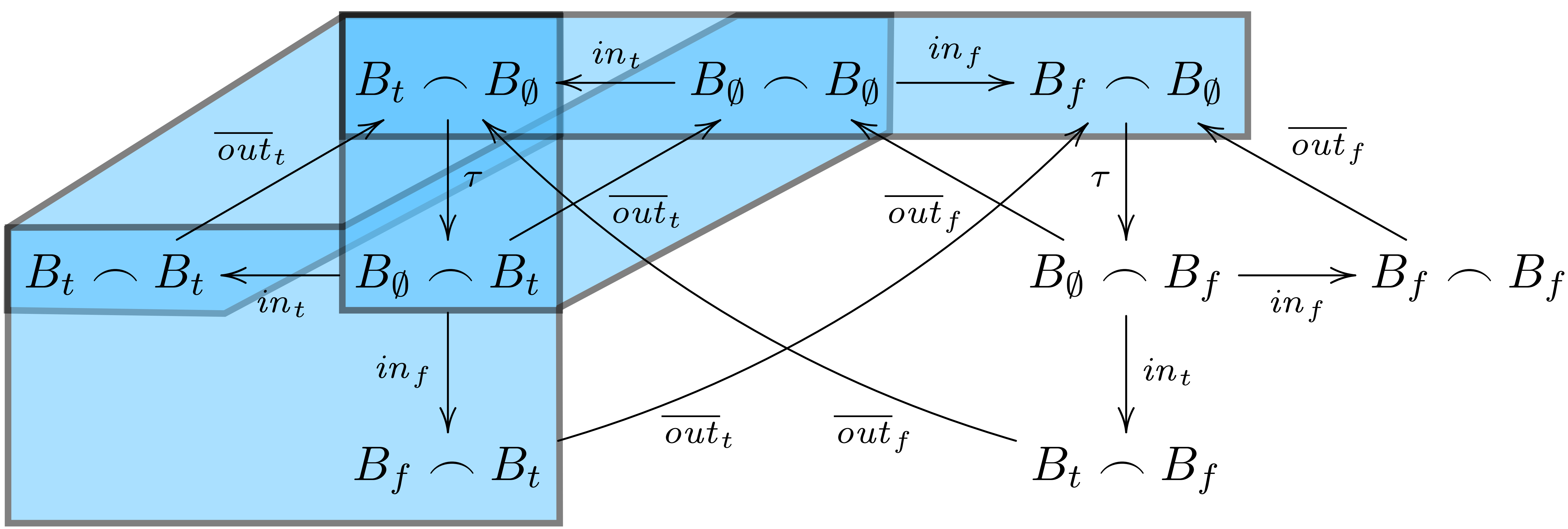


Linked boolean buffers

$$B_\emptyset \triangleq in_t.B_t + in_f.B_f \qquad \eta(out_t) = c_t \qquad \phi(in_t) = c_t$$

$$B_t \triangleq \overline{out}_t.B_\emptyset \qquad \eta(out_f) = c_f \qquad \phi(in_f) = c_f$$

$$B_f \triangleq \overline{out}_f.B_\emptyset \qquad p \frown q \triangleq (p[\eta] | q[\phi]) \backslash \{c_t, c_f\}$$



Linked boolean buffers

$$B_\emptyset \triangleq in_t.B_t + in_f.B_f \qquad \eta(out_t) = c_t \qquad \phi(in_t) = c_t$$

$$B_t \triangleq \overline{out}_t.B_\emptyset \qquad \eta(out_f) = c_f \qquad \phi(in_f) = c_f$$

$$B_f \triangleq \overline{out}_f.B_\emptyset \qquad p \frown q \triangleq (p[\eta]|q[\phi])\backslash\{c_t, c_f\}$$

$$\begin{tikzcd}
 & & B_t \frown B_\emptyset \arrow[d, "\tau"] & & B_\emptyset \frown B_\emptyset \arrow[ll, "in_t"'] \arrow[rr, "in_f"] & & B_f \frown B_\emptyset \arrow[d, "\tau"'] & & \\
B_t \frown B_t \arrow[urr, "\overline{out}_t"] & & B_\emptyset \frown B_t \arrow[ll, "in_t"] \arrow[urr, "\overline{out}_t"'] \arrow[d, "in_f"'] & & & & B_\emptyset \frown B_f \arrow[ull, "\overline{out}_f"] \arrow[rr, "in_f"'] \arrow[d, "in_t"] & & B_f \frown B_f \arrow[ull, "\overline{out}_f"'] \\
 & & B_f \frown B_t \arrow[uurrrr, bend right, "\overline{out}_t"'] & & & & B_t \frown B_f \arrow[uullll, bend left, "\overline{out}_f"] & &
\end{tikzcd}$$

CCS with value passing

$$\frac{}{\alpha!v.p \xrightarrow{\overline{\alpha_v}} p} \qquad \frac{}{\alpha?x.p \xrightarrow{\alpha_v} p[^v/_x]}$$

when the set of values is finite $V \triangleq \{v_1, ..., v_n\}$

$$\alpha!v.p \equiv \overline{\alpha_v}.p$$

$$\alpha?x.p \equiv \alpha_{v_1}.p[^{v_1}/_x] + \cdots + \alpha_{v_n}.p[^{v_n}/_x]$$

```
receive
        v -> p
        w -> q
        _ -> r
end
```

$$\equiv \alpha_v.p + \alpha_w.q + \sum_{z \neq v,w} \alpha_z.r$$

Exercise: LTS?

$$P \triangleq (\mathbf{rec}\ x.\ \alpha.x) + (\mathbf{rec}\ x.\ \beta.x)$$

$$\begin{tikzcd}
 & P \arrow[dl, "\alpha"'] \arrow[dr, "\beta"] & \\
\mathbf{rec}\ x.\ \alpha.x \arrow[loop below, "\alpha"] & & \mathbf{rec}\ x.\ \beta.x \arrow[loop below, "\beta"]
\end{tikzcd}$$

Exercise: LTS?

$$Q \triangleq \mathbf{rec}\ x.\ (\alpha.x + \beta.x)$$

$$Q \triangleq \mathbf{rec}\ x.\ \alpha.x + \beta.x$$

$$Q \triangleq \alpha.Q + \beta.Q$$

$$\begin{tikzcd} Q \arrow[loop left, "\alpha"] \arrow[loop right, "\beta"] \end{tikzcd}$$

Exercise: LTS?

$$R \triangleq \textbf{rec } x.\ (\alpha.x + \beta.\textbf{nil})$$

$$R \triangleq \textbf{rec } x.\ \alpha.x + \beta$$

$$R \triangleq \alpha.R + \beta$$

$$\begin{tikzcd} R \arrow[loop left, "\alpha"] \arrow[d, "\beta"] \\ \textbf{nil} \end{tikzcd}$$

Exercise: LTS?

$$T \triangleq \mathbf{rec}\ x.\ ((\alpha.\mathbf{nil}|x) + \beta.\mathbf{nil})$$

$$T \triangleq \mathbf{rec}\ x.\ (\alpha|x) + \beta$$

$$T \triangleq (\alpha|T) + \beta$$

$$\begin{tikzcd}
\mathbf{nil} & \alpha|\mathbf{nil} & \alpha|\alpha|\mathbf{nil} & \cdots \\
T \arrow[u, "\beta"] \arrow[ur, "\beta"] \arrow[urr, "\beta"] \arrow[d, "\alpha"'] \arrow[dr, "\alpha"] \arrow[drr, "\alpha"] & & & \\
\mathbf{nil}|T & \alpha|\mathbf{nil}|T & \alpha|\alpha|\mathbf{nil}|T & \cdots
\end{tikzcd}$$

Exercise: LTS?

$$U \triangleq \mathbf{rec}\ x.\ ((\alpha.\mathbf{nil})|\beta.x)$$

$$U \triangleq \mathbf{rec}\ x.\ \alpha|\beta.x$$

$$U \triangleq \alpha|\beta.U$$

$$\begin{tikzcd}
U \arrow[rr, "\beta"] \arrow[d, "\alpha"'] & & \alpha|U \\
\mathbf{nil}|\beta.U & &
\end{tikzcd}$$

Exercise: LTS?

$$U \triangleq \mathbf{rec}\ x.\ ((\alpha.\mathbf{nil})|\beta.x)$$

$$U \triangleq \mathbf{rec}\ x.\ \alpha|\beta.x$$

$$U \triangleq \alpha|\beta.U$$

$$\begin{tikzcd}
U \arrow[r, "\beta"] \arrow[d, "\alpha"'] & \alpha|U \\
\mathbf{nil}|\beta.U \arrow[d, "\beta"'] & \\
\mathbf{nil}|U &
\end{tikzcd}$$

Exercise: LTS?

$$U \triangleq \mathbf{rec}\ x.\ ((\alpha.\mathbf{nil})|\beta.x)$$

$$U \triangleq \mathbf{rec}\ x.\ \alpha|\beta.x$$

$$U \triangleq \alpha|\beta.U$$

$$\begin{tikzcd}
U \arrow[r, "\beta"] \arrow[d, "\alpha"'] & \alpha|U \arrow[r, "\beta"] \arrow[d, "\alpha"] \arrow[ddl, "\alpha"'] & \alpha|\alpha|U \\
\mathbf{nil}|\beta.U \arrow[d, "\beta"'] & \alpha|\mathbf{nil}|\beta.U & \\
\mathbf{nil}|U & &
\end{tikzcd}$$

Exercise: LTS?

$$U \triangleq \mathbf{rec}\ x.\ ((\alpha.\mathbf{nil})|\beta.x)$$

$$U \triangleq \mathbf{rec}\ x.\ \alpha|\beta.x$$

$$U \triangleq \alpha|\beta.U$$

$$\begin{tikzcd}
U \arrow[r, "\beta"] \arrow[d, "\alpha"'] & \alpha|U \arrow[r, "\beta"] \arrow[d, "\alpha"'] \arrow[dl, "\alpha"', bend right=0, shift left=0] & \alpha|\alpha|U \\
\mathbf{nil}|\beta.U \arrow[d, "\beta"'] & \alpha|\mathbf{nil}|\beta.U \arrow[r, "\beta"'] \arrow[d, "\alpha"'] & \alpha|\mathbf{nil}|U \\
\mathbf{nil}|U & \mathbf{nil}|\mathbf{nil}|\beta.U &
\end{tikzcd}$$

Exercise: LTS?

$$U \triangleq \mathbf{rec}\ x.\ ((\alpha.\mathbf{nil})|\beta.x)$$

$$U \triangleq \mathbf{rec}\ x.\ \alpha|\beta.x$$

$$U \triangleq \alpha|\beta.U$$

$$\begin{tikzcd}
U \arrow[r, "\beta"] \arrow[d, "\alpha"'] & \alpha|U \arrow[r, "\beta"] \arrow[d, "\alpha"'] \arrow[ddl, "\alpha"'] & \alpha|\alpha|U \arrow[r, "\beta"] \arrow[d, "\alpha"'] \arrow[dr, "\alpha"] & \cdots \\
\mathbf{nil}|\beta.U \arrow[d, "\beta"'] & \alpha|\mathbf{nil}|\beta.U \arrow[r, "\beta"'] \arrow[d, "\alpha"'] & \alpha|\mathbf{nil}|U & \cdots \\
\mathbf{nil}|U & \mathbf{nil}|\mathbf{nil}|\beta.U & &
\end{tikzcd}$$

Exercise: LTS?

$$U \triangleq \mathbf{rec}\ x.\ ((\alpha.\mathbf{nil})|\beta.x)$$

$$U \triangleq \mathbf{rec}\ x.\ \alpha|\beta.x$$

$$U \triangleq \alpha|\beta.U$$

$$\begin{tikzcd}
U \arrow[r, "\beta"] \arrow[d, "\alpha"'] & \alpha|U \arrow[r, "\beta"] \arrow[d, "\alpha"'] \arrow[ddl, "\alpha"'] & \alpha|\alpha|U \arrow[r, "\beta"] \arrow[d, "\alpha"'] \arrow[dr, "\alpha"] & \cdots \\
\mathbf{nil}|\beta.U \arrow[d, "\beta"'] & \alpha|\mathbf{nil}|\beta.U \arrow[r, "\beta"'] \arrow[d, "\alpha"'] & \alpha|\mathbf{nil}|U \arrow[r, "\beta"'] \arrow[d, "\alpha"'] \arrow[dr, "\alpha"] & \cdots \\
\mathbf{nil}|U & \mathbf{nil}|\mathbf{nil}|\beta.U & \mathbf{nil}|\mathbf{nil}|U & \cdots
\end{tikzcd}$$

Exercise: LTS?

let's ignore **nil**

$$U \triangleq \mathbf{rec}\ x.\ ((\alpha.\mathbf{nil})|\beta.x)$$

$$U \triangleq \mathbf{rec}\ x.\ \alpha|\beta.x$$

$$U \triangleq \alpha|\beta.U$$

$$\begin{tikzcd}
U \arrow[r, bend left, "\beta"] \arrow[d, bend right, "\alpha"'] & \alpha|U \\
\beta.U &
\end{tikzcd}$$

Exercise: LTS?

let's ignore **nil**

$$U \triangleq \mathbf{rec}\ x.\ ((\alpha.\mathbf{nil})|\beta.x)$$

$$U \triangleq \mathbf{rec}\ x.\ \alpha|\beta.x$$

$$U \triangleq \alpha|\beta.U$$

$$\begin{tikzcd}
U \arrow[r, bend left, "\beta"] \arrow[d, bend right, "\alpha"'] & \alpha|U \\
\beta.U \arrow[u, bend right, "\beta"']
\end{tikzcd}$$

Exercise: LTS?

let's ignore **nil**

$$U \triangleq \textbf{rec}\ x.\ ((\alpha.\textbf{nil})|\beta.x)$$

$$U \triangleq \textbf{rec}\ x.\ \alpha|\beta.x$$

$$U \triangleq \alpha|\beta.U$$

$$\begin{tikzcd}
U \arrow[r, bend left, "\beta"] \arrow[d, bend right, "\alpha"'] & \alpha|U \arrow[l, bend left, "\alpha"] \arrow[r, "\beta"] \arrow[d, "\alpha"'] & \alpha|\alpha|U \\
\beta.U \arrow[u, bend right, "\beta"'] & \alpha|\beta.U &
\end{tikzcd}$$

Exercise: LTS?

let's ignore **nil**

$$U \triangleq \mathbf{rec}\ x.\ ((\alpha.\mathbf{nil})|\beta.x)$$

$$U \triangleq \mathbf{rec}\ x.\ \alpha|\beta.x$$

$$U \triangleq \alpha|\beta.U$$

$$\begin{tikzcd}
U \arrow[r, bend left, "\beta"] \arrow[d, bend right, "\alpha"'] & \alpha|U \arrow[l, bend left, "\alpha"] \arrow[r, bend left, "\beta"] \arrow[d, bend right, "\alpha"'] & \alpha|\alpha|U \\
\beta.U \arrow[u, bend right, "\beta"'] & \alpha|\beta.U \arrow[l, "\alpha"] \arrow[u, bend right, "\beta"']
\end{tikzcd}$$

Exercise: LTS?

let's ignore **nil**

$$U \triangleq \mathbf{rec}\ x.\ ((\alpha.\mathbf{nil})|\beta.x)$$

$$U \triangleq \mathbf{rec}\ x.\ \alpha|\beta.x$$

$$U \triangleq \alpha|\beta.U$$

$$\begin{tikzcd}
U \arrow[r, bend left, "\beta"] \arrow[d, bend right, "\alpha"'] & \alpha|U \arrow[l, bend left, "\alpha"] \arrow[r, bend left, "\beta"] \arrow[d, bend right, "\alpha"'] & \alpha|\alpha|U \arrow[l, bend left, "\alpha"] \arrow[r, bend left, "\beta"] \arrow[d, bend right, "\alpha"'] & {} \arrow[l, bend left, "\alpha"] & \cdots \\
\beta.U \arrow[u, bend right, "\beta"'] & \alpha|\beta.U \arrow[l, "\alpha"] \arrow[u, bend right, "\beta"'] & \alpha|\alpha|\beta.U \arrow[l, "\alpha"] \arrow[u, bend right, "\beta"'] & {} \arrow[l, "\alpha"] & \cdots
\end{tikzcd}$$

Badge exercise

Write an interactive counter modulo 4 in CCS

The counter process has four input channels:

inc, *val*, *reset*, *stop*

and four output channels:

c_0, c_1, c_2, c_3

used to display the current value of the counter

Draw the LTS of the counter process.