

MPP 2026/27 (0077A, 9CFU)

Models for Programming Paradigms

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

https://didawiki.di.unipi.it/doku.php/magistraleinformatica/mpp/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)

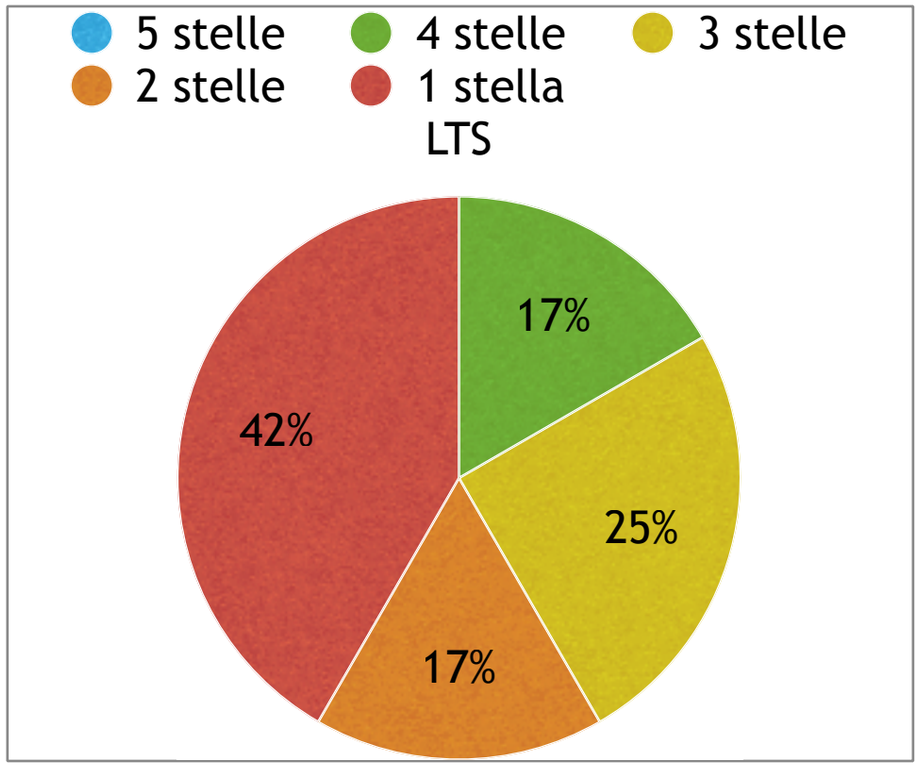
Communication

- 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 12 answers)

Labelled transitions

ongoing interaction
with the environment
(with other processes)

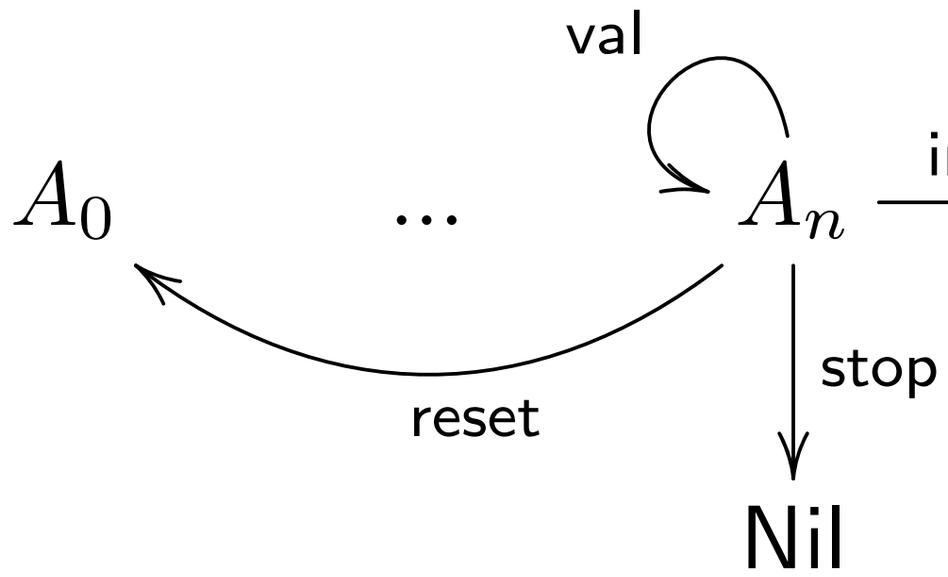
$$p \xrightarrow{\mu} q$$

a process
in its current state

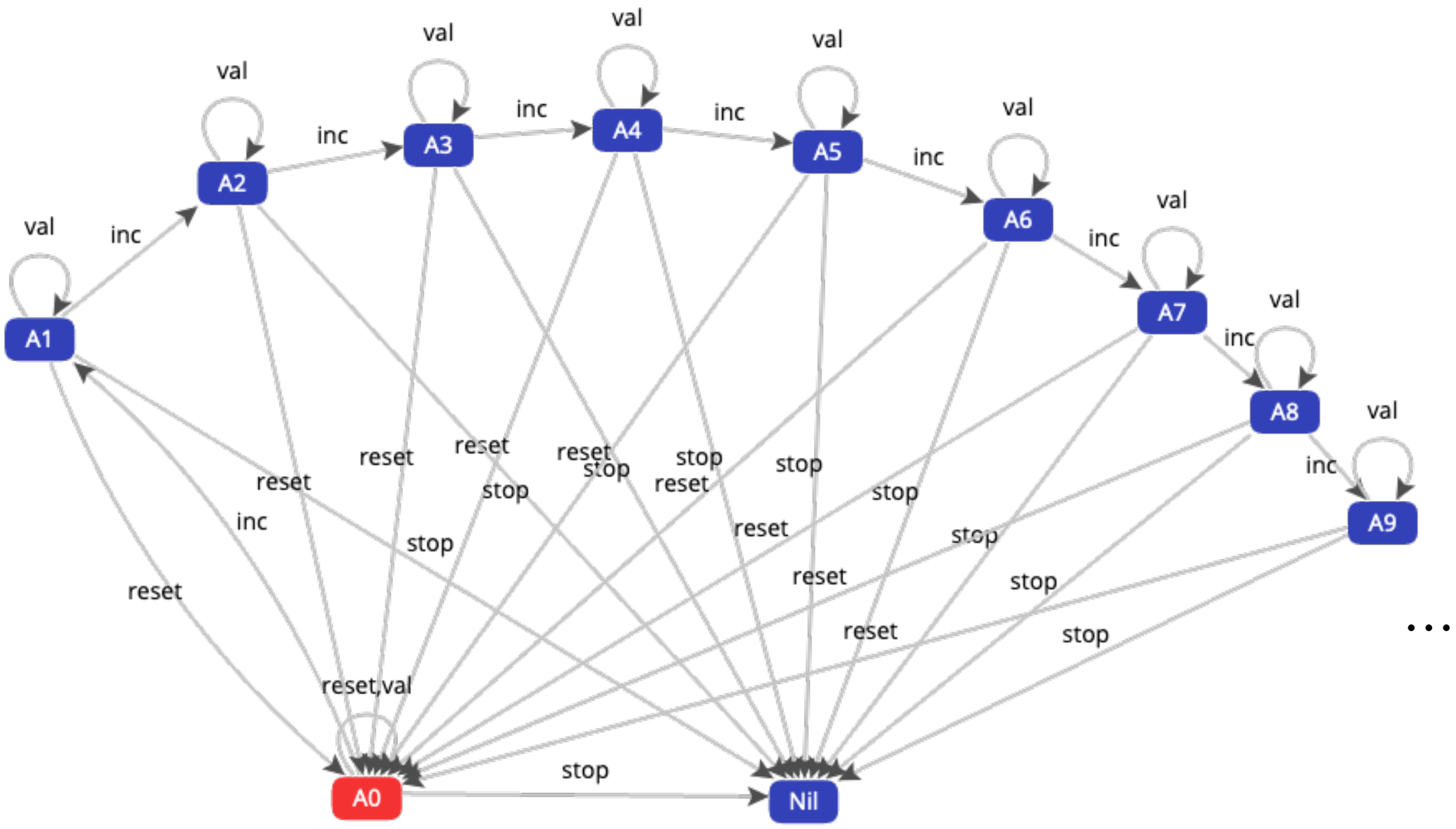
the process
state after the
interaction

number of states/transitions
can be infinite

Example: counter



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) $\alpha!v$ — send v on channel α
a dual action (e.g. an input) $\alpha?v$ — receive v on channel α

an internal action (silent action) τ — concluded communication
(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)

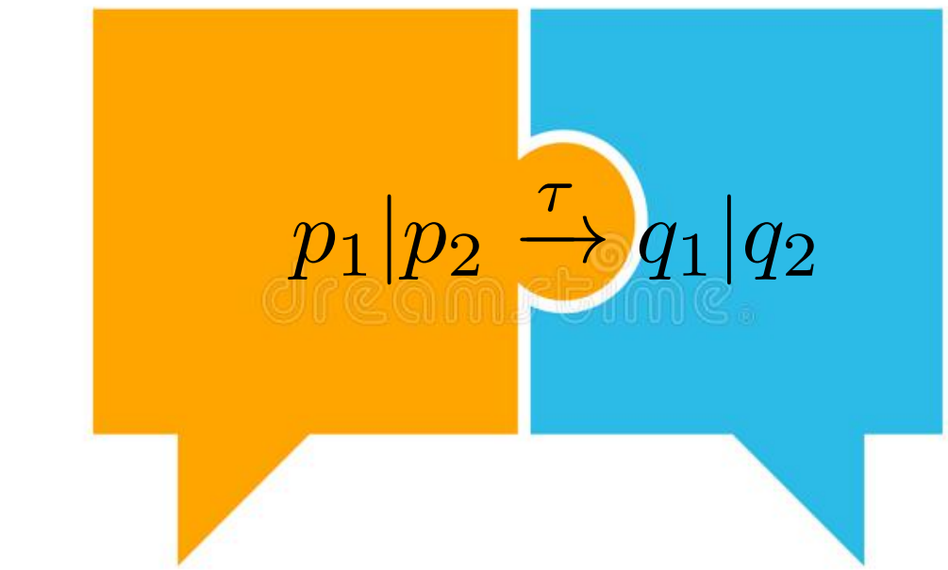
$\alpha!v$ becomes just $\overline{\alpha_v}$ or just $\overline{\alpha}$

$\alpha?v$ becomes just α_v or just α

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

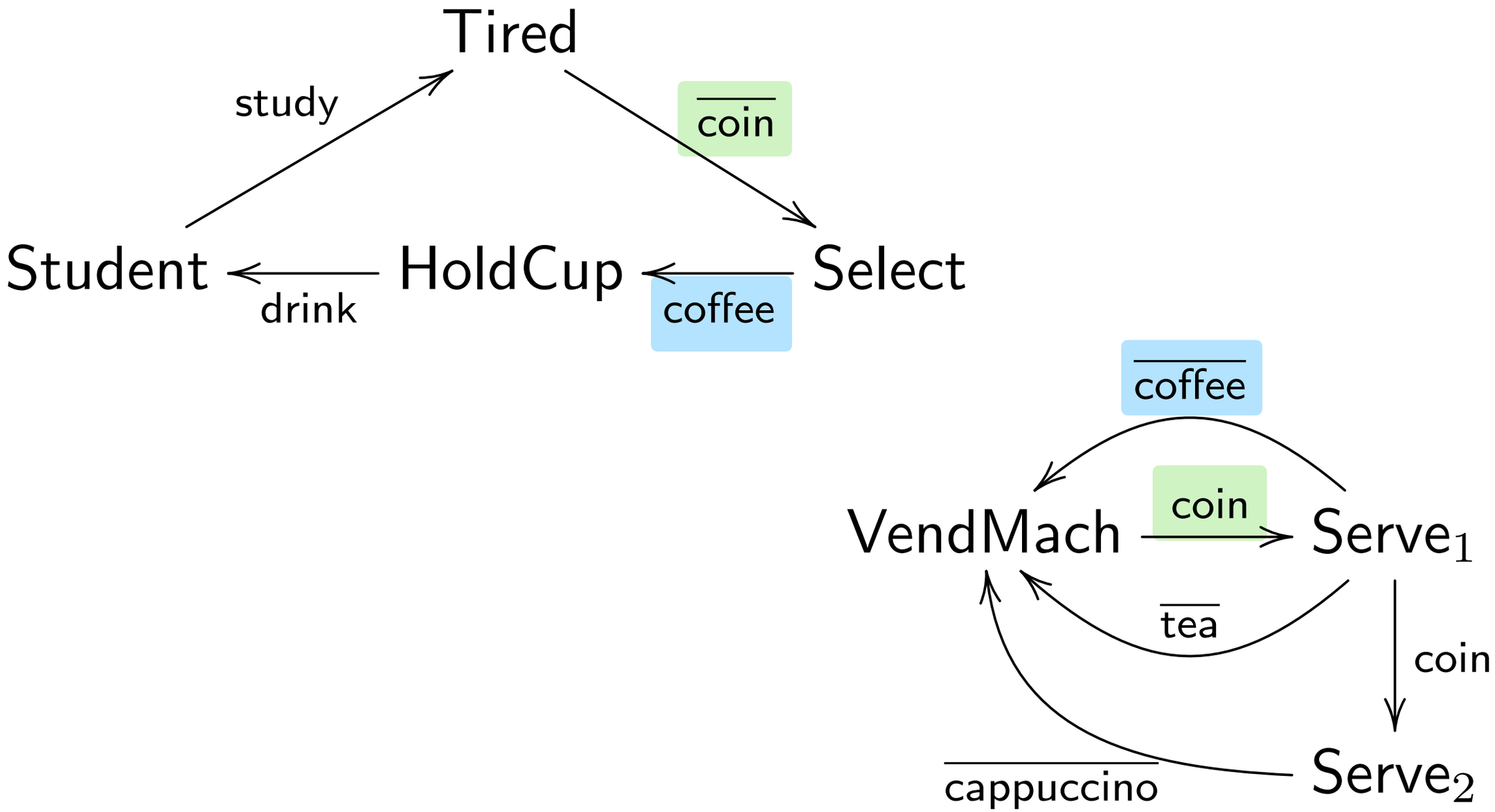
$\overline{\lambda}$ denotes its dual (assume $\overline{\overline{\alpha}} = \alpha$)

CCS: communication



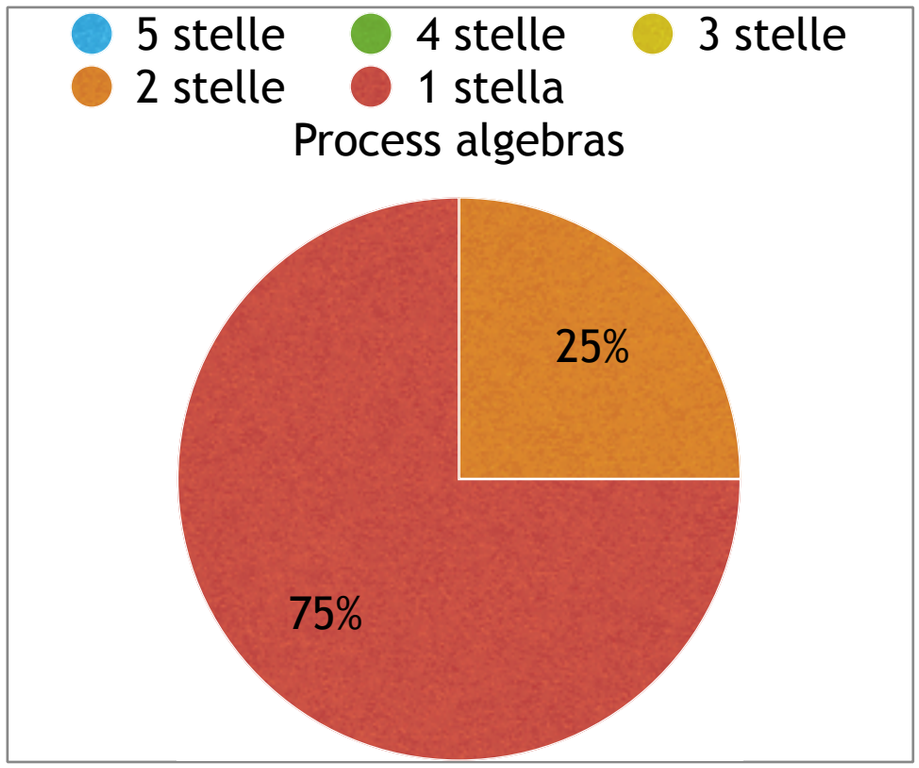
$p_1 \xrightarrow{\lambda} q_1$ $\quad$ $p_2 \xrightarrow{\overline{\lambda}} q_2$

Example: vending machine



CCS syntax

From your forms



(over 12 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)
	$\mid$	$p \mid q$	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

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

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 $\overline{\alpha}$

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

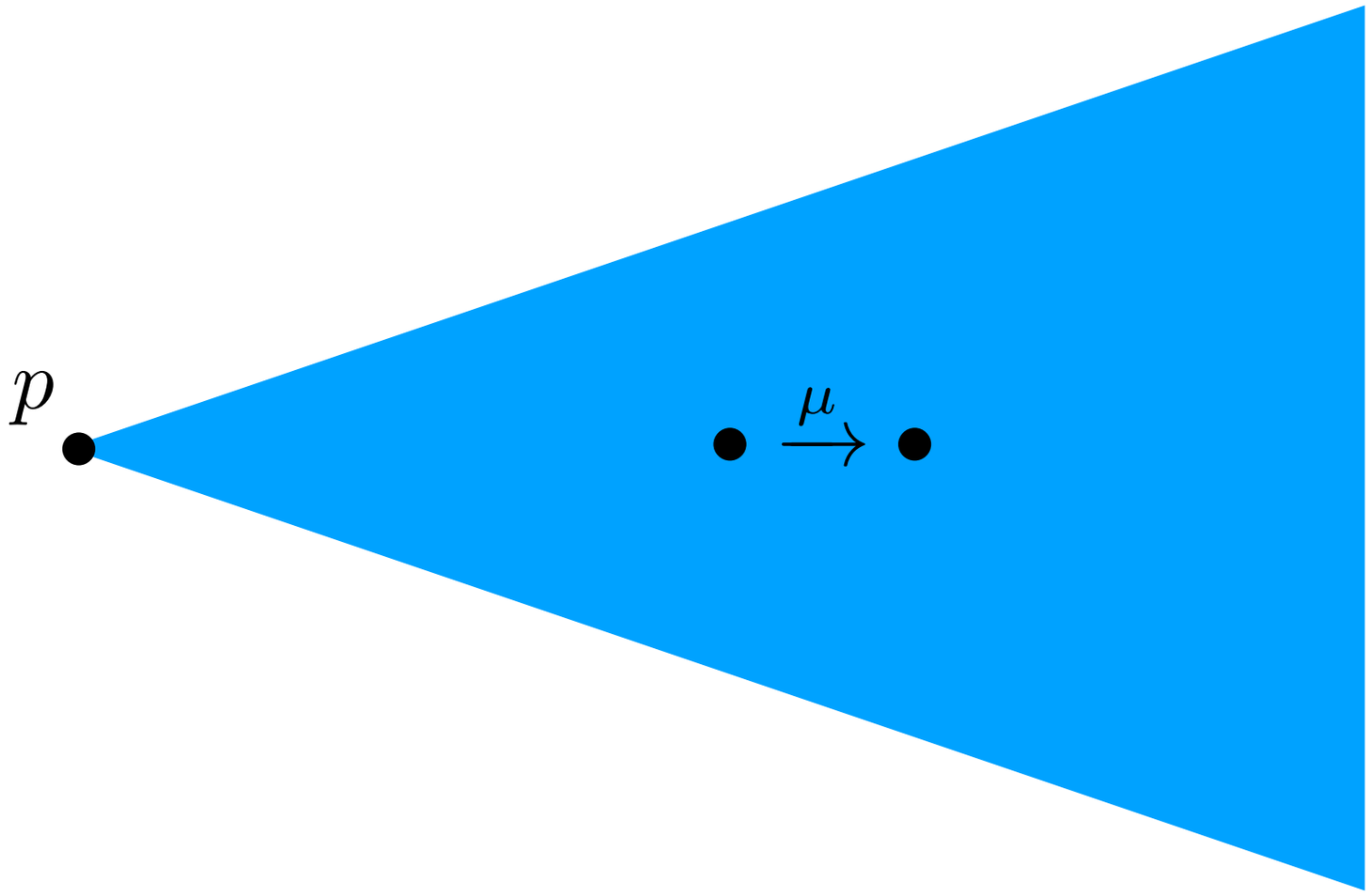
$\Lambda = \mathcal{C} \cup \overline{\mathcal{C}}$ set of observable actions, ranged by $\lambda \quad \overline{\lambda}$

$\tau \notin \Lambda$ 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

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

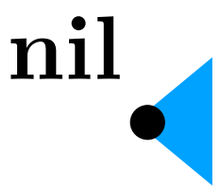
the inactive process does nothing

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

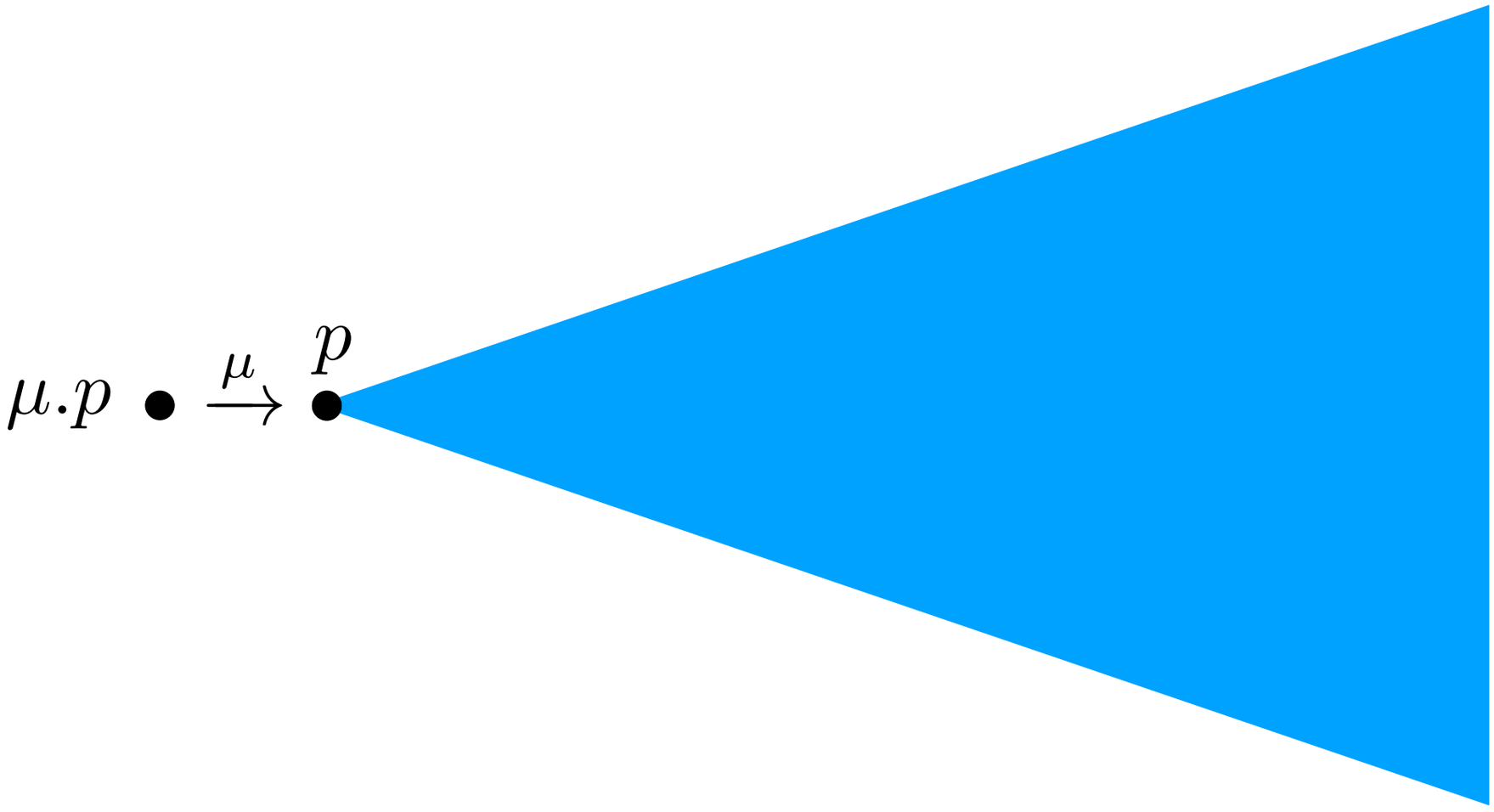
the action may involve an interaction with the environment

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

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



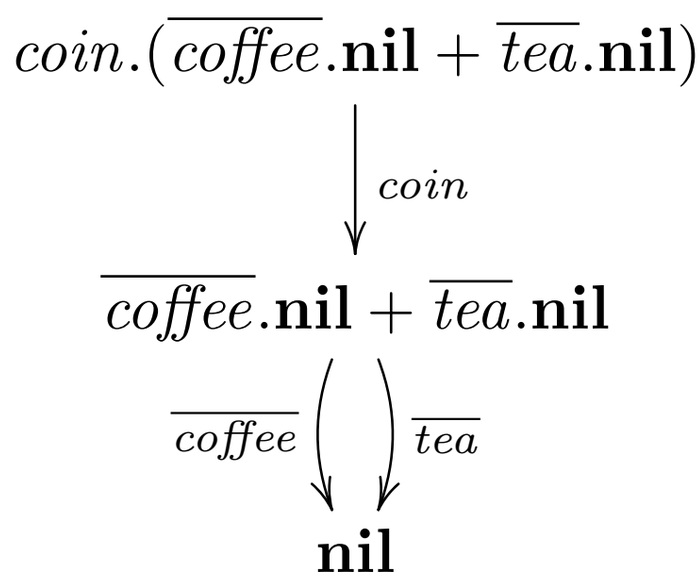
Nondeterministic choice

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

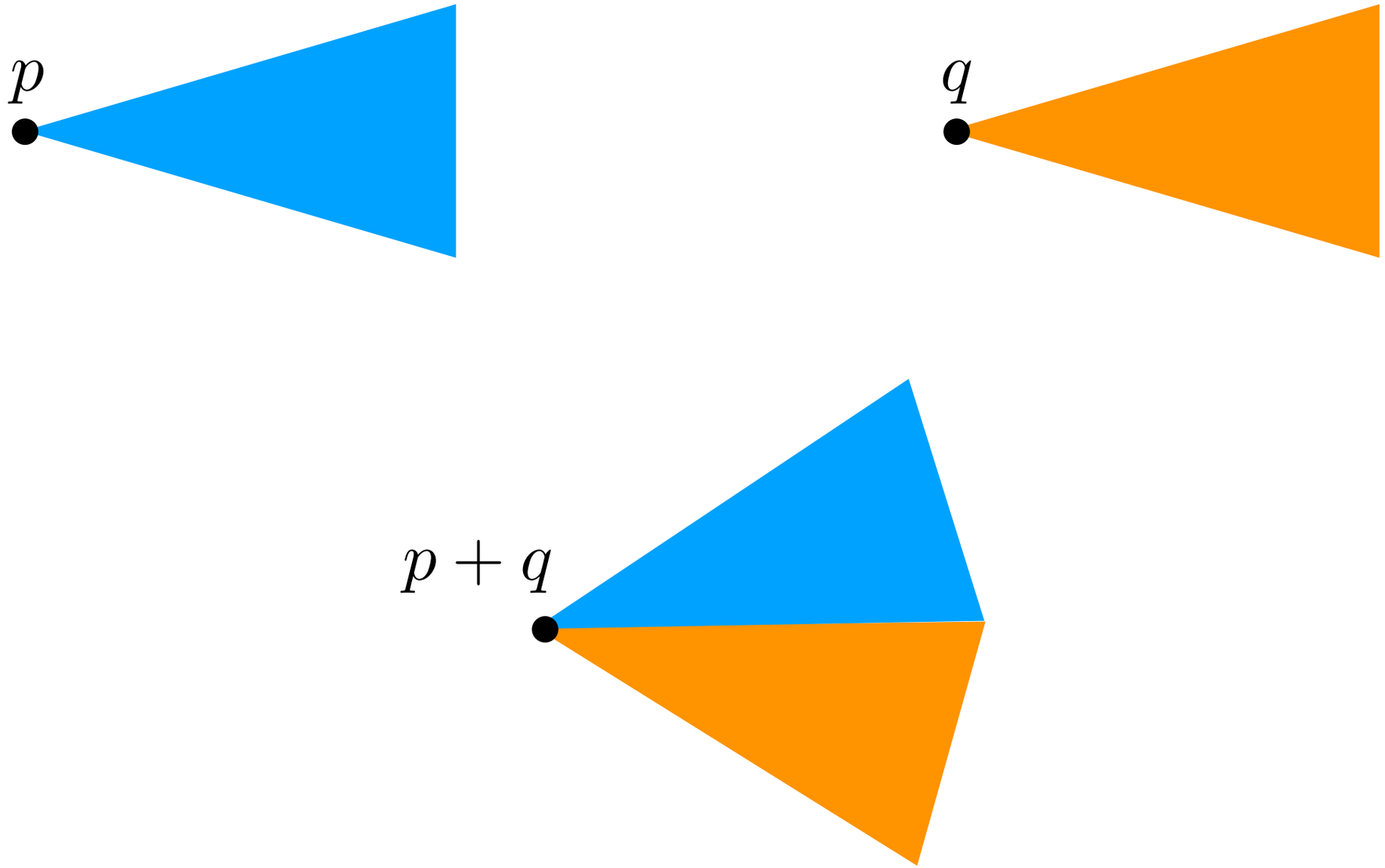
process $p_1 + p_2$ can behave either as p_1 or as p_2

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

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



LTS of a process



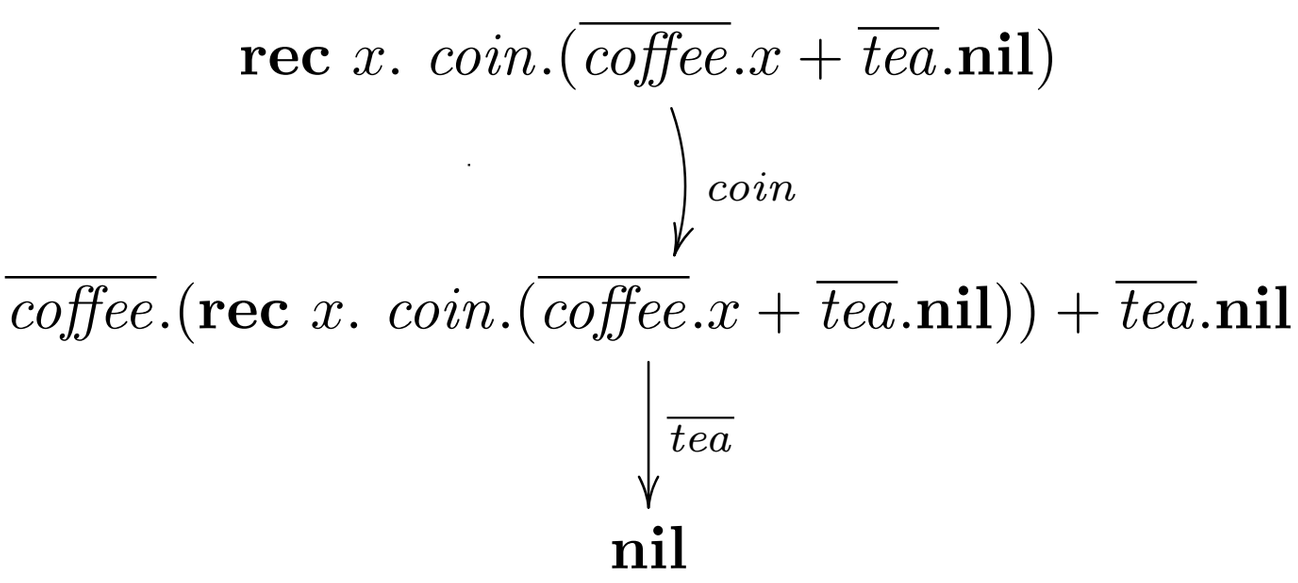
Recursion

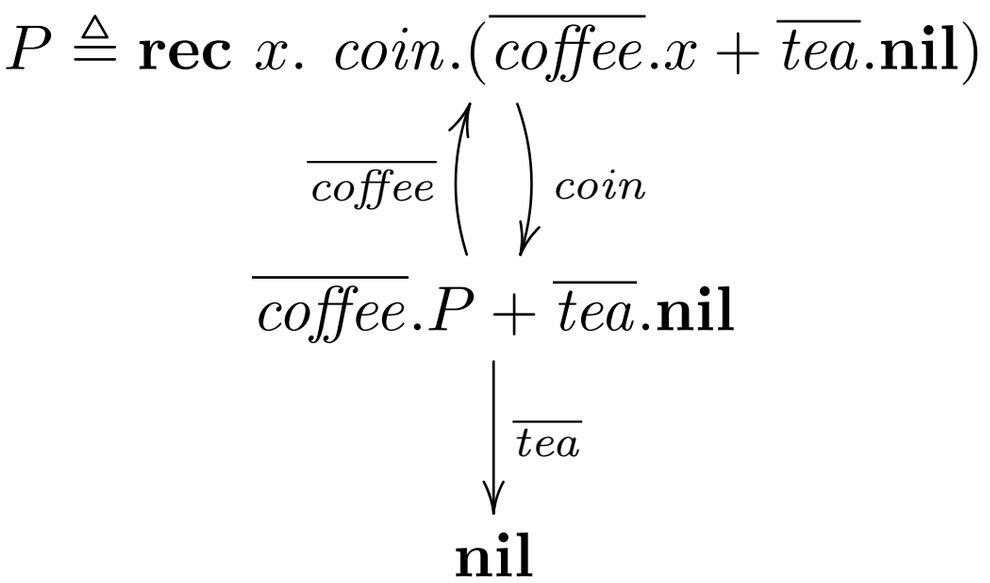
$$\text{Rec)}\ \frac{p\left[{}^{\mathbf{rec}\ x.\ p}/_{x}\right] \xrightarrow{\mu} q}{\mathbf{rec}\ x.\ p \xrightarrow{\mu} q}$$

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





Recursion via process constants

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

$$A \triangleq p$$

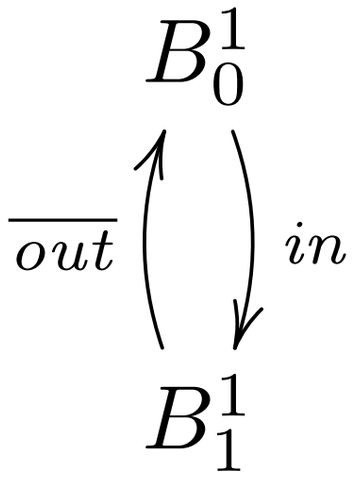
one for each constant

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

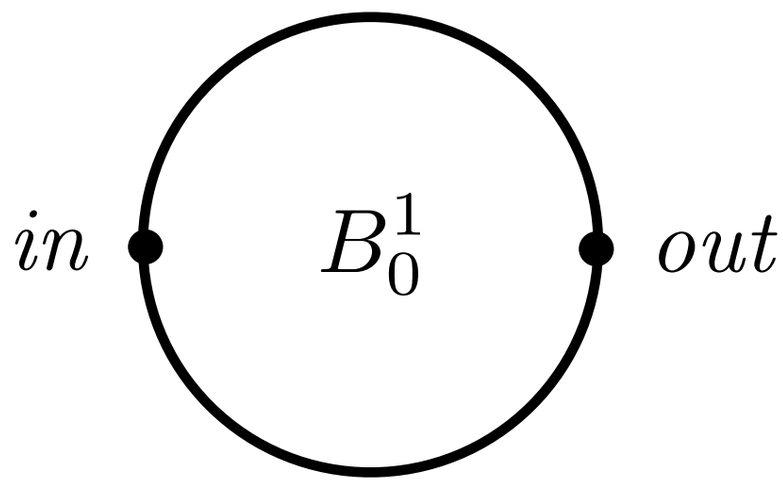
$$P \triangleq coin.(\overline{coffee}.P + \overline{tea}.\mathbf{nil})$$

CCS: capacity 1 buffer

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



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

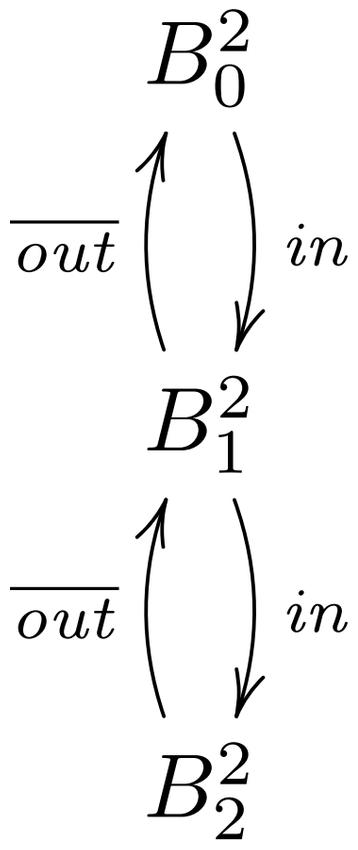


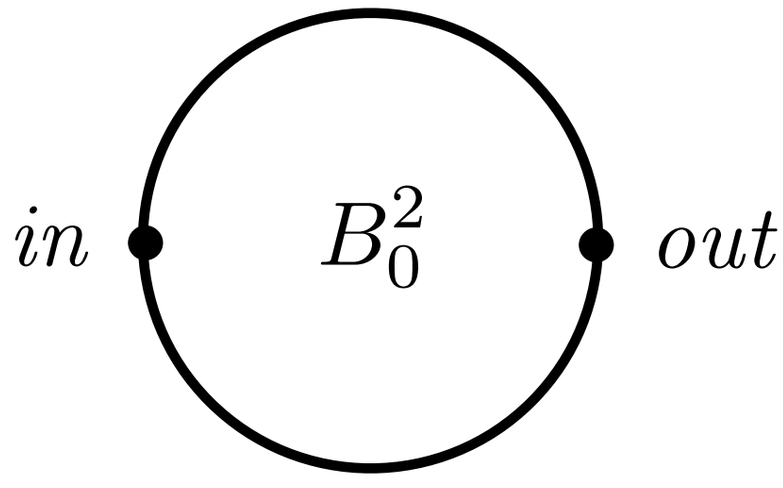
CCS: capacity 2 buffer

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

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

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



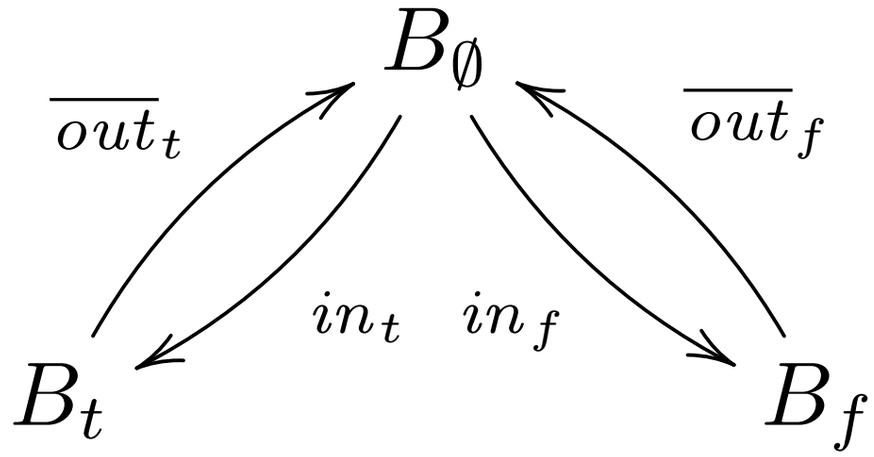


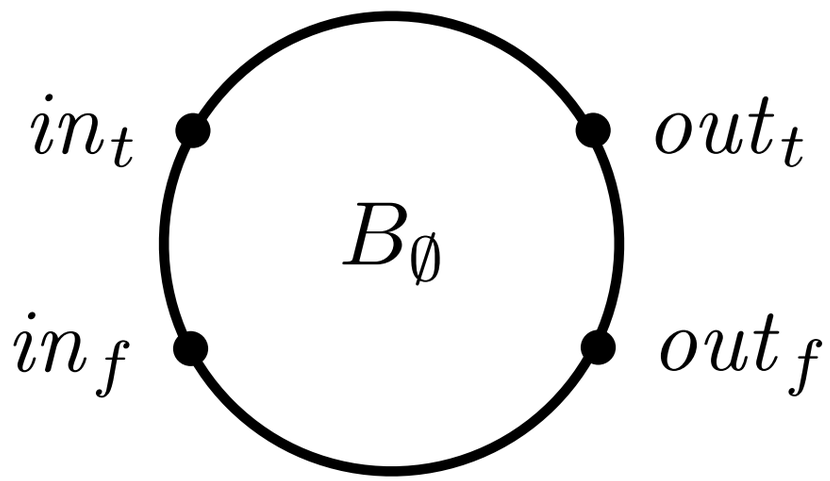
CCS: boolean buffer

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

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

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





Parallel composition

ParL) $\dfrac{p_1 \xrightarrow{\mu} q_1}{p_1|p_2 \xrightarrow{\mu} q_1|p_2}$ Com) $\dfrac{p_1 \xrightarrow{\lambda} q_1 \quad p_2 \xrightarrow{\overline{\lambda}} q_2}{p_1|p_2 \xrightarrow{\tau} q_1|q_2}$ ParR) $\dfrac{p_2 \xrightarrow{\mu} q_2}{p_1|p_2 \xrightarrow{\mu} p_1|q_2}$

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

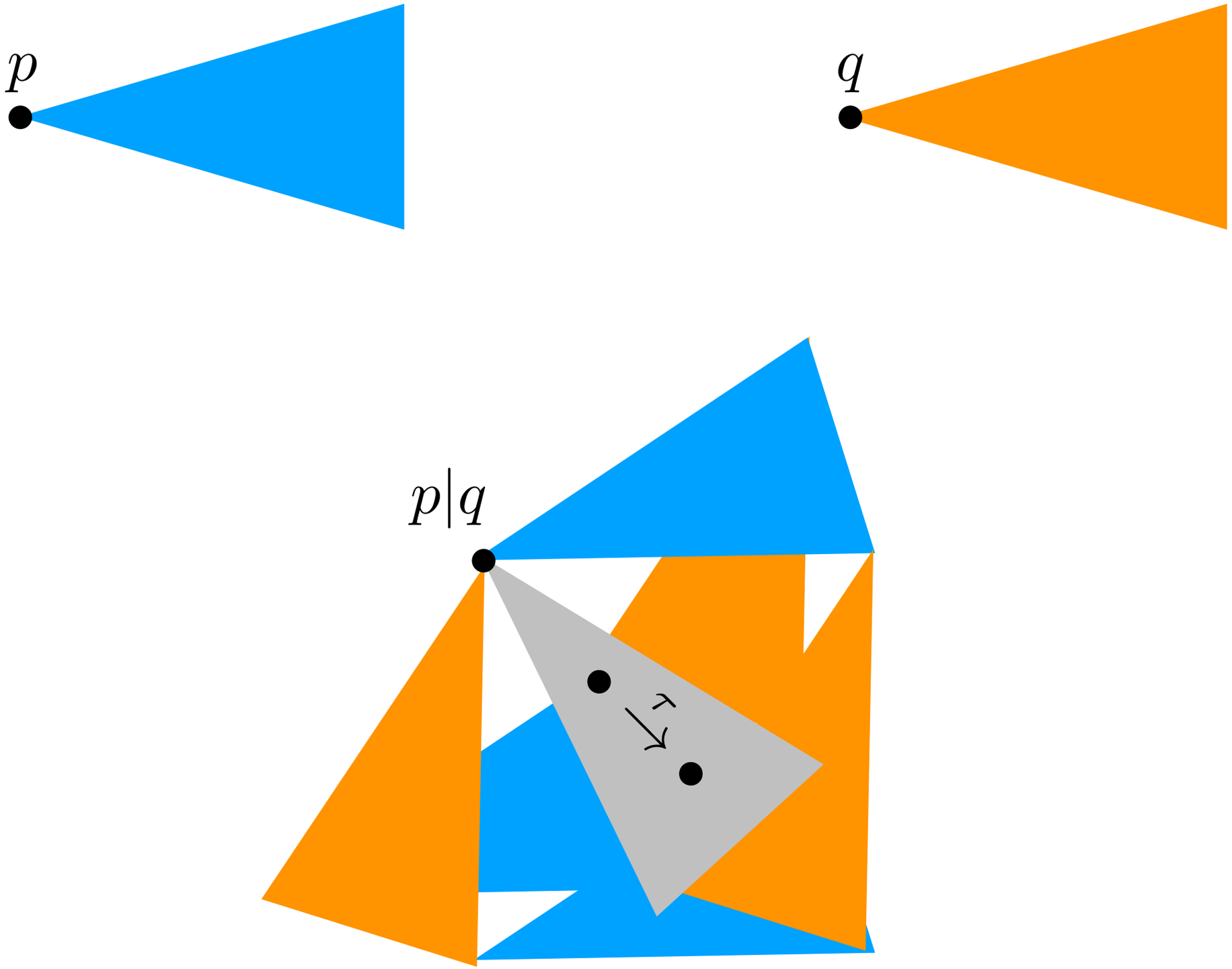
$P \triangleq \overline{coin}.coffee.\mathbf{nil}$ $\quad M \triangleq coin.(\overline{coffee}.\mathbf{nil} + \overline{tea}.\mathbf{nil})$

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

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

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

LTS of a process



CCS: parallel buffers

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

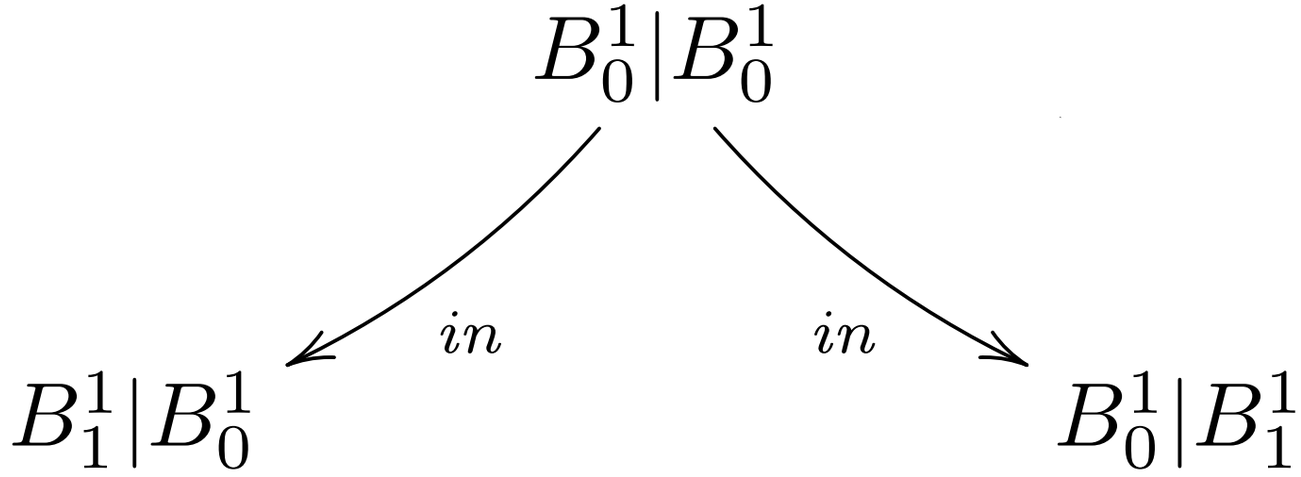
$$B_1^1 \triangleq \overline{out}.B_0^1$$

$$B_0^1 | B_0^1$$

CCS: parallel buffers

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

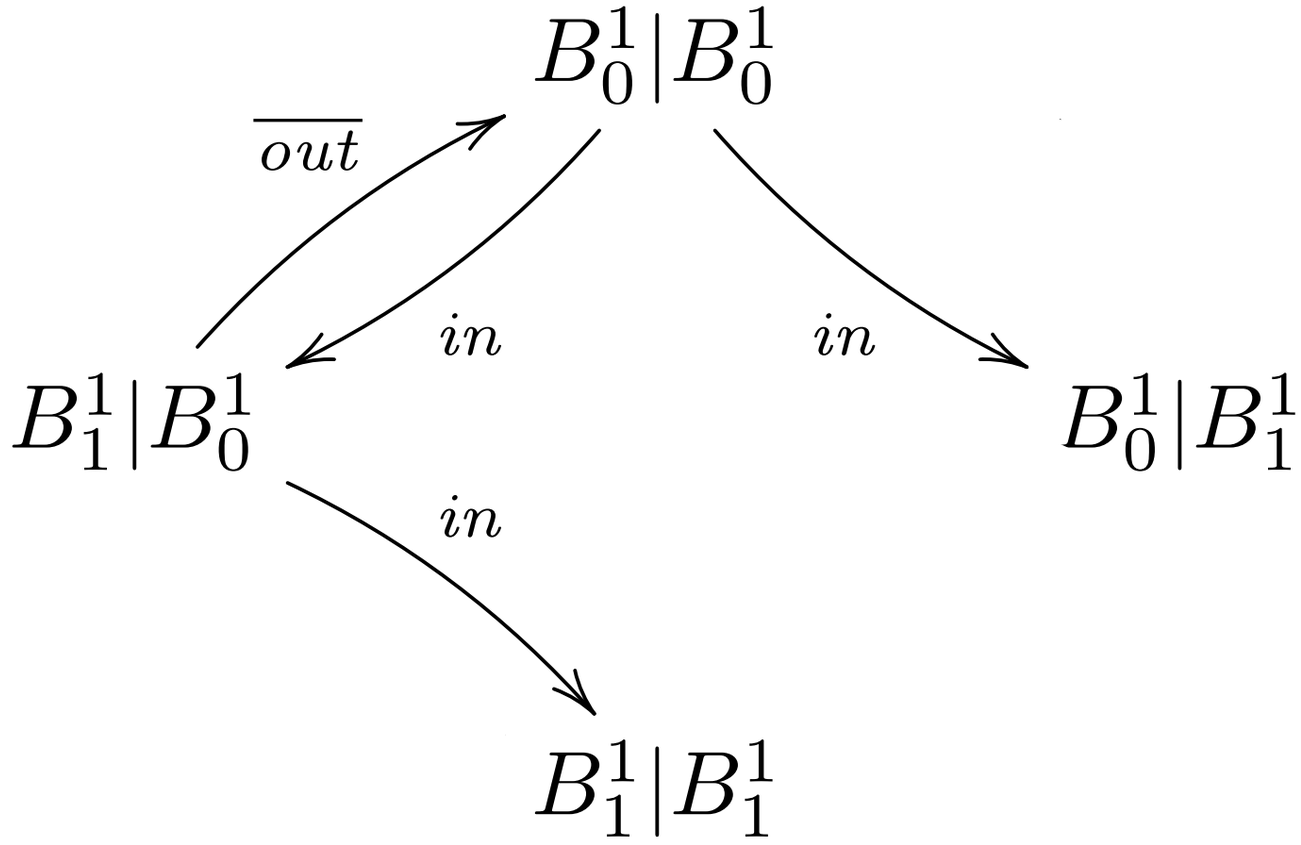
$$B_1^1 \triangleq \overline{out}.B_0^1$$



CCS: parallel buffers

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

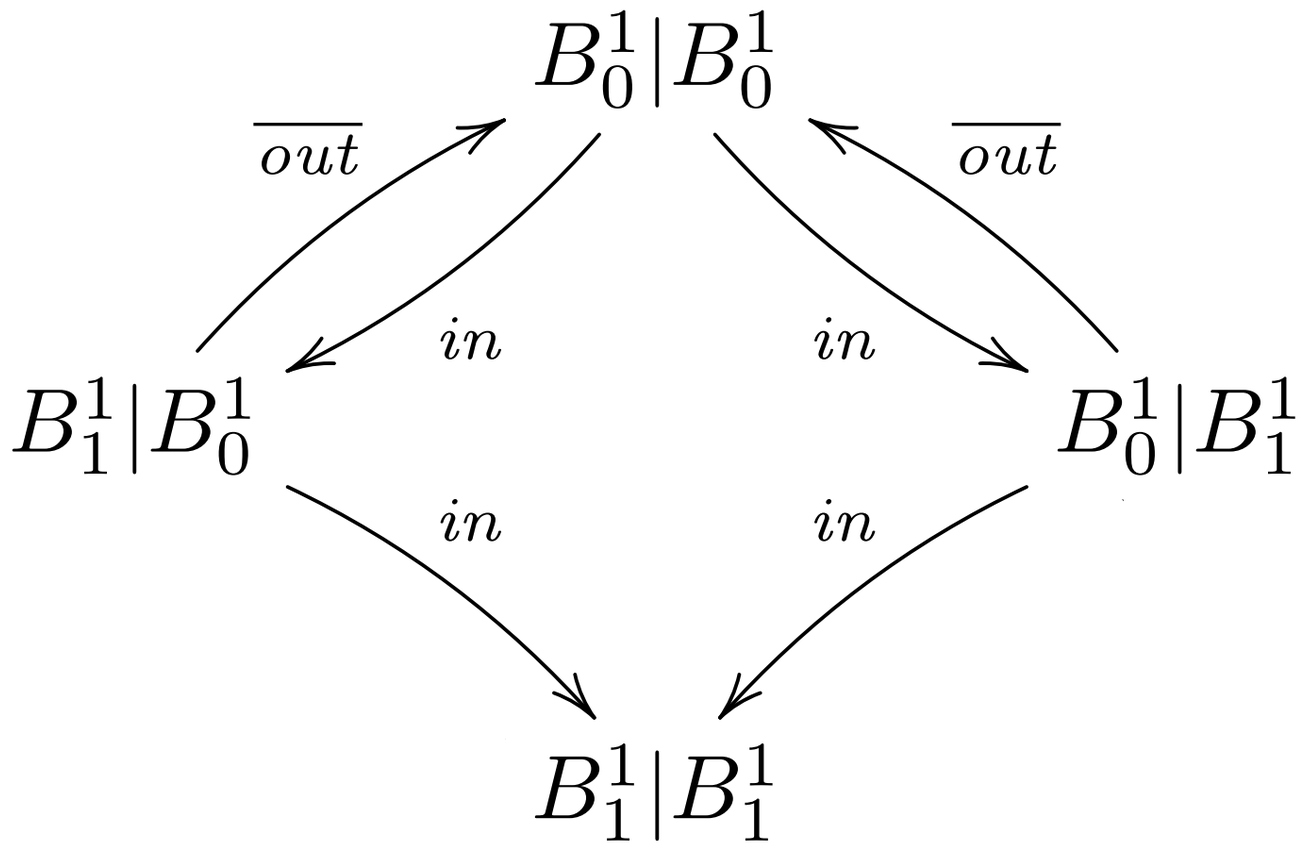
$$B_1^1 \triangleq \overline{out}.B_0^1$$



CCS: parallel buffers

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

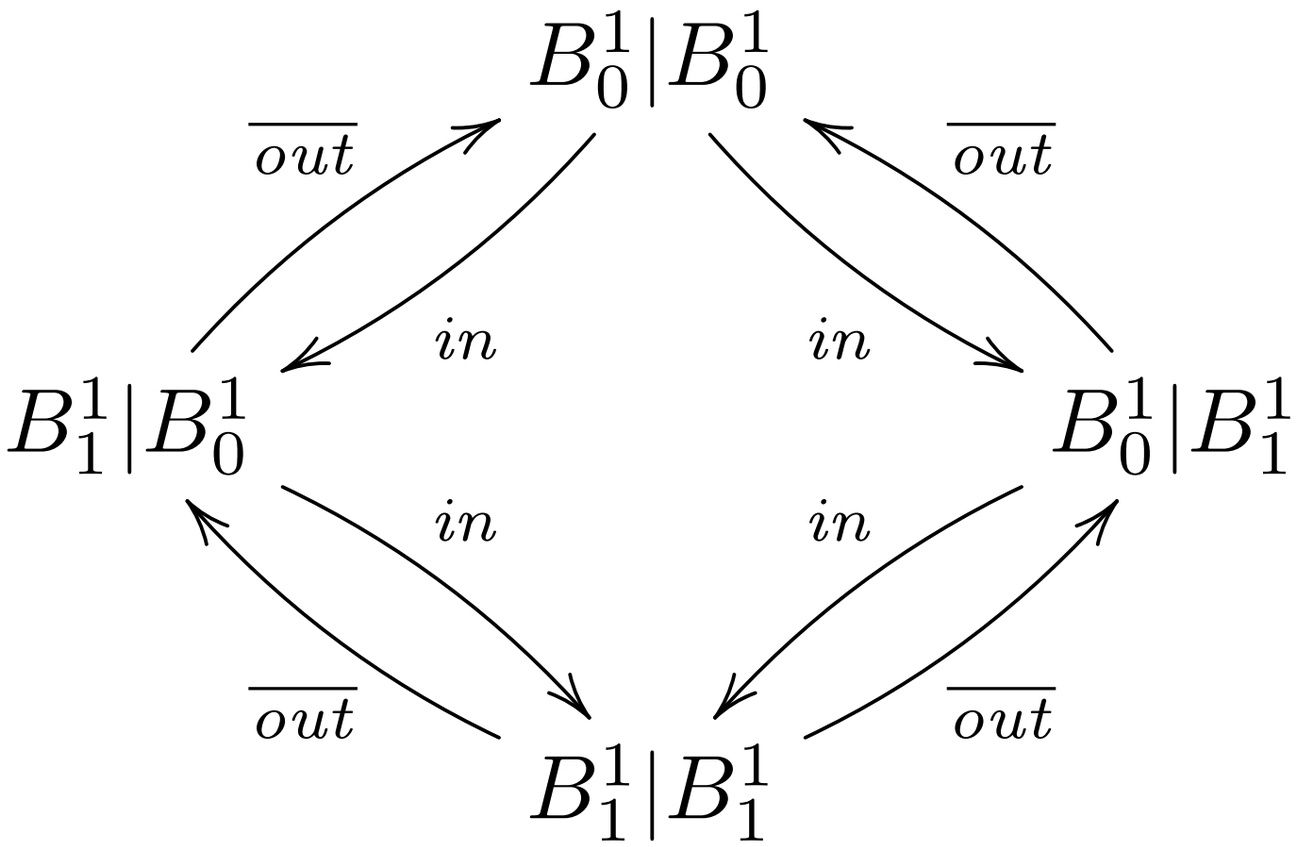
$$B_1^1 \triangleq \overline{out}.B_0^1$$

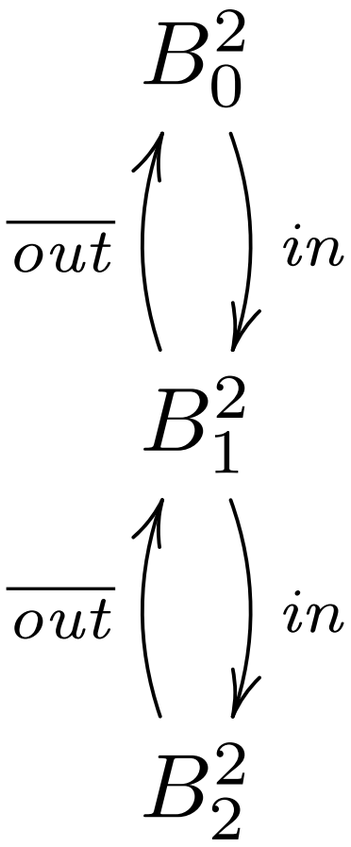


CCS: parallel buffers

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

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





compare with the 2-capacity buffer

Restriction

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

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

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

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

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

Restriction: shorthand

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

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

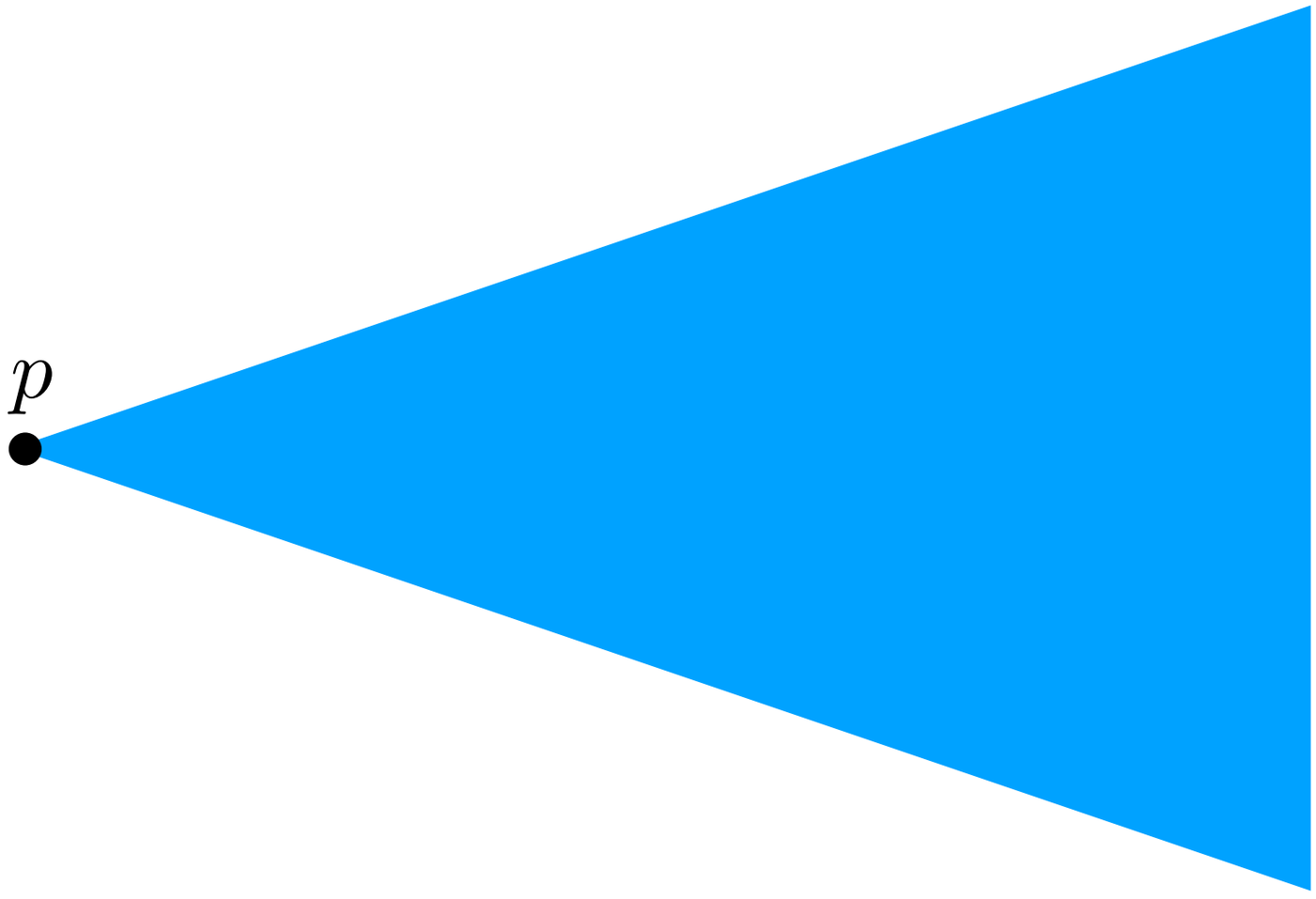
we omit trailing $\mathbf{nil}$

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

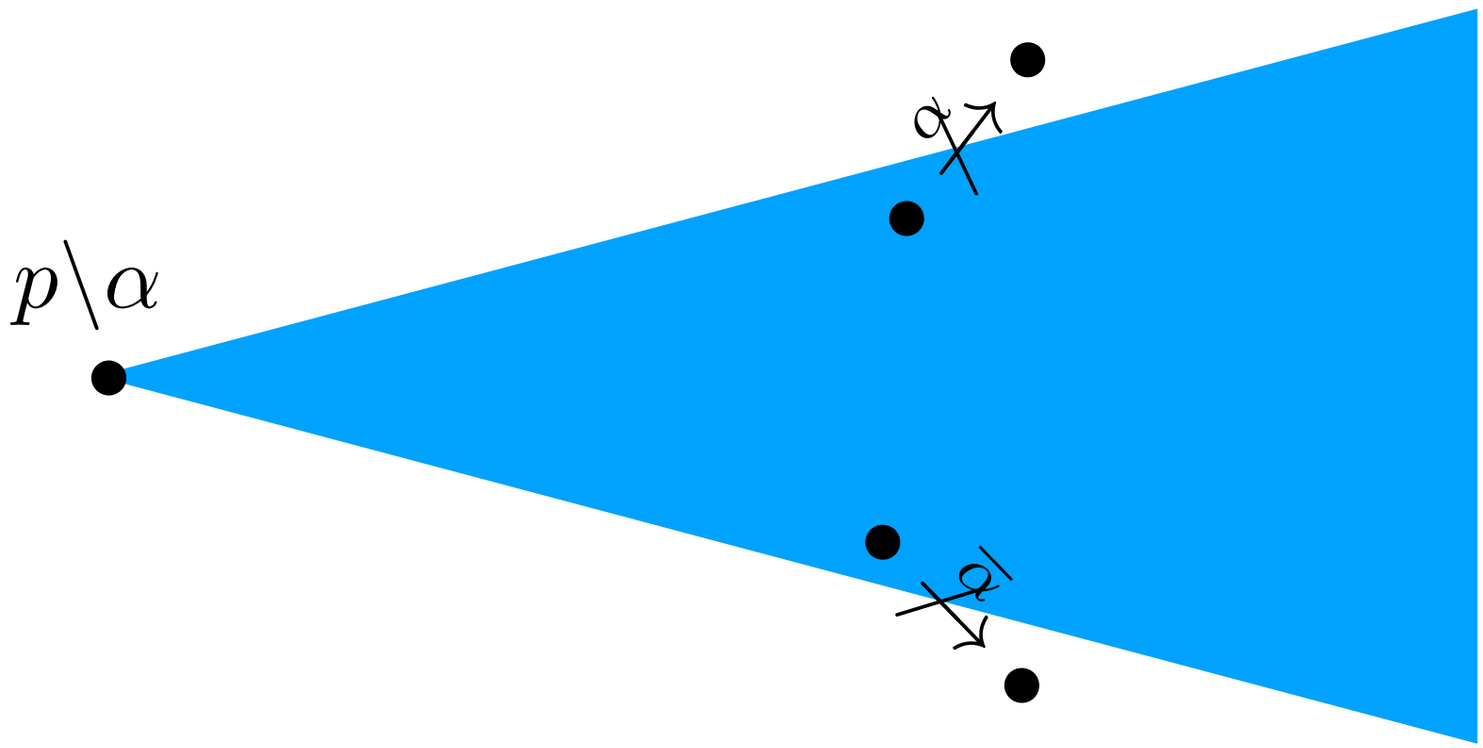
$$S \triangleq \{coin, coffee, tea\}$$

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

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

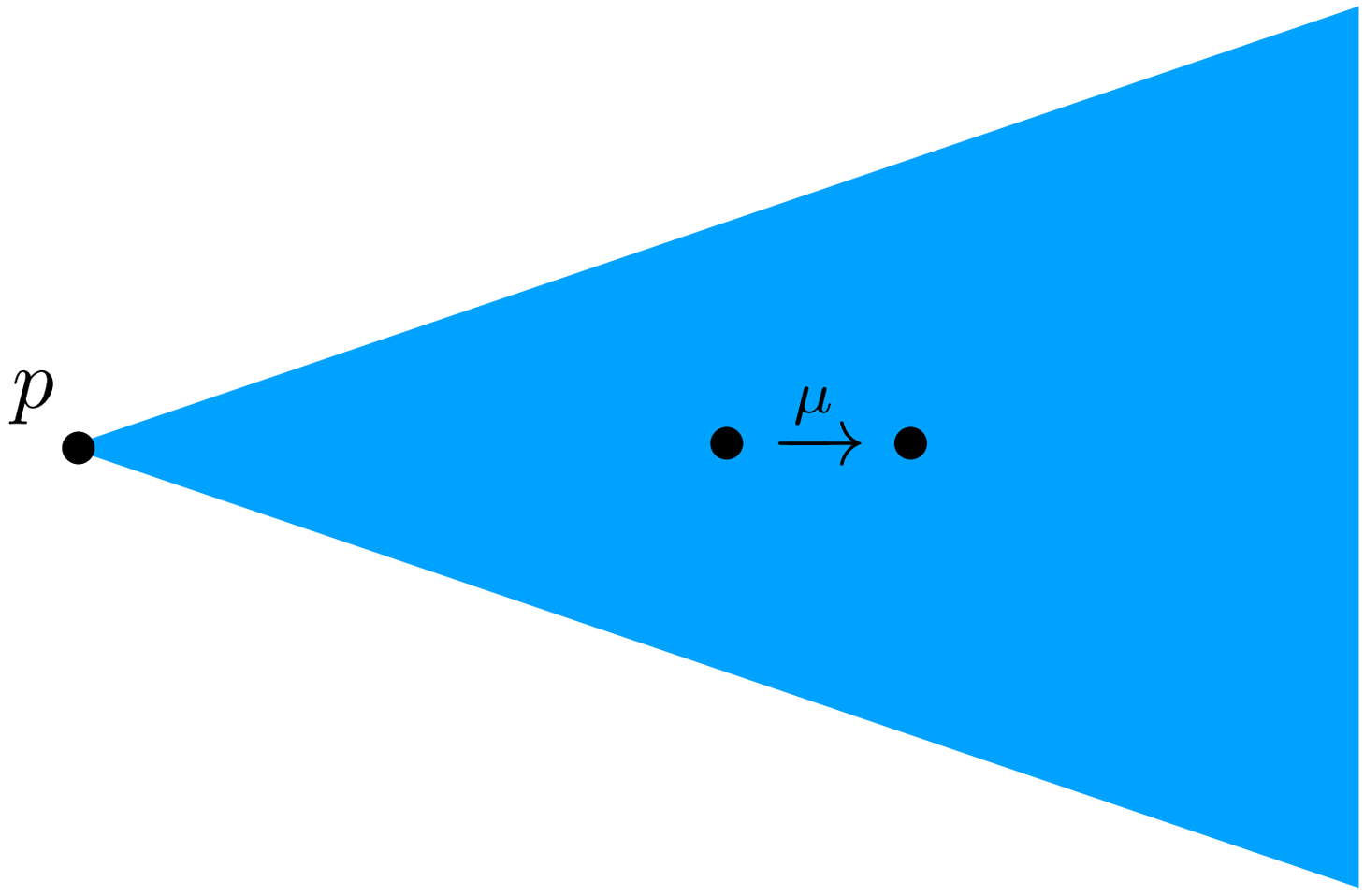
$P \triangleq \overline{coin}.coffee$

$\phi(coin) = moneta$

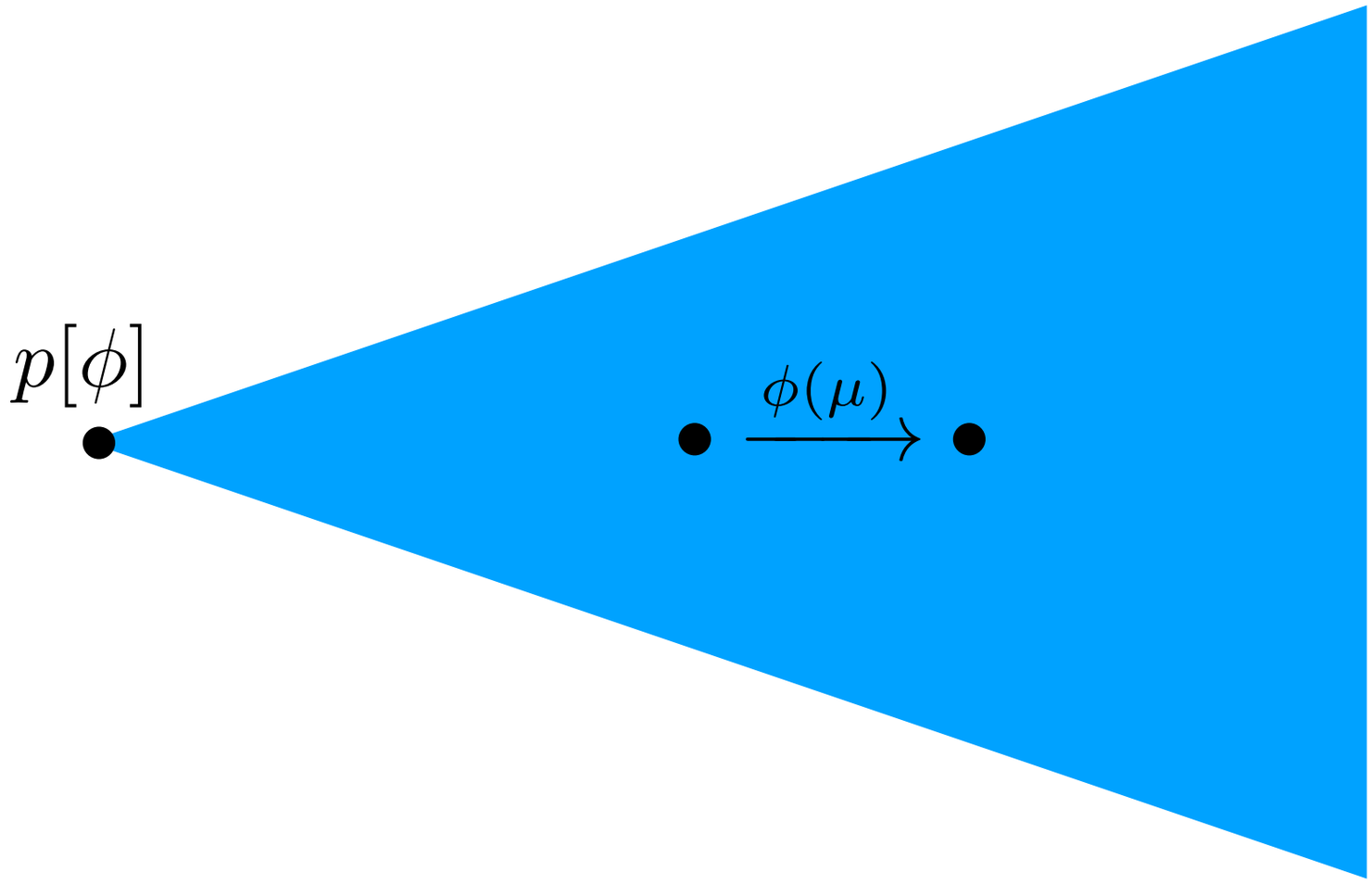
$\phi(coffee) = caffè$

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

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

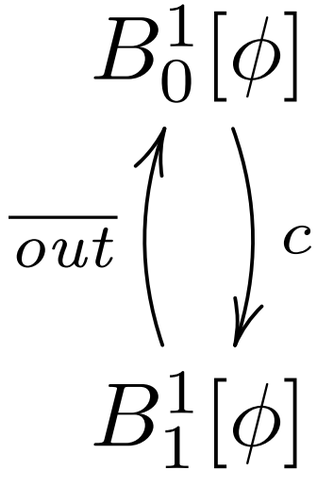
ParR) $$\frac{p_2 \xrightarrow{\mu} q_2}{p_1|p_2 \xrightarrow{\mu} p_1|q_2}$$

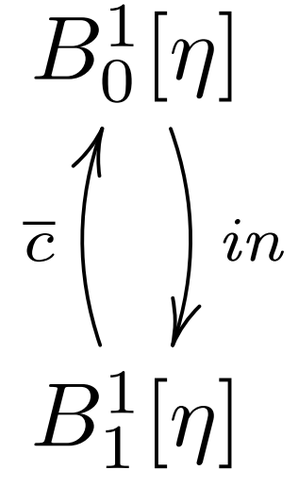
Rec) $$\frac{p\left[{}^{\mathbf{rec}\ x.\ p}/_{x}\right] \xrightarrow{\mu} q}{\mathbf{rec}\ x.\ p \xrightarrow{\mu} q}$$

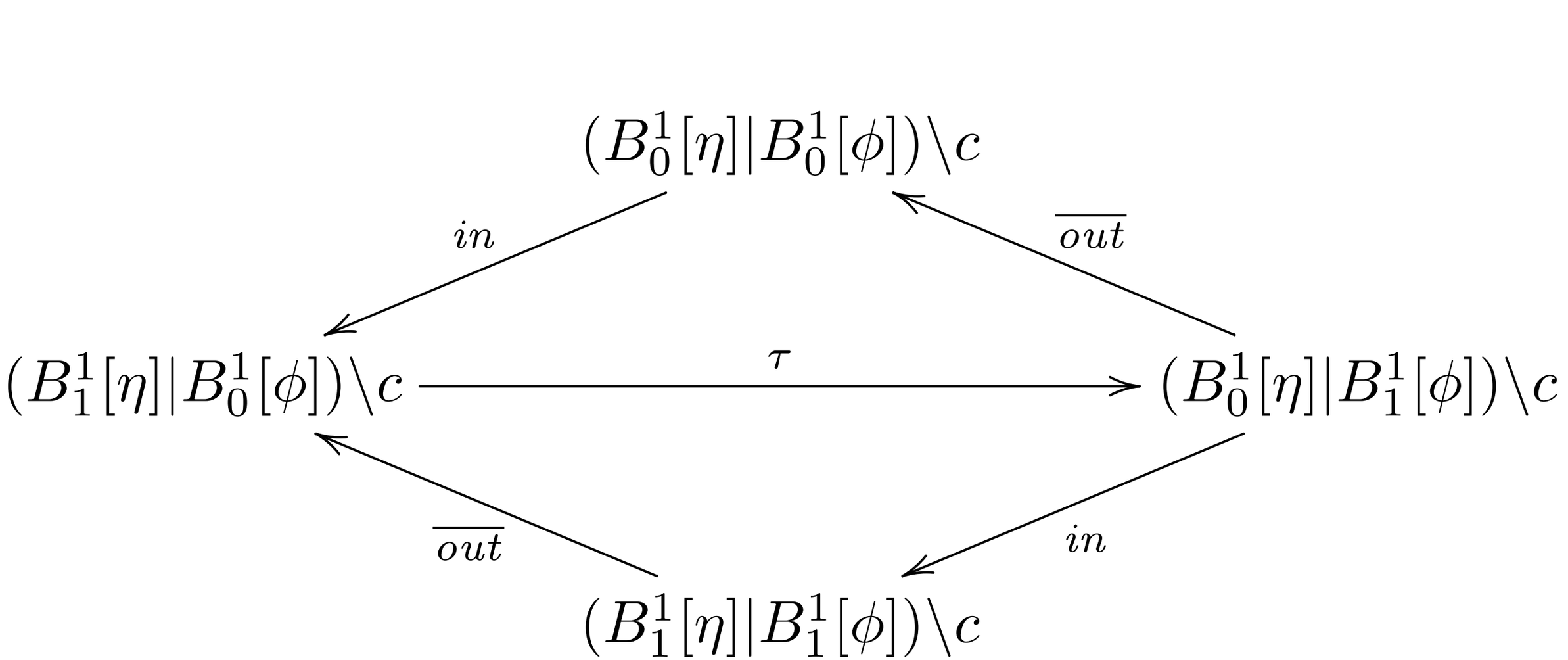
Linked buffers

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

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



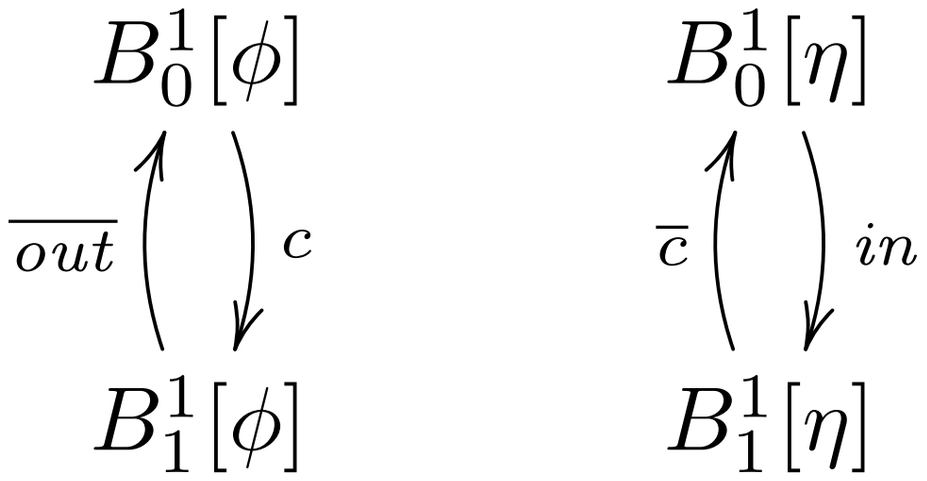


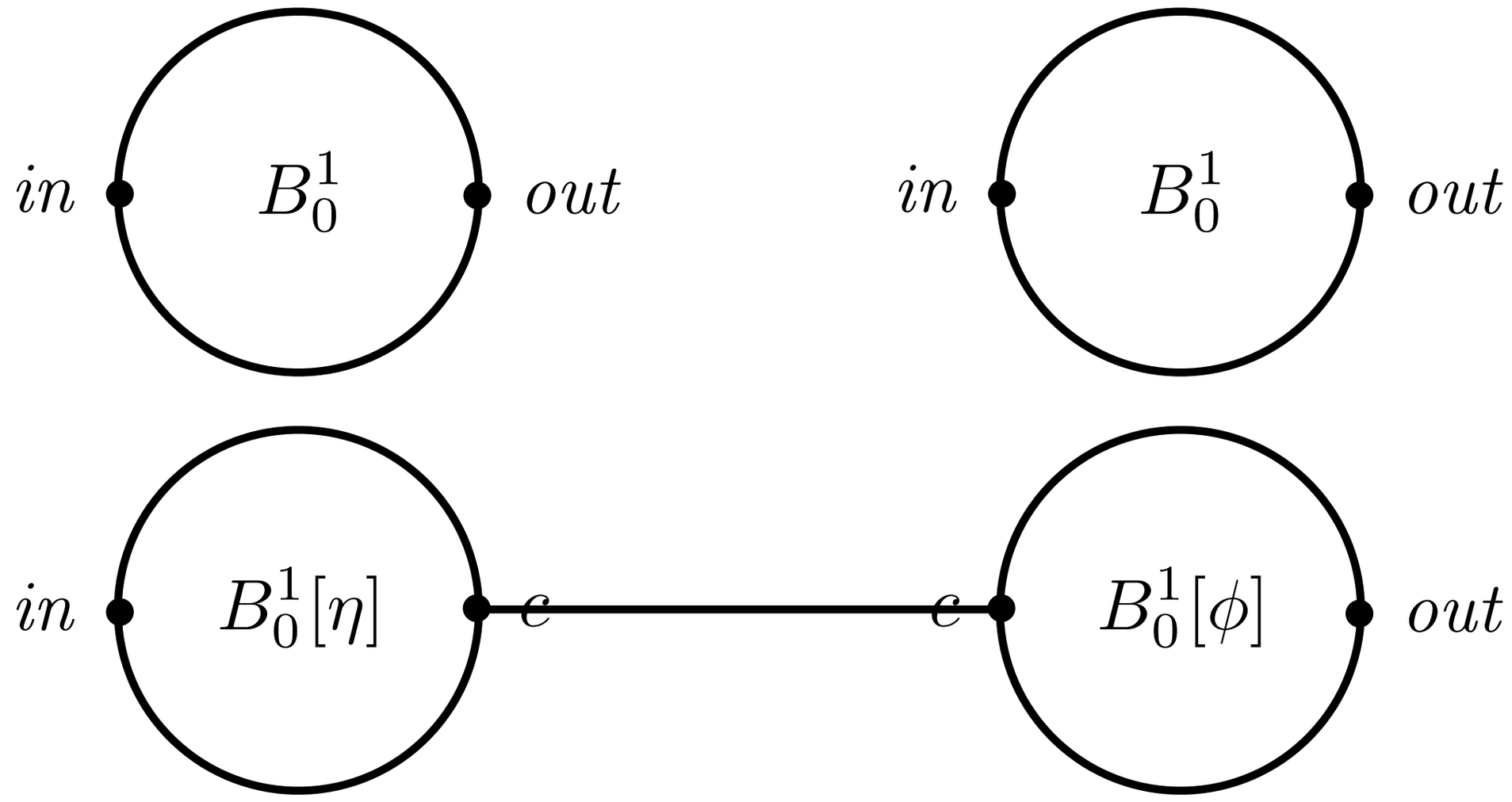


Linked buffers

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

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



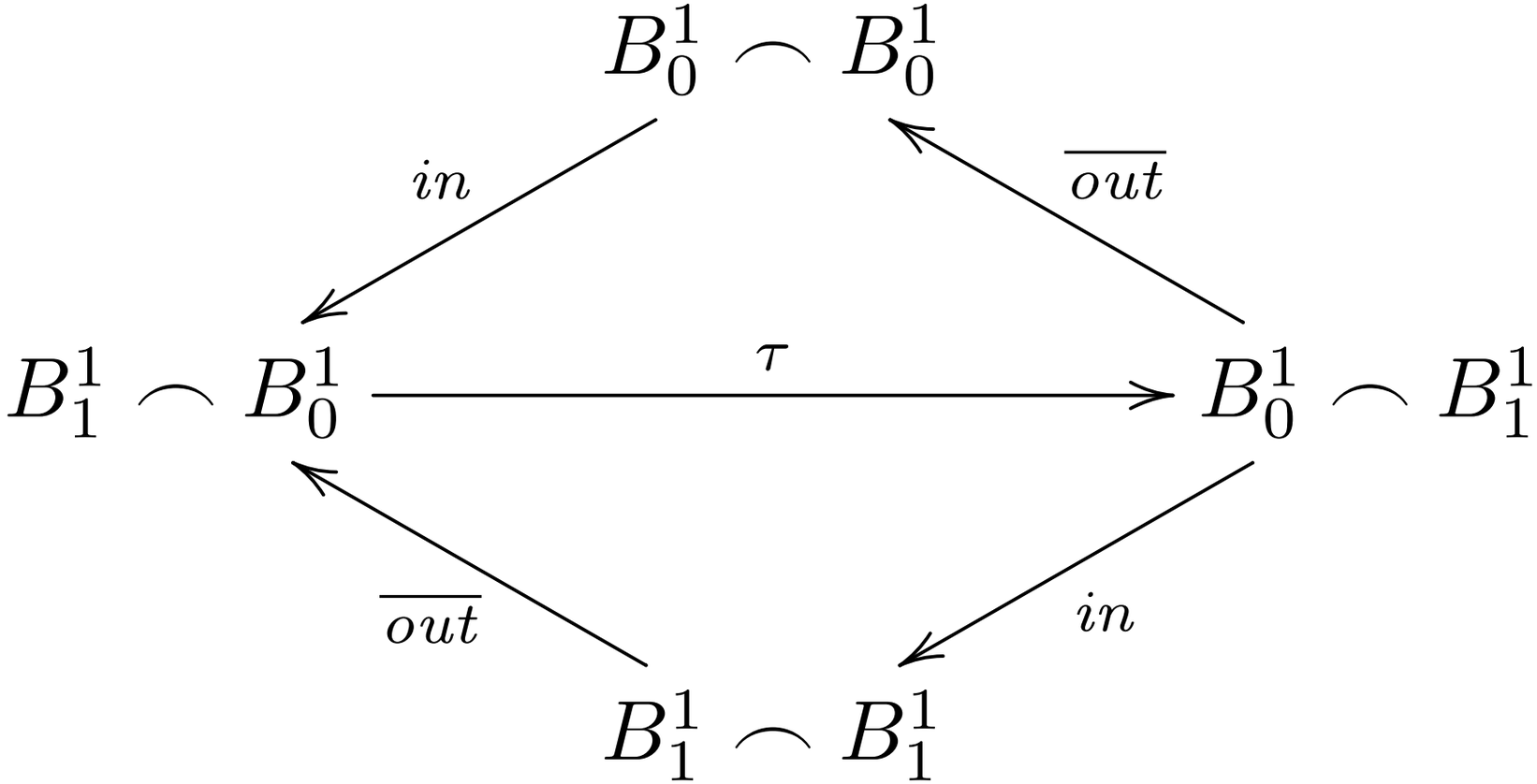


Linked buffers

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

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

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



Linked boolean buffers

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

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

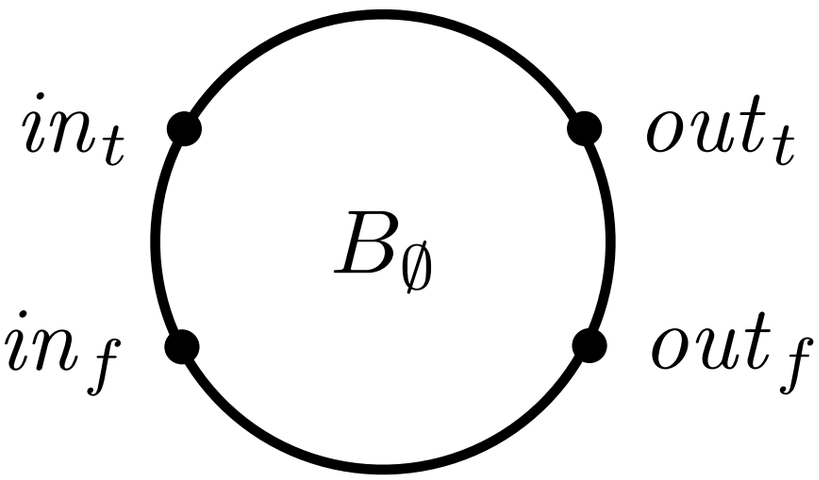
$$B_f \triangleq \overline{out}_f.B_\emptyset \qquad p \frown q \triangleq (p[\eta]|q[\phi])\backslash\{c_t, c_f\}$$

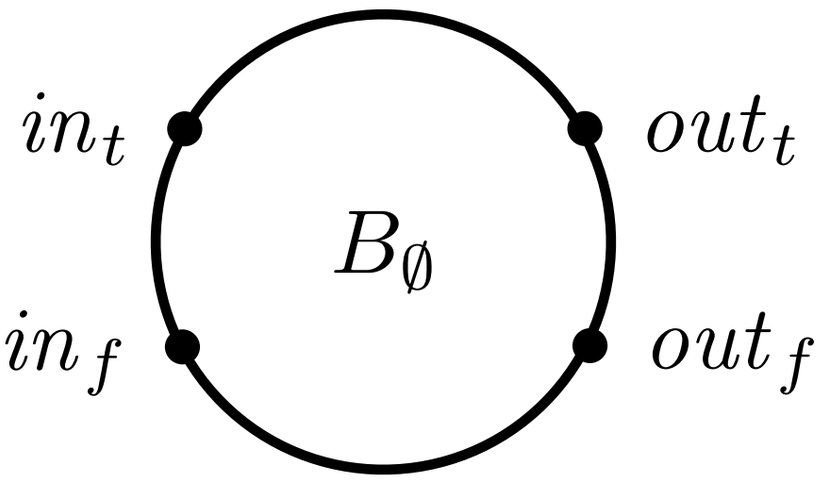
Linked boolean buffers

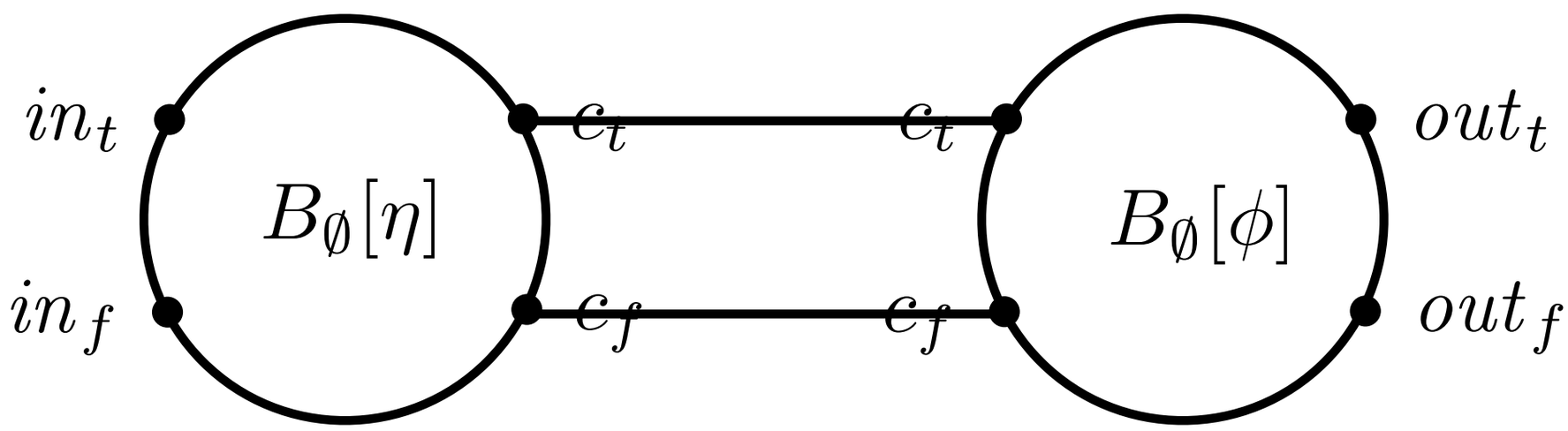
$$B_\emptyset \triangleq in_t.B_t + in_f.B_f$$

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

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





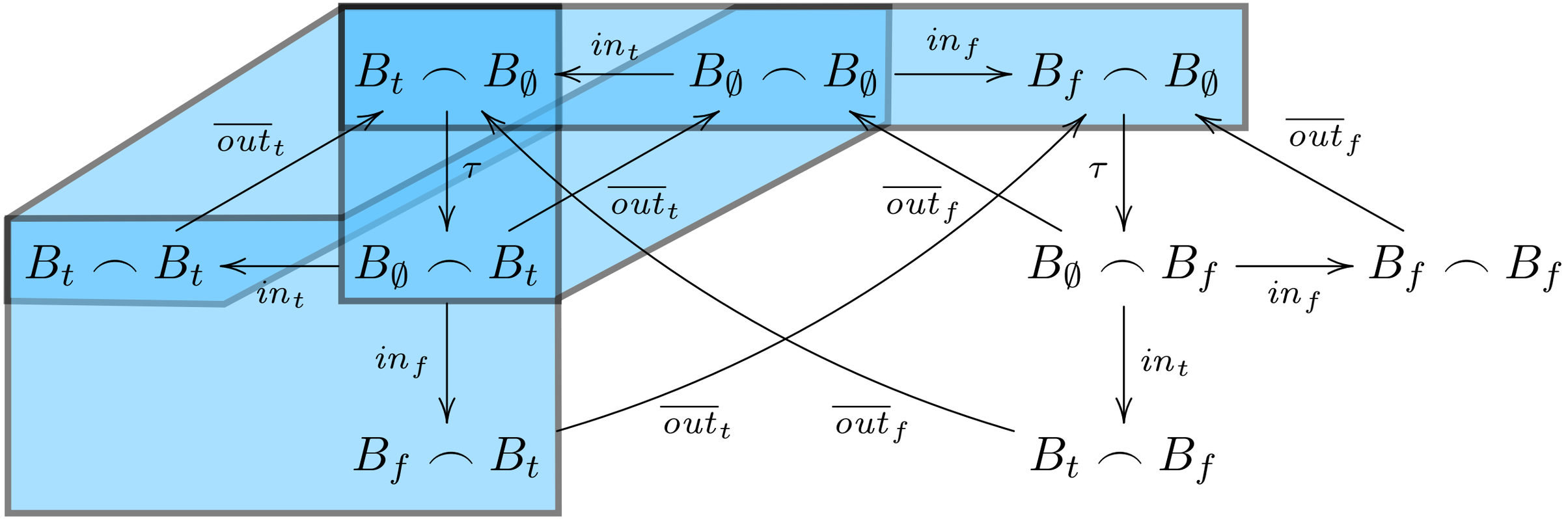


Linked boolean buffers

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

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

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

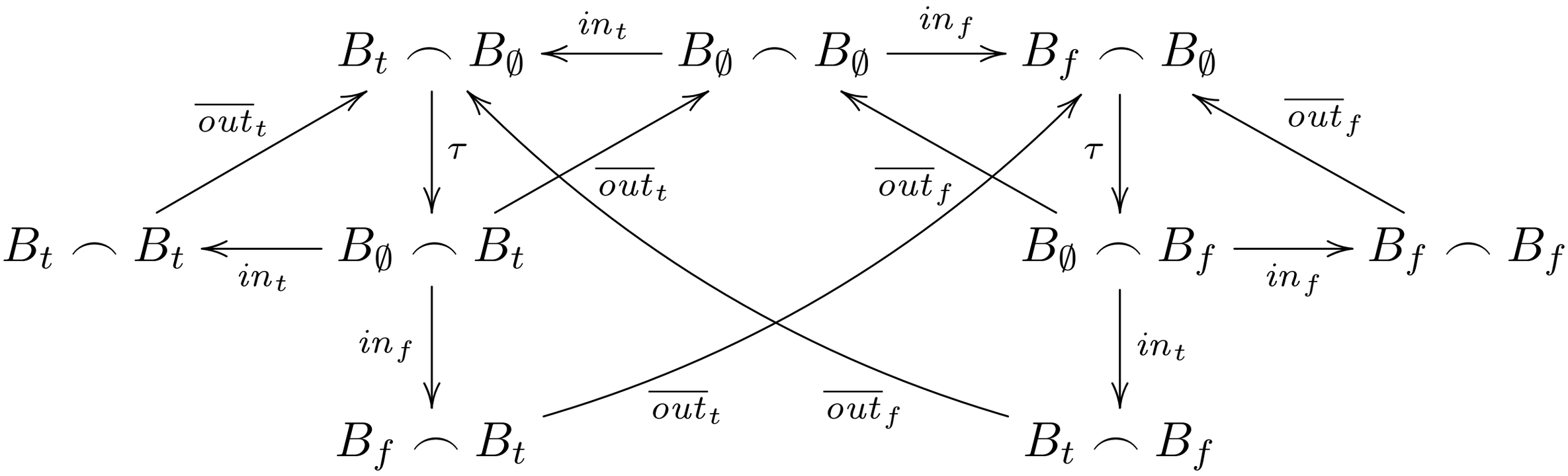


Linked boolean buffers

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

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

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



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 $V \triangleq \{v_1, ..., v_n\}$

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

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

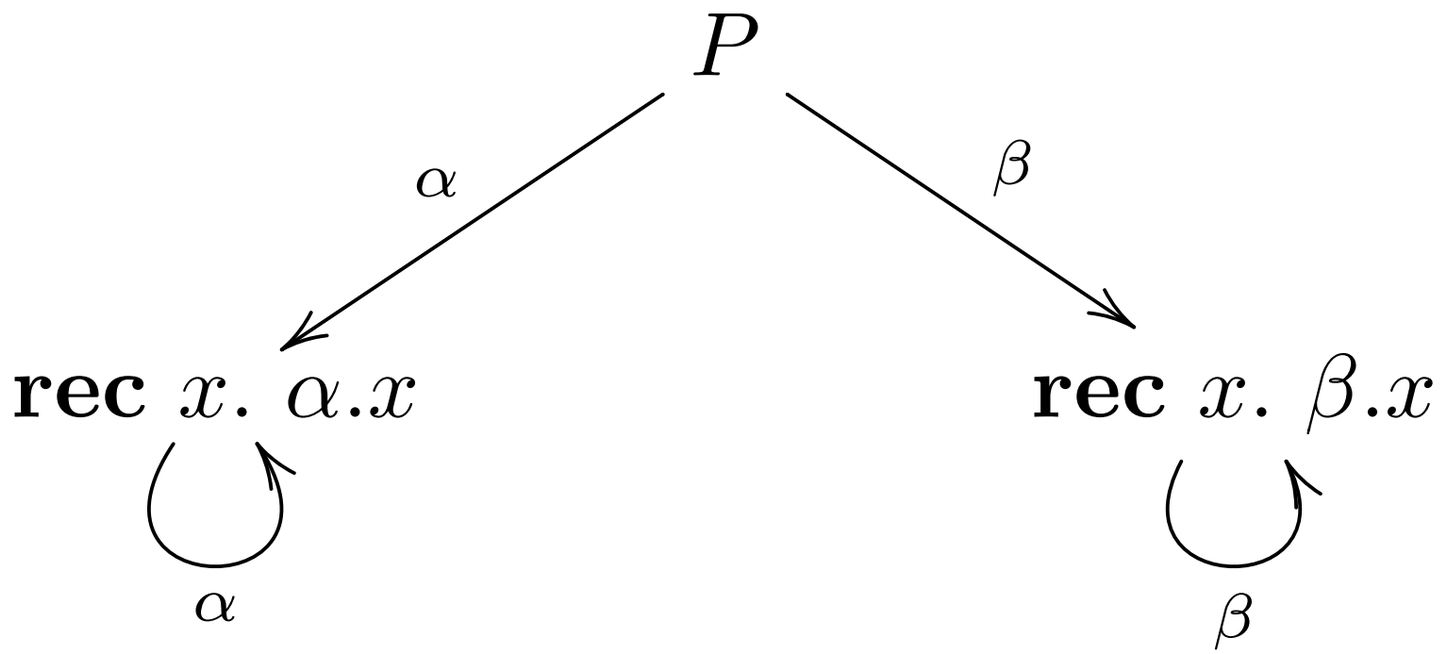
```
receive
        v -> p
        w -> q
        _ -> r
end
```

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



Exercise: LTS?

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



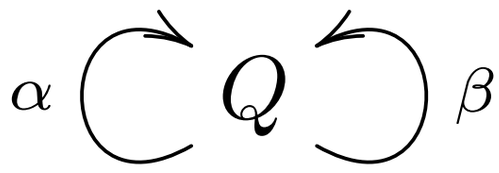


Exercise: LTS?

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

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

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



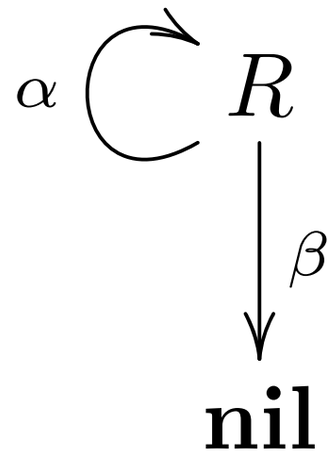


Exercise: LTS?

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

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

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



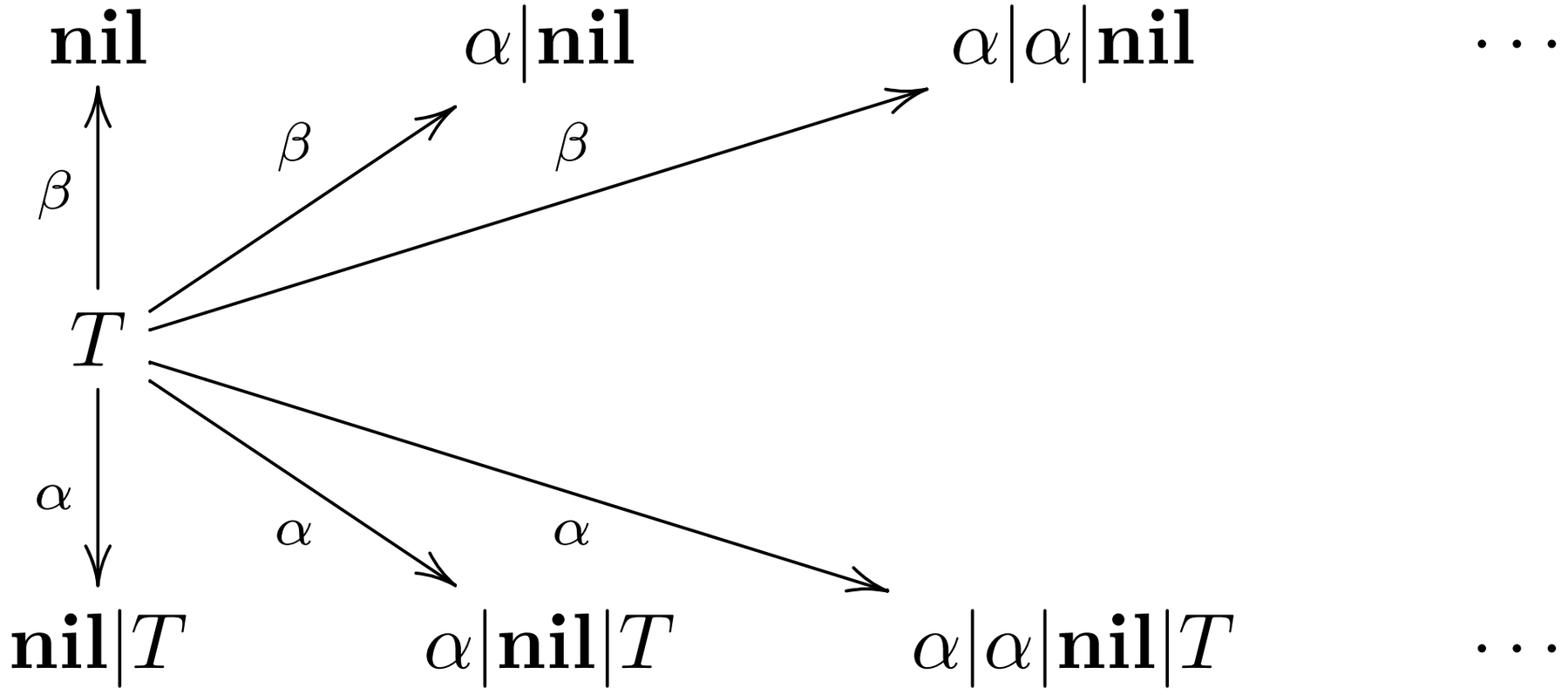


Exercise: LTS?

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

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

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



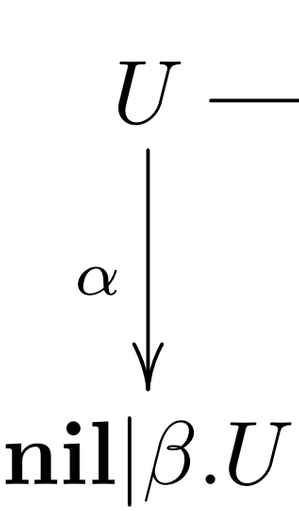


Exercise: LTS?

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

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

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



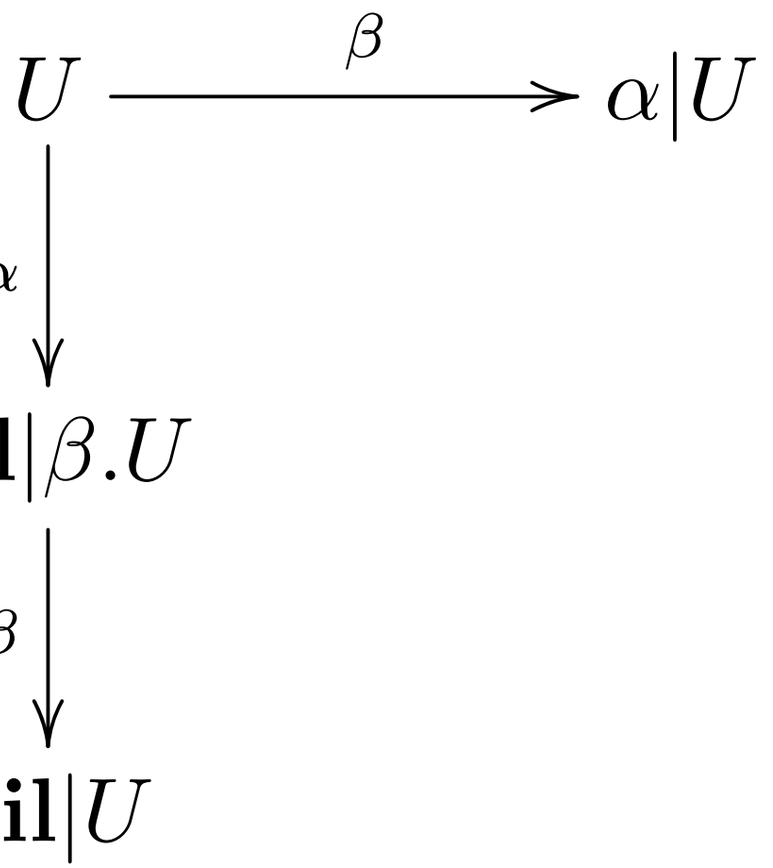


Exercise: LTS?

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

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

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



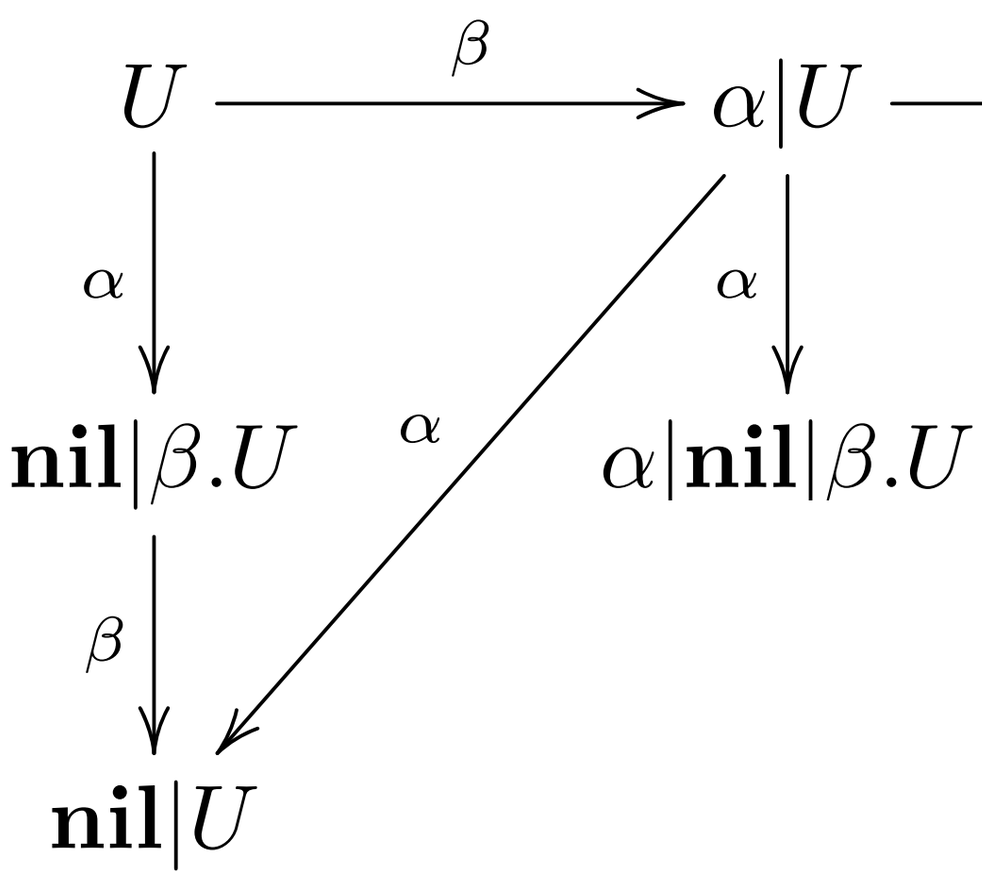


Exercise: LTS?

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

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

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



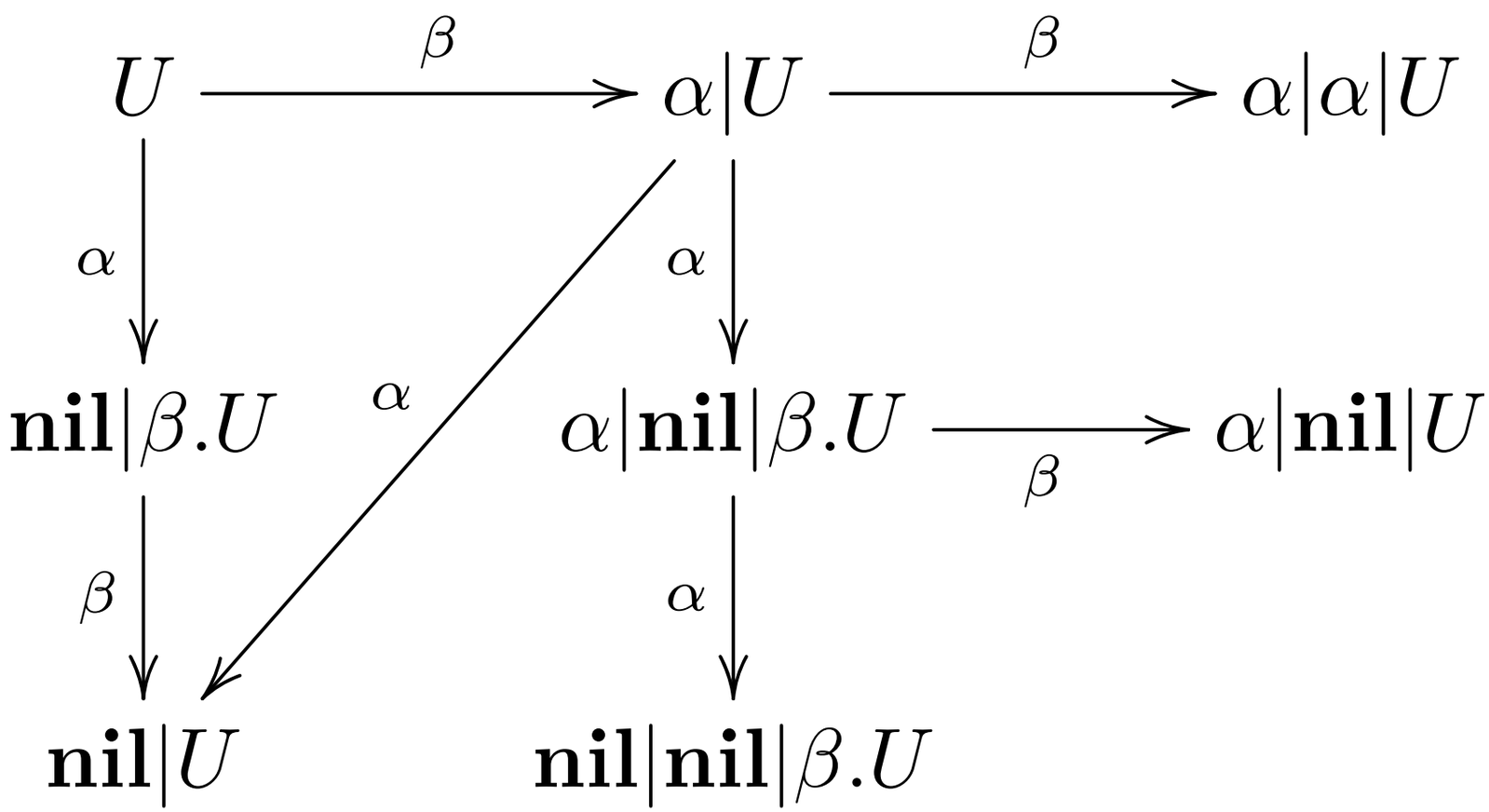


Exercise: LTS?

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

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

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



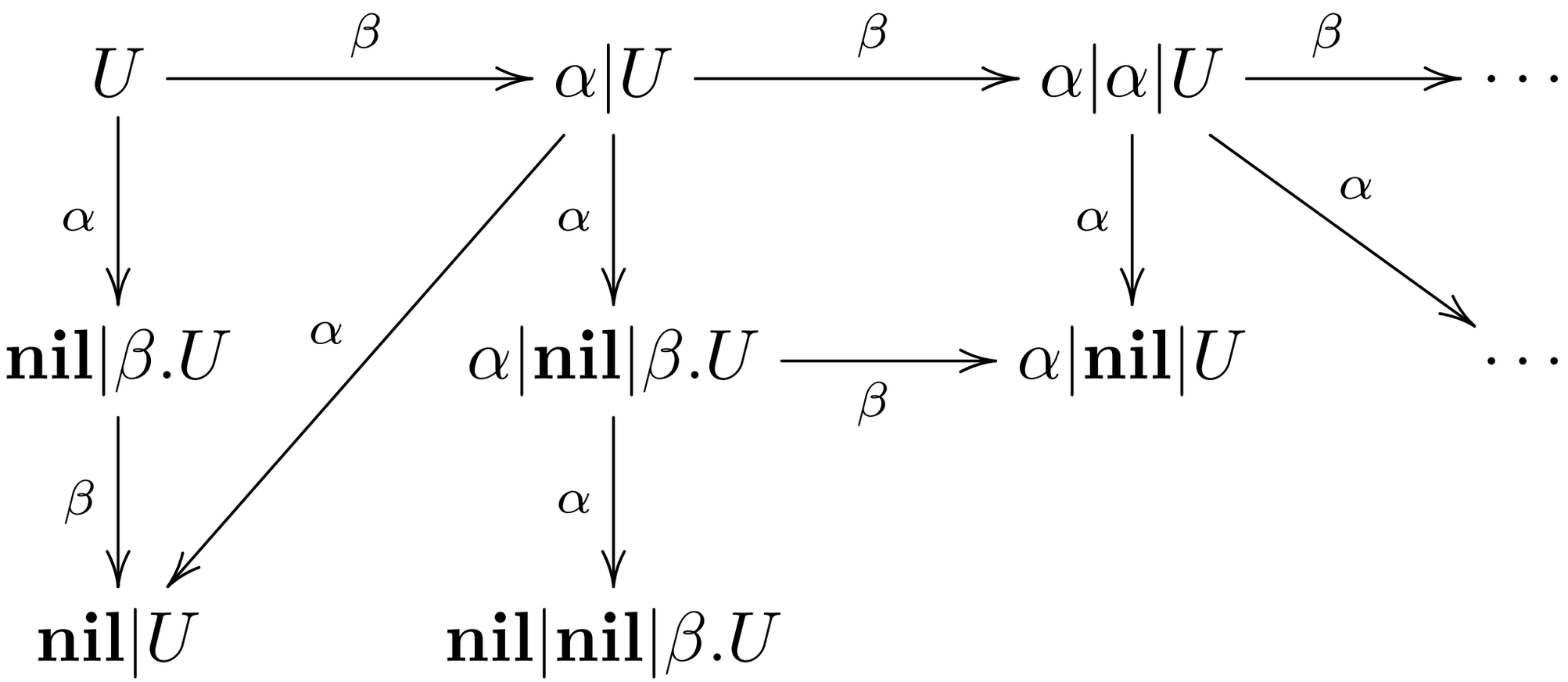


Exercise: LTS?

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

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

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



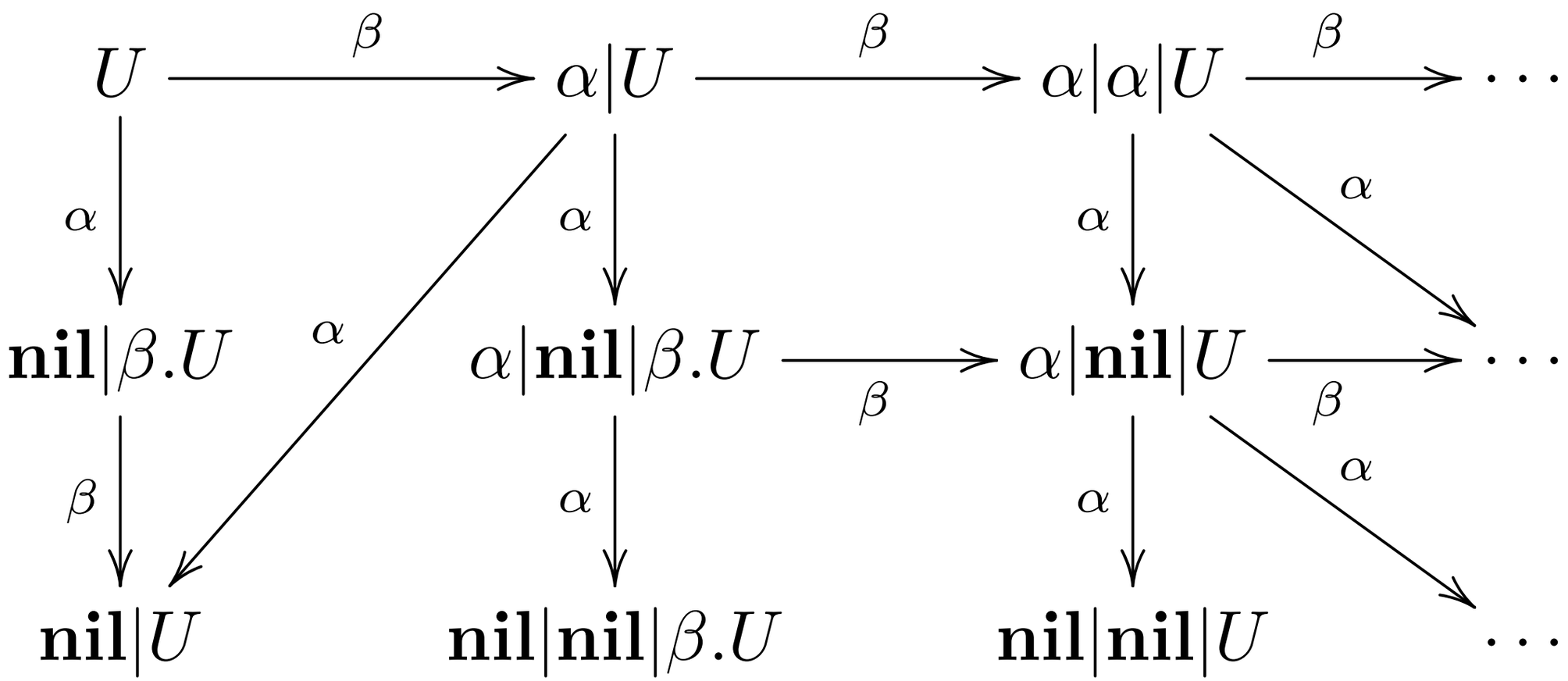


Exercise: LTS?

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

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

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





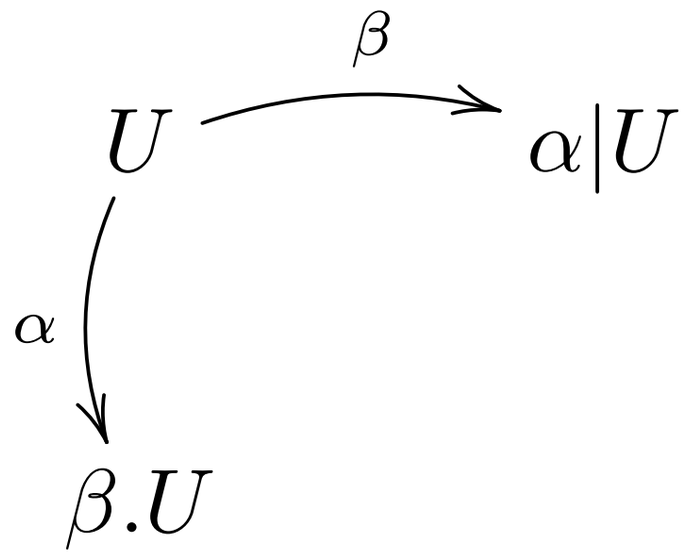
Exercise: LTS?

let's ignore **nil**

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

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

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





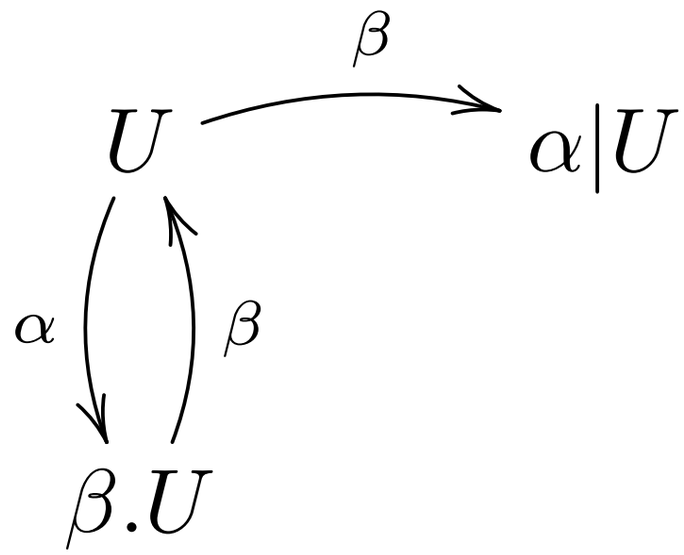
Exercise: LTS?

let's ignore **nil**

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

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

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





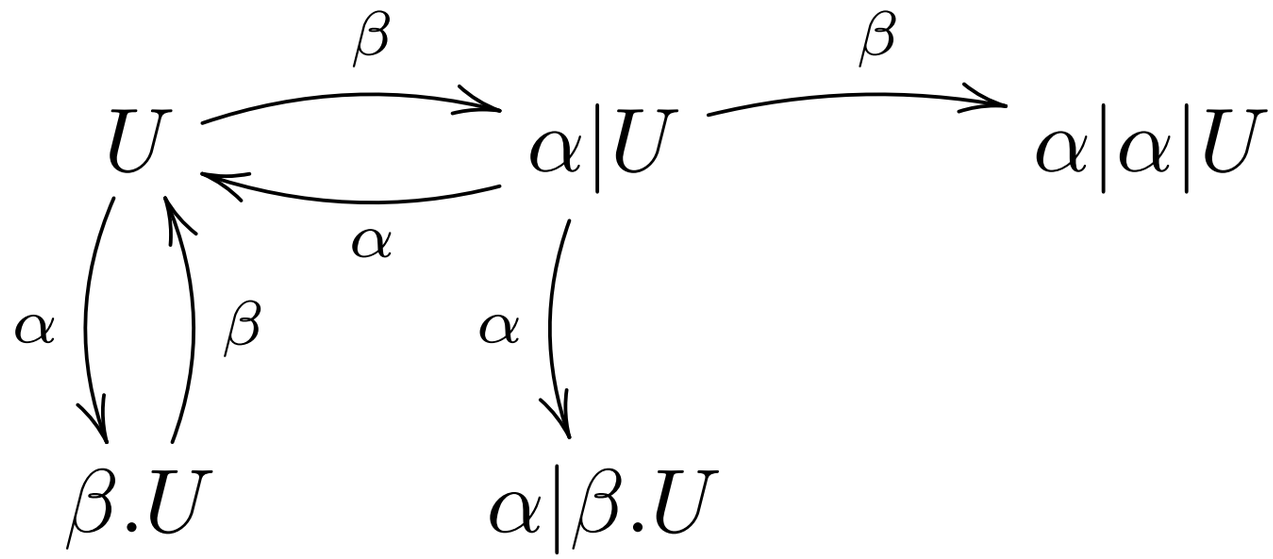
Exercise: LTS?

let's ignore **nil**

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

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

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





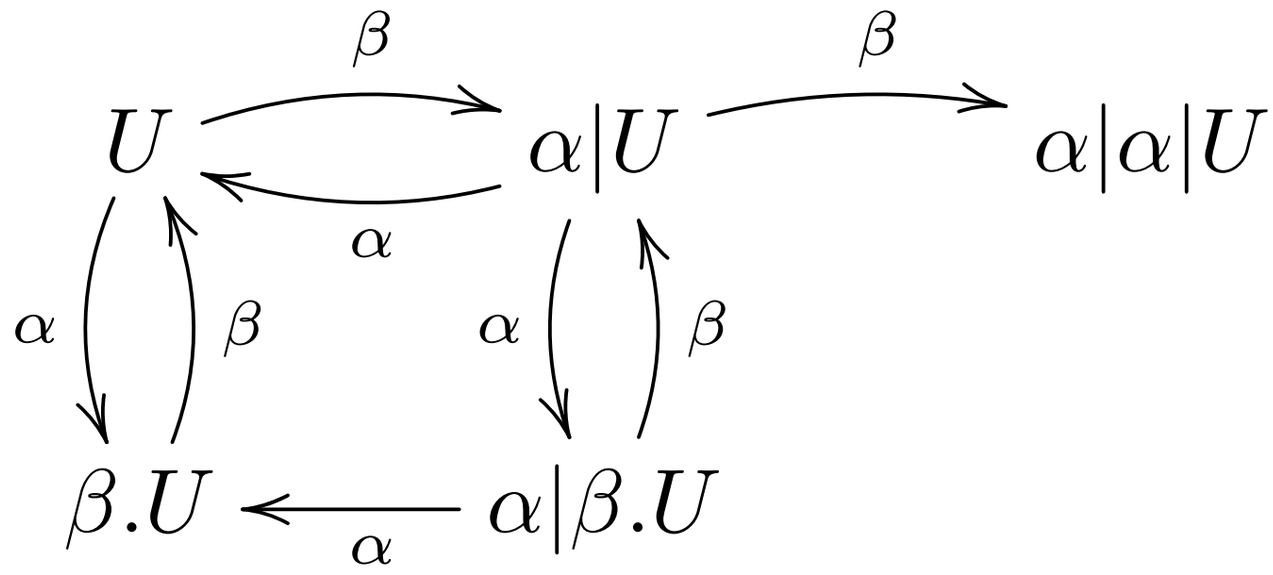
Exercise: LTS?

let's ignore **nil**

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

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

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





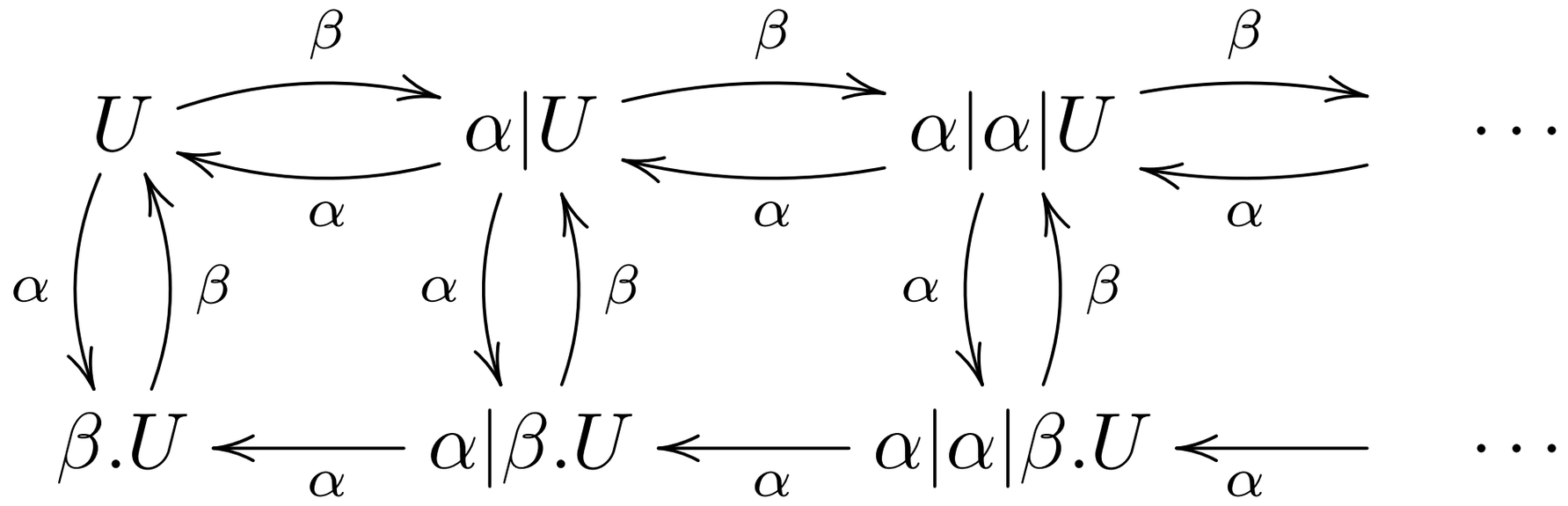
Exercise: LTS?

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:

c_0, c_1, c_2, c_3

used to display the current value of the counter

Draw the LTS of the counter process.

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)
	$\mid$	$p \mid q$	parallel composition
	$\mid$	$\mathbf{rec}\ x.\ p$	recursion

(operators are listed in order of precedence)

Some notation

write $\sum_{i=1}^{n} p_i$ instead of $p_1 + \cdots + p_n$

write $\prod_{i=1}^{n} p_i$ instead of $p_1 | \cdots | p_n$

write $p \backslash \{a_1, ..., a_n\}$ instead of $p \backslash a_1 \cdots \backslash a_n$

write $\mu^n.p$ instead of $\underbrace{\mu.\mu....\mu.}_{n} p$

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

CCS

Encoding imperative languages

Preliminaries: termination

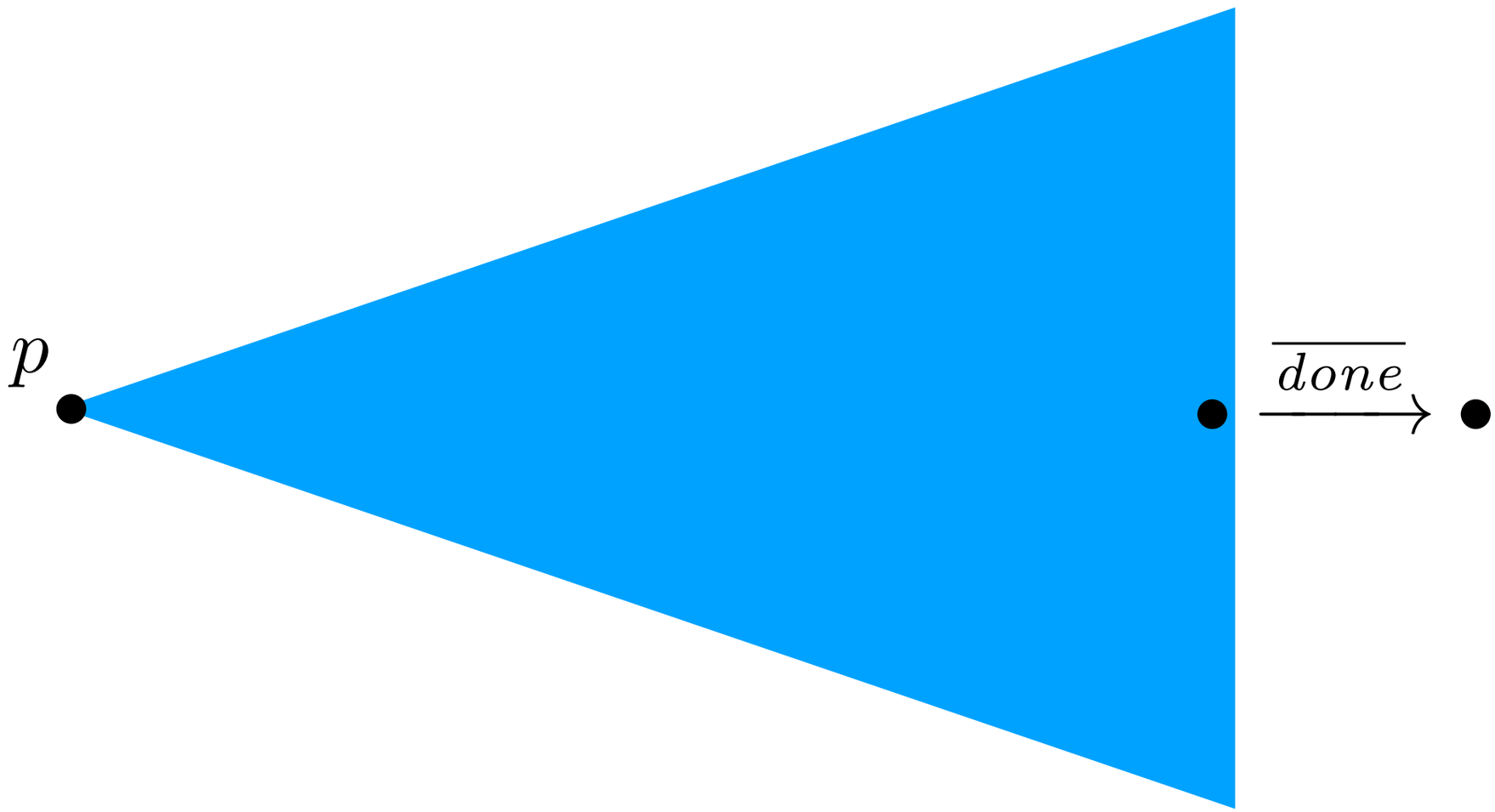
A dedicated channel *done*:

a message is sent when the current command terminates

$$\text{Done} \triangleq \overline{done}$$

$$\text{Done} \xrightarrow{\overline{done}} \mathbf{nil}$$

Termination



Skip

skip

does nothing and sends *done*

$$\tau.\text{Done}$$

$$\bullet \xrightarrow{\tau} \text{Done} \xrightarrow{\overline{done}} \mathbf{nil}$$

Variables

x ranging over $V = \{v_1, ..., v_n\}$

a dedicated process for managing each variable

we can read its current value (channel xr_i)

we can write any value (channel xw_i)

$$\mathsf{XW} \triangleq \sum_{i=1}^{n} xw_i.\mathsf{X}_i$$
$$= xw_1.\mathsf{X}_1 + \cdots + xw_n.\mathsf{X}_n$$

$$\mathsf{X}_i \triangleq \overline{xr}_i.\mathsf{X}_i + \mathsf{XW}$$

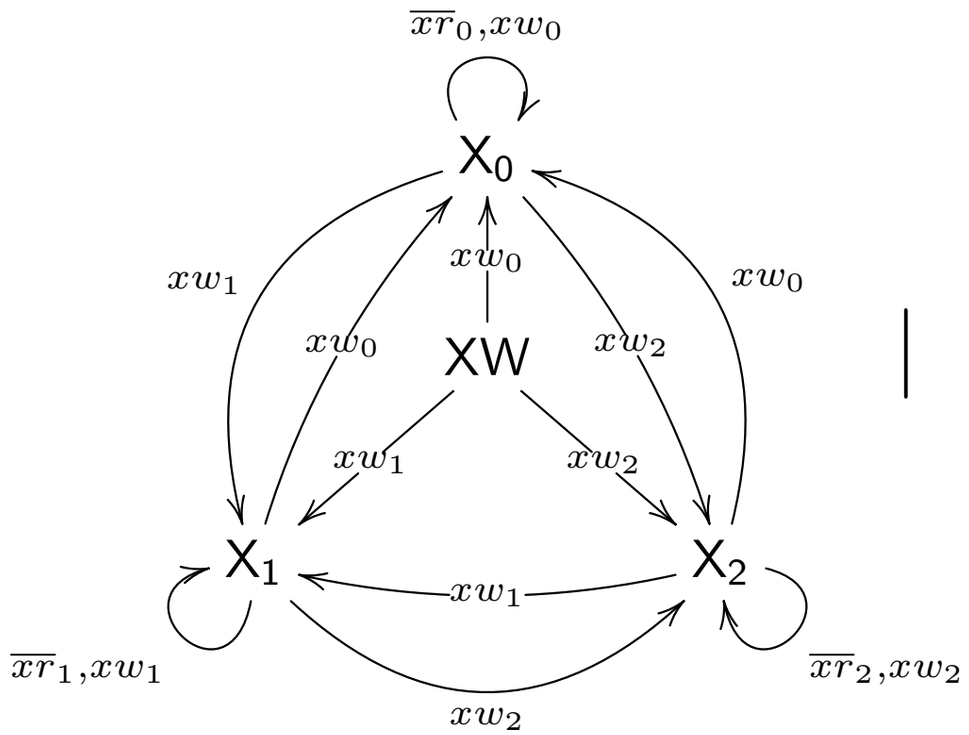
Variable declaration

var x

releases an uninitialised variable and terminates

XW|Done

$V = \{v_0, v_1, v_2\}$



|

Done $\xrightarrow{\overline{done}}$ **nil**

Assignment

$$x := v_i$$

sends a message to change the state of the variable and then terminates

$$\overline{xw}_i.\text{Done}$$



Sequential composition

$$c_1 \,;\, c_2$$

suppose p_1 models c_1

p_2 models c_2

$$p_1 | done.p_2$$

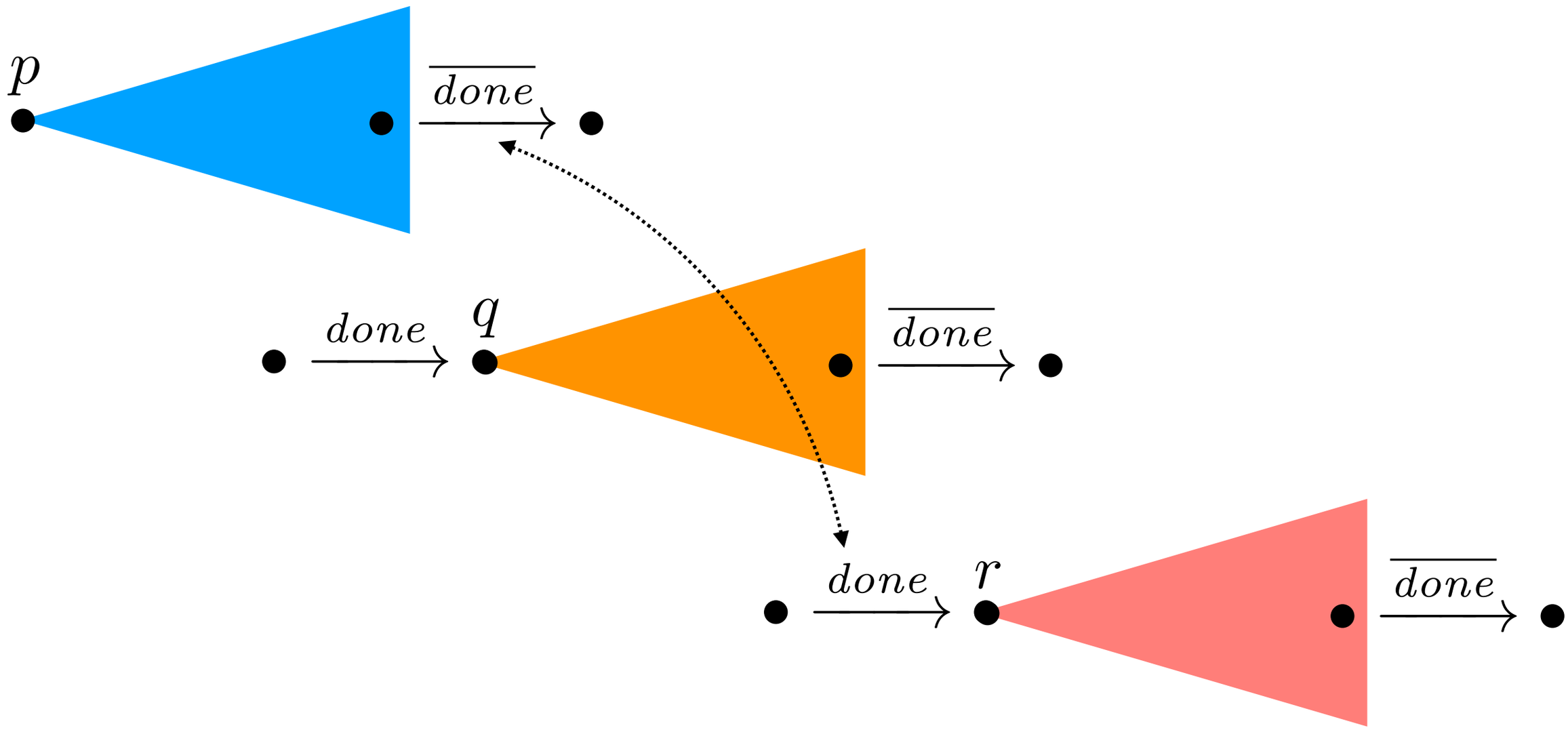
unfortunate choice: does not scale

$$c_1; c_2; c_3$$

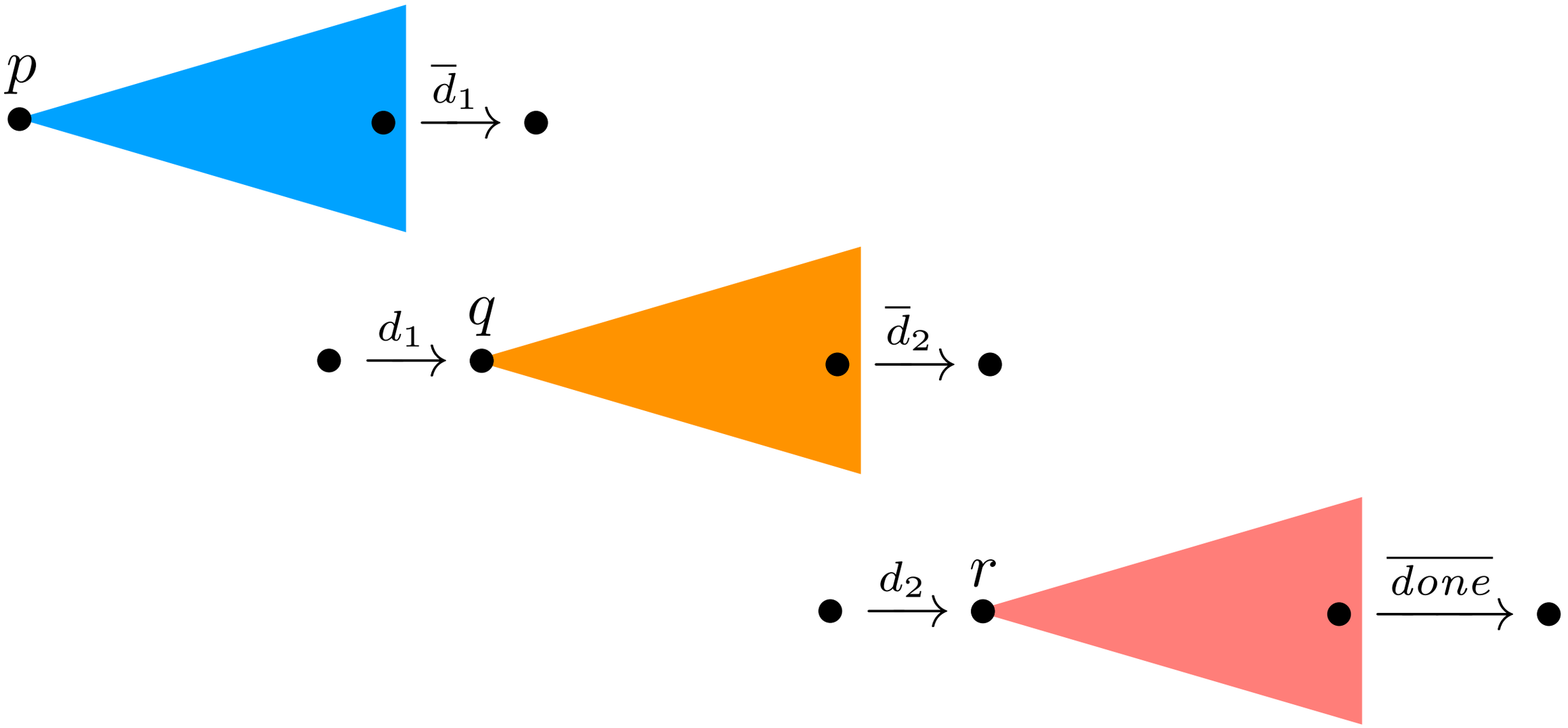
$$p_1 | done.p_2 | done.p_3$$

p_3 can start after p_1

Sequential?



Sequential!



Sequential compositon

$$c_1 \,;\, c_2$$

suppose p_1 models c_1

p_2 models c_2

$$\phi_d(done) = d$$

$$p_1 \frown p_2 \triangleq (p_1[\phi_d] | d.p_2)\backslash d$$

now d is local to p_1 and p_2

$$(\ ((p_1[\phi_{d_1}] \mid d_1.p_2)\backslash d_1)[\phi_{d_2}] \mid d_2.p_3)\backslash d_2$$

$$(\ ((p_1[\phi_d] \mid d.p_2)\backslash d)[\phi_d] \mid d.p_3)\backslash d$$

$$(p_1 \frown p_2) \frown p_3$$

Conditional

if $x = v_i$ then c_1 else c_2

suppose p_1 models c_1

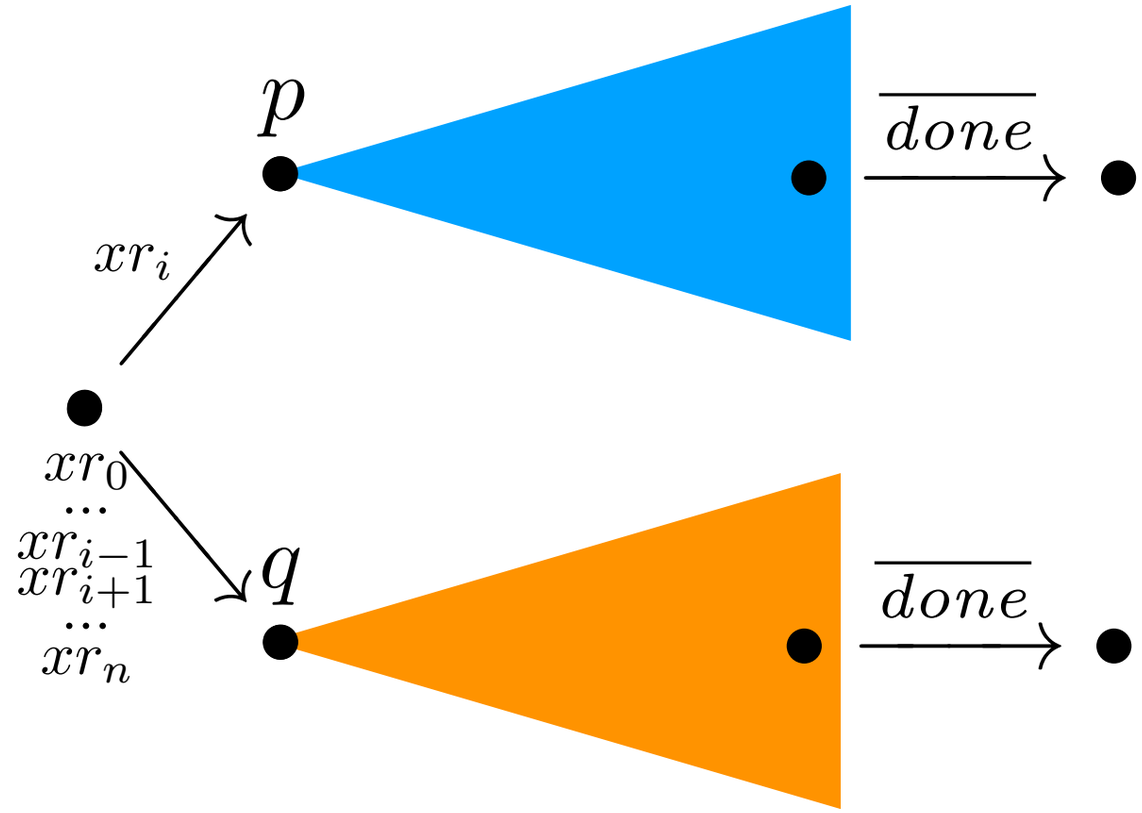
p_2 models c_2

receives the state of the variable

and then chooses accordingly

$$xr_i.p_1 + \sum_{j \neq i} xr_j.p_2$$

Conditional



Iteration

while $x = v_i$ do c

suppose p models c

receives the state of the variable
and then chooses accordingly, possibly recurring

$$\mathbf{rec}\ y.\ xr_i.(p[\phi_d]|d.y)\backslash d + \sum_{j \neq i} xr_j.\mathsf{Done}$$

$$Y \triangleq xr_i.(p[\phi_d]|d.Y)\backslash d + \sum_{j \neq i} xr_j.\mathsf{Done}$$

$$Y \triangleq xr_i.(p \frown Y) + \sum_{j \neq i} xr_j.\mathsf{Done}$$

Concurrency

$$c_1 \mid c_2$$

suppose

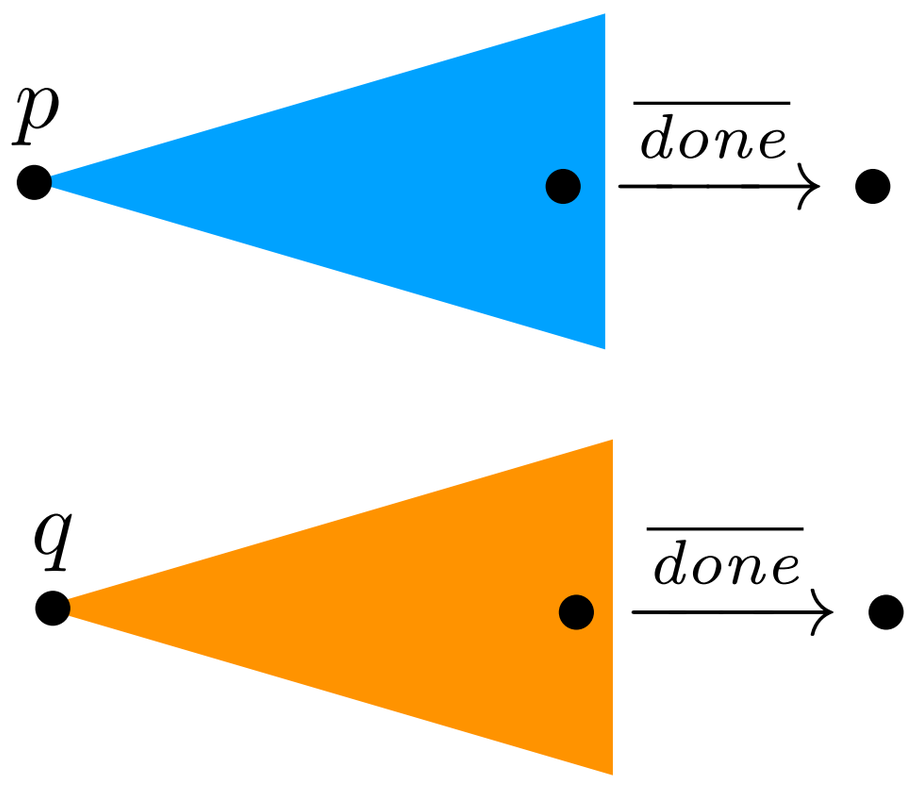
p_1 models c_1

p_2 models c_2

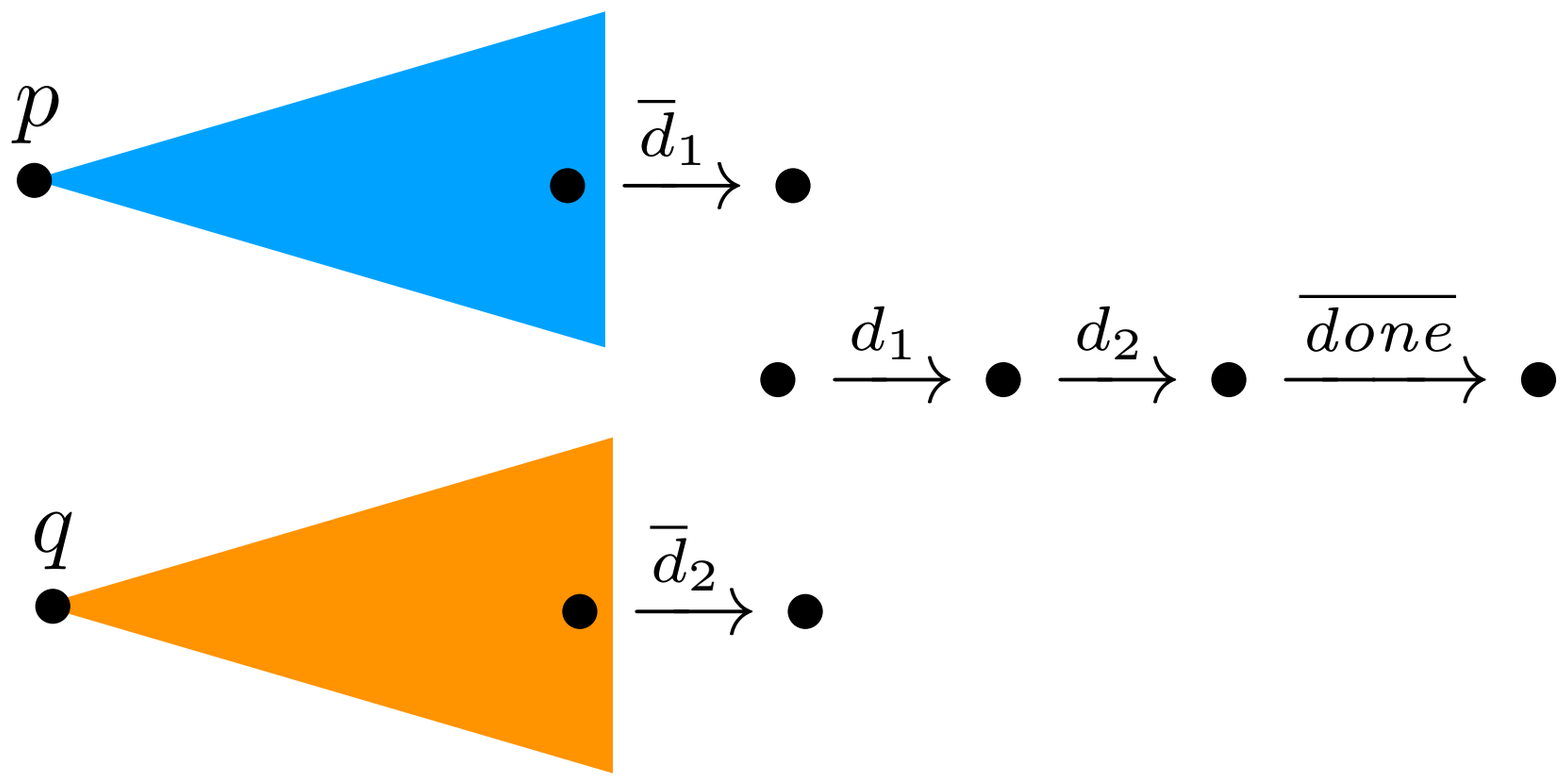
$$p_1 \mid p_2$$

unfortunate choice: *done* is possibly issued twice

Concurrent termination



Joint termination



Concurrency

$$c_1 \mid c_2$$

suppose p_1 models c_1

p_2 models c_2

$$(p_1[\phi_{d_1}] \mid p_2[\phi_{d_2}] \mid d_1.d_2.\mathsf{Done})\backslash d_1 \backslash d_2$$

the order is not important: both must be done to terminate

$$(p_1[\phi_d] \mid p_2[\phi_d] \mid d.d.\mathsf{Done})\backslash d$$

$$(p_1[\phi_d] \mid p_2[\phi_d] \mid d^2.\mathsf{Done})\backslash d$$

Finally

all channels for communicating with variables
must be restricted at the top
to guarantee read/write requests are synchronised

$$p \backslash \{xw_1, xr_1, \cdots\}$$

several optimisations are possible:
action prefix instead of linking for sequential composition
allows more expressive guards
remove silent transitions
introduce constants for loops
implement expressions
…