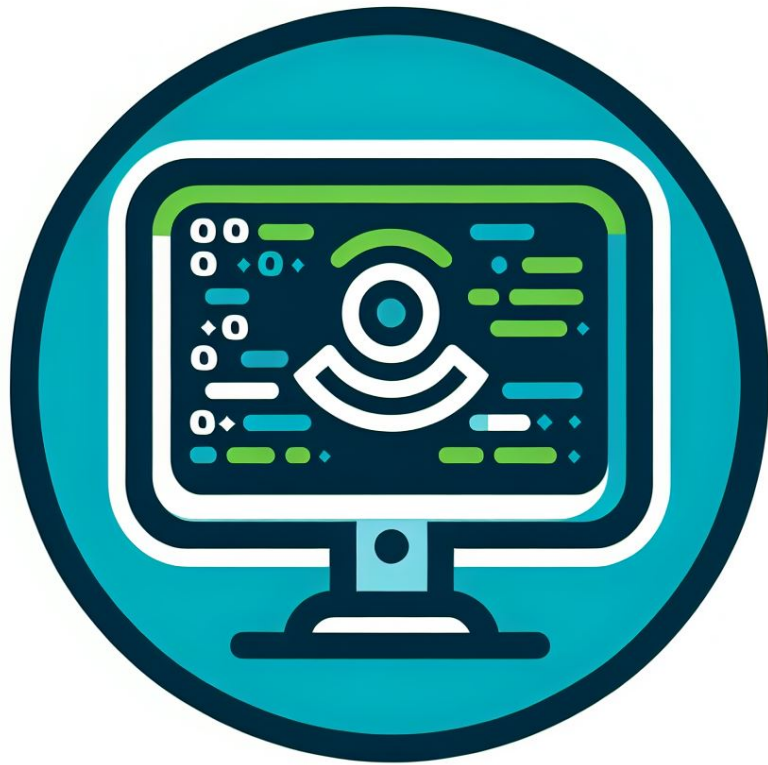




PSC 2024/25 (375AA, 9CFU)

Principles for Software Composition

Roberto Bruni

<http://www.di.unipi.it/~bruni/>

[http://didawiki.di.unipi.it/doku.php/
magistraleinformatica/psc/start](http://didawiki.di.unipi.it/doku.php/magistraleinformatica/psc/start)

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)

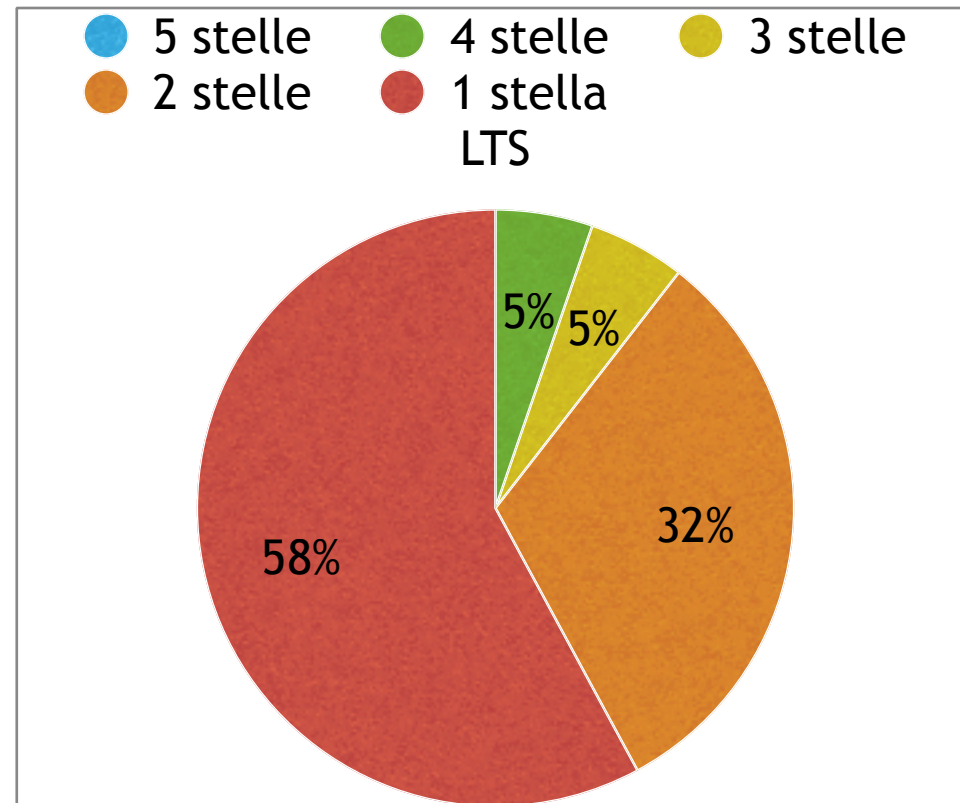
Communication

- 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

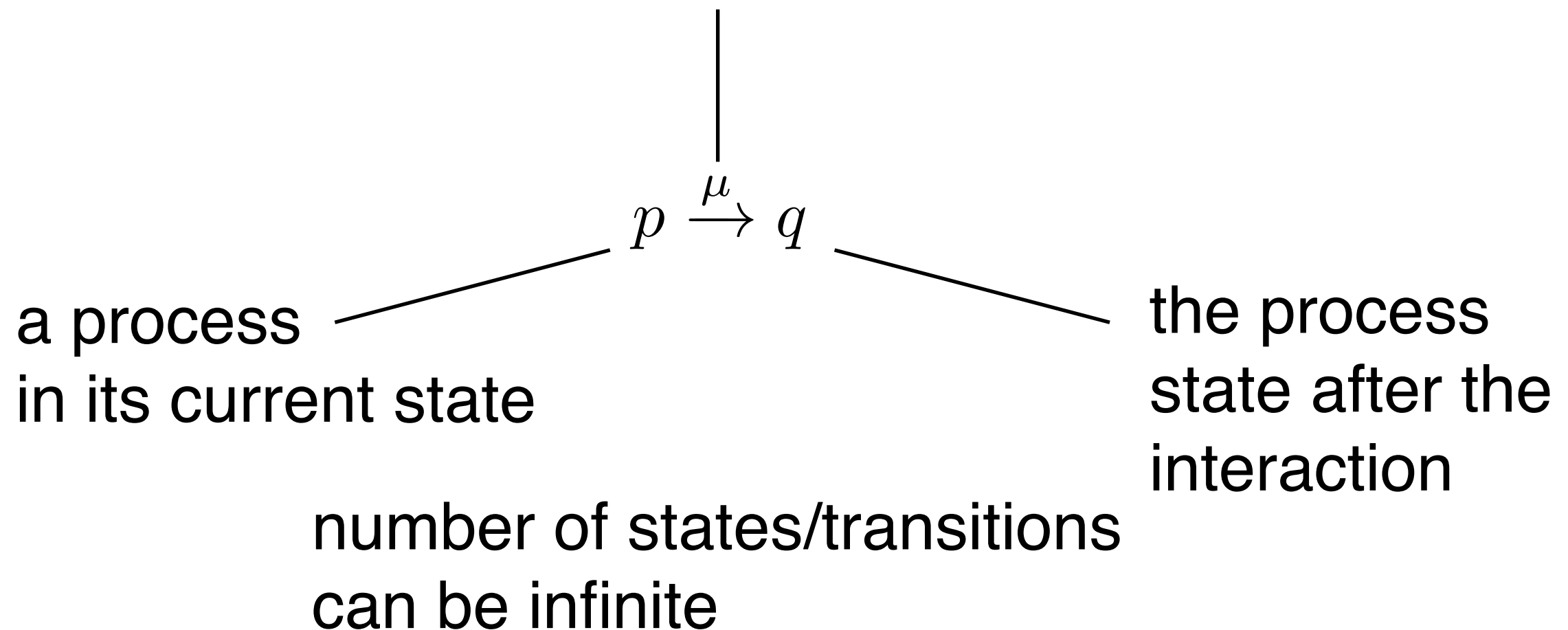
From your forms



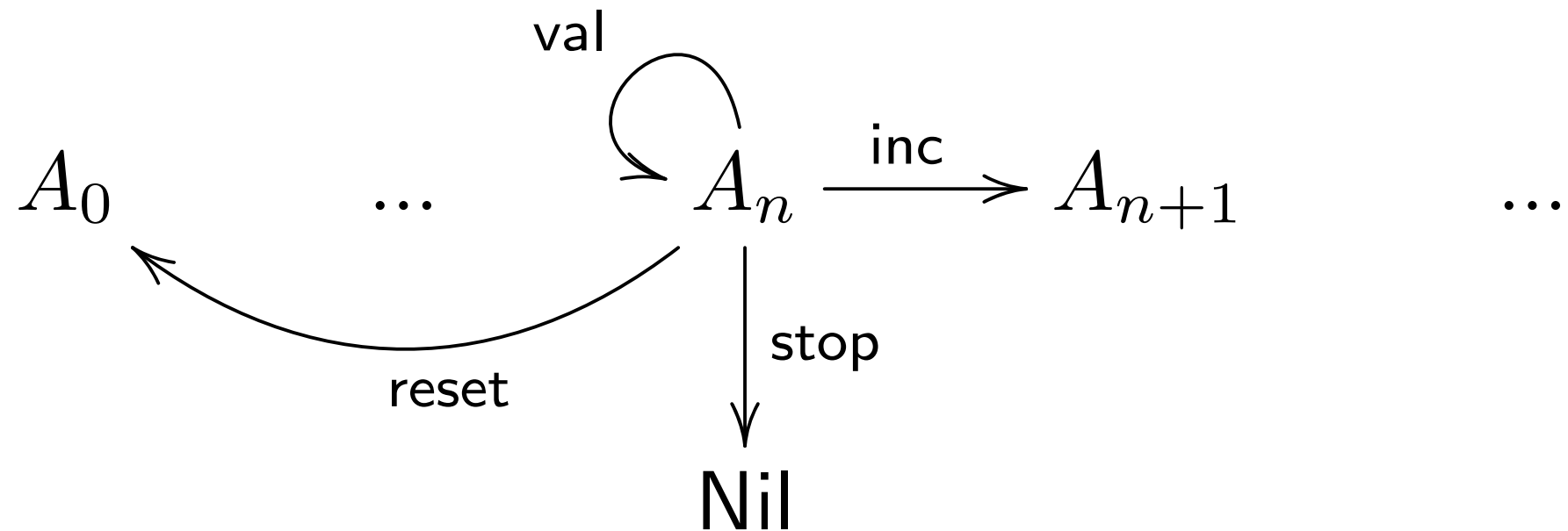
(over 19 answers)

Labelled transitions

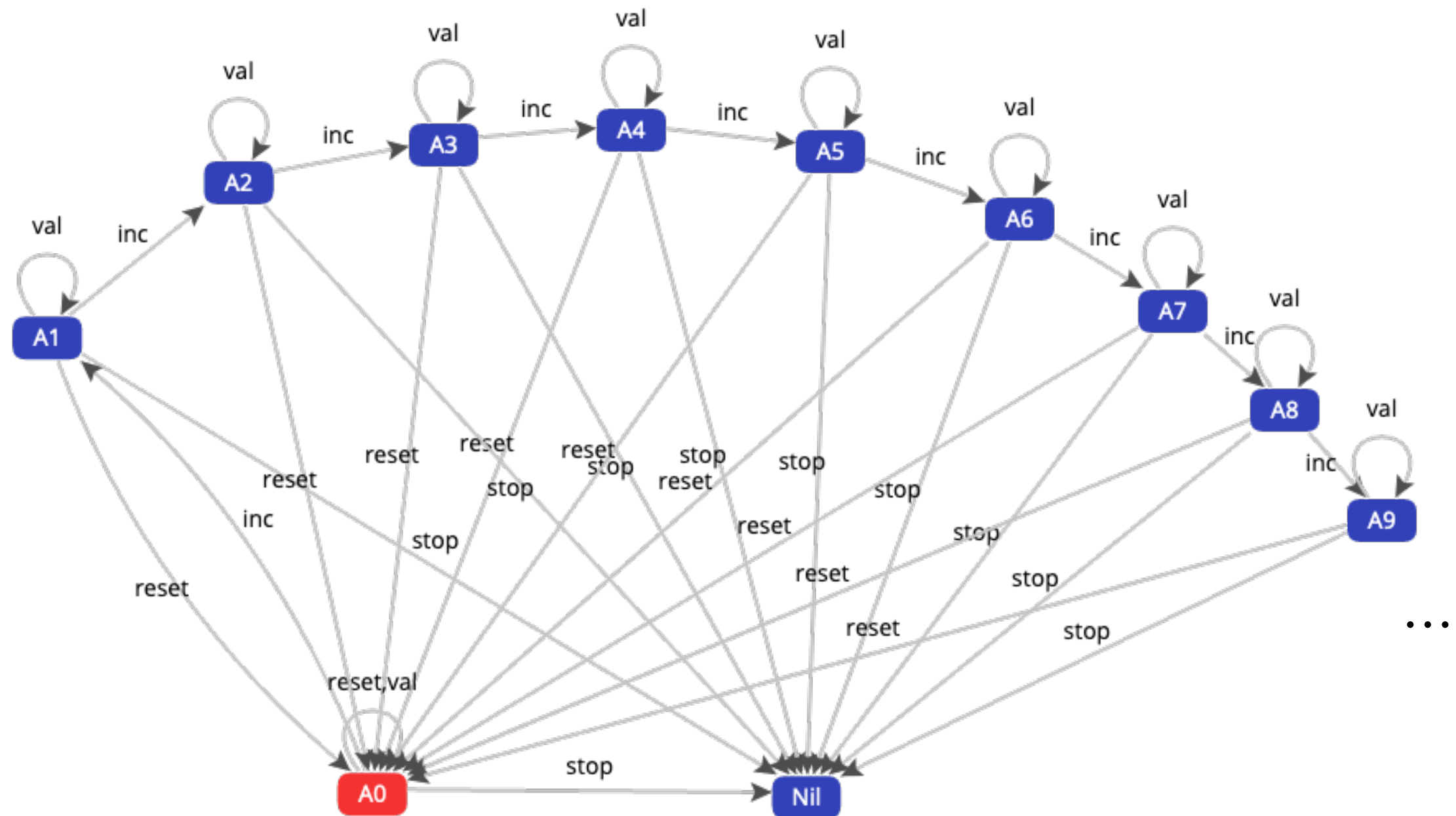
ongoing interaction
with the environment
(with other processes)



Example: counter



LTS: Labelled Transition System



CCS: states and labels

What is a process p ?

a sequential agent

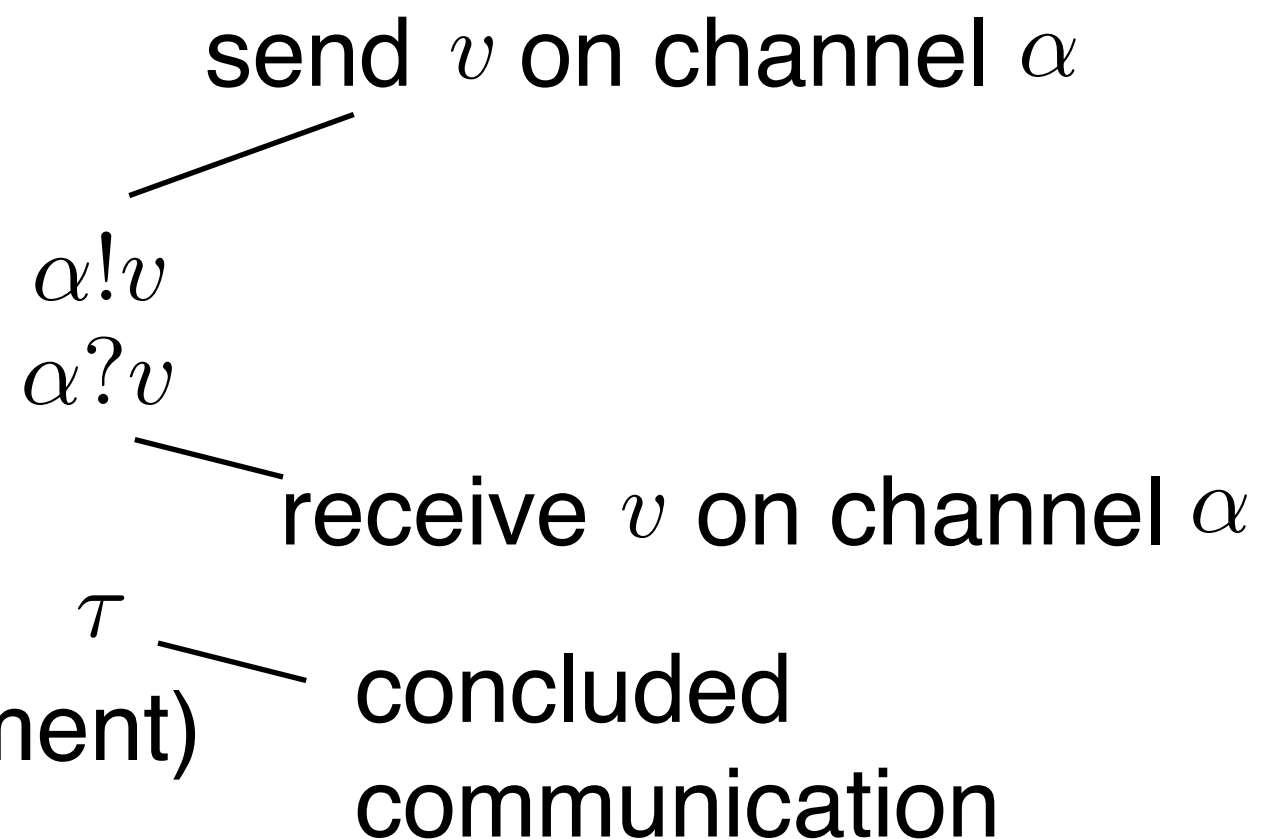
a system where many sequential agents interact

What is a label μ ?

an action (e.g. an output)

a dual action (e.g. an input)

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



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)

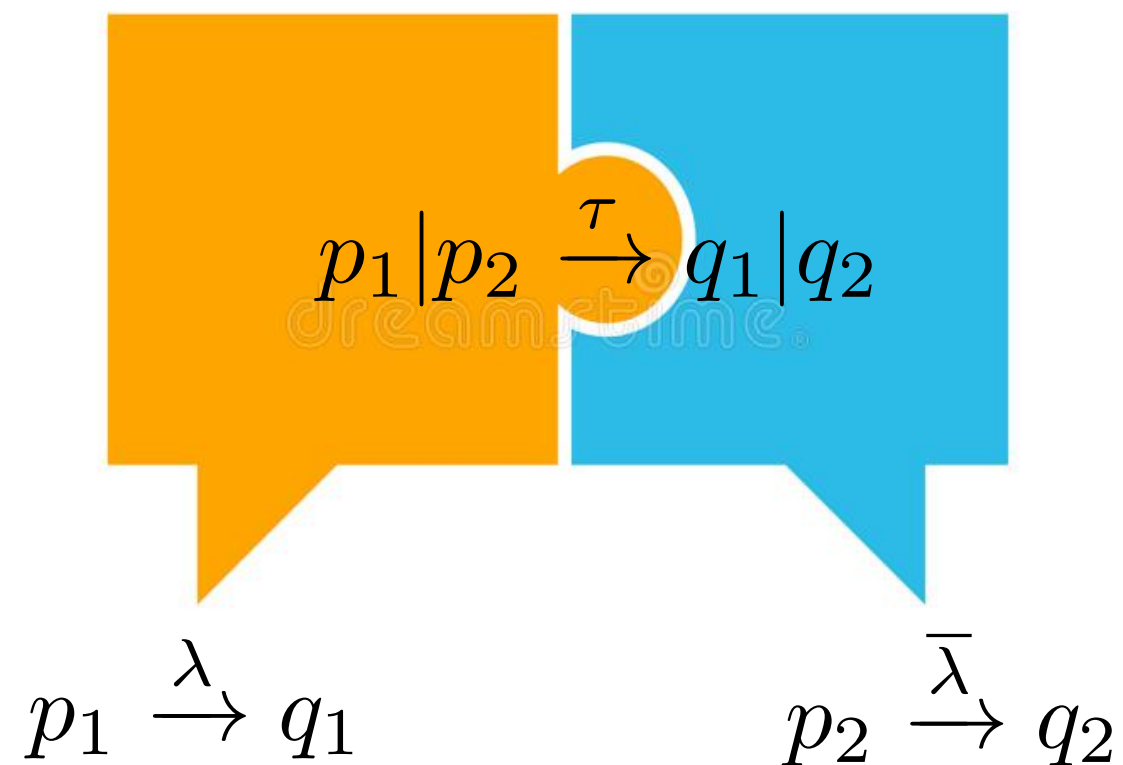
$\alpha!v$ becomes just $\overline{\alpha_v}$ or just $\overline{\alpha}$

$\alpha?v$ becomes just α_v or just α

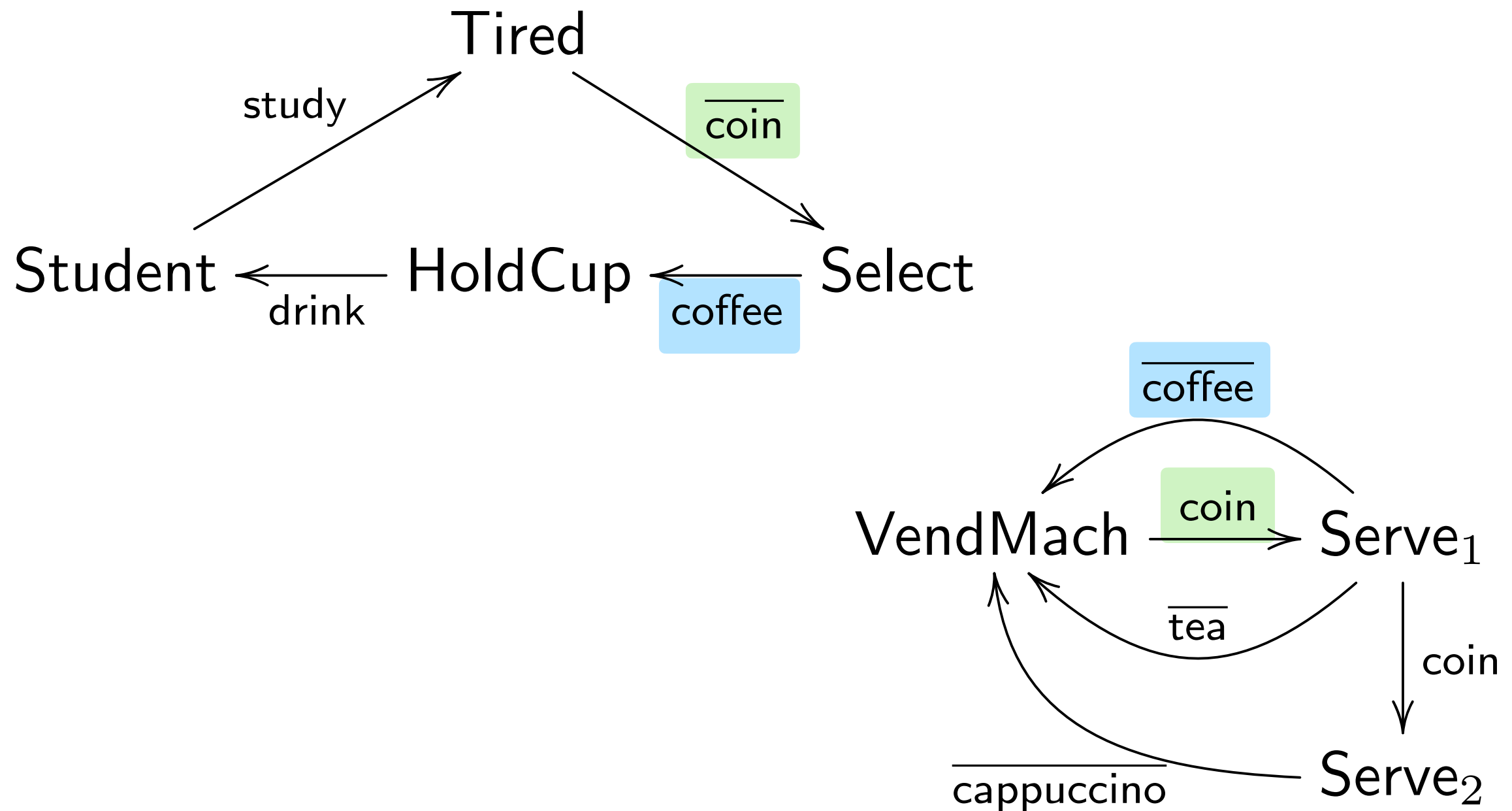
λ denotes either α or $\overline{\alpha}$

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

CCS: communication

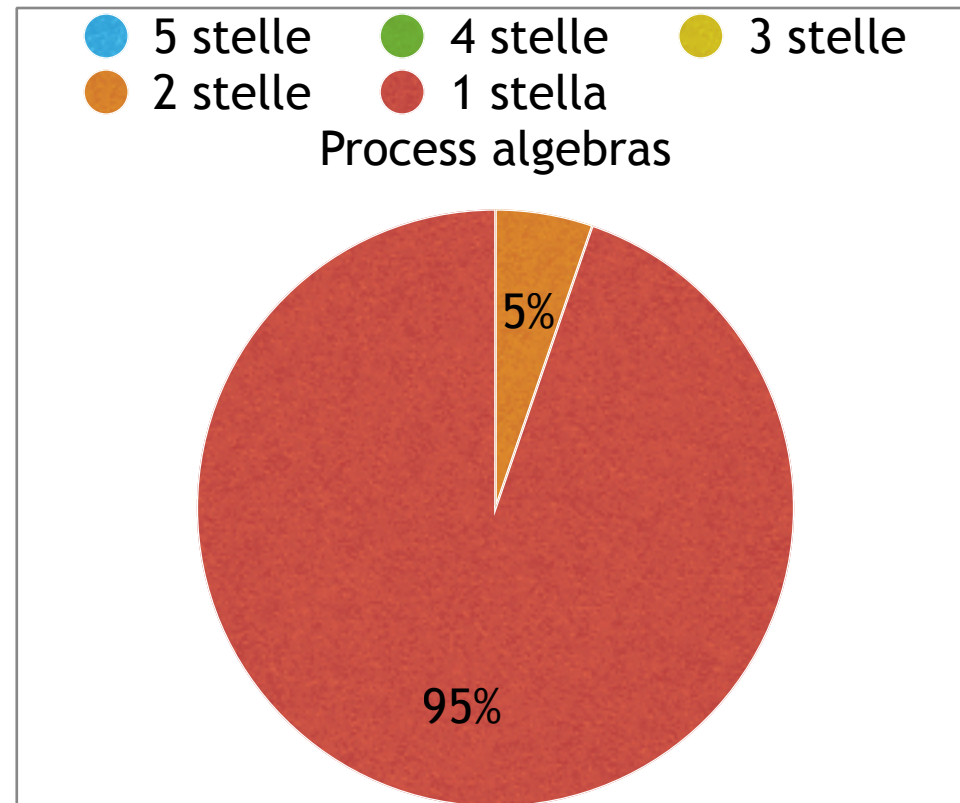


Example: vending machine



CCS syntax

From your forms



(over 8 answers)

CCS: syntax

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

(operators are listed in order of precedence)

CCS: syntax

$p, q ::=$

- $\mathbf{nil}$
- x
- $\mu.p$
- $p \backslash \alpha$
- $p[\phi]$
- $p + q$
- $p | q$
- $\mathbf{rec } x. p$

$\mathbf{rec } x. \text{coffee}.x + \text{tea}.\mathbf{nil} \mid \text{water}.\mathbf{nil}$

to be read as

$\mathbf{rec } x. (((\text{coffee}.x) + \text{tea}.\mathbf{nil}) \mid \text{water}.\mathbf{nil})$

(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

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

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

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

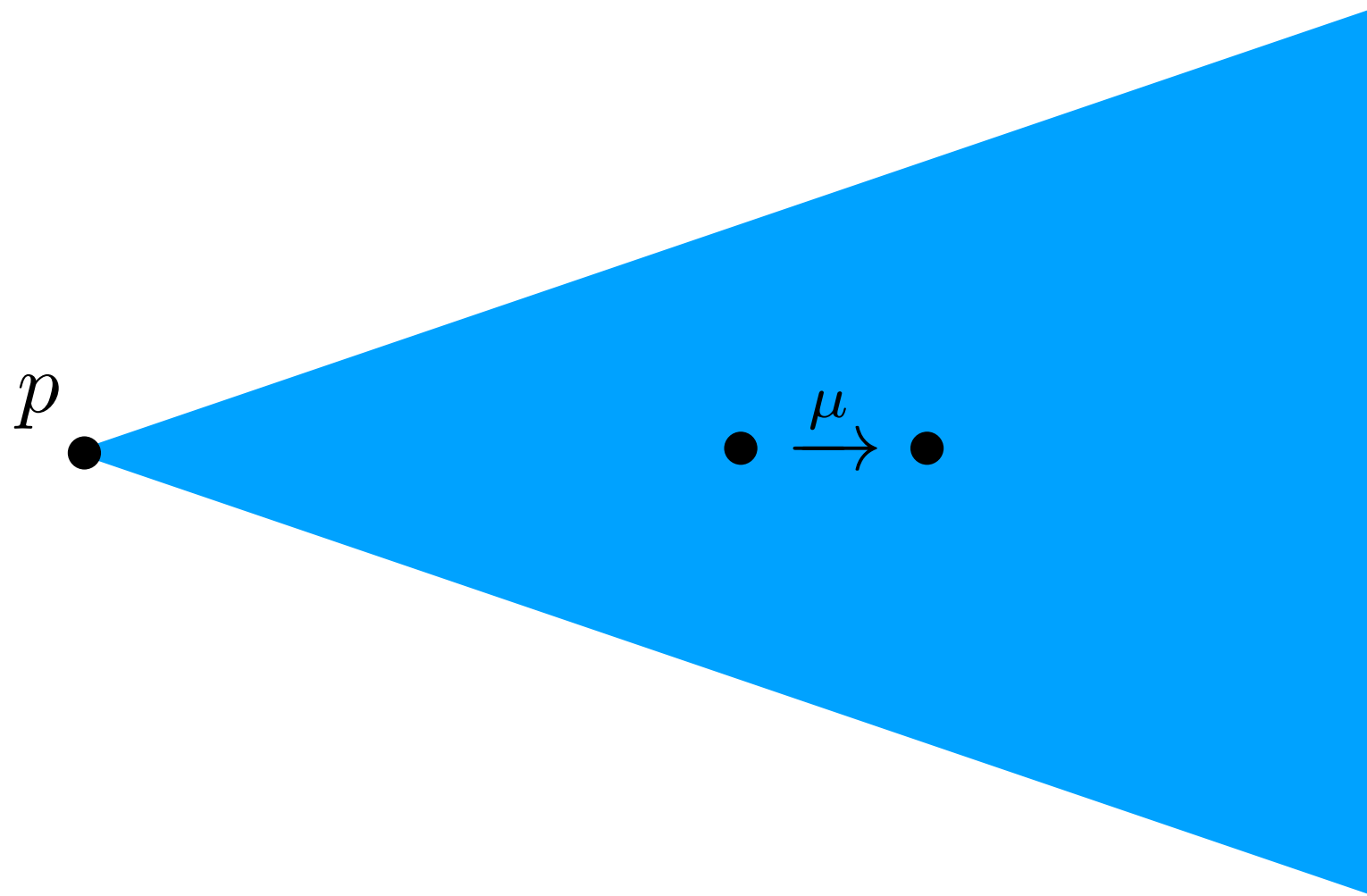
CCS operational semantics

CCS: labels

- $\mathcal{C}$ set of (input) actions, ranged by α
- $\bar{\mathcal{C}}$ set of (output) co-actions, ranged by $\bar{\alpha}$
- $\mathcal{C} \cap \bar{\mathcal{C}} = \emptyset$
- $\Lambda = \mathcal{C} \cup \bar{\mathcal{C}}$ set of observable actions, ranged by λ $\bar{\lambda}$
- $\tau \notin \Lambda$ 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

$\mathbf{nil} \nrightarrow$

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

nil



Action prefix

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

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

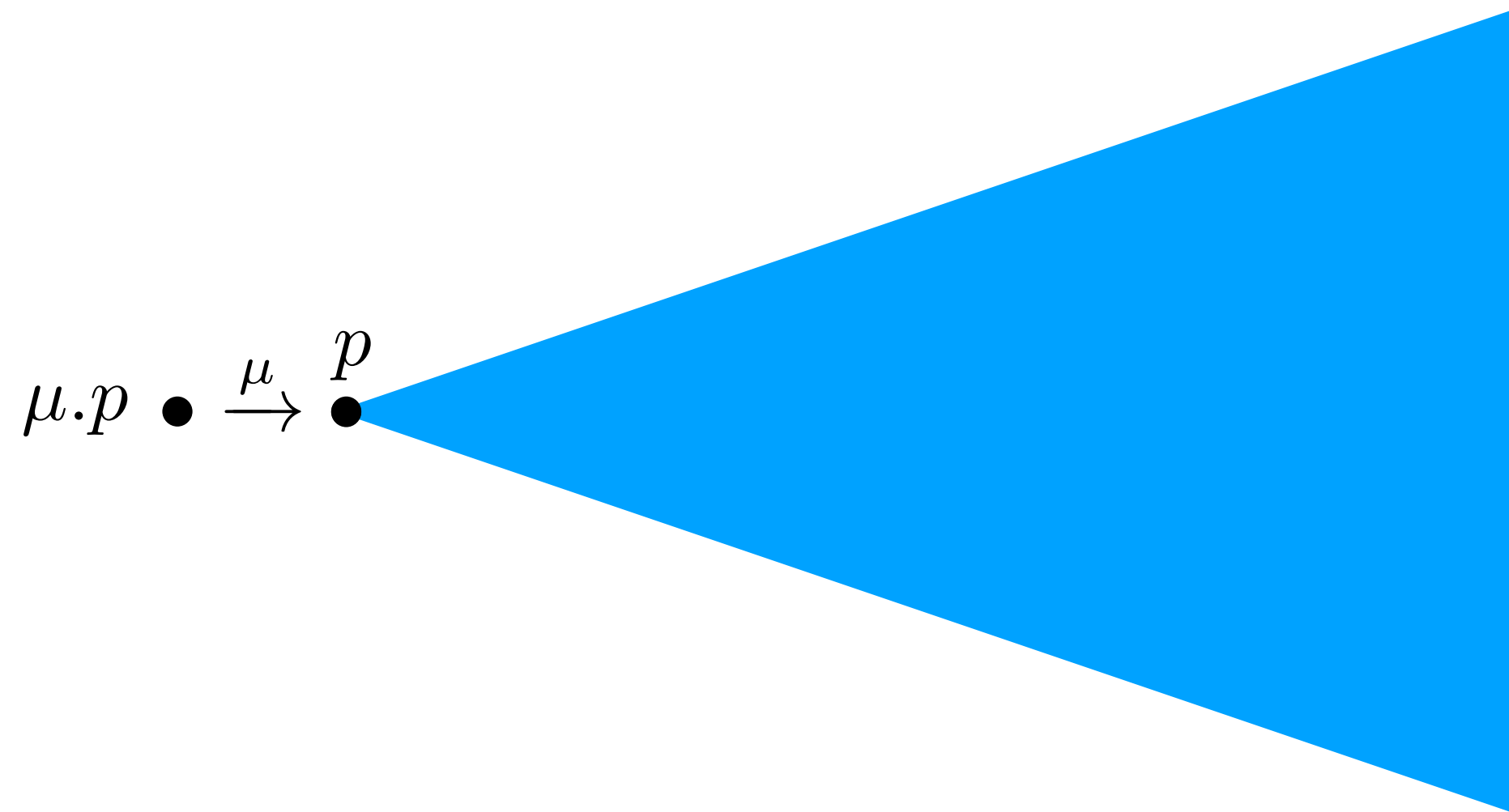
the action may involve an interaction with the environment

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

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

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

LTS of a process



Nondeterministic choice

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

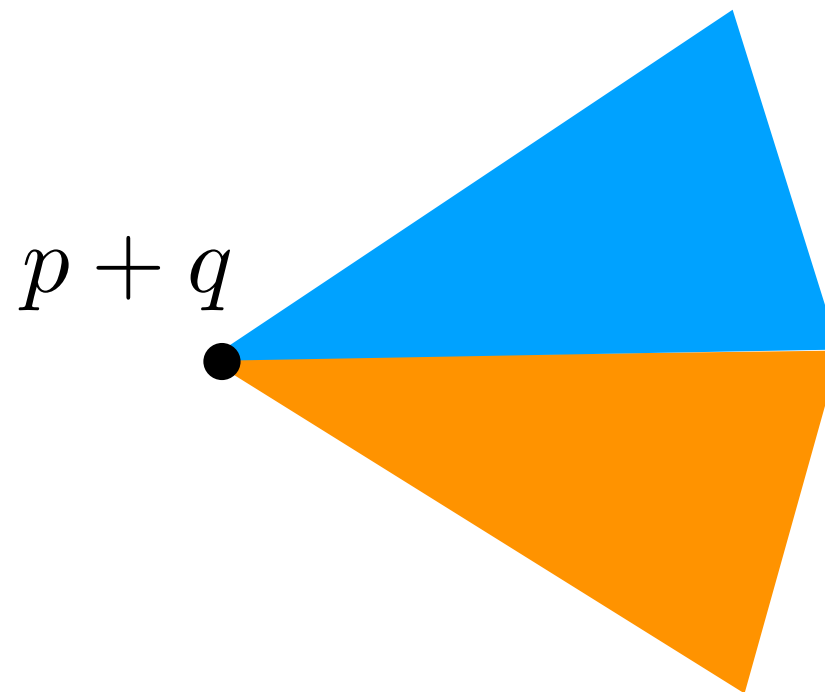
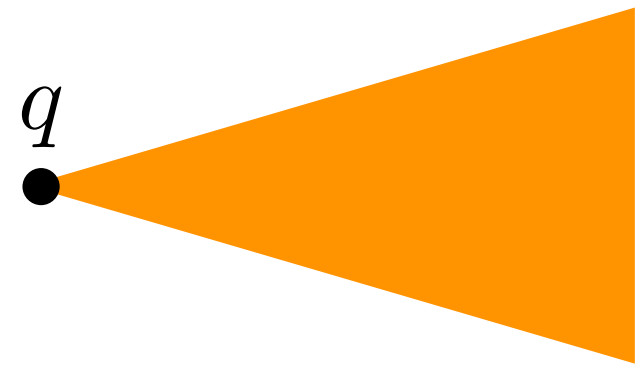
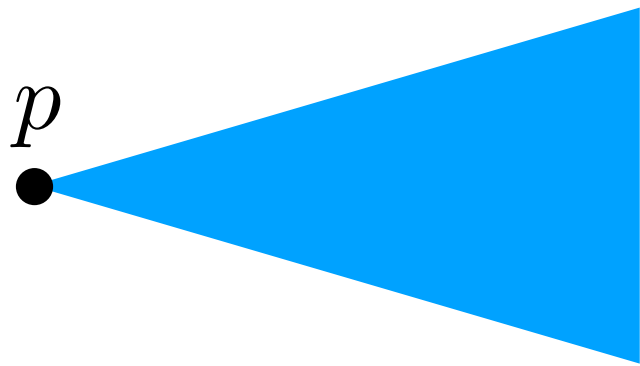
process $p_1 + p_2$ can behave either as p_1 or as p_2

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

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

$$\begin{array}{c} \text{coin}.\overline{\text{coffee}}.\mathbf{nil} + \overline{\text{tea}}.\mathbf{nil} \\ \downarrow \text{coin} \\ \overline{\text{coffee}}.\mathbf{nil} + \overline{\text{tea}}.\mathbf{nil} \\ \begin{array}{c} \overline{\text{coffee}} \quad \overline{\text{tea}} \\ \downarrow \quad \downarrow \\ \mathbf{nil} \end{array} \end{array}$$

LTS of a process



Recursion

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

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

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

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

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

$$\begin{array}{c} \overline{coffee}.\mathbf{rec} \ x. \ coin.(\overline{coffee.x} + \overline{tea.nil}) + \overline{tea.nil} \\ \downarrow \overline{tea} \\ \mathbf{nil} \end{array}$$

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

$$\begin{array}{c} \overline{coffee}.P + \overline{tea.nil} \\ \downarrow \overline{tea} \\ \mathbf{nil} \end{array}$$

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

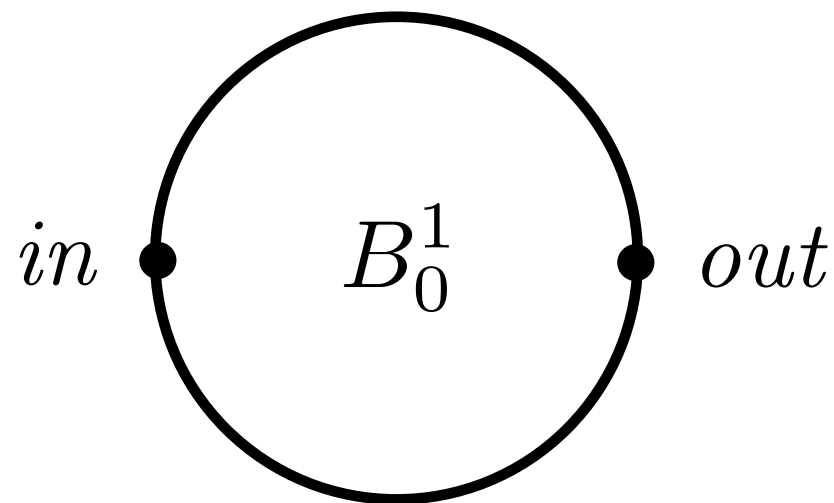
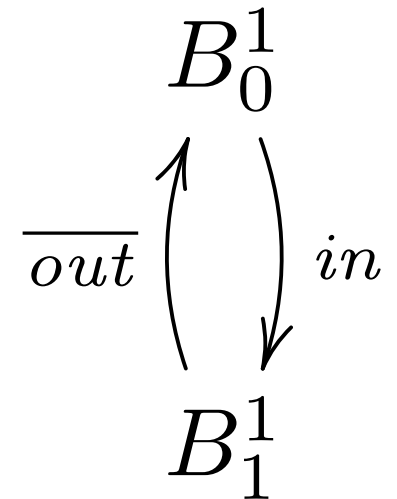
$$\text{Const) } \frac{A \triangleq p \in \Delta \quad p \xrightarrow{\mu} q}{A \xrightarrow{\mu} q}$$

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

CCS: capacity 1 buffer

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

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

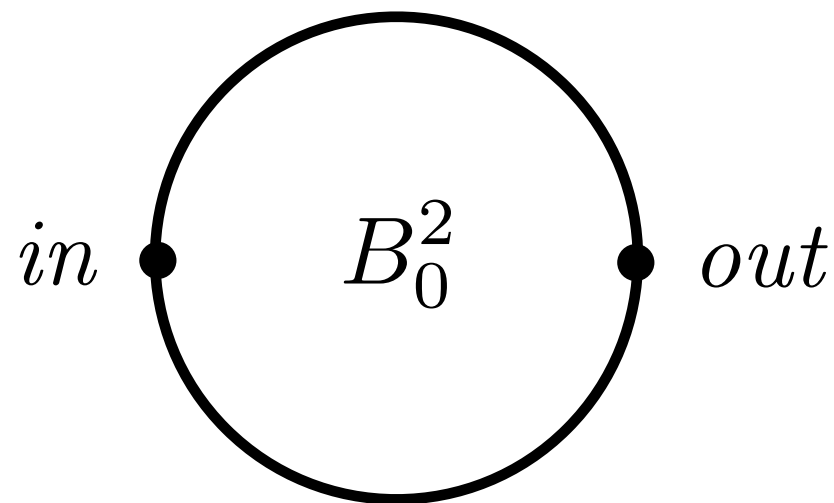
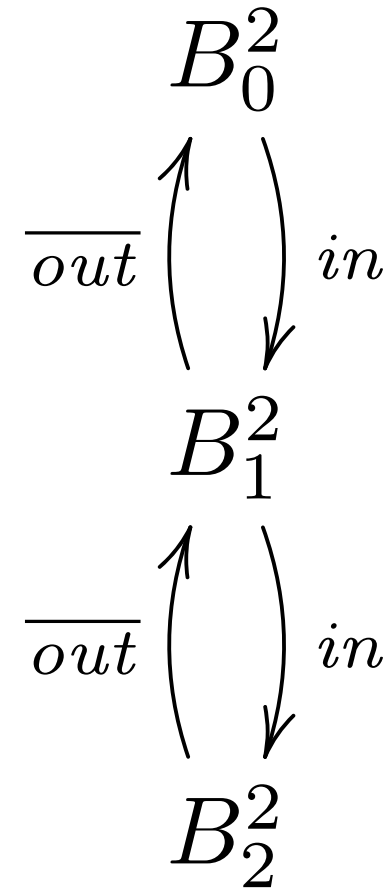


CCS: capacity 2 buffer

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

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

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

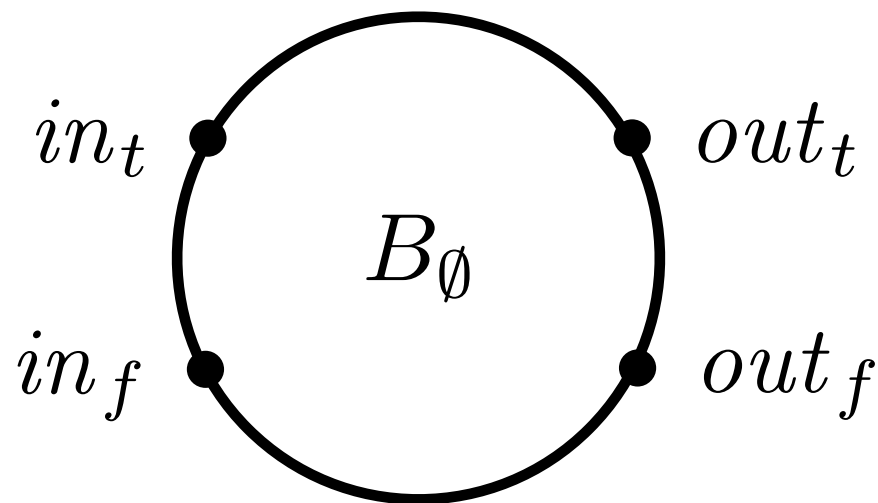
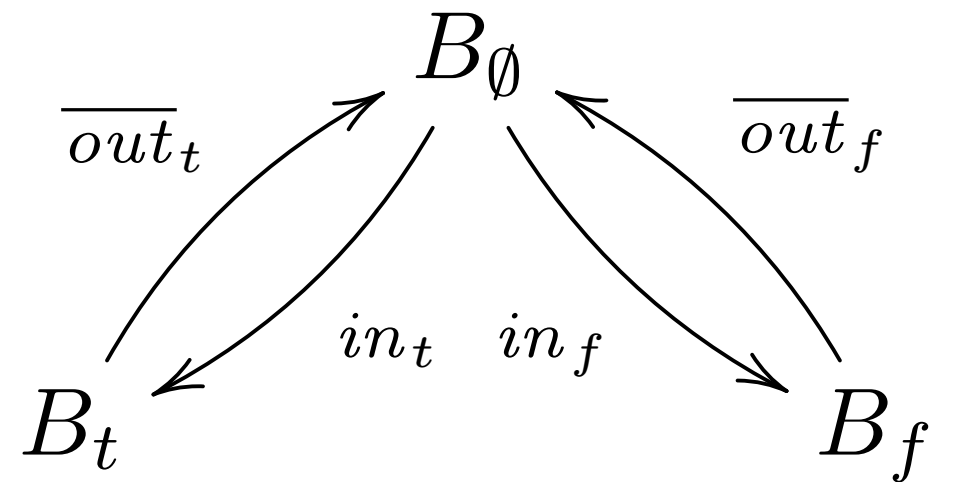


CCS: boolean buffer

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

$$B_t \triangleq \overline{out_t}.B_{\emptyset}$$

$$B_f \triangleq \overline{out_f}.B_{\emptyset}$$



Parallel composition

$$\begin{array}{c} \text{ParL)} \frac{p_1 \xrightarrow{\mu} q_1}{p_1 | p_2 \xrightarrow{\mu} q_1 | p_2} \quad \text{Com)} \frac{p_1 \xrightarrow{\lambda} q_1 \quad p_2 \xrightarrow{\bar{\lambda}} q_2}{p_1 | p_2 \xrightarrow{\tau} q_1 | q_2} \quad \text{ParR)} \frac{p_2 \xrightarrow{\mu} q_2}{p_1 | p_2 \xrightarrow{\mu} p_1 | q_2} \end{array}$$

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

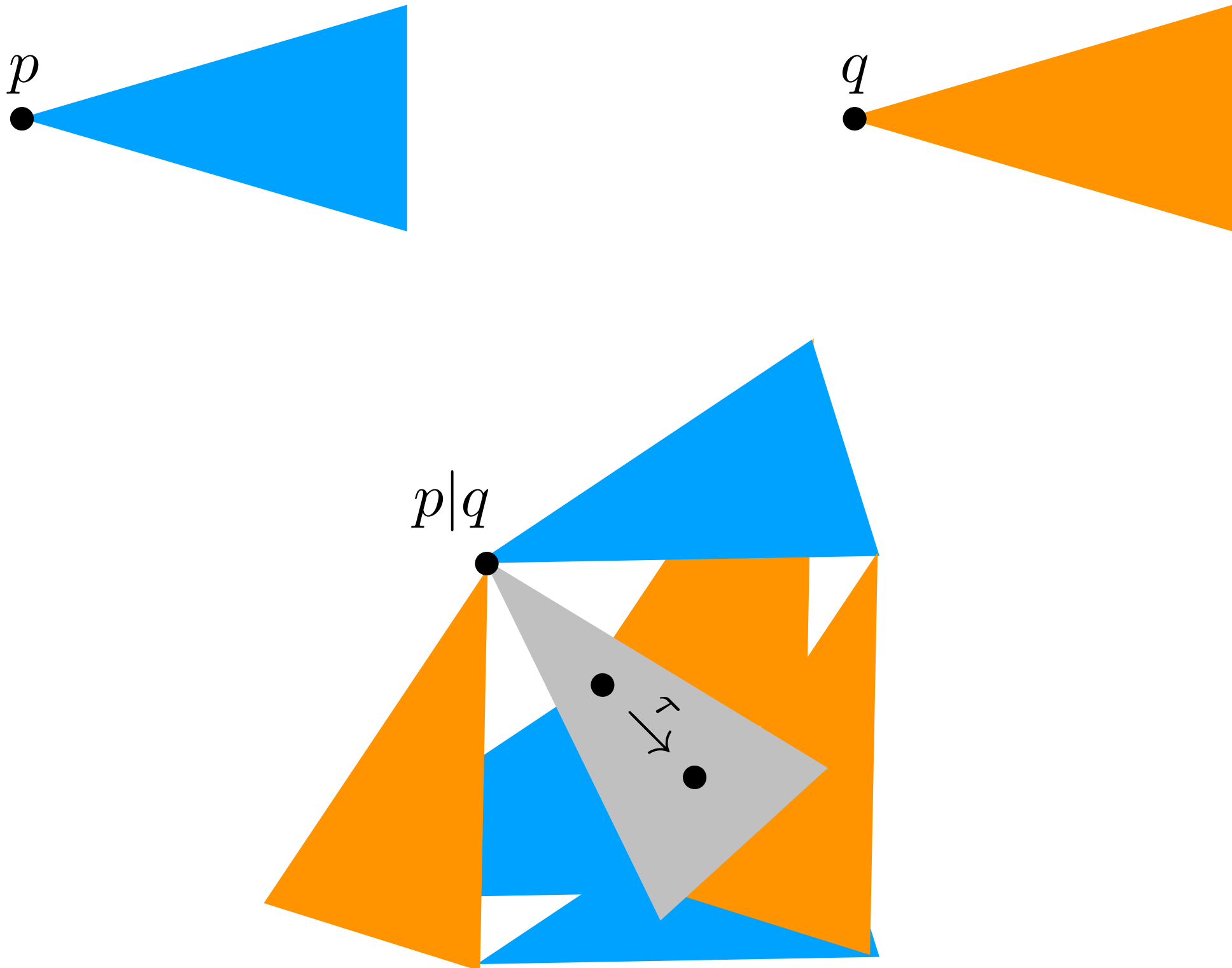
$$P \triangleq \overline{coin}.coffee.nil \qquad M \triangleq coin.(\overline{coffee}.nil + \overline{tea}.nil)$$

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

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

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

LTS of a process



CCS: parallel buffers

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

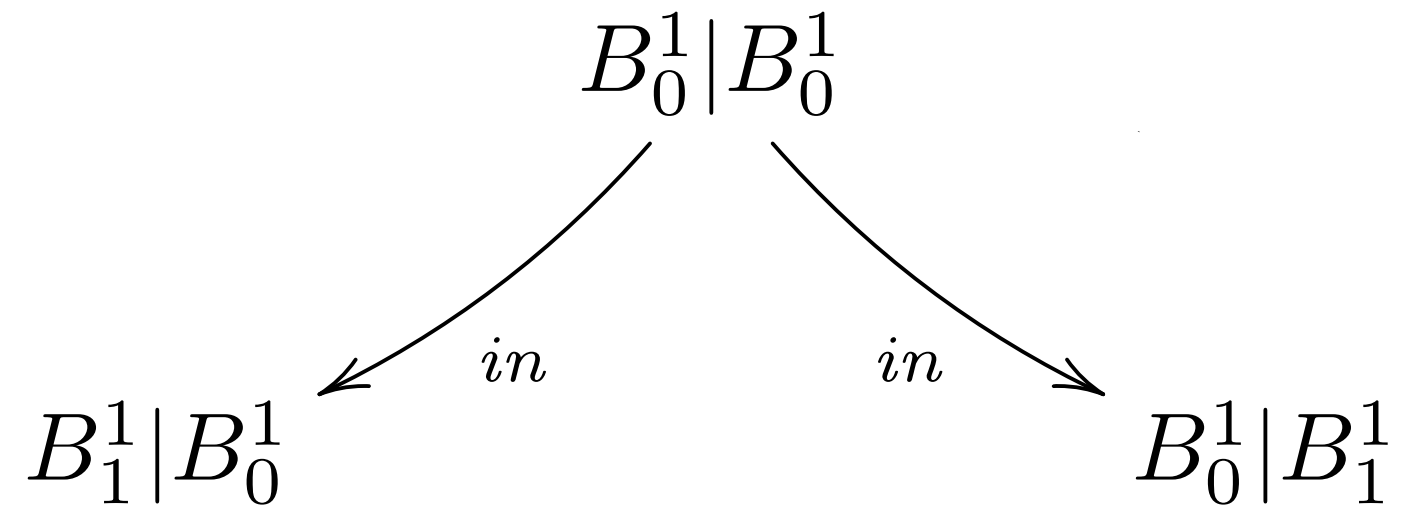
$$B_1^1 \triangleq \overline{out}.B_0^1$$

$$B_0^1 | B_0^1$$

CCS: parallel buffers

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

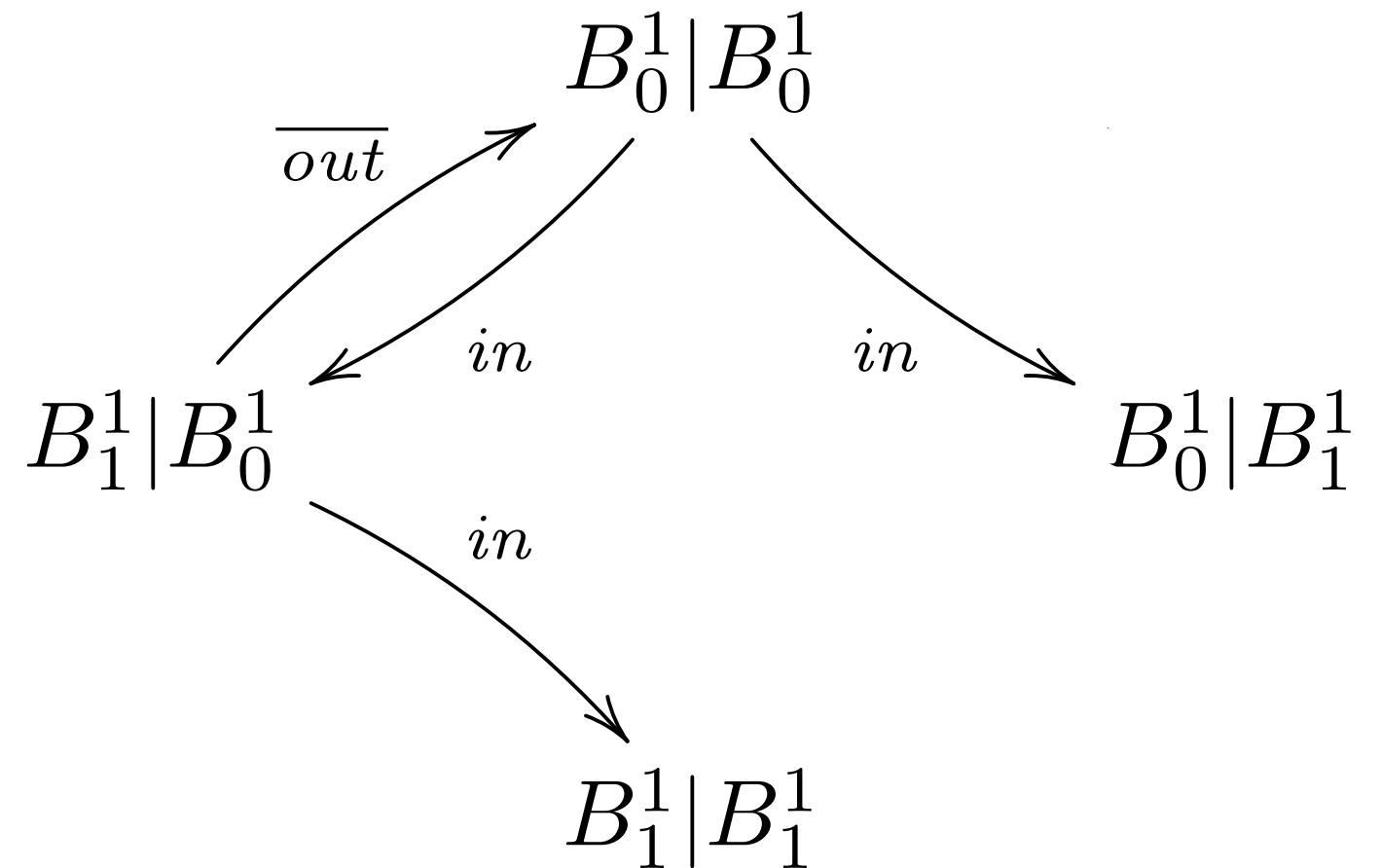
$$B_1^1 \triangleq \overline{out}.B_0^1$$



CCS: parallel buffers

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

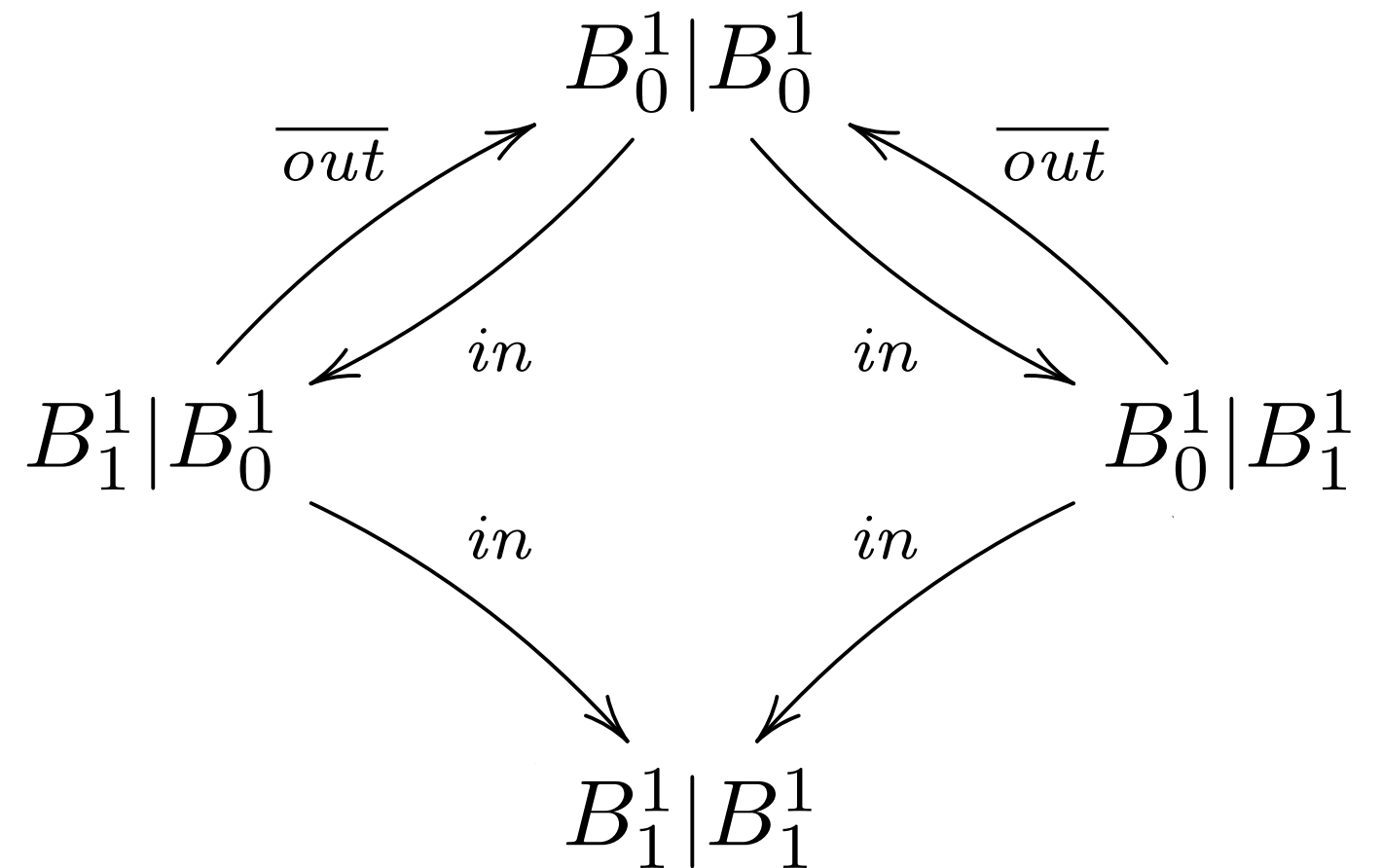
$$B_1^1 \triangleq \overline{out}.B_0^1$$



CCS: parallel buffers

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

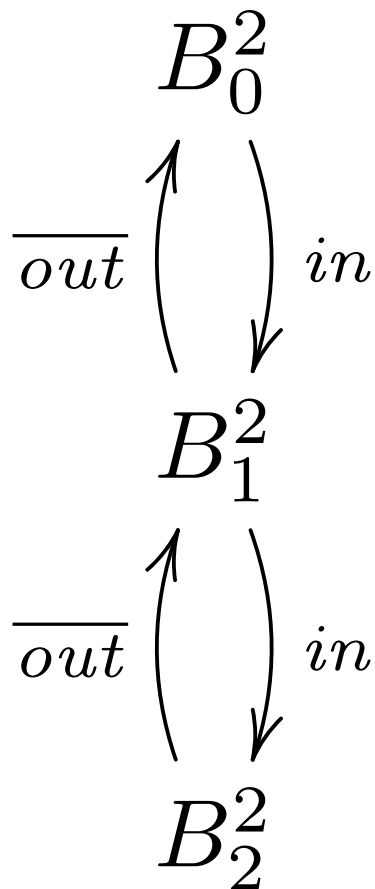
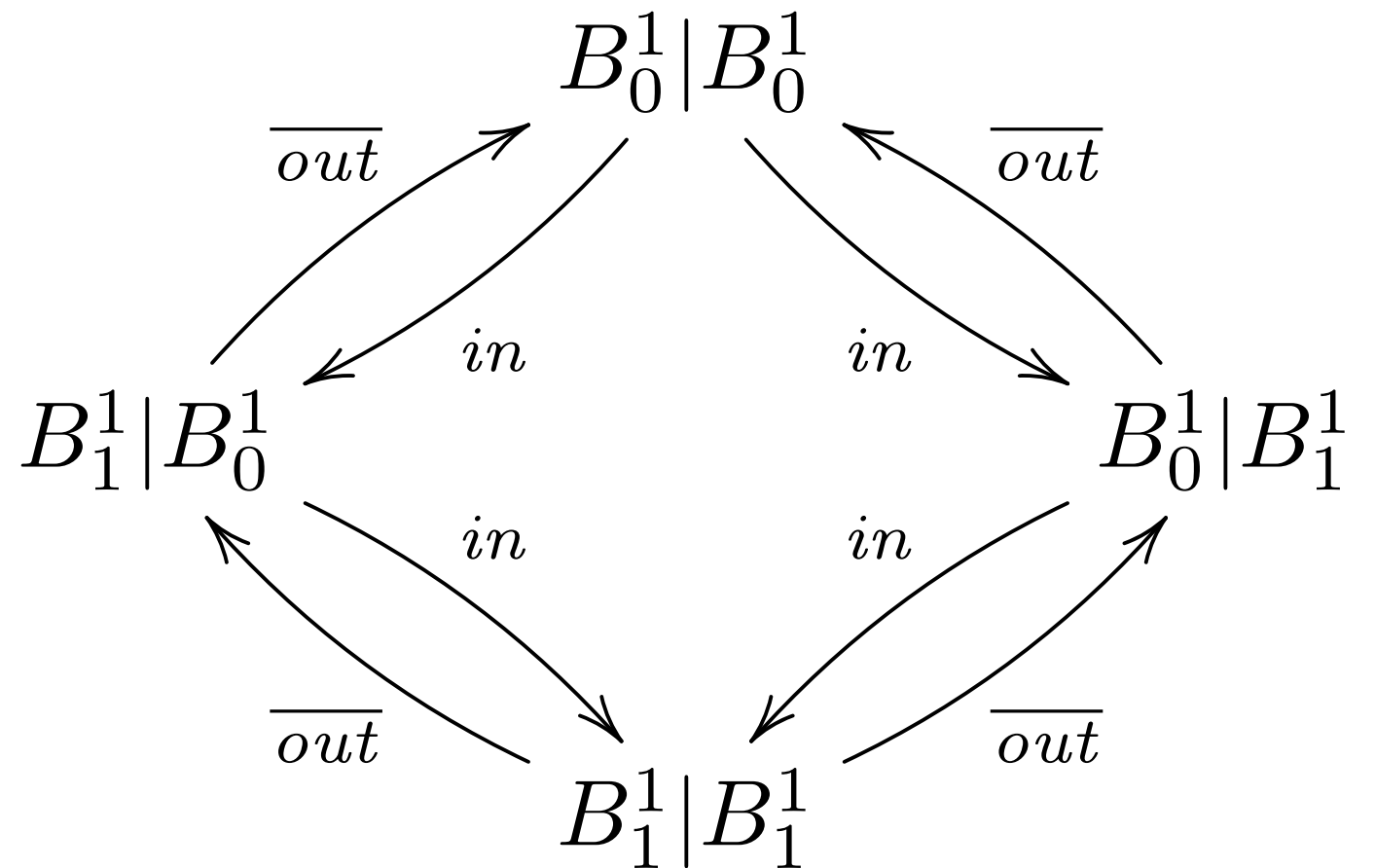
$$B_1^1 \triangleq \overline{out}.B_0^1$$



CCS: parallel buffers

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

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



compare with the 2-capacity buffer

Restriction

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

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

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

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

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

Restriction: shorthand

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

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

we omit trailing **nil**

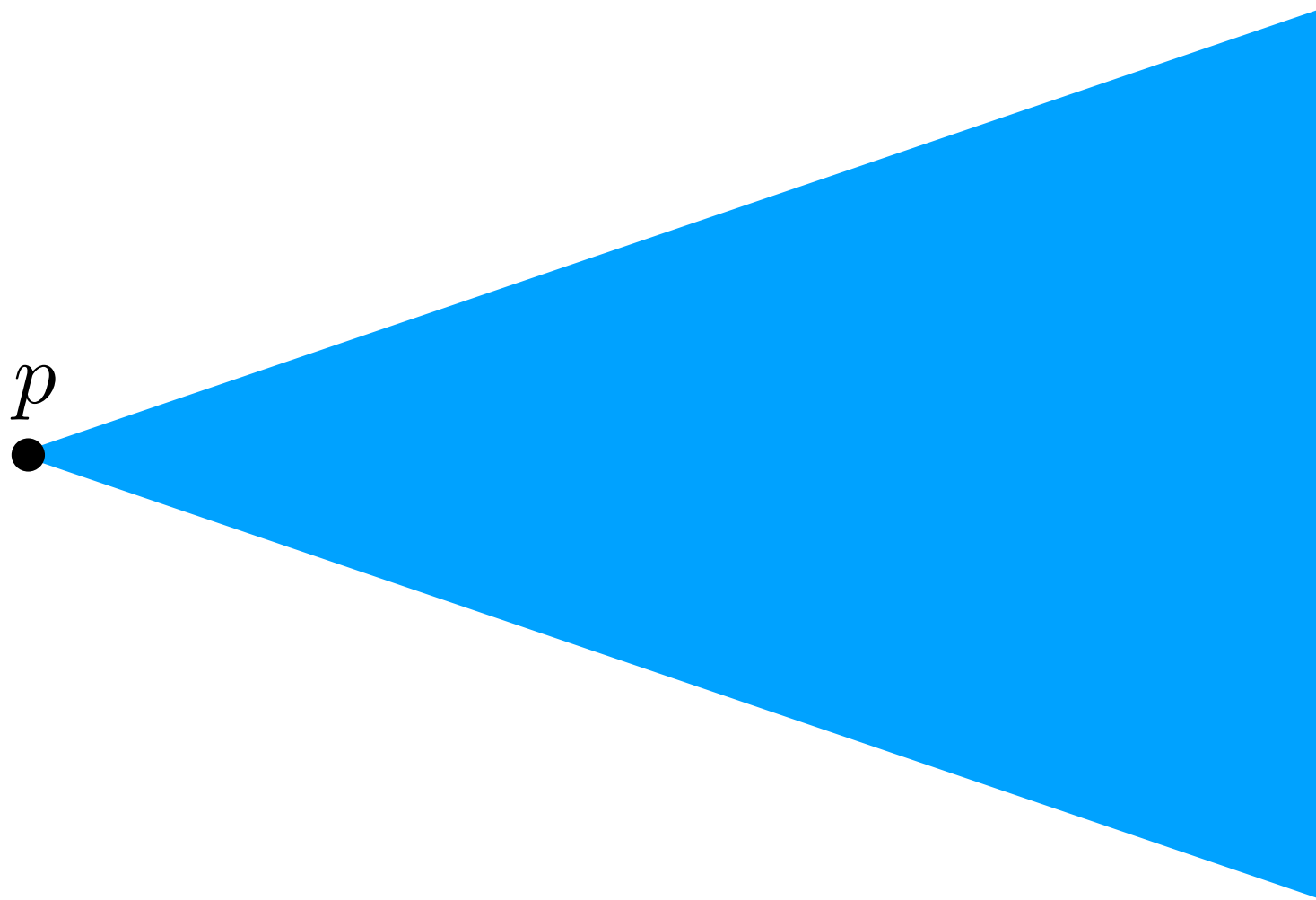
$$P \triangleq \overline{coin}.coffee$$

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

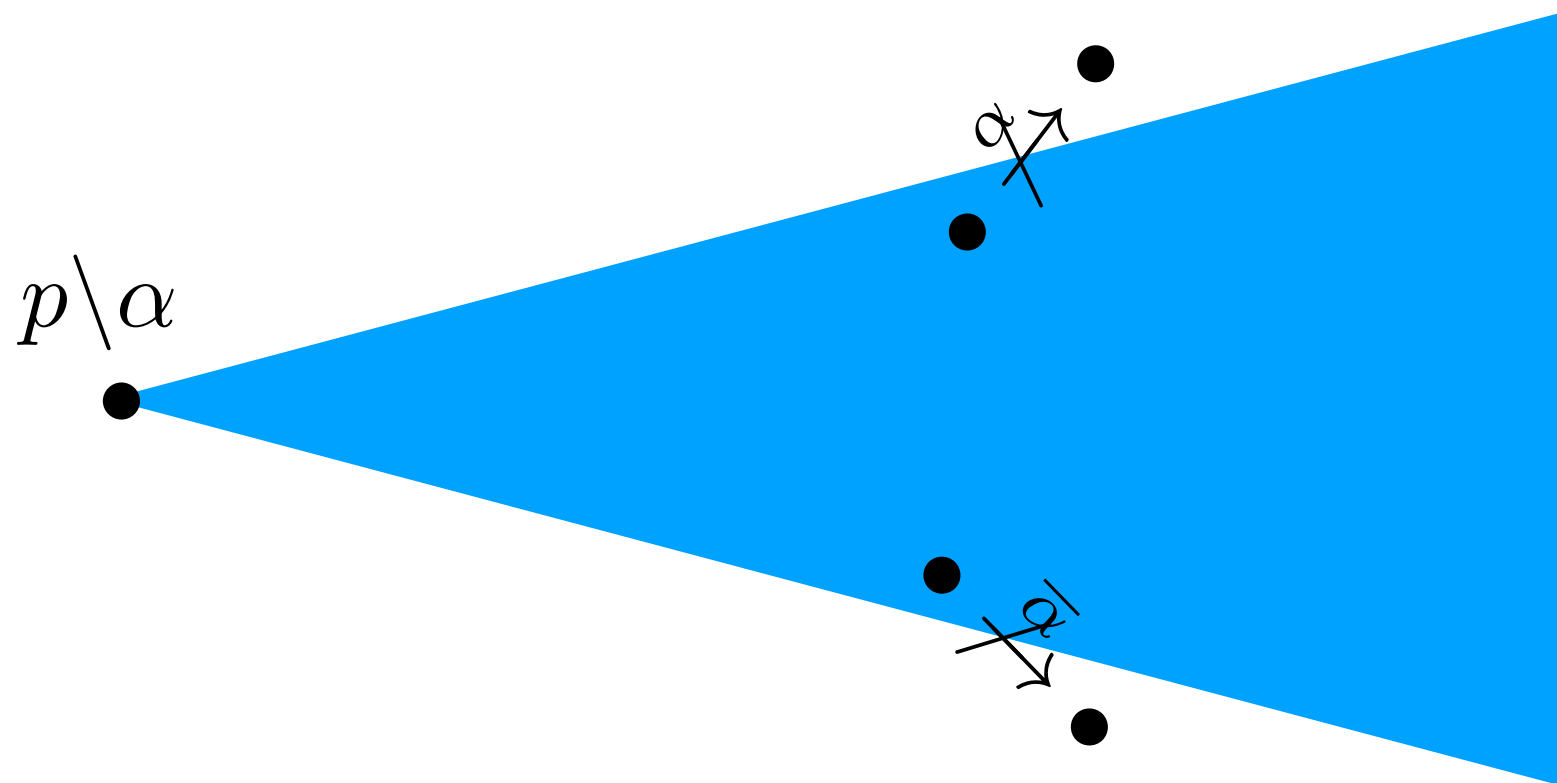
$$S \triangleq \{coin, coffee, tea\}$$

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

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

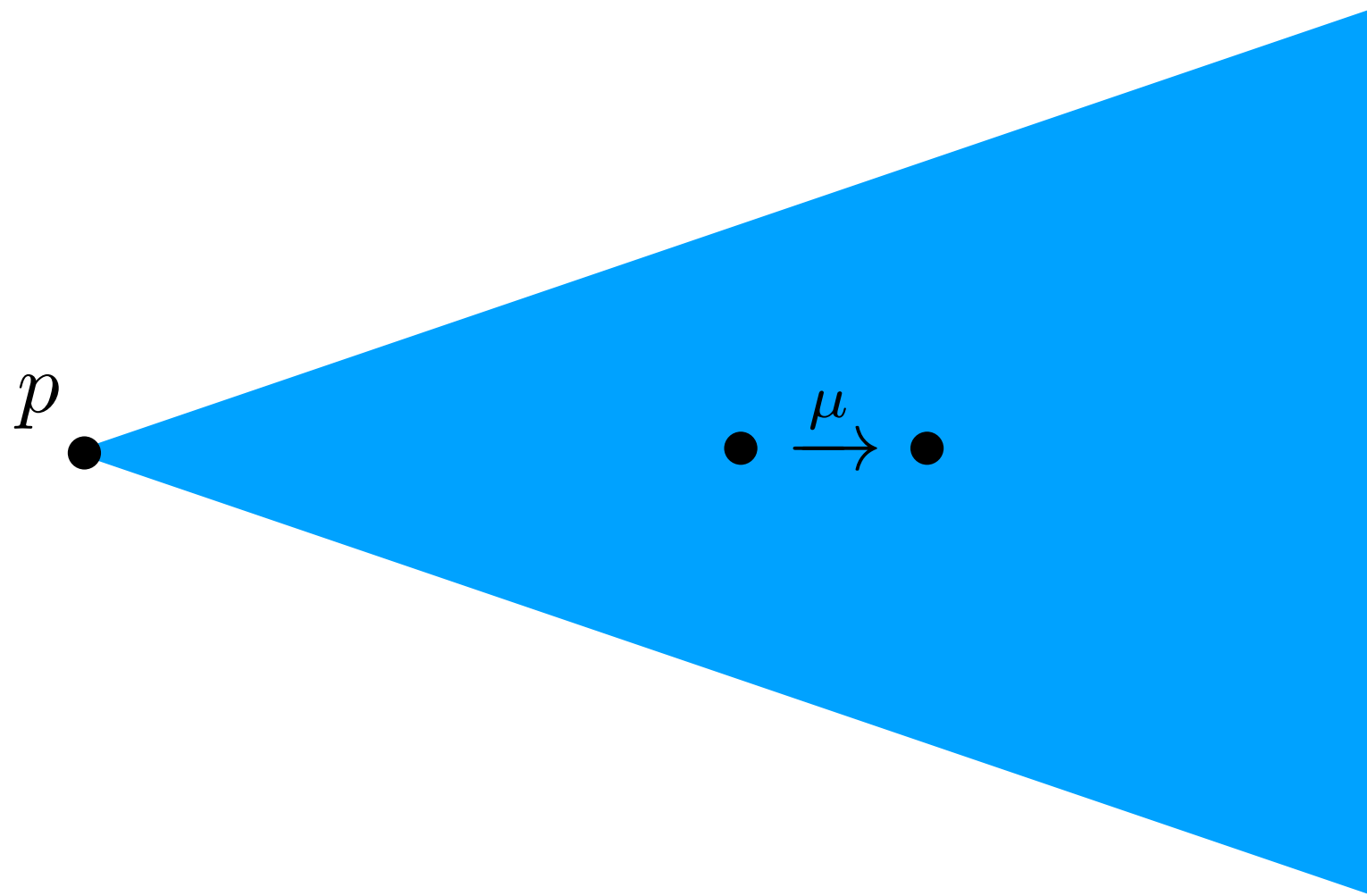
we assume $\phi(\tau) = \tau$ $\phi(\bar{\lambda}) = \overline{\phi(\lambda)}$

allows one to reuse processes

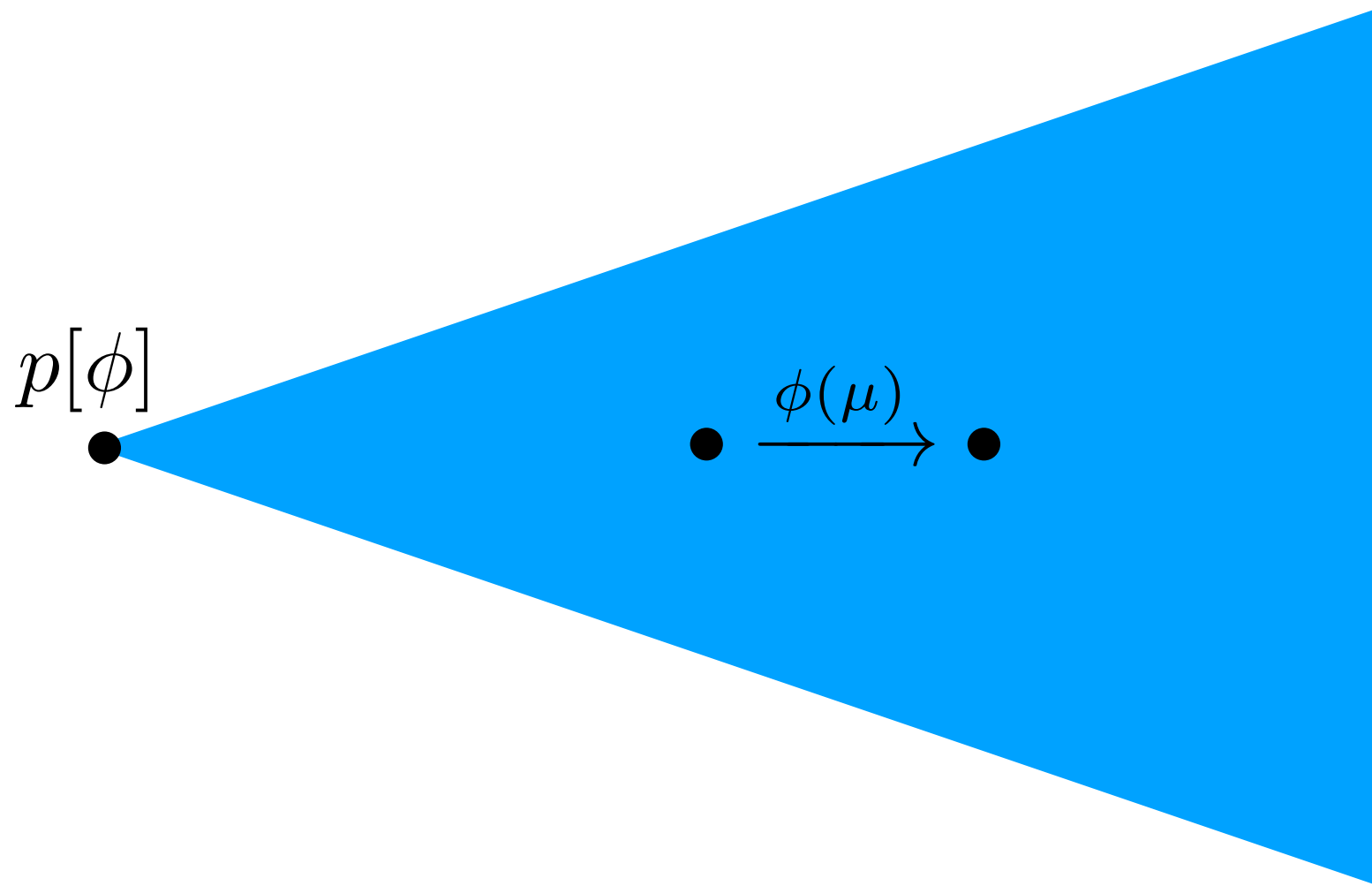
$P \triangleq \overline{coin}.coffee$ $\phi(coin) = moneta$
 $\phi(coffee) = caffè$

$$P[\phi] \xrightarrow{\overline{moneta}} coffee[\phi] \xrightarrow{caffè} \mathbf{nil}[\phi]$$

LTS of a process



LTS of a process



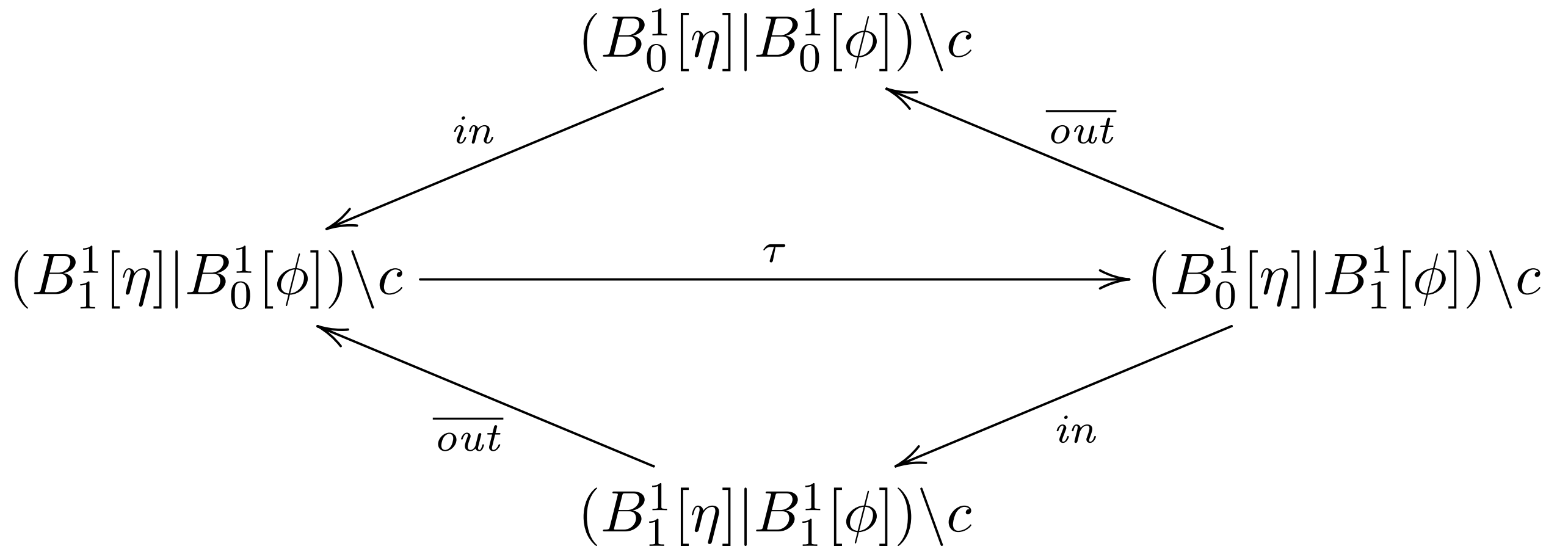
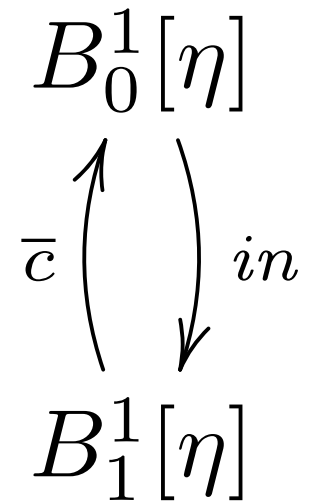
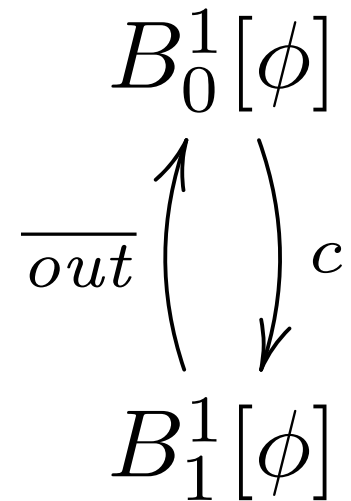
CCS op. semantics

$$\begin{array}{c}
 \text{Act)} \frac{}{\mu.p \xrightarrow{\mu} p} \qquad \text{Res)} \frac{p \xrightarrow{\mu} q \quad \mu \notin \{\alpha, \bar{\alpha}\}}{p \setminus \alpha \xrightarrow{\mu} q \setminus \alpha} \qquad \text{Rel)} \frac{p \xrightarrow{\mu} q}{p[\phi] \xrightarrow{\phi(\mu)} q[\phi]} \\
 \\
 \text{SumL)} \frac{p_1 \xrightarrow{\mu} q}{p_1 + p_2 \xrightarrow{\mu} q} \qquad \text{SumR)} \frac{p_2 \xrightarrow{\mu} q}{p_1 + p_2 \xrightarrow{\mu} q} \\
 \\
 \text{ParL)} \frac{p_1 \xrightarrow{\mu} q_1}{p_1 | p_2 \xrightarrow{\mu} q_1 | p_2} \qquad \text{Com)} \frac{p_1 \xrightarrow{\lambda} q_1 \quad p_2 \xrightarrow{\bar{\lambda}} q_2}{p_1 | p_2 \xrightarrow{\tau} q_1 | q_2} \qquad \text{ParR)} \frac{p_2 \xrightarrow{\mu} q_2}{p_1 | p_2 \xrightarrow{\mu} p_1 | q_2} \\
 \\
 \text{Rec)} \frac{p[\mathbf{rec} \ x. \ p / x] \xrightarrow{\mu} q}{\mathbf{rec} \ x. \ p \xrightarrow{\mu} q}
 \end{array}$$

Linked buffers

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

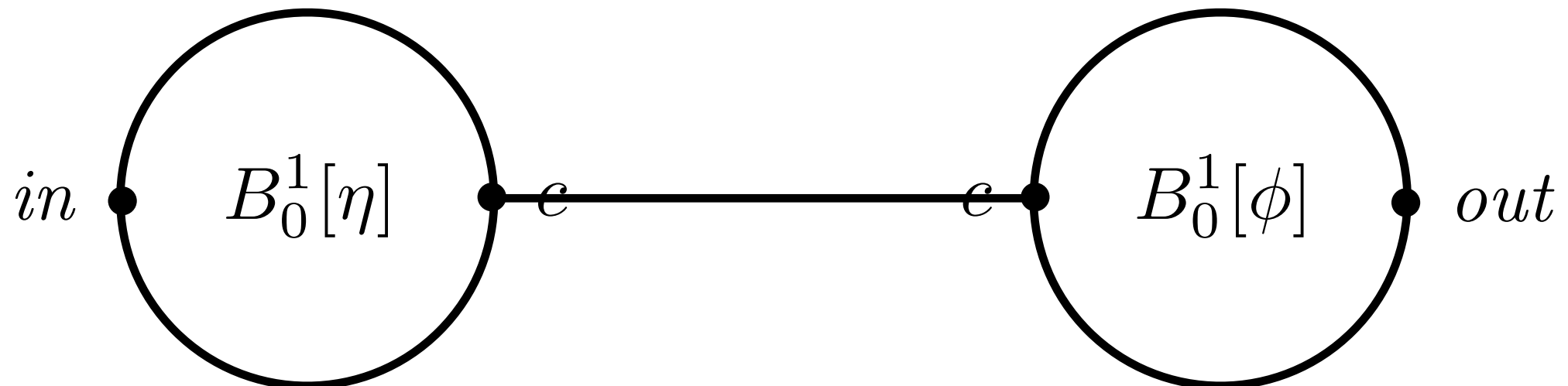
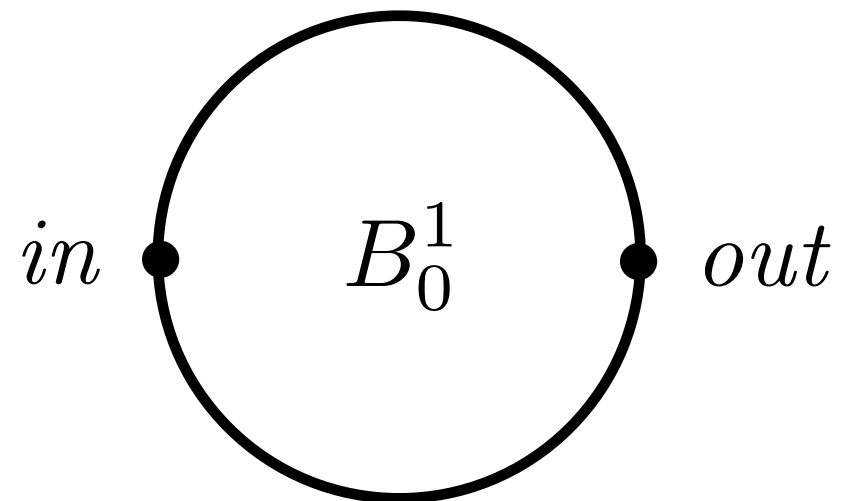
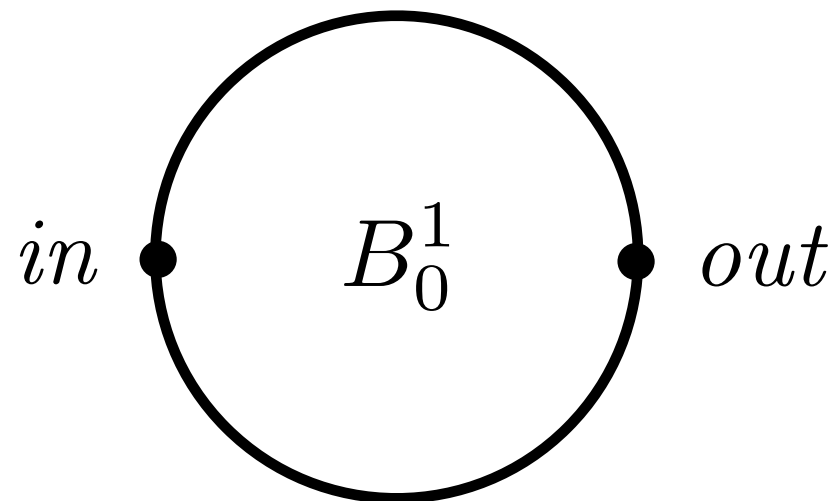
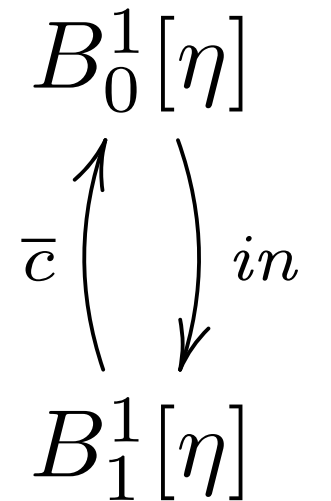
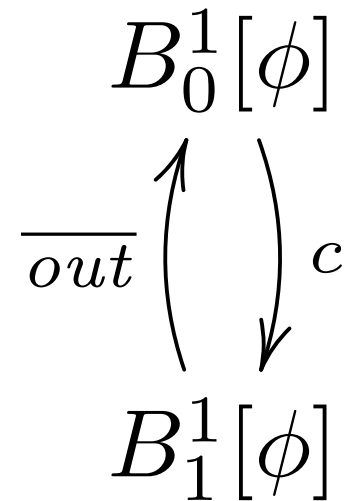
$$B_1^1 \triangleq \overline{out}.B_0^1 \quad \phi(in) = c$$



Linked buffers

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

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

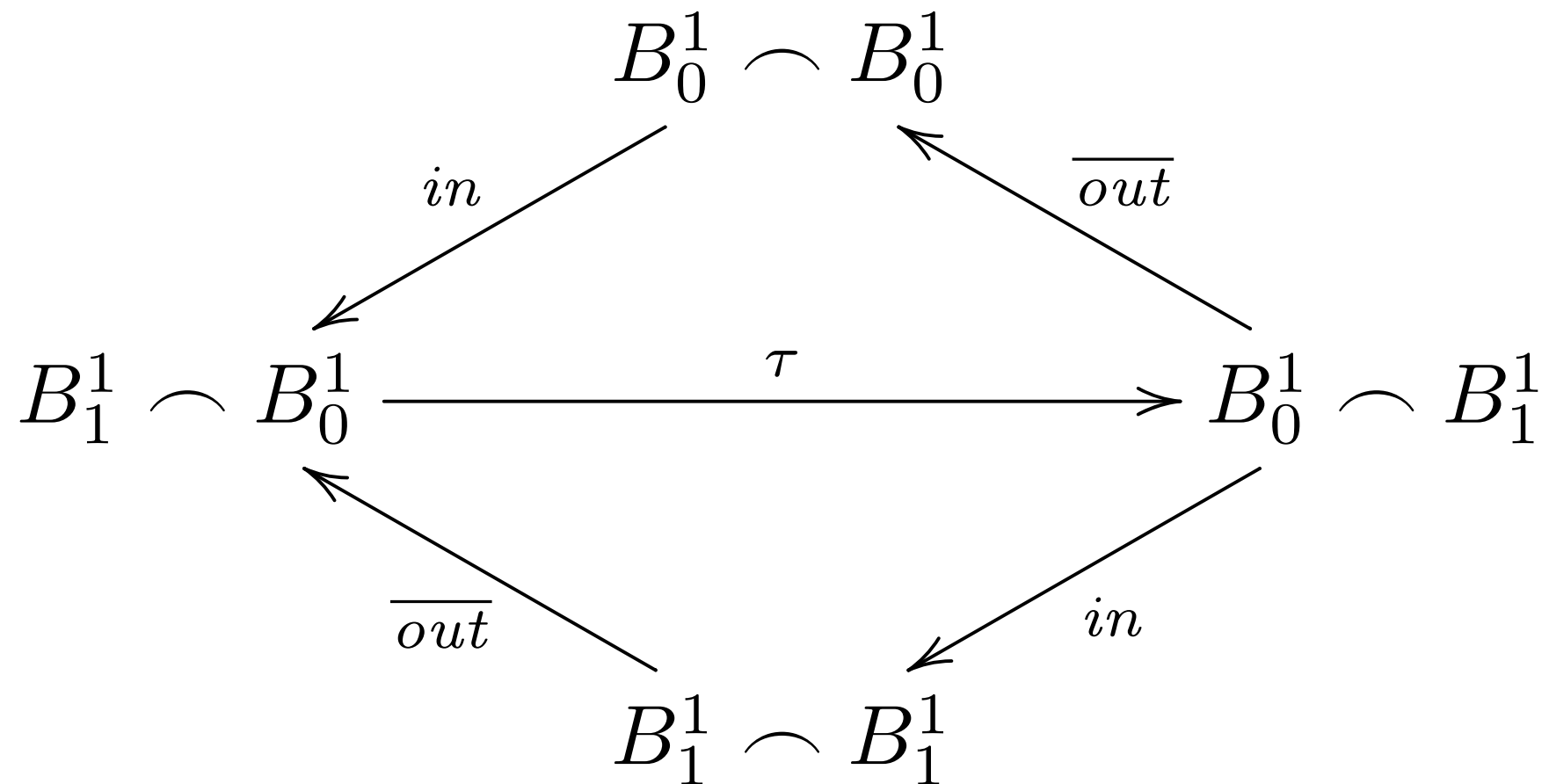


Linked buffers

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

$$p \frown q \triangleq (p[\eta] \parallel q[\phi]) \setminus c$$

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



Linked boolean buffers

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

$$B_t \triangleq \overline{out_t}.B_{\emptyset} \quad \eta(out_f) = c_f \quad \phi(in_f) = c_f$$

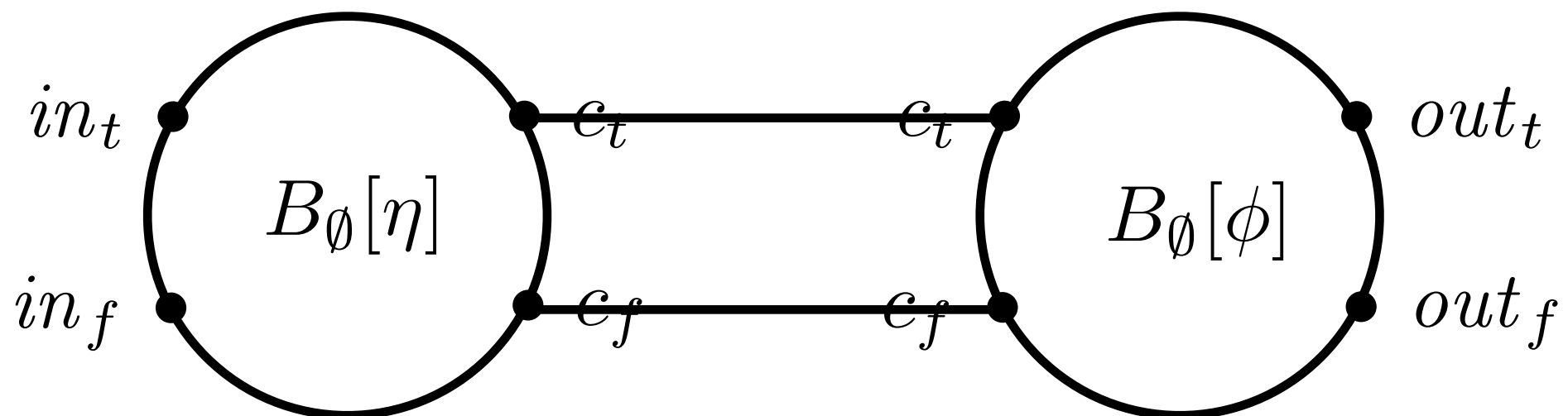
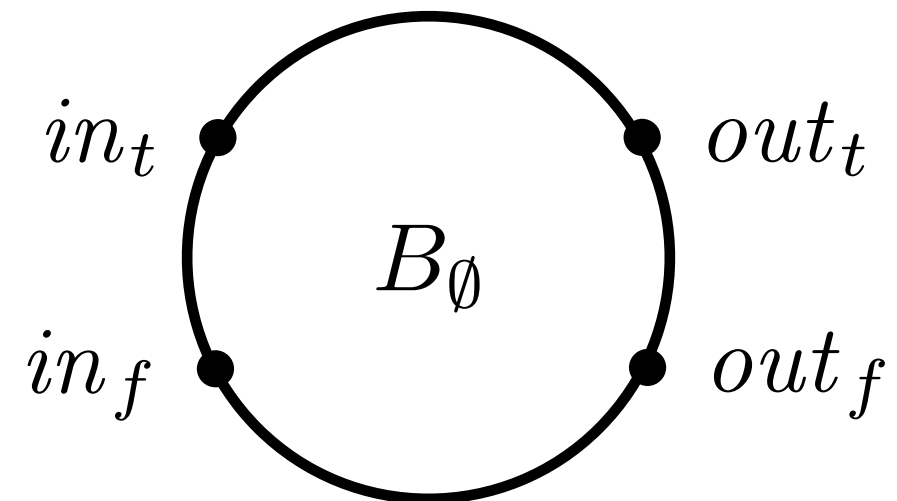
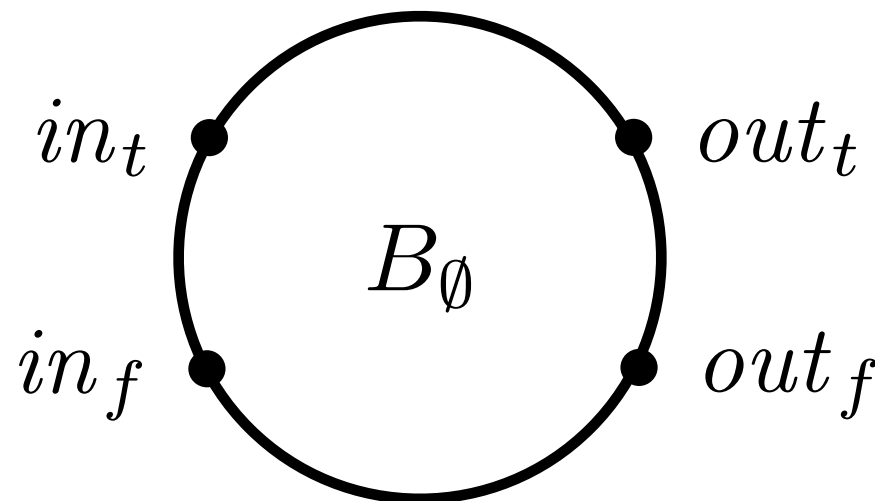
$$B_f \triangleq \overline{out_f}.B_{\emptyset} \quad p \frown q \triangleq (p[\eta]|q[\phi]) \setminus \{c_t, c_f\}$$

Linked boolean buffers

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

$$B_t \triangleq \overline{out_t}.B_{\emptyset}$$

$$B_f \triangleq \overline{out_f}.B_{\emptyset}$$

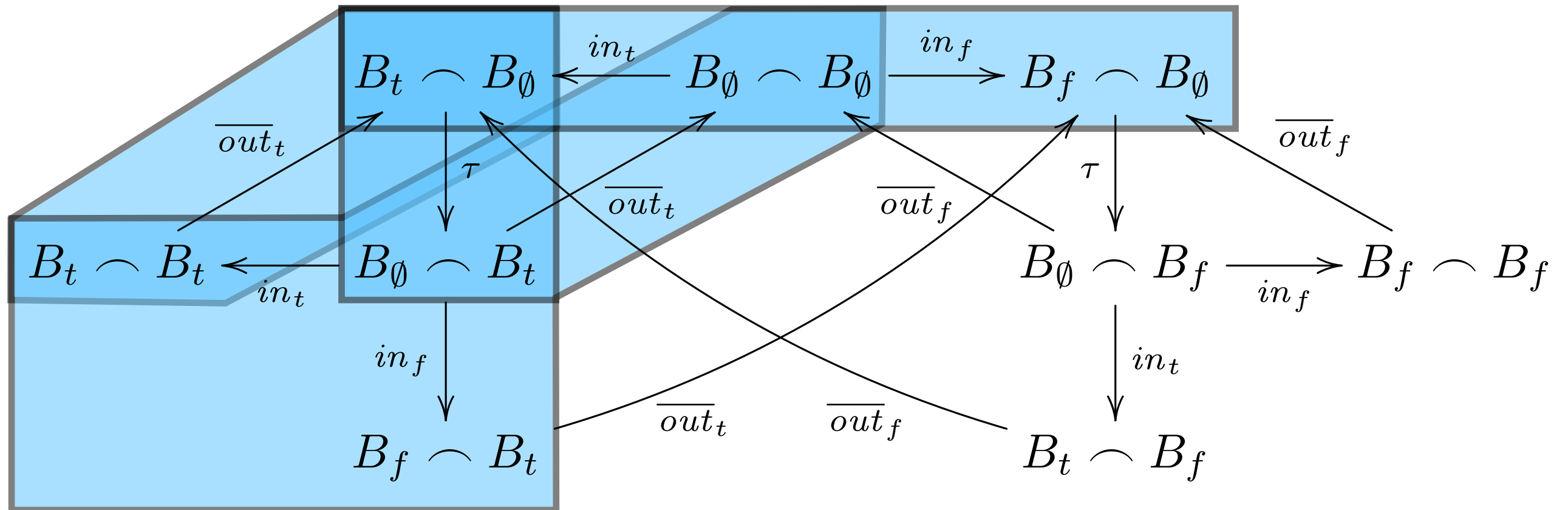


Linked boolean buffers

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

$$B_t \triangleq \overline{out_t}.B_\emptyset \quad \eta(out_f) = c_f \quad \phi(in_f) = c_f$$

$$B_f \triangleq \overline{out_f}.B_\emptyset \quad p \frown q \triangleq (p[\eta] | q[\phi]) \setminus \{c_t, c_f\}$$

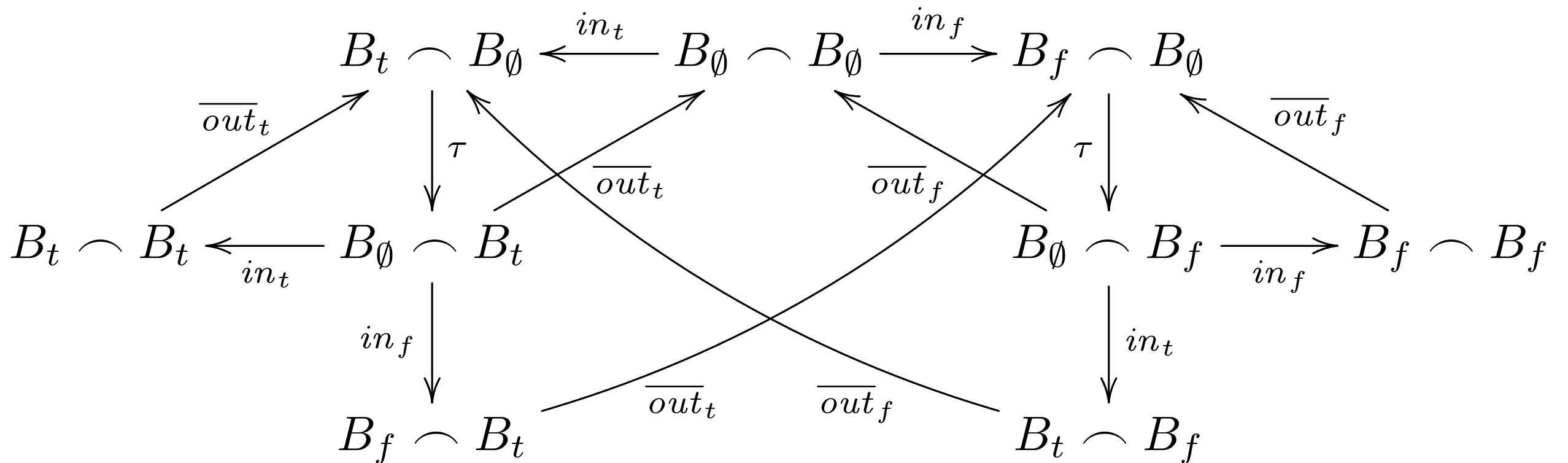


Linked boolean buffers

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

$$B_t \triangleq \overline{out_t}.B_\emptyset \quad \eta(out_f) = c_f \quad \phi(in_f) = c_f$$

$$B_f \triangleq \overline{out_f}.B_\emptyset \quad p \frown q \triangleq (p[\eta] | q[\phi]) \setminus \{c_t, c_f\}$$



CCS with value passing

$$\overline{\alpha!v.p} \xrightarrow{\quad} p$$

$$\overline{\alpha?x.p} \xrightarrow{\alpha_v} p[v/x]$$

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

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

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

receive

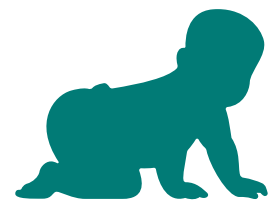
v -> p

w -> q

_ -> r

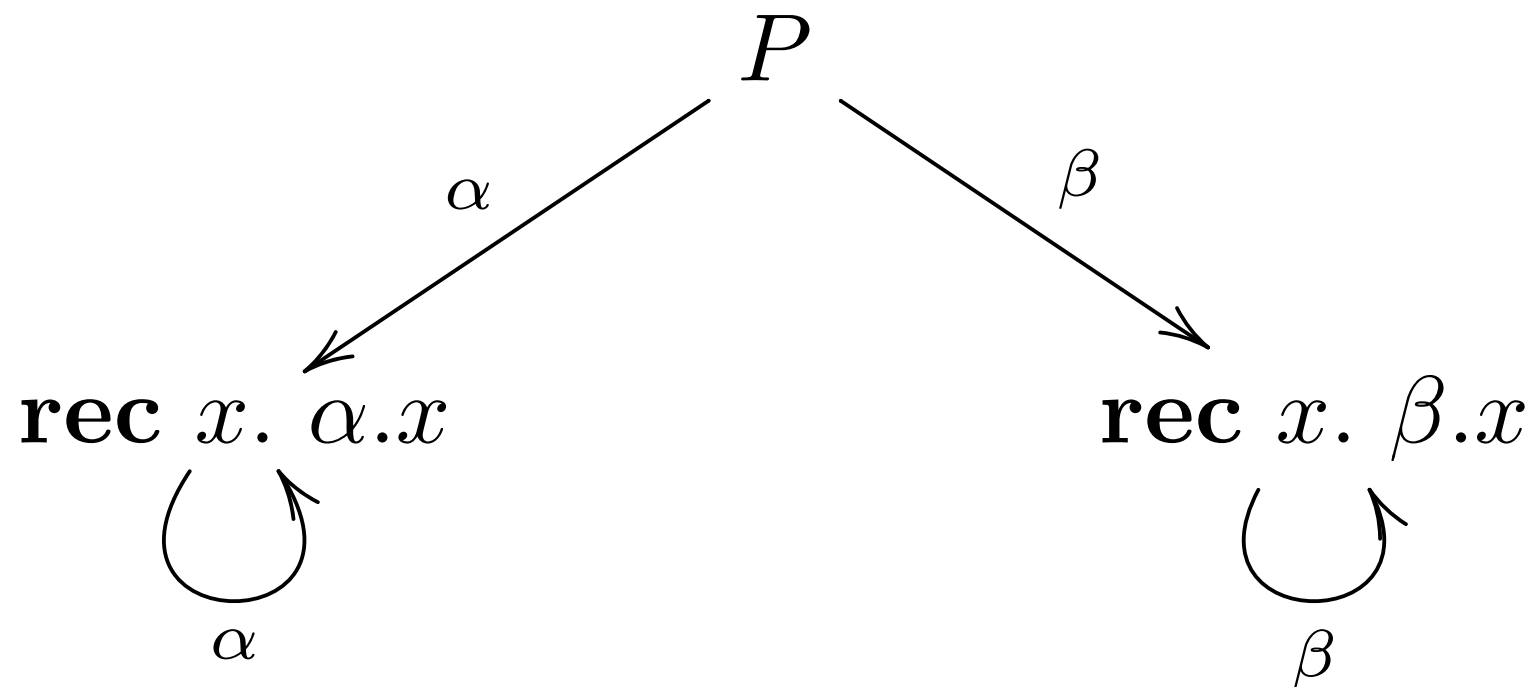
end

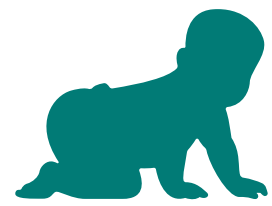
$$\equiv \alpha_v.p + \alpha_w.q + \sum_{z \neq v, w} \alpha_z.r$$



Exercise: LTS?

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





Exercise: LTS?

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

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

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



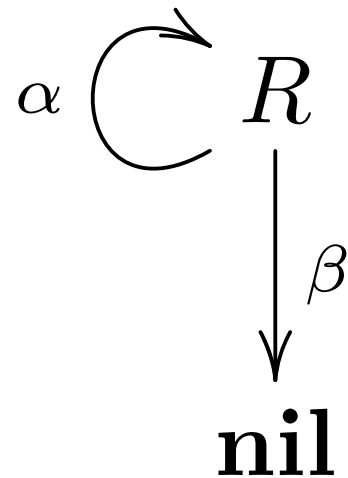


Exercise: LTS?

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

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

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



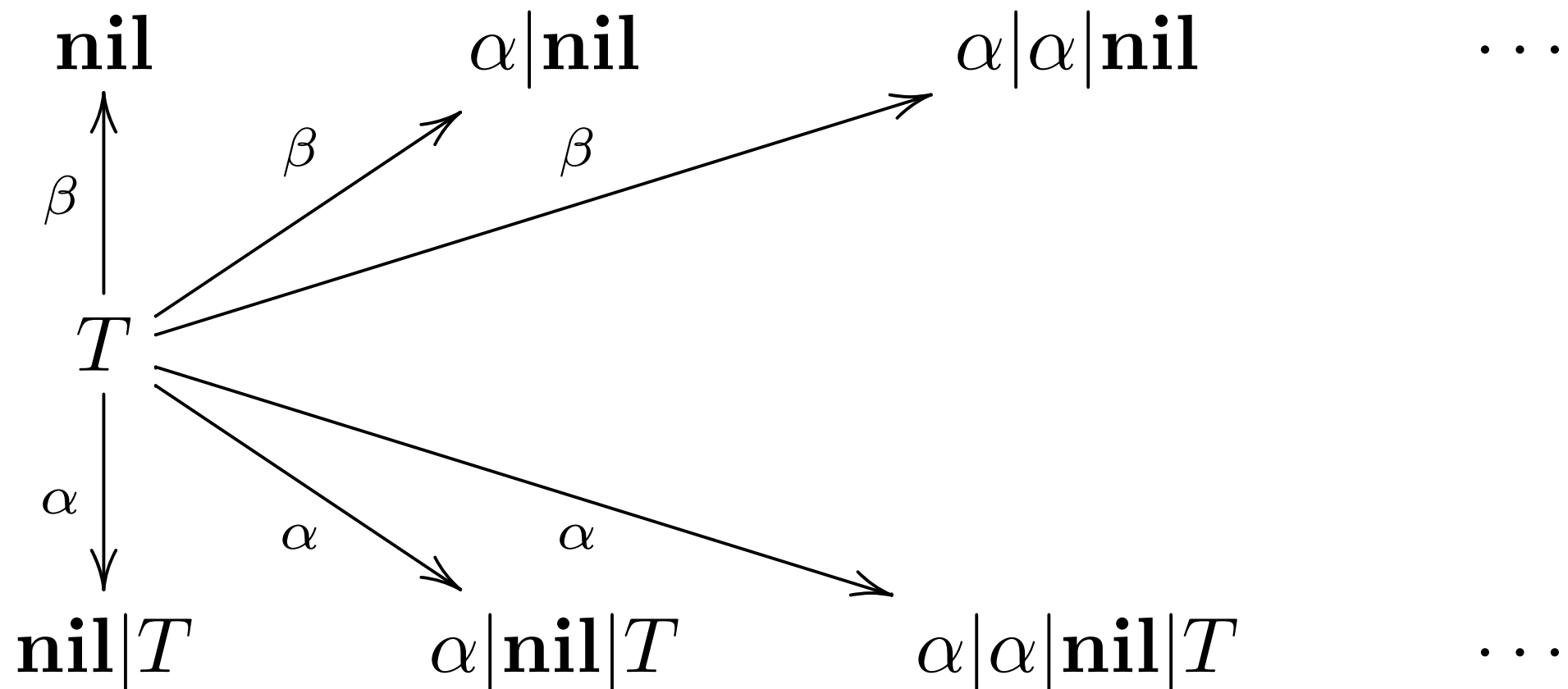


Exercise: LTS?

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

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

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





Exercise: LTS?

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

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

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

$$\begin{array}{ccc} U & \xrightarrow{\beta} & \alpha|U \\ \alpha \downarrow & & \\ \mathbf{nil}|\beta.U & & \end{array}$$

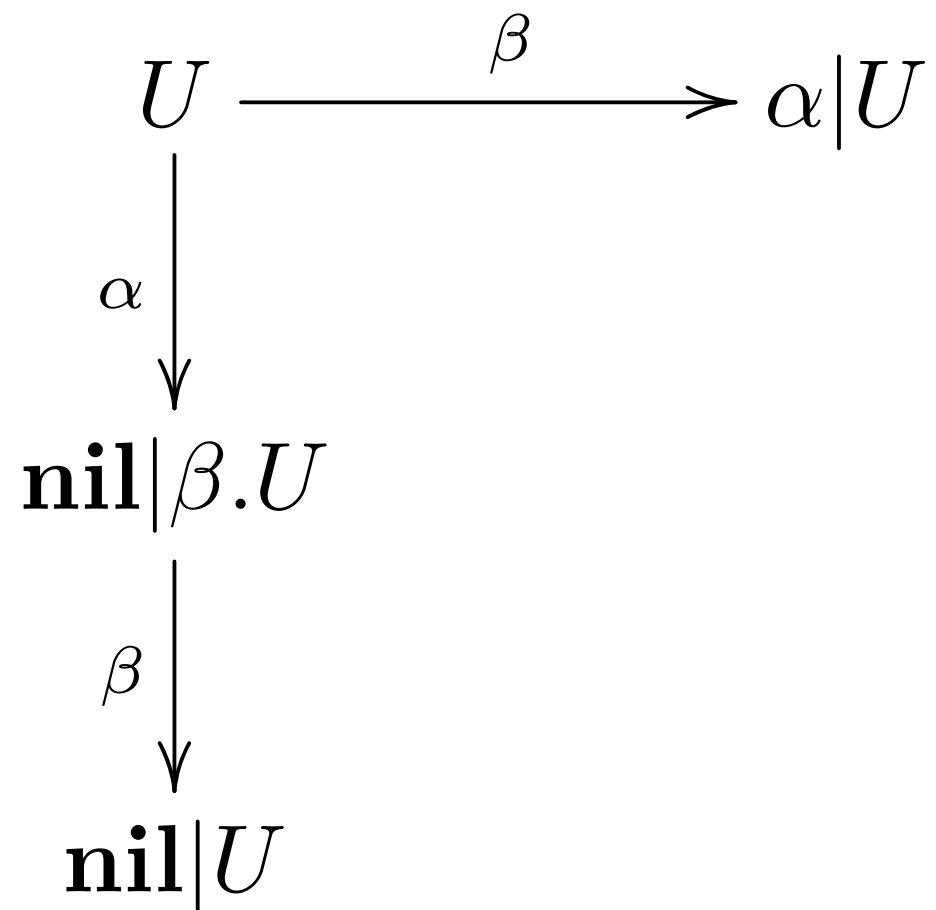


Exercise: LTS?

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

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

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



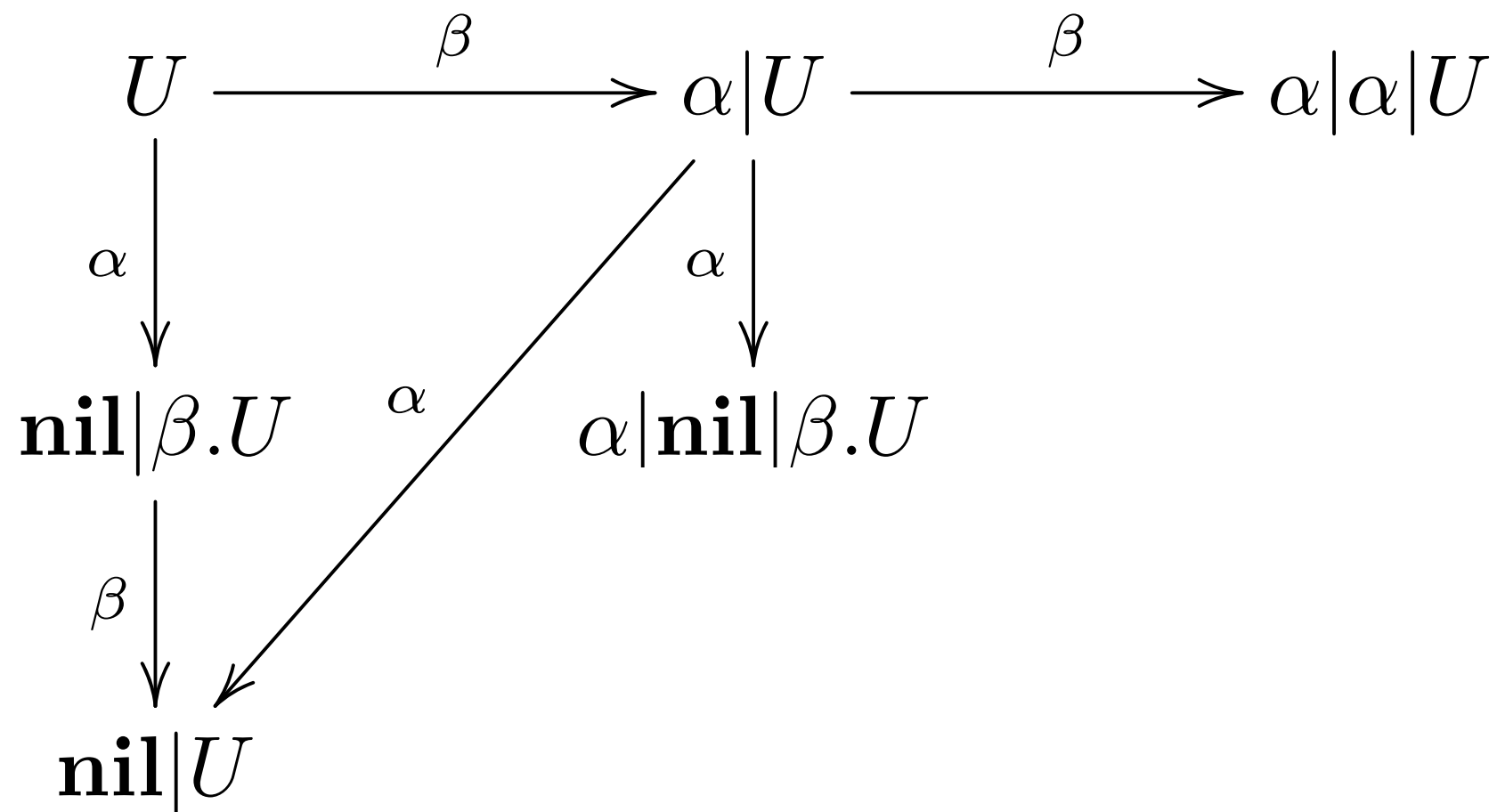


Exercise: LTS?

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

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

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



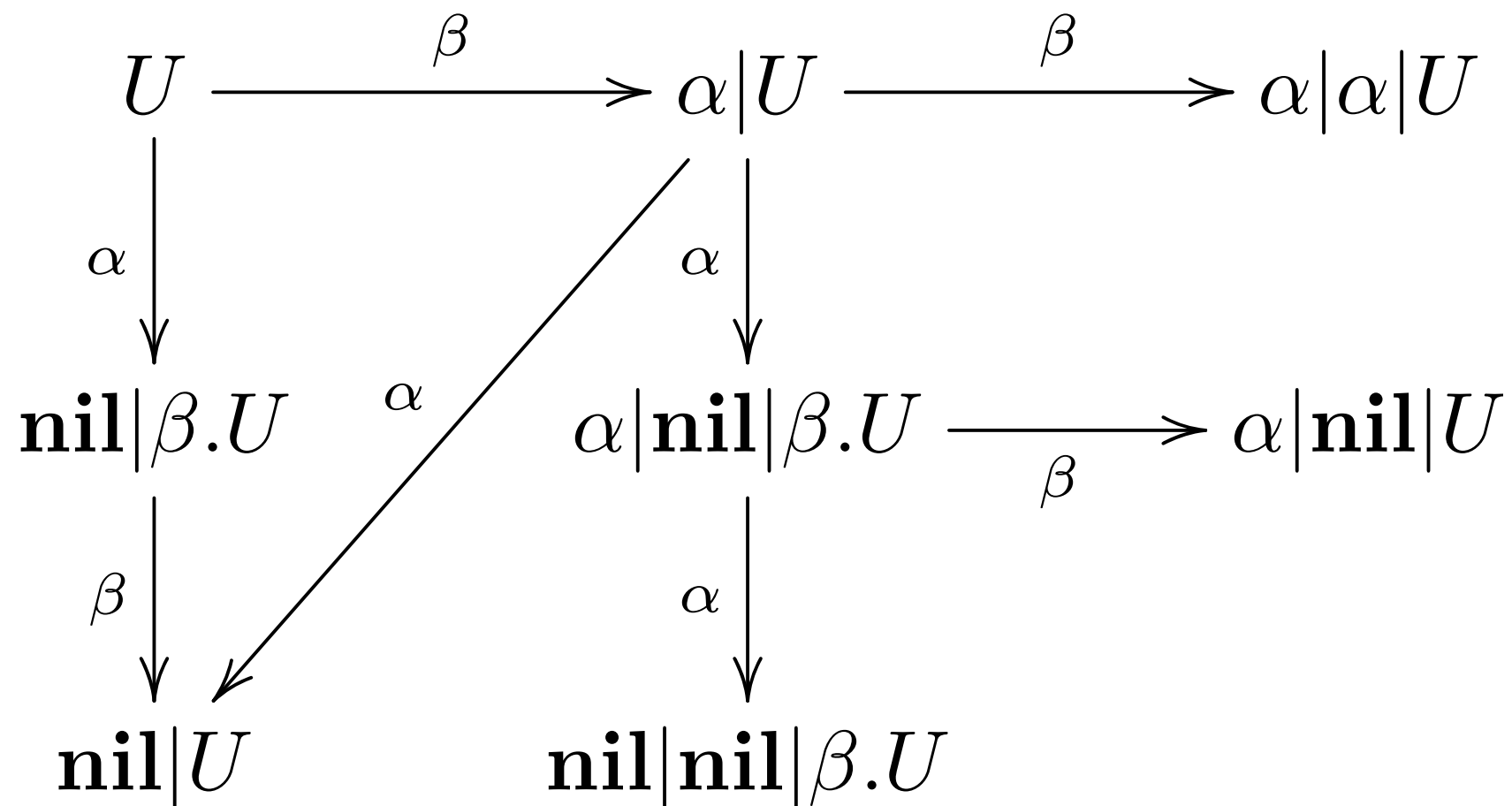


Exercise: LTS?

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

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

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



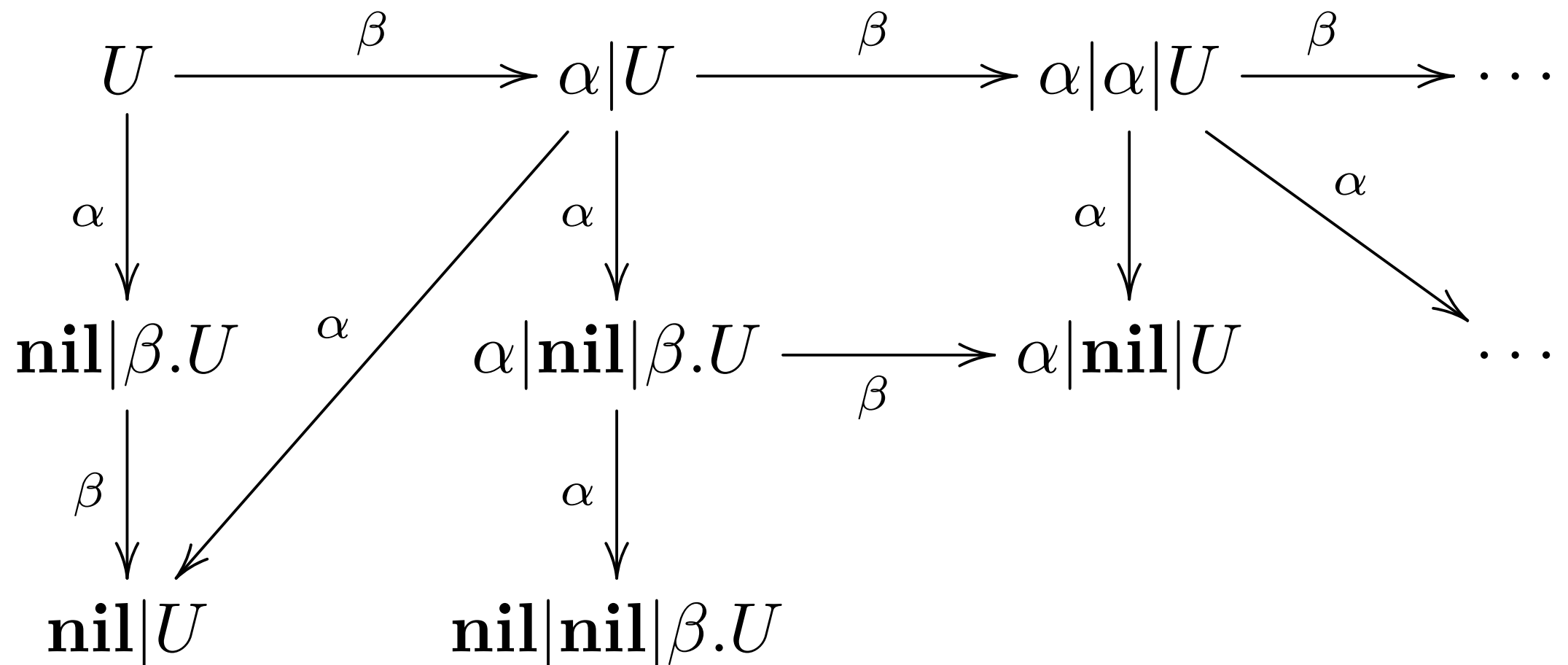


Exercise: LTS?

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

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

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



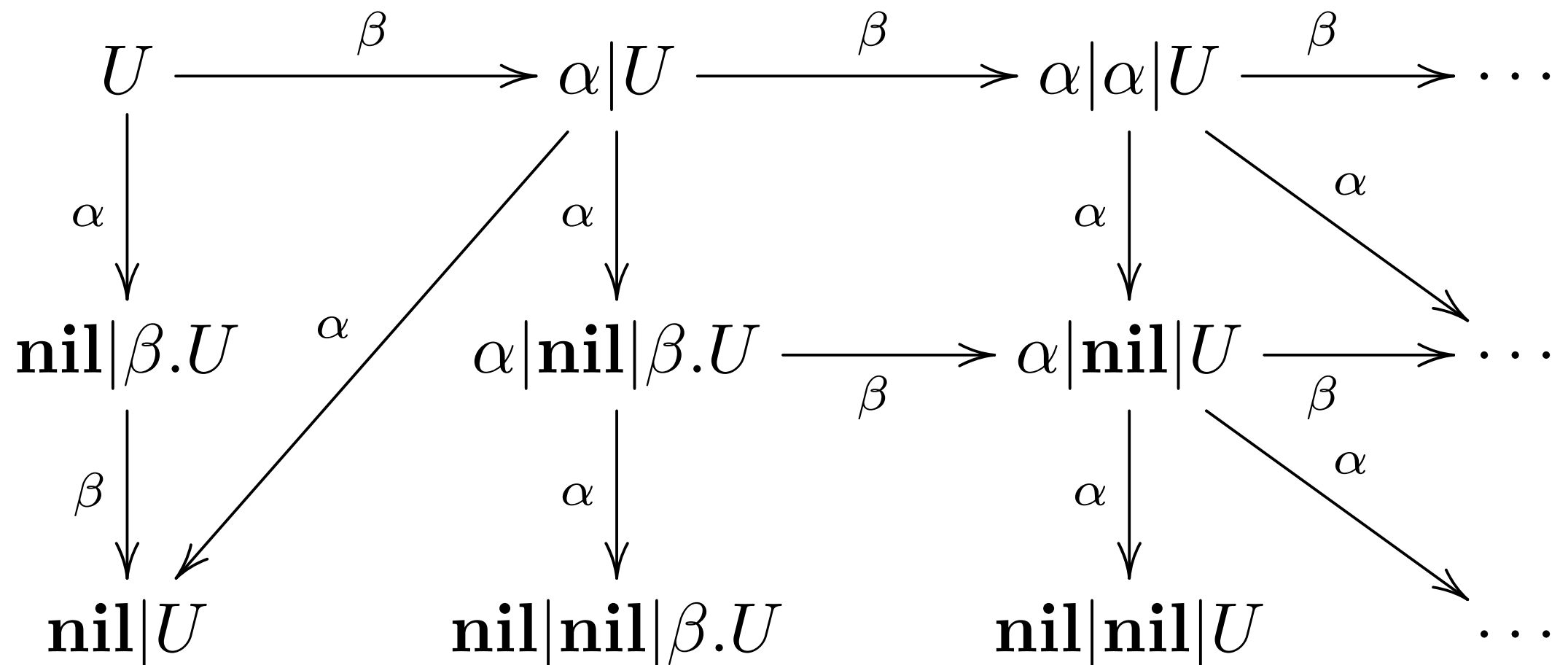


Exercise: LTS?

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

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

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





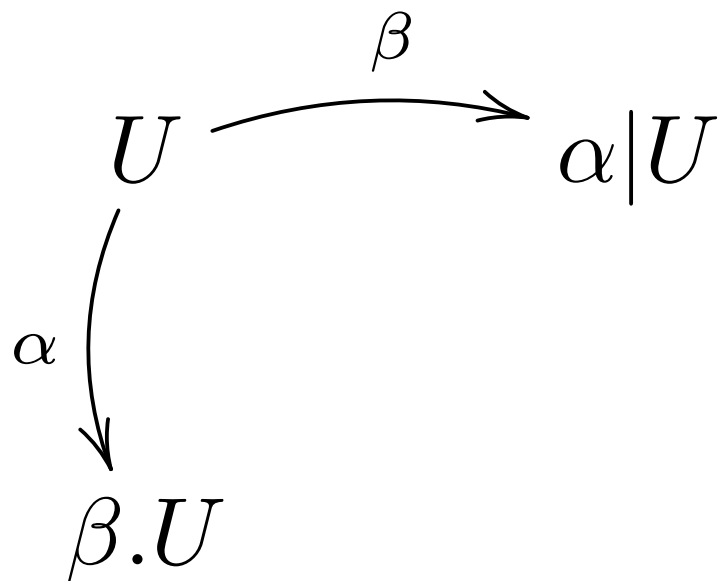
Exercise: LTS?

let's ignore **nil**

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

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

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





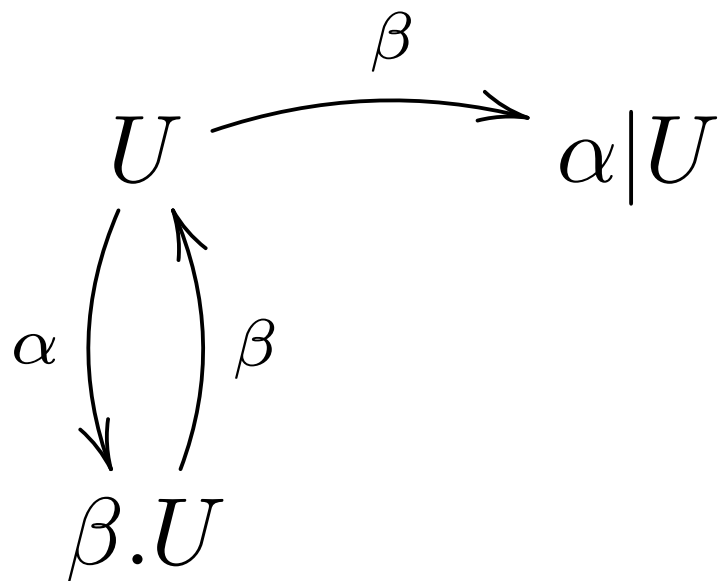
Exercise: LTS?

let's ignore **nil**

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

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

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





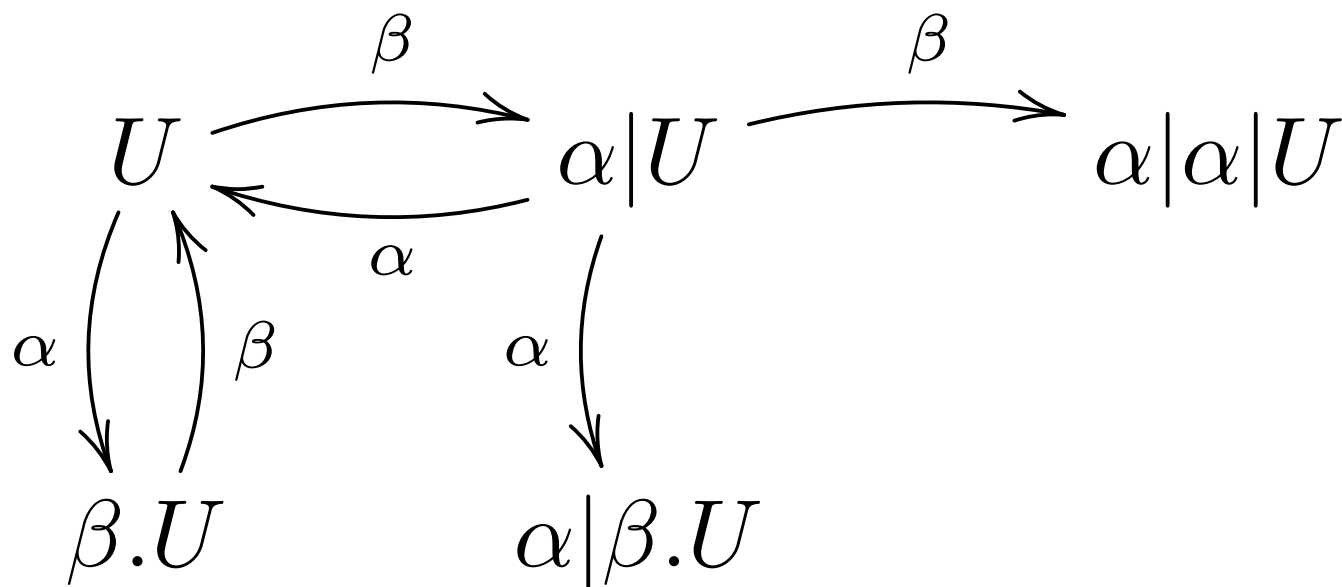
Exercise: LTS?

let's ignore **nil**

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

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

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





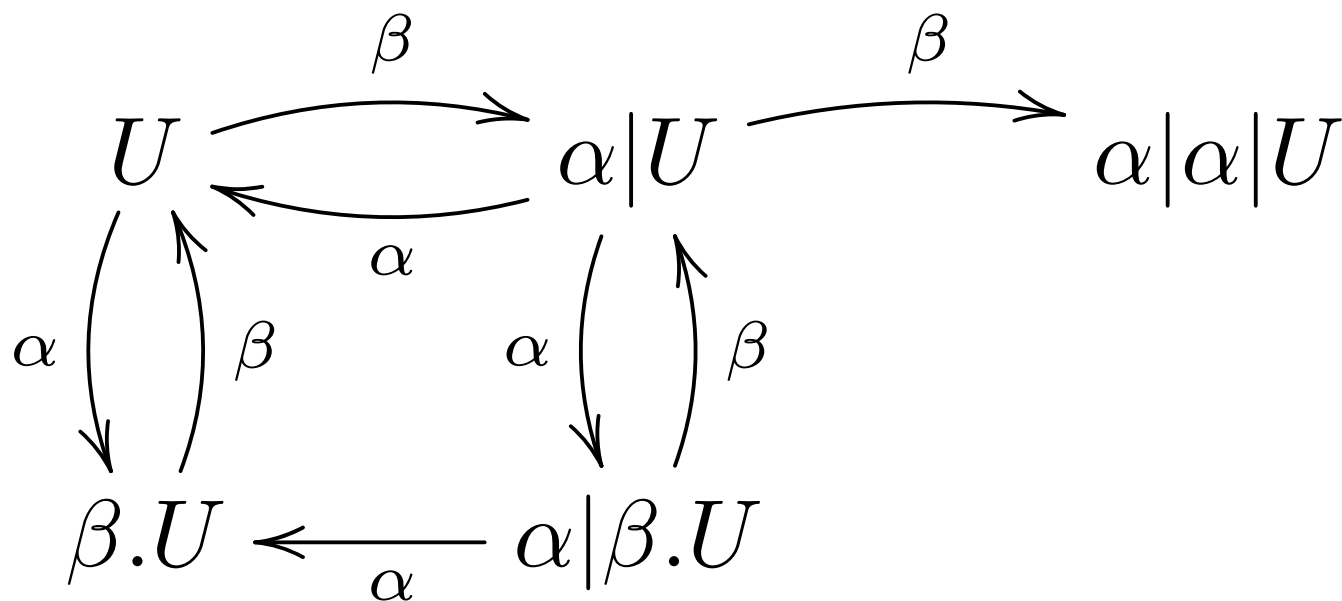
Exercise: LTS?

let's ignore **nil**

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

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

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





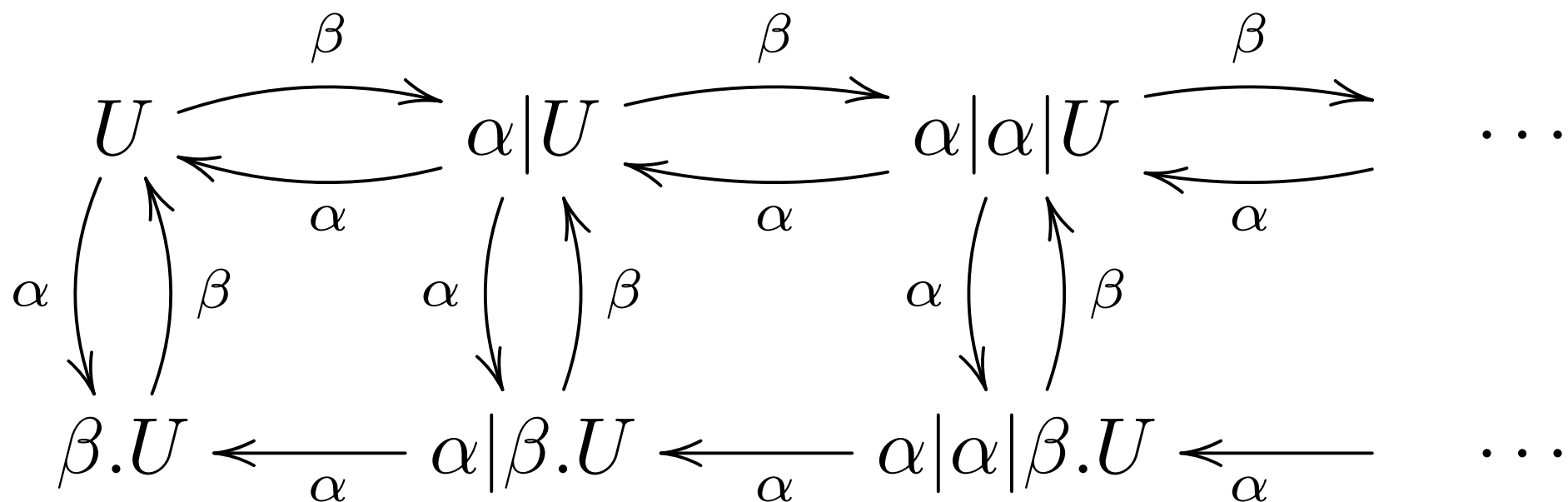
Exercise: LTS?

let's ignore **nil**

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

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

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



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:

C₀, C₁, C₂, C₃

used to display the current value of the counter

Draw the LTS of the counter process.