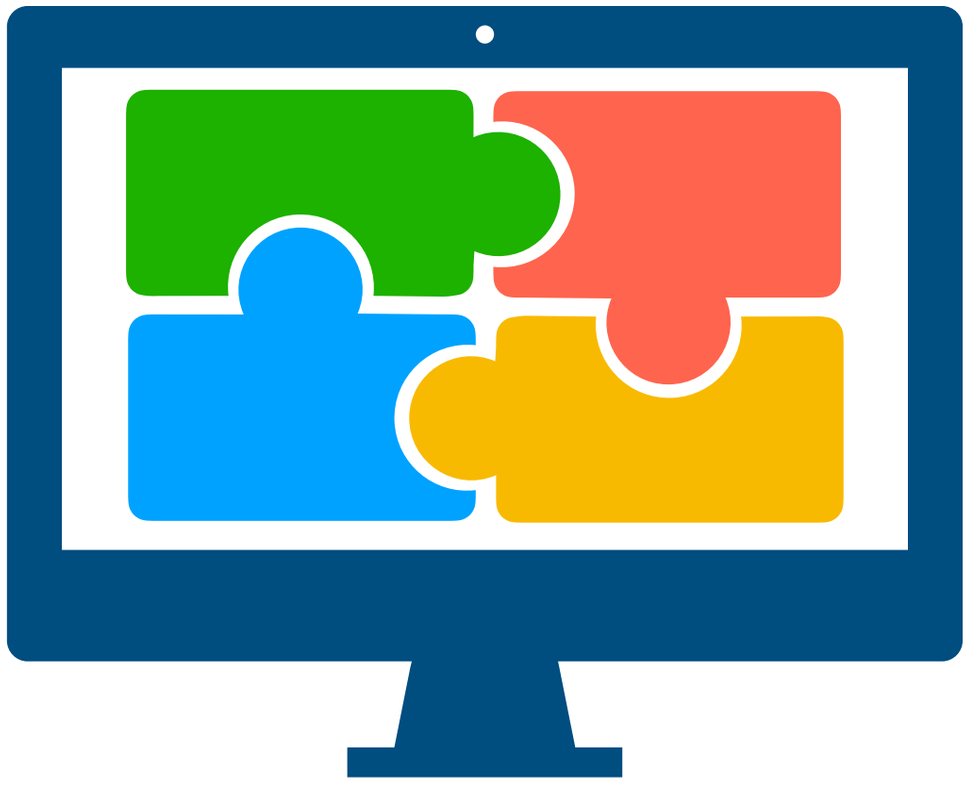


Linguaggi di Programmazione

Roberta Gori

CCS sintassi & semantica operazionale-II.1.,II.2,II.3

CCS

Calculus of Communicating Systems

Sequenziale vs Concorrente

Esempio

- Consideriamo un programma con 2 thread che stampano le sequenze di 3 lettere abc e xyz rispettivamente.

```
1. #include <pthread.h>
2. #include <stdio.h>
3. void * tf1()
4. {
5. printf("x");
6. printf("y");
7. printf("z");
8. return NULL;
9. }
10. void * tf2()
11. {
12. printf("a");
13. printf("b");
14. printf("c");
15. return NULL;
16. }
17. int main(void)
18. {
19. pthread_t tID1;
20. pthread_t tID2;
21. pthread_create(&tID1, NULL, &tf1, NULL);
22. pthread_create(&tID2, NULL, &tf2, NULL);
23. pthread_join(tID1, NULL);
24. pthread_join(tID2, NULL);
25. printf("\nfine\n");
26. return 0; }
```

per compilare
gcc main.c -lpthread

Aggiungendo un ritardo tipo
for (i=0; i<N; i++);
tra le istruzioni si possono ottenere tutte le seguenti stampe

abcxyz fine xyzabc fine axbycz fine

ma non

aybcxz fine

Concorrenza

IMP/HOFL (paradigmi sequenziali)
- determinatezza
- due programmi che non terminano sono equivalenti

paradigmi concorrenti
- presentano un nondeterminismo intrinseco agli osservatori esterni
- la non terminazione può essere una caratteristica desiderabile (ad esempio nei server)
- non tutti i processi non terminati sono equivalenti
- l'interazione è un problema primario
- sono necessarie nuove nozioni di comportamento / equivalenza

CCS: le basi

Algebra di processi

- focus su pochi operatori primitivi (caratteristiche essenziali)
- sintassi concisa per costruire e comporre processi
- non è un vero e proprio linguaggio di programmazione
- piena potenza di calcolo (equivalente di Turing)

Comunicazione

- binaria, passaggio di messaggi su canali

Semantica operativa strutturale

- small steps (Labelled Transition System)
- processi come stati
- interazioni come etichette
- definito da regole di inferenza
- definito per induzione sulla struttura dei processi

Transizioni con etichette

interazione continua
con l'ambiente
(con altri processi)

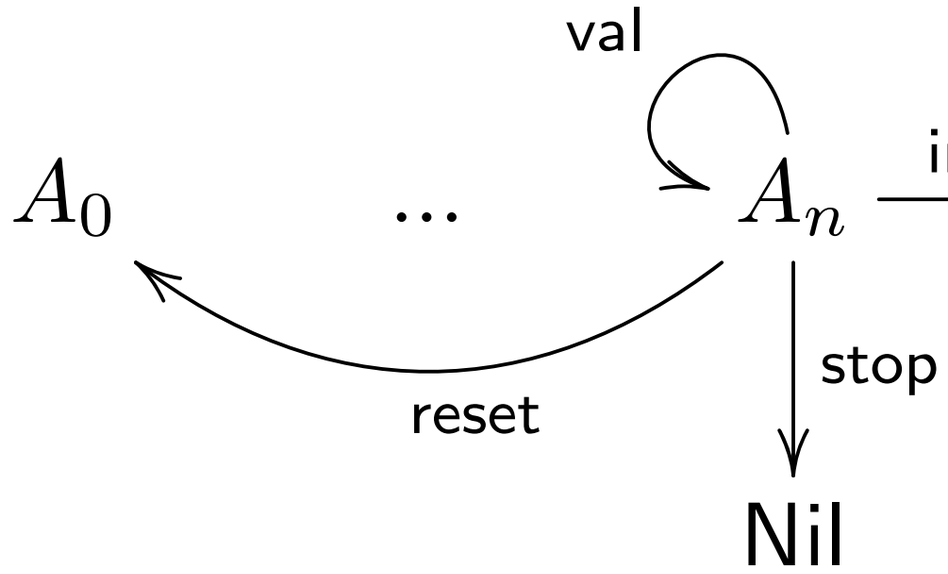
$$p \xrightarrow{\mu} q$$

un processo
nel suo stato attuale

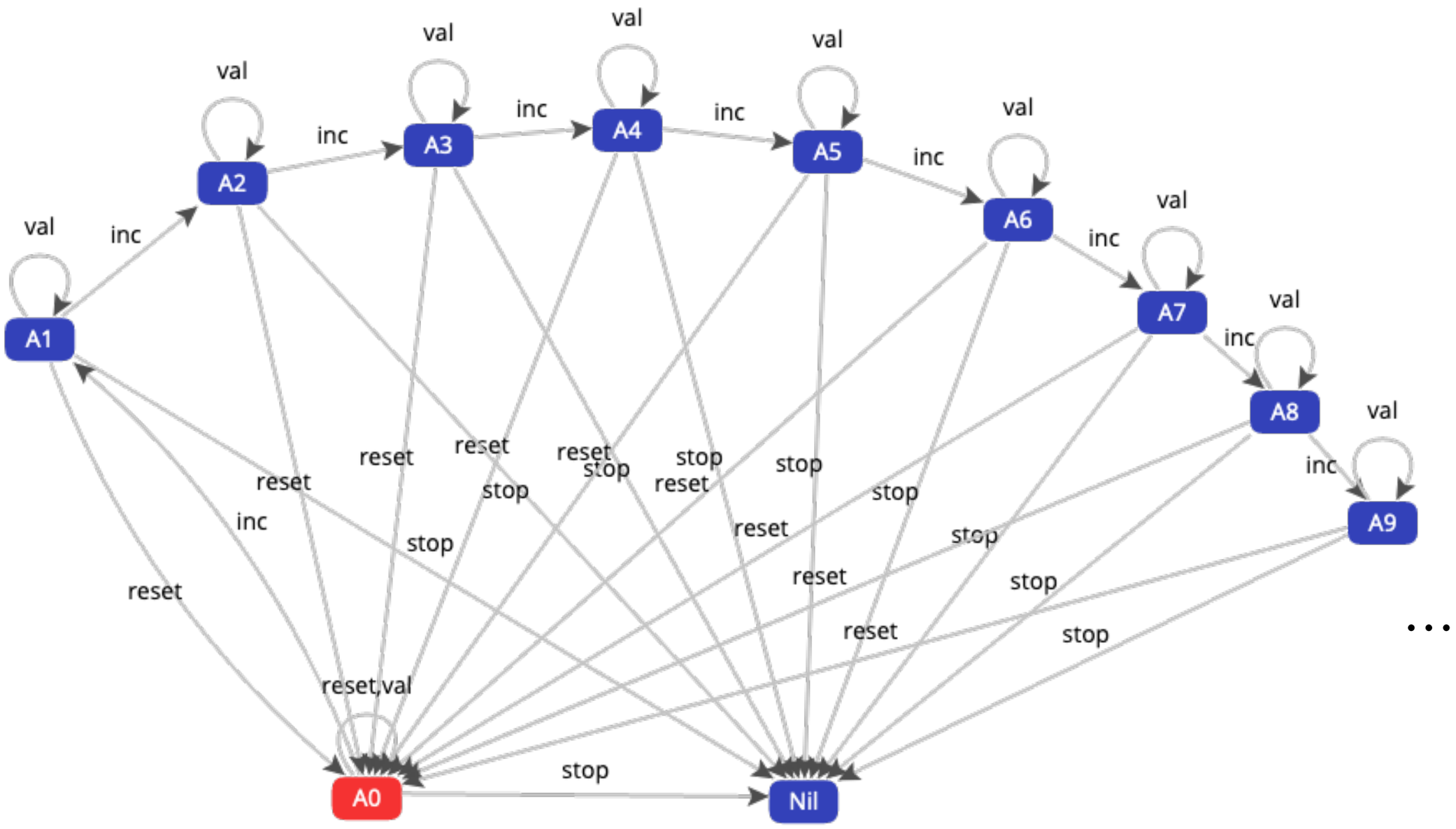
il processo
dopo l'interazione

numero di stati/transizioni
può essere infinito

Esempio: contatore



LTS: Labelled Transition System



CCS: stati e etichette

Cos'è un processo p?

un agente sequenziale
un sistema in cui molti agenti sequenziali interagiscono

Cos'è un'etichetta μ ?

un'azione (ad esempio un'uscita) $\alpha!v$ — inviato v sul canale α

un'azione doppia (ad esempio un ingresso) $\alpha?v$ — inviato v sul canale α

un'azione interna (azione silenziosa) (nessuna interazione con l'ambiente) τ — comunicazione conclusa

CCS: azioni & coazioni

Possiamo essere ancora più astratti di così
senza perdere espressività computazionale

non teniamo conto dei valori comunicati
(immaginate che ci sia un canale dedicato per ogni valore)

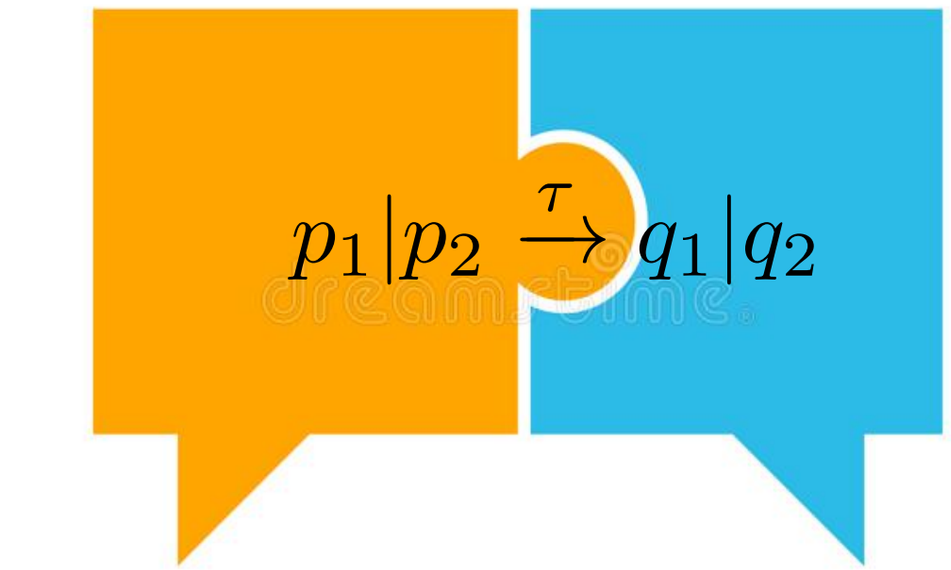
$\alpha!v$ diventa solo $\overline{\alpha_v}$ o solo $\overline{\alpha}$

$\alpha?v$ diventa solo α_v o solo α

λ denota o α o $\overline{\alpha}$

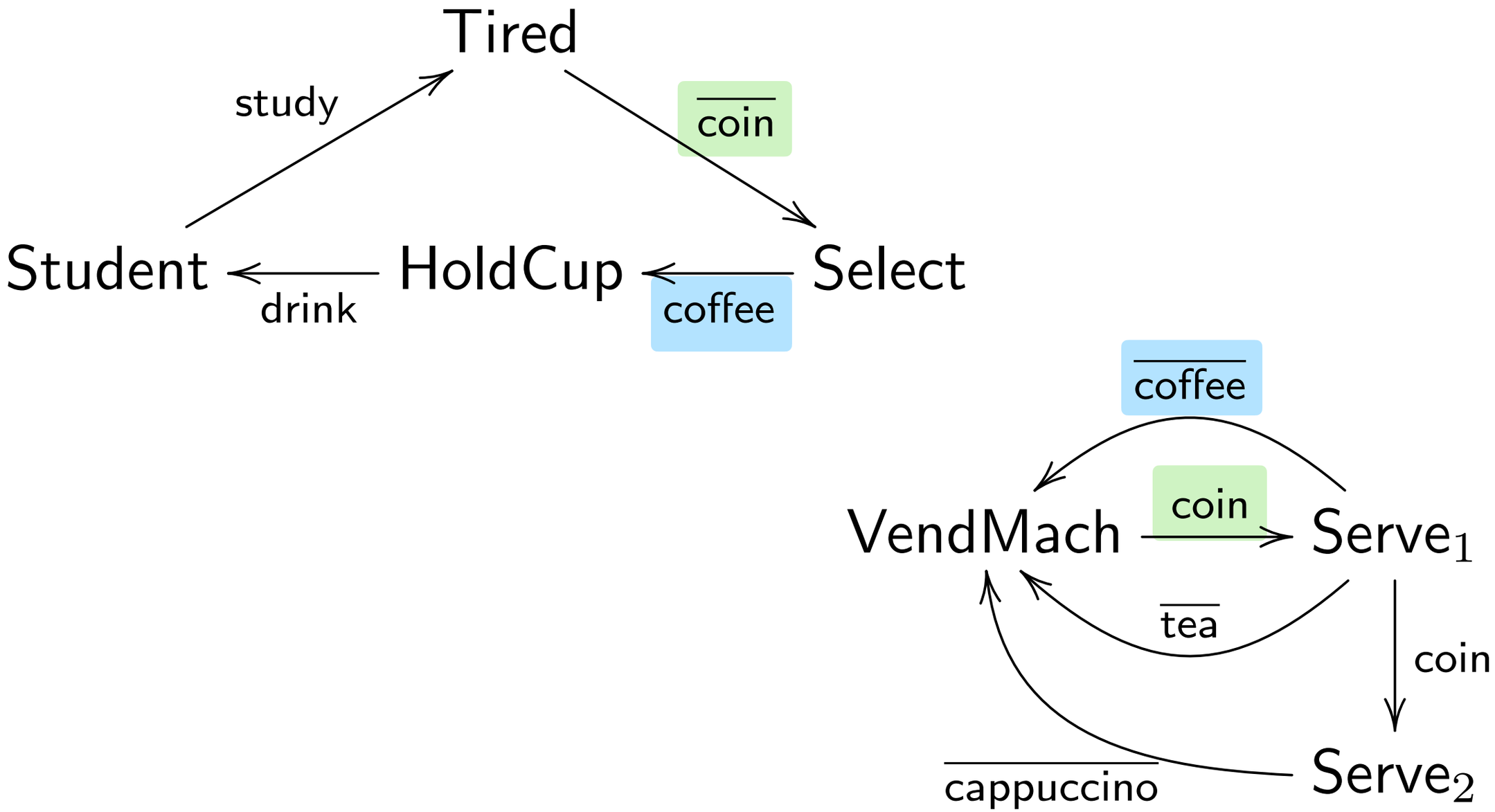
$\overline{\lambda}$ denota il suo duale (assumiamo $\overline{\overline{\alpha}} = \alpha$)

CCS: comunicazione



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

Esempio: distributore automatico



Sintassi CCS

CCS: sintassi

p, q	$::=$	$\mathbf{nil}$	processo inattivo
	$\mid$	x	variabile di processo (per la ricorsione)
	$\mid$	$\mu.p$	prefisso azione
	$\mid$	$p\backslash\alpha$	canale ristretto
	$\mid$	$p[\phi]$	rietichettatura del canale
	$\mid$	$p + q$	scelta nondeterministica (somma)
	$\mid$	$p \mid q$	compozione parallela
	$\mid$	$\mathbf{rec}\ x.\ p$	ricorsione

(gli operatori sono elencati in ordine di precedenza)

CCS: sintassi

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

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

che si legge

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

(gli operatori sono elencati in ordine di precedenza)

CCS: sintassi

l'unico binder è l'operatore di ricorsione

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

la nozione di variabile libera (di processo) è definita come al solito

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

un processo è detto chiuso se non ha variabili libere

la nozione di capture avoiding substitution è definita come al solito

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

i processi sono considerati alfa-rinominati rispetto alle variabili vincolate

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

semantica operazionale del CCS

CCS: etichette

$\mathcal{C}$ insieme di azioni (di ingresso), denotate da α

$\overline{\mathcal{C}}$ insieme di azioni (di uscita), denotate da $\overline{\alpha}$ $\quad \mathcal{C} \cap \overline{\mathcal{C}} = \emptyset$

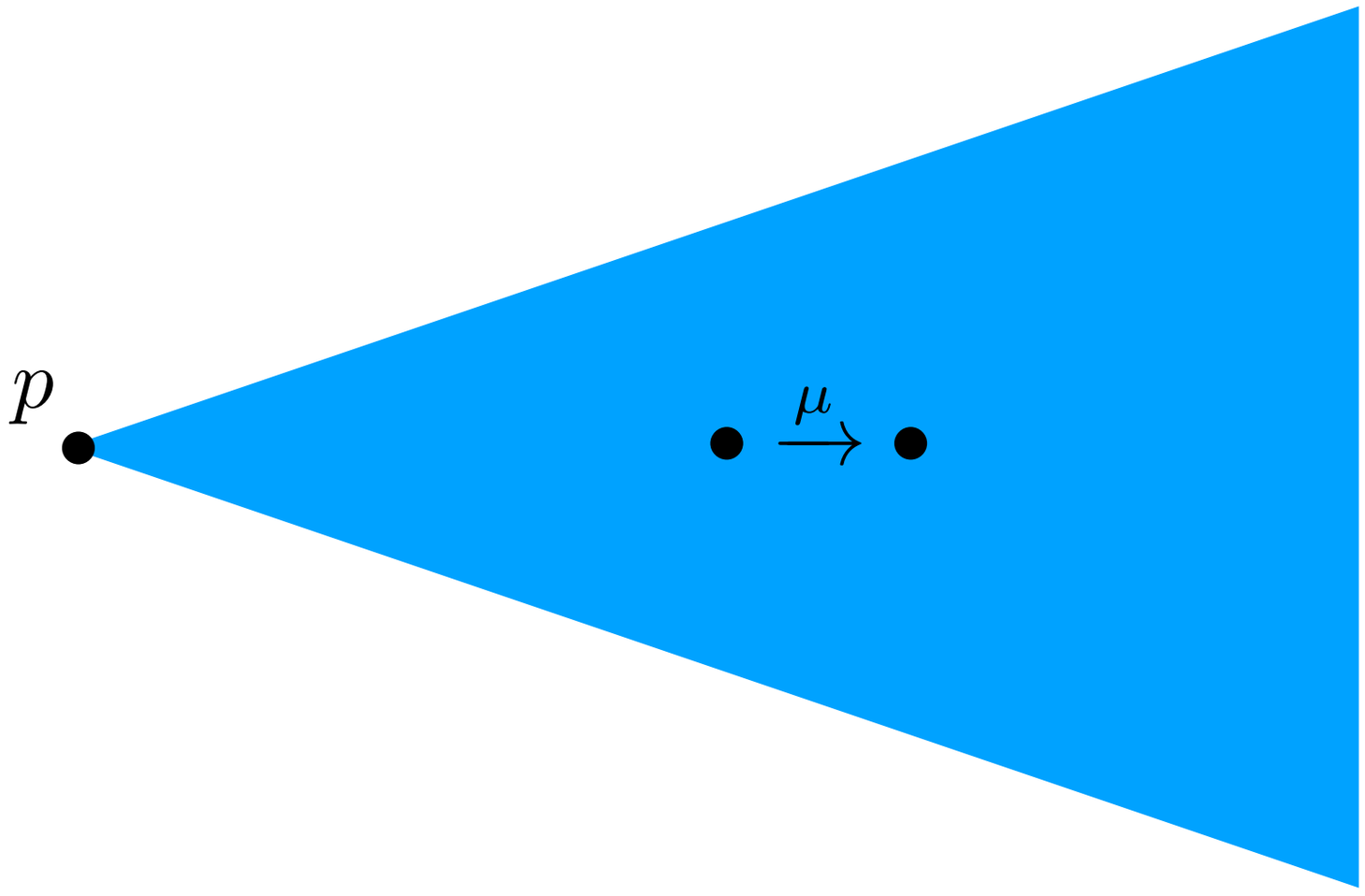
$\Lambda = \mathcal{C} \cup \overline{\mathcal{C}}$ insieme di azioni osservabili, denotate da $\lambda\ \overline{\lambda}$

$\tau \notin \Lambda$ una separata azione silenziosa

$\mathcal{L} = \Lambda \cup \{\tau\}$ insiemi di azioni, denotate da μ

LTS di un processo

l'LTS di CCS è infinito (uno stato per ogni processo)



a partire da p considera tutti gli stati raggiungibili:

l'LTS di un processo può essere finito/infinito

Processo Nil

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

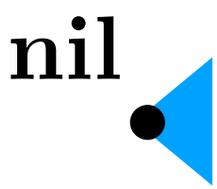
il processo inattivo non fa nulla

nessuna interazione è possibile con l'ambiente

rappresenta un agente terminato

nessuna regola di semantica operativa associata a nil

LTS di un processo



prefisso azione

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

un processo con prefisso un’azione può eseguire l'azione e continuare come previsto

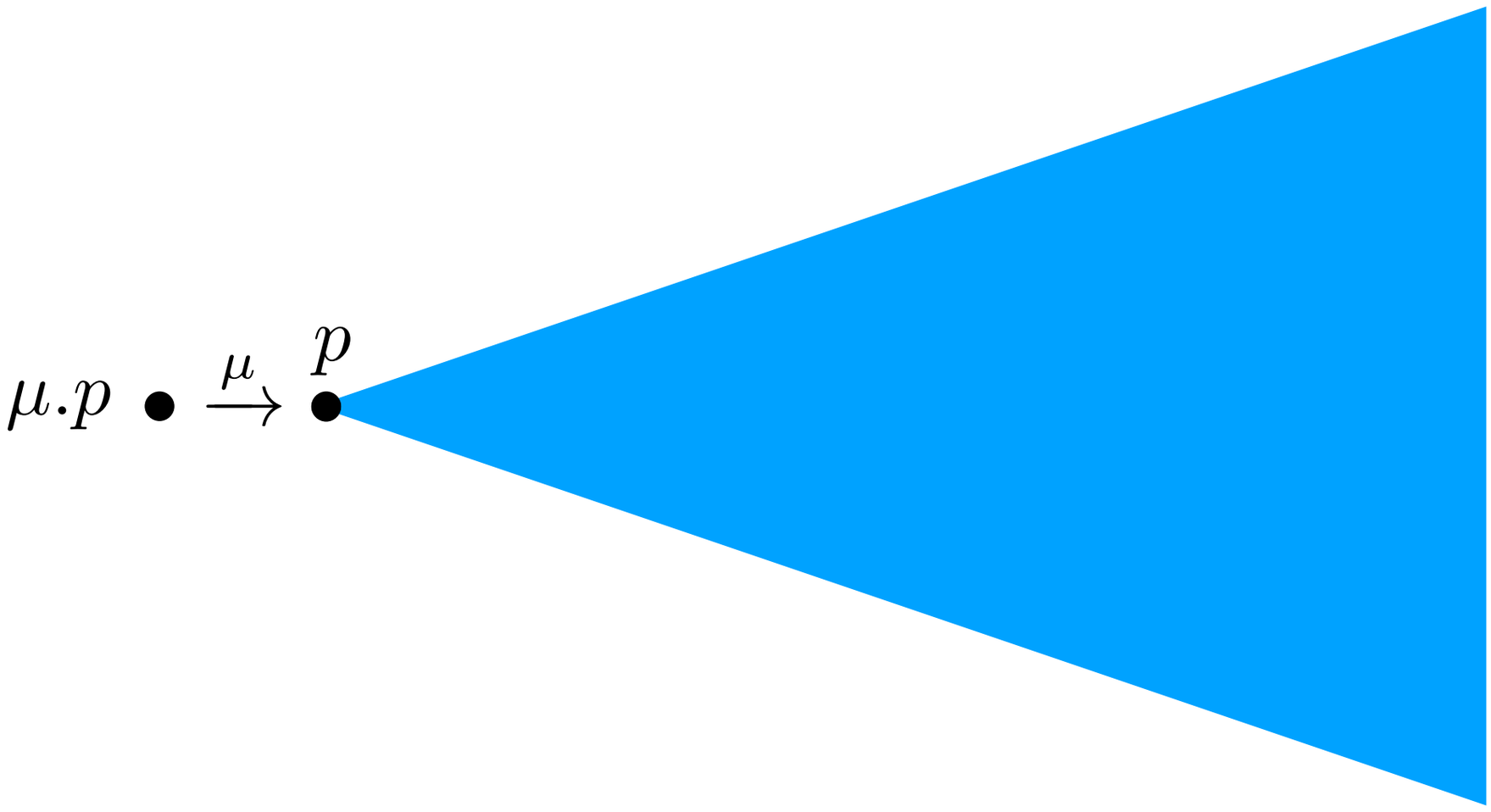
l'azione può comportare un'interazione con l'ambiente

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

aspetta una moneta, poi dà un caffè e poi si ferma

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

LTS del processo



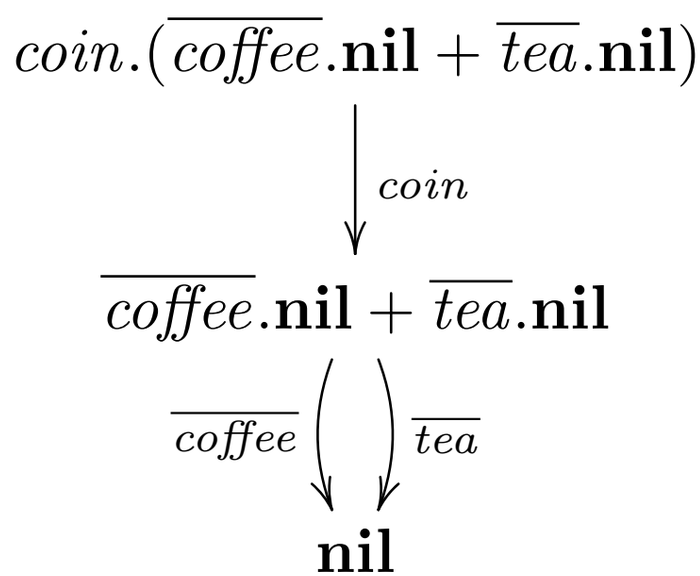
Scelta non deterministica

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

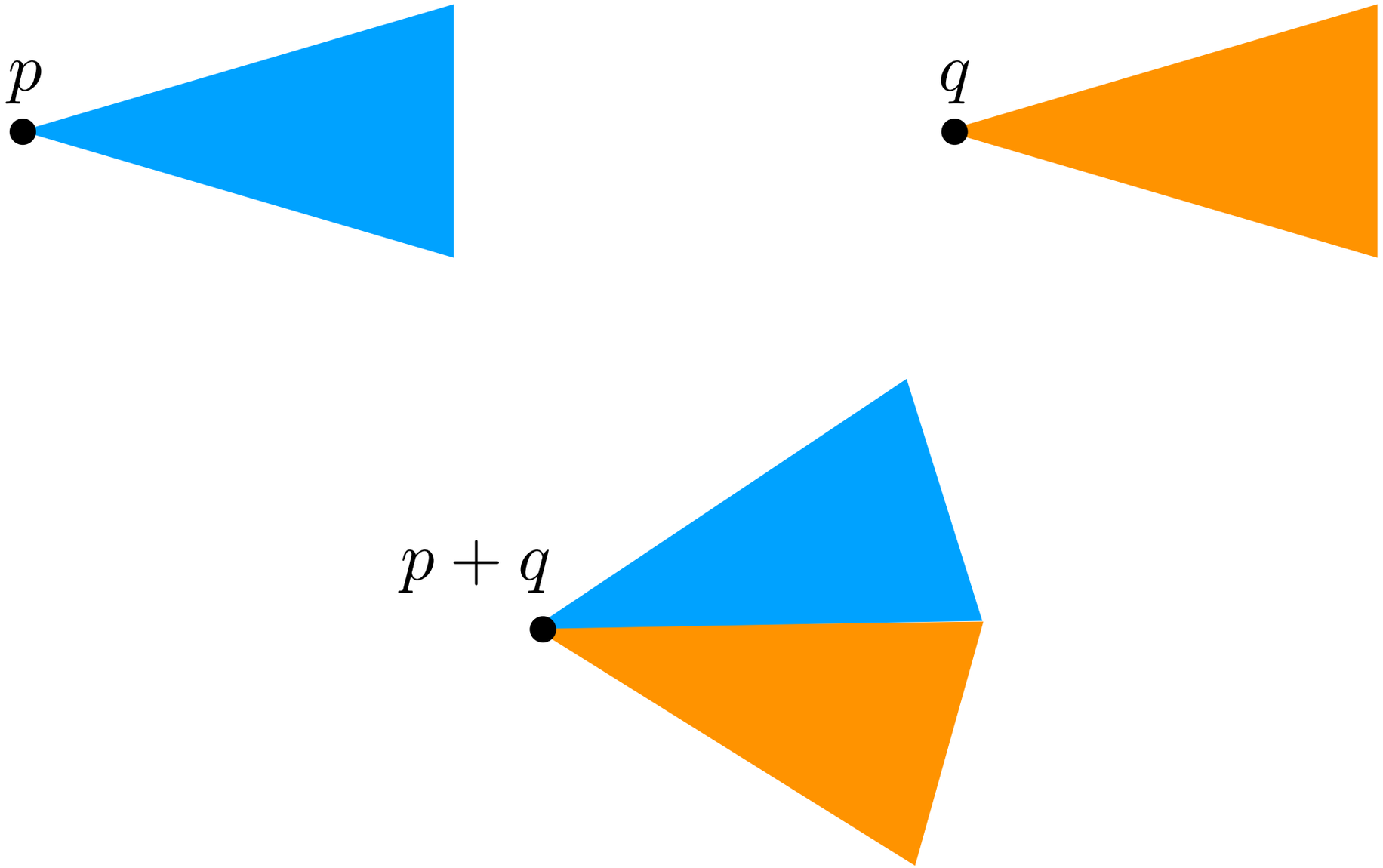
Il processo $p_1 + p_2$ può comportarsi come p_1 o come p_2

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

aspetta una moneta, poi dà un caffè o un tè, poi si ferma



LTS del processo



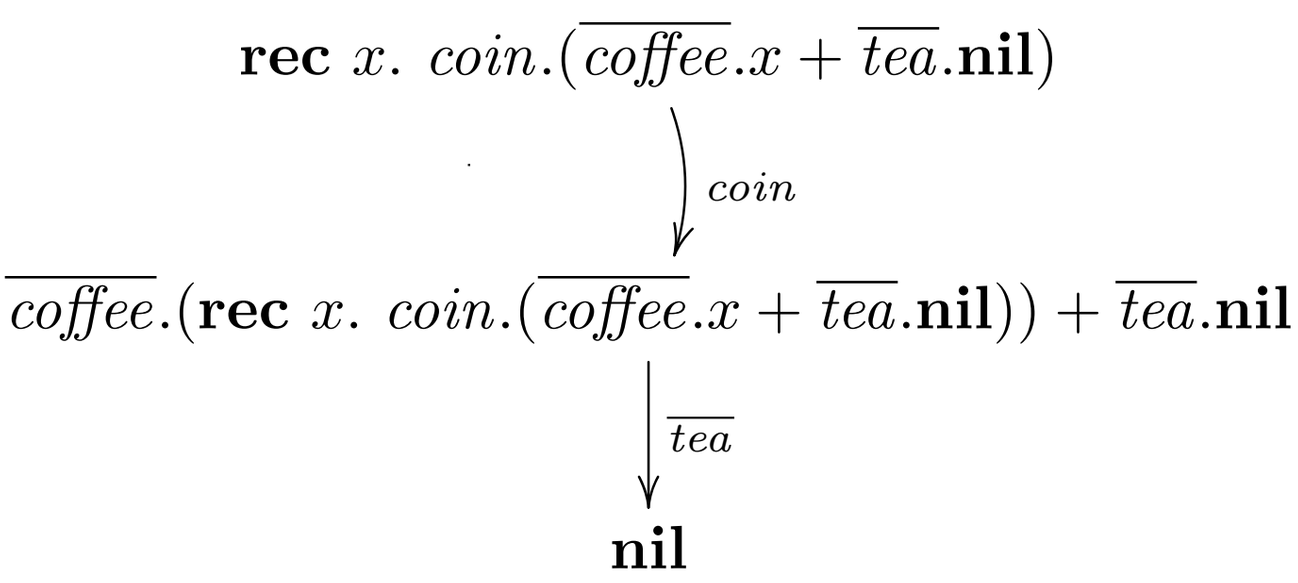
Ricorsione

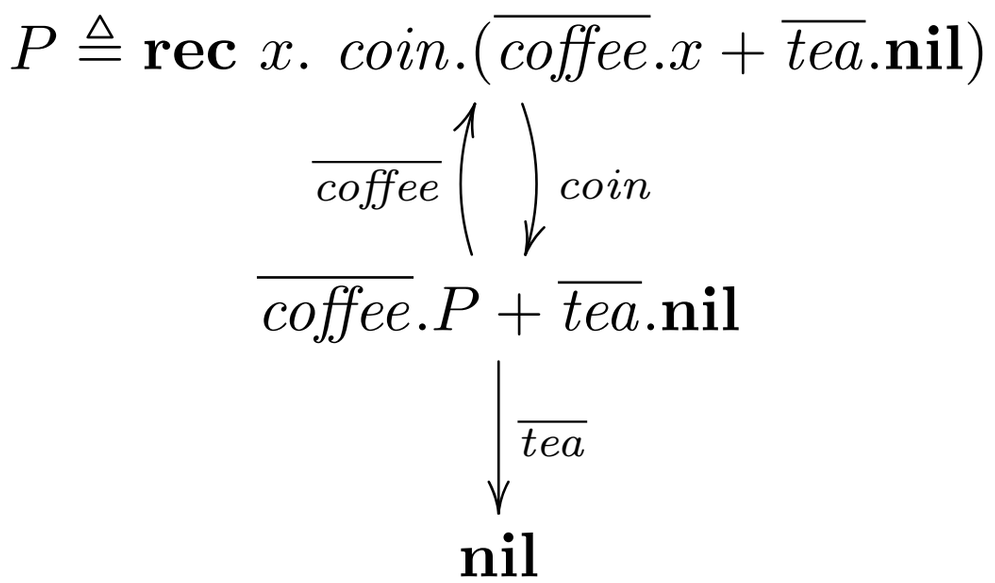
$$\text{Rec)}\ \frac{p\left[{}^{\mathbf{rec}\ x.\ p}/_{x}\right] \xrightarrow{\mu} q}{\mathbf{rec}\ x.\ p \xrightarrow{\mu} q}$$

come una definizione ricorsiva let $x = p$ in x

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

aspetta una moneta, poi dà un caffè ed è di nuovo pronto o un tè e si ferma





Ricorsione tramite costanti di processo

immaginate che alcune costanti A di processo siano disponibili insieme ad Δ un insieme di dichiarazioni della forma

$$A \triangleq p$$

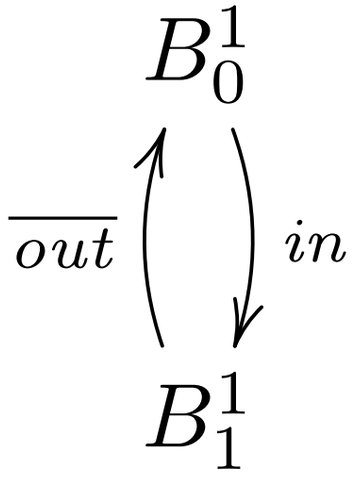
per ogni costante

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

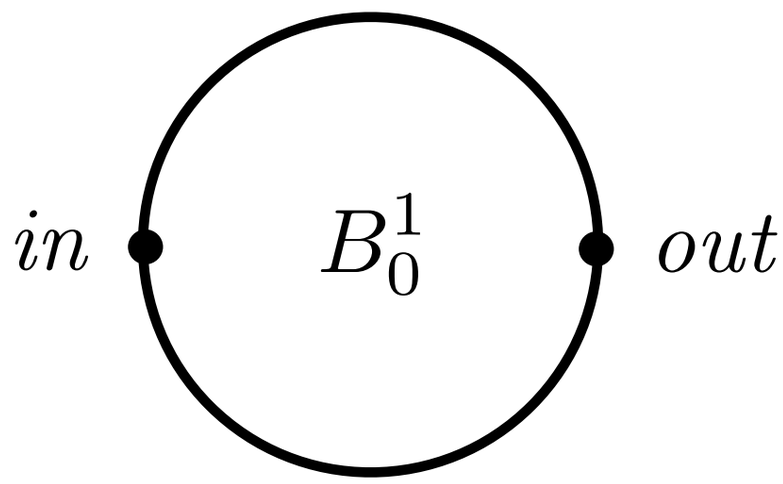
$$P \triangleq coin.(\overline{coffee}.P + \overline{tea}.\mathbf{nil})$$

CCS: buffer di capacita' 1

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



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

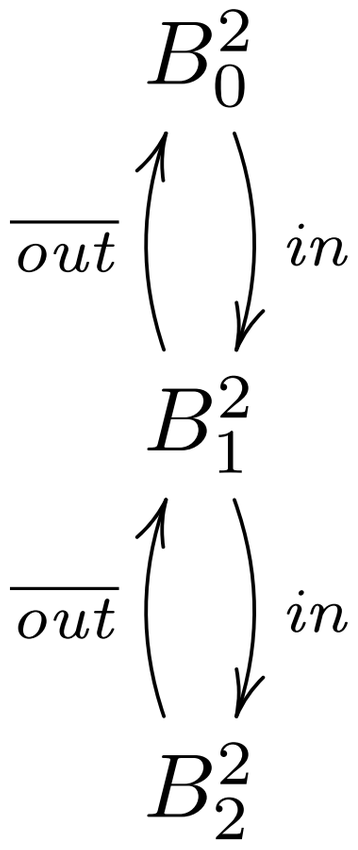


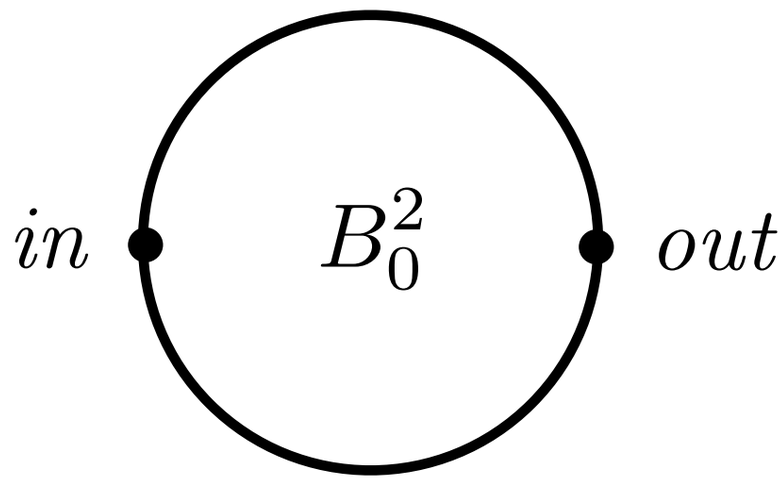
CCS: buffer di capacita' 2

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

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

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



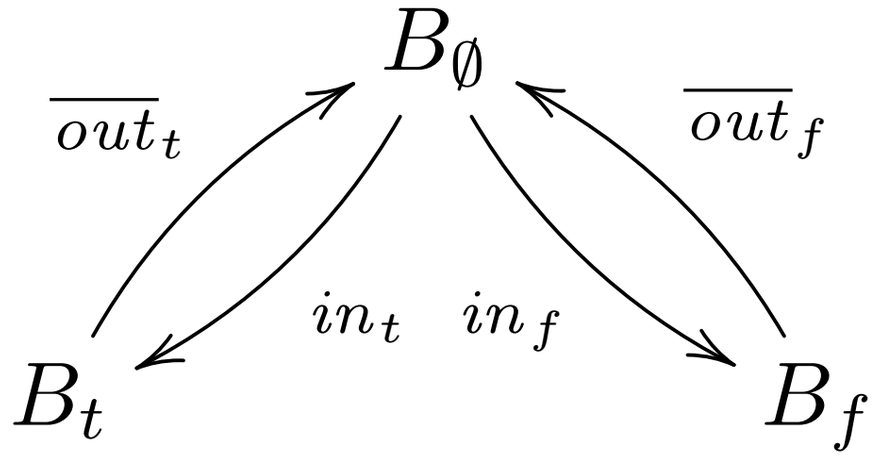


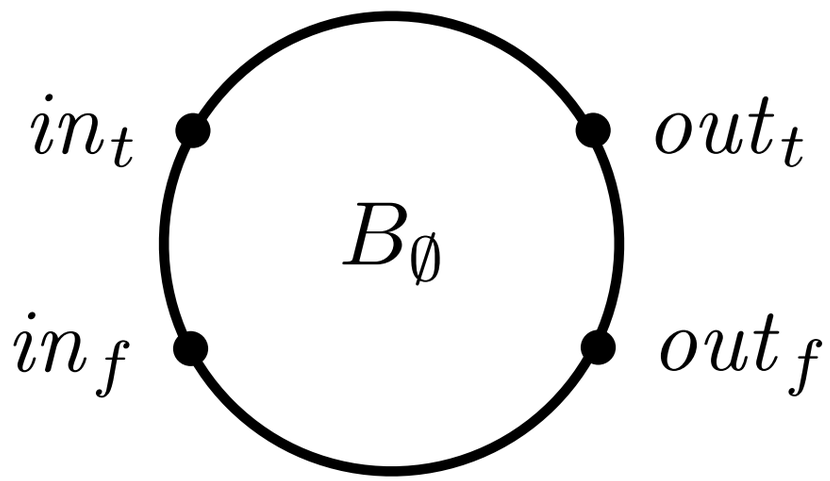
CCS: buffer booleano

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

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

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





Composizione parallela

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) $\dfrac{p_2 \xrightarrow{\mu} q_2}{p_1|p_2 \xrightarrow{\mu} p_1|q_2}$

i processi che girano in parallelo possono intrecciare le loro azioni o sincronizzarsi quando vengono eseguite due azioni

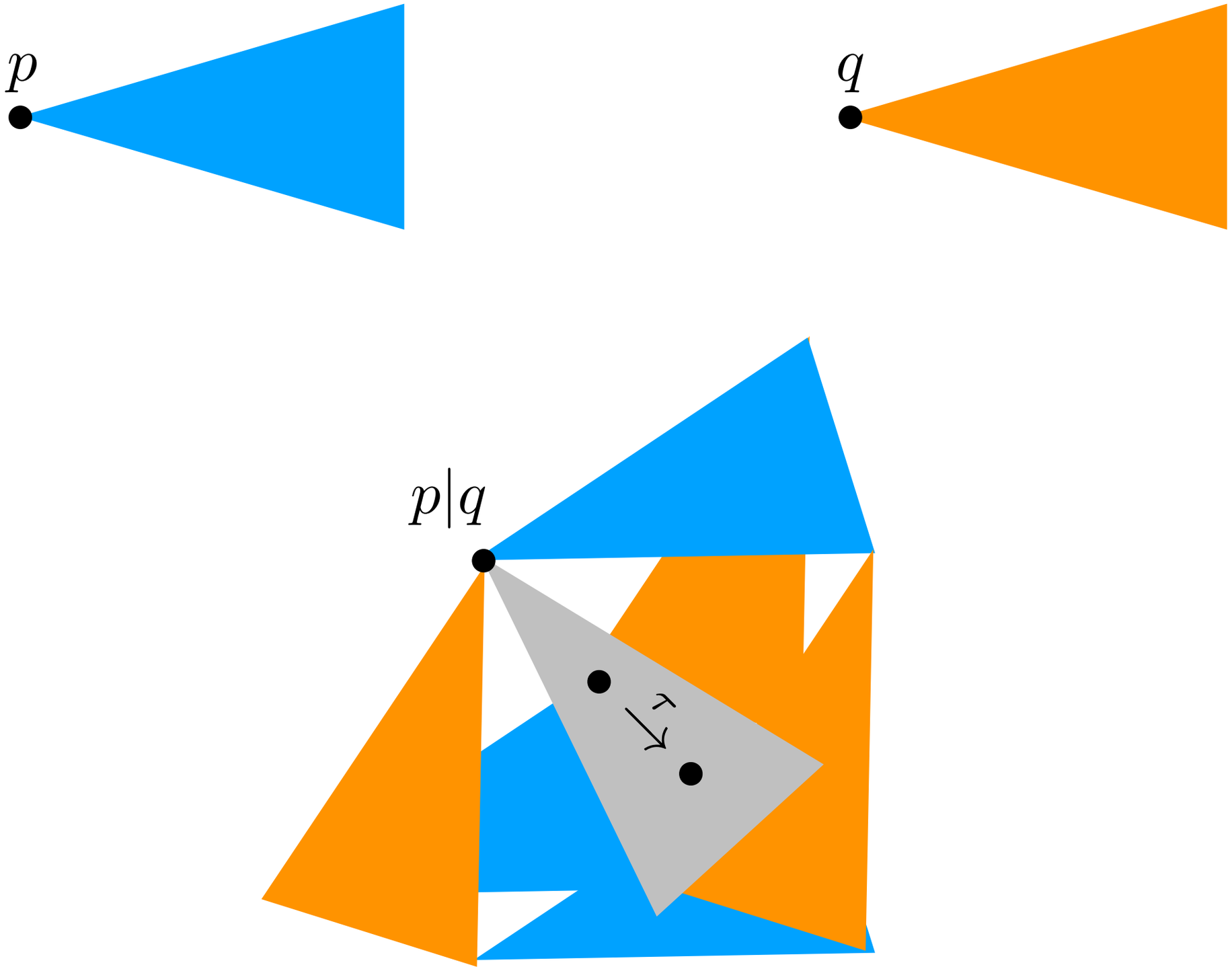
$P \triangleq \overline{coin}.coffee.\mathbf{nil}$ $M \triangleq coin.(\overline{coffee}.\mathbf{nil} + \overline{tea}.\mathbf{nil})$

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

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

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

LTS del processo



CCS: buffer paralleli

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

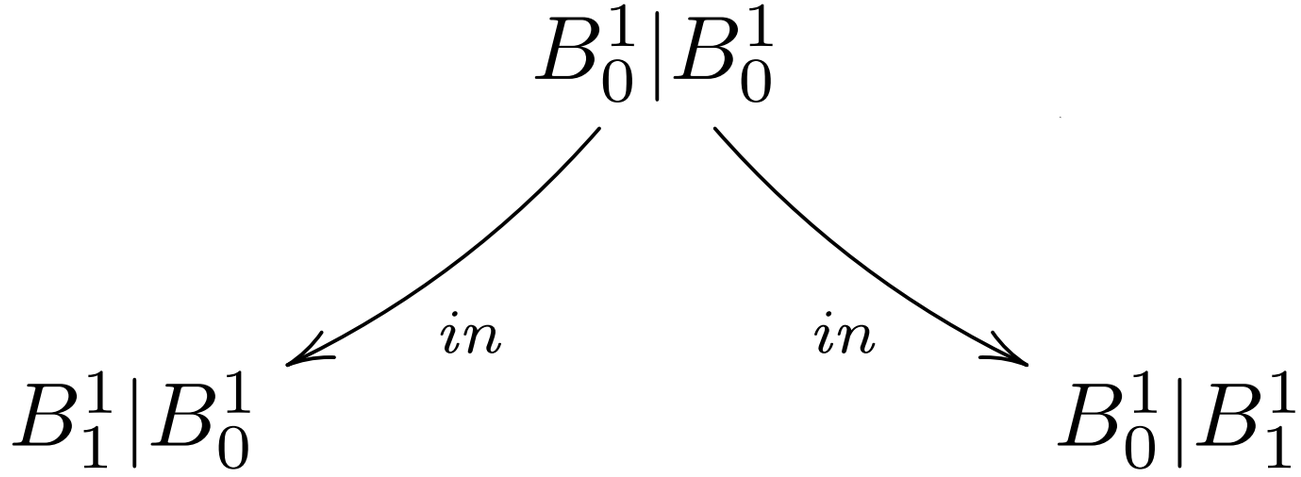
$$B_1^1 \triangleq \overline{out}.B_0^1$$

$$B_0^1 | B_0^1$$

CCS: buffer paralleli

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

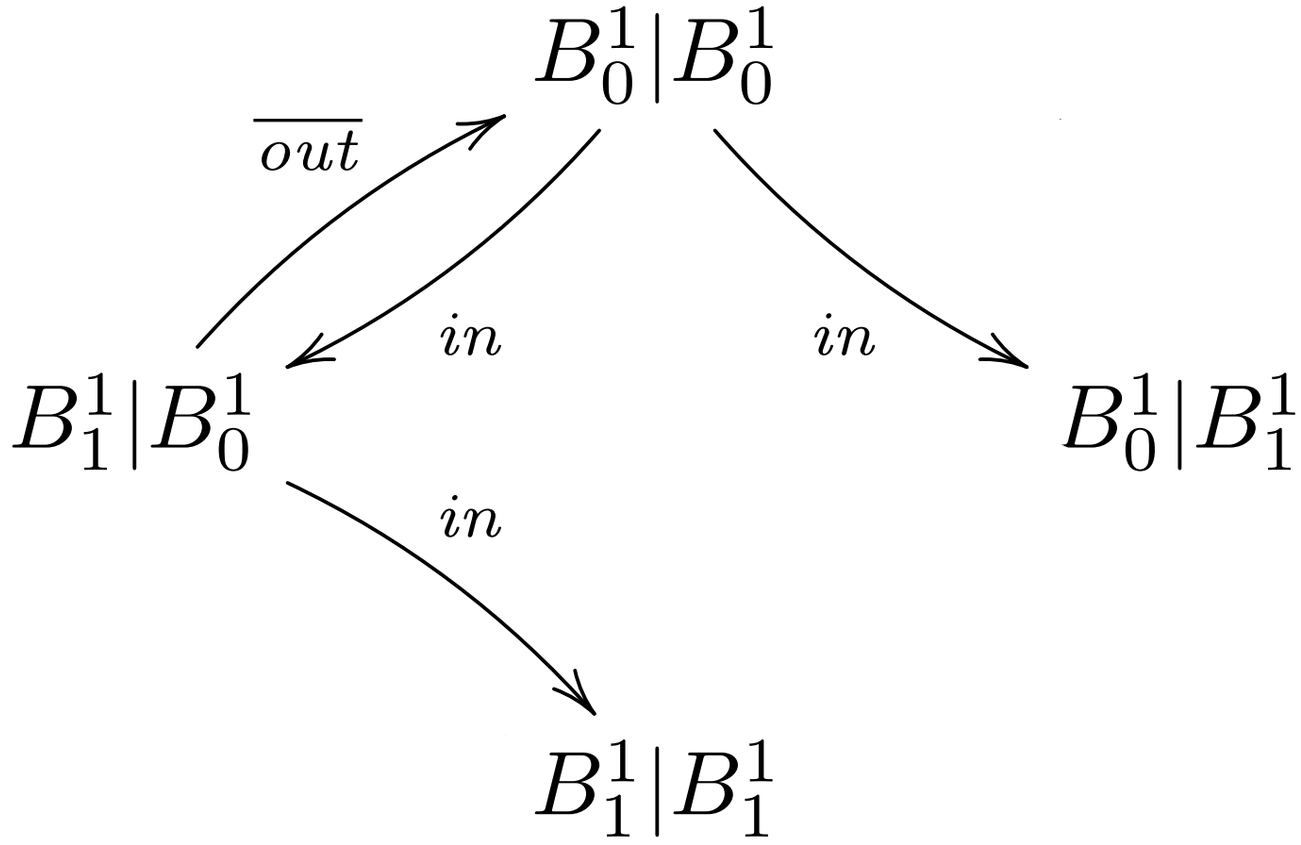
$$B_1^1 \triangleq \overline{out}.B_0^1$$



CCS: buffer paralleli

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

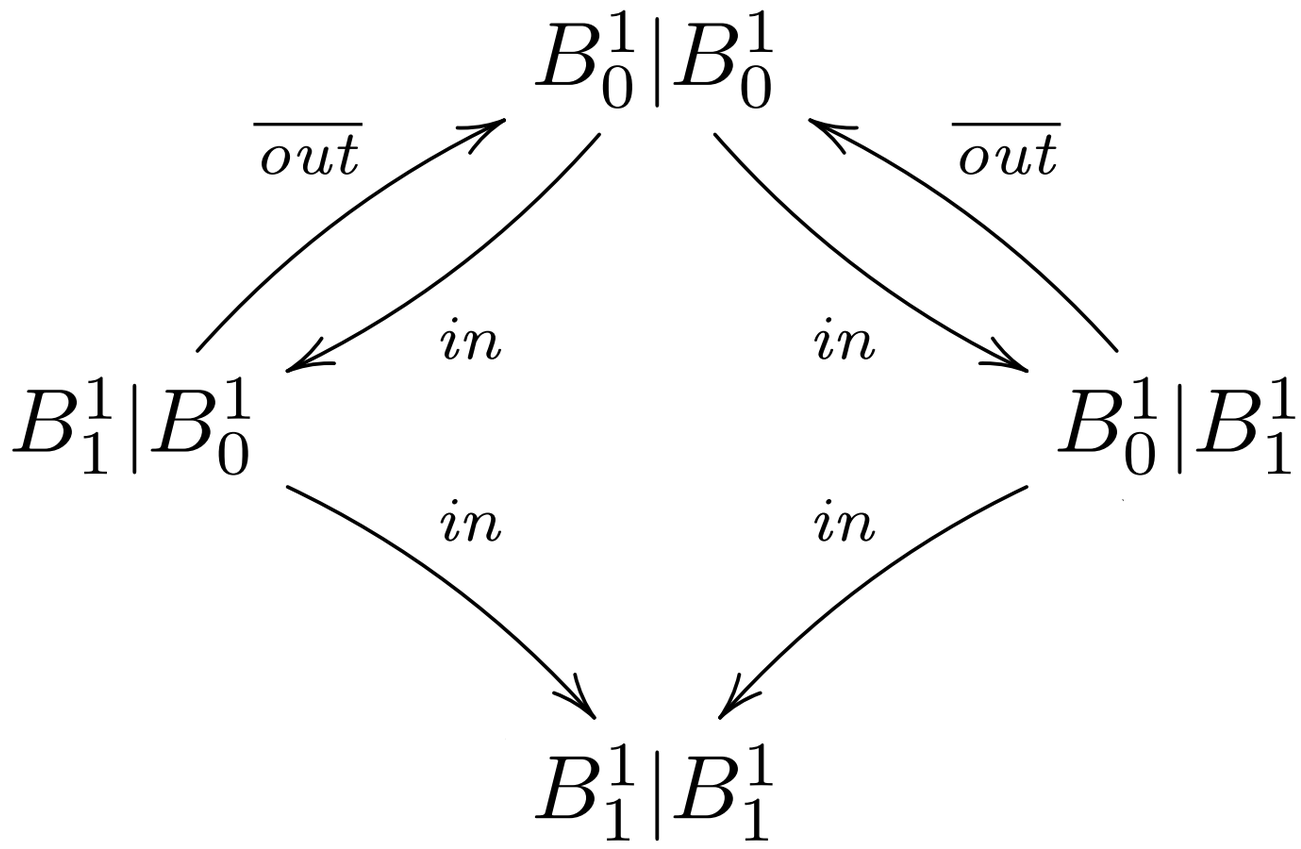
$$B_1^1 \triangleq \overline{out}.B_0^1$$



CCS: buffer paralleli

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

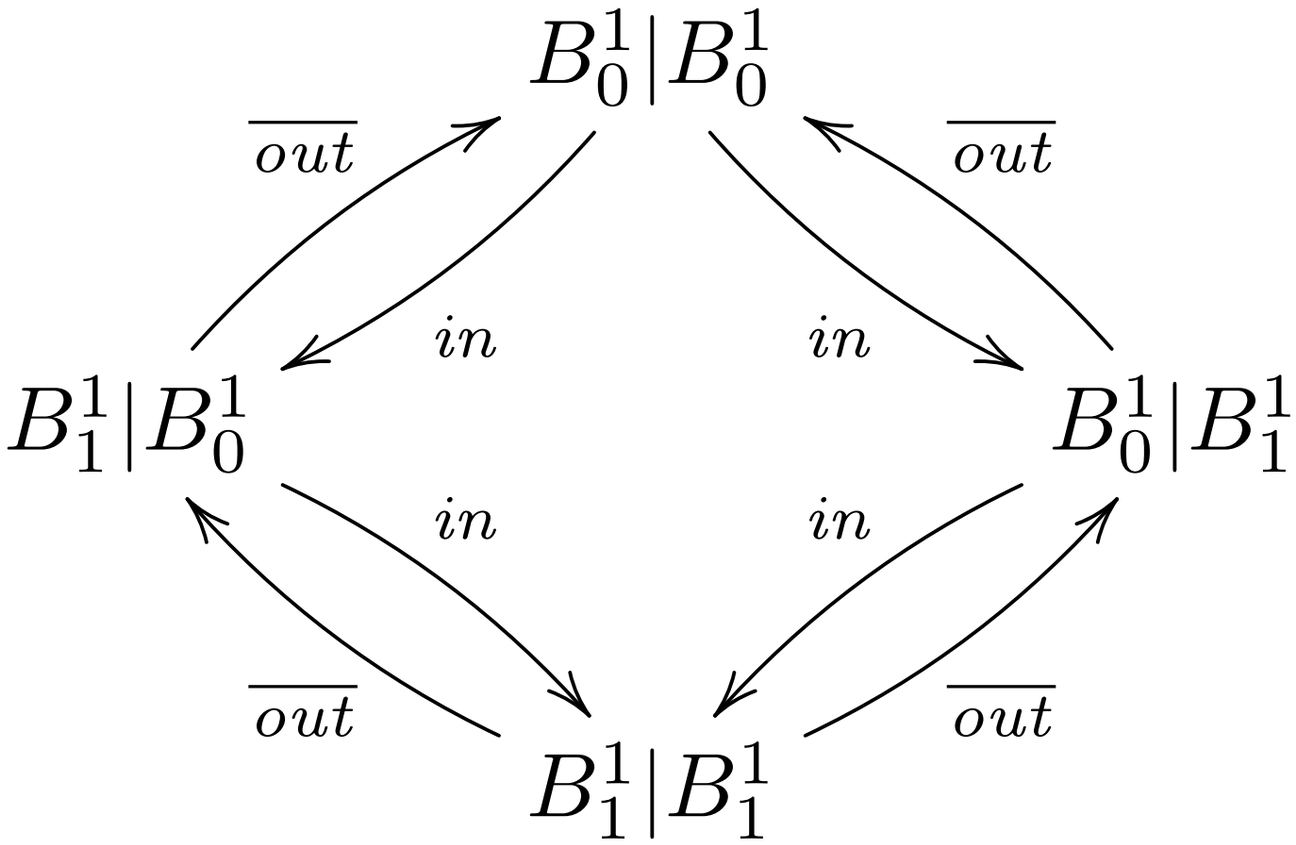
$$B_1^1 \triangleq \overline{out}.B_0^1$$

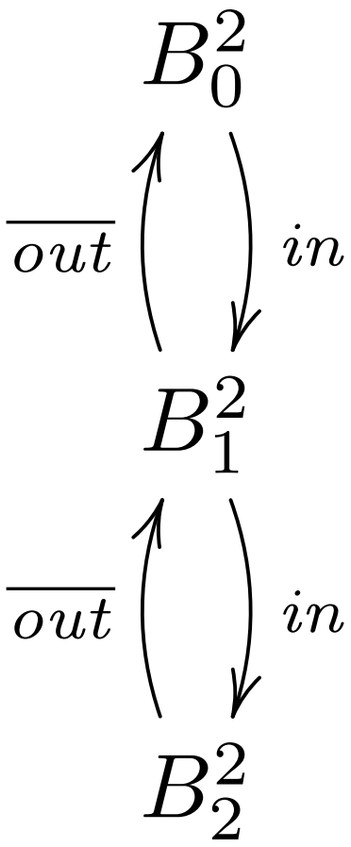


CCS: buffer paralleli

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

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





confrontare con il buffer a capacità 2

Restrizione

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

rende il canale α privato a p

nessuna interazione sul canale α con l'ambiente

se p è la composizione parallela dei processi, allora possono sincronizzarsi su α

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

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

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

Restrizione: shorthand

dato $S = \{\alpha_1, ..., \alpha_n\}$ scriviamo $p\backslash S$

invece di $p\backslash\alpha_1 \ldots \backslash\alpha_n$

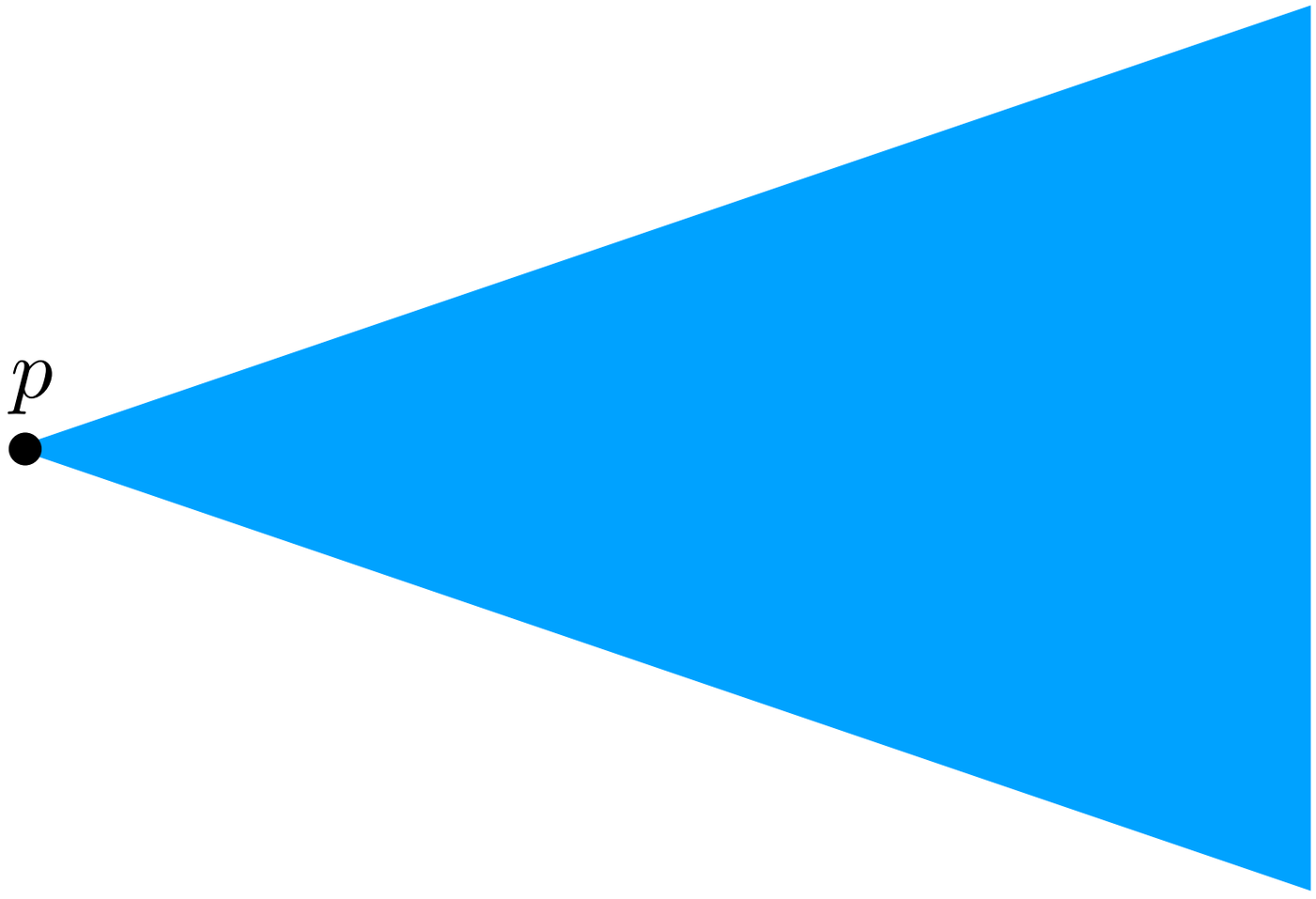
omettiamo $\mathbf{nil}$

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

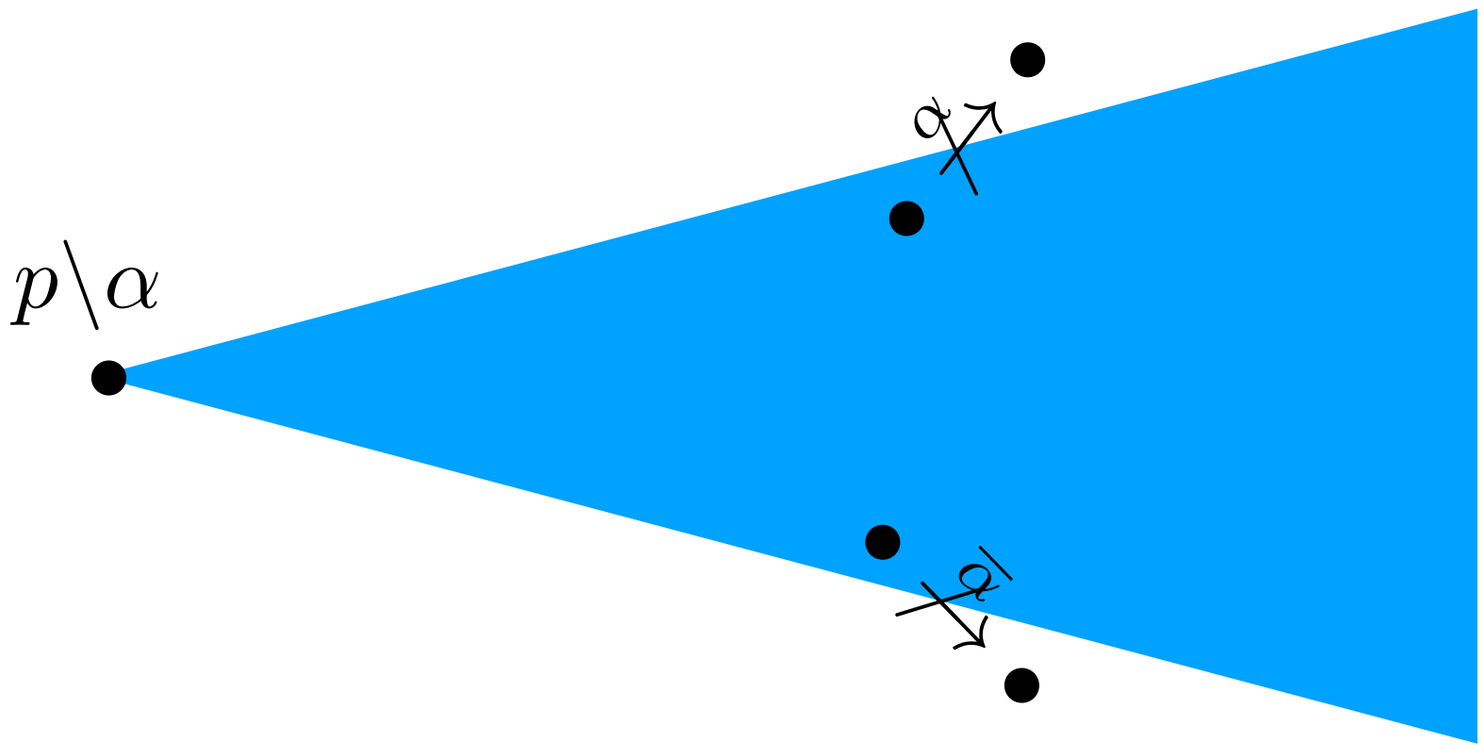
$$S \triangleq \{coin, coffee, tea\}$$

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

LTS del processo



LTS del processo



ridenominazione

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

rinomina i canali di un'azione secondo ϕ

assumiamo $\phi(\tau) = \tau$ $\phi(\overline{\lambda}) = \overline{\phi(\lambda)}$

ci permette di riusare i processi

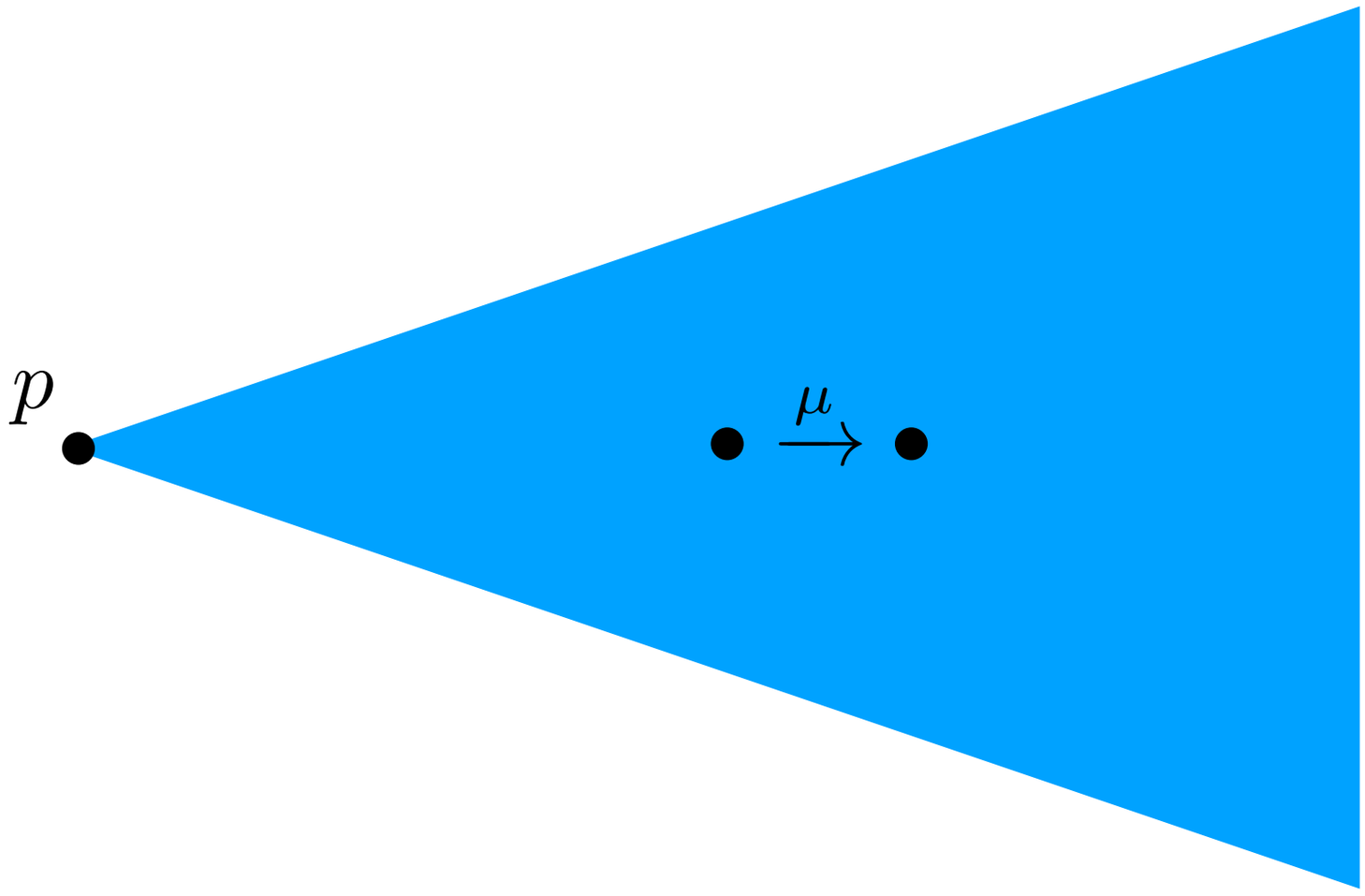
$P \triangleq \overline{coin}.coffee$

$\phi(coin) = moneta$

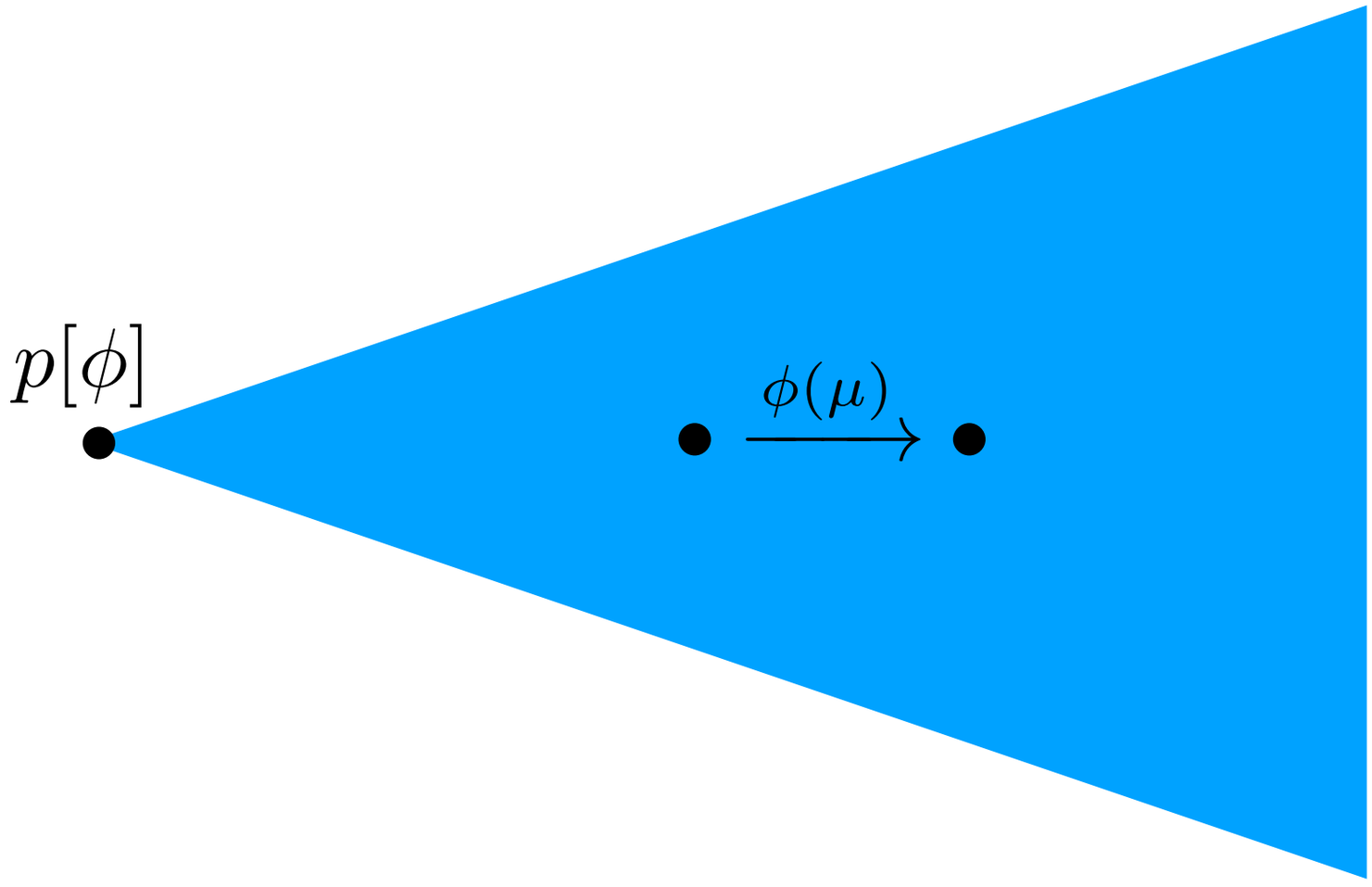
$\phi(coffee) = caffè$

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

LTS del processo



LTS del processo



CCS semantica operazionale

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

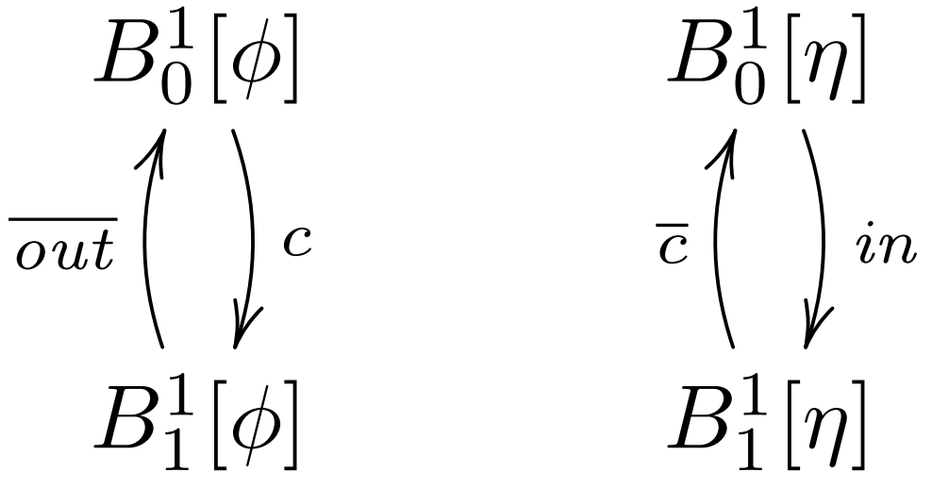
ParR) $$\frac{p_2 \xrightarrow{\mu} q_2}{p_1 | p_2 \xrightarrow{\mu} p_1 | q_2}$$

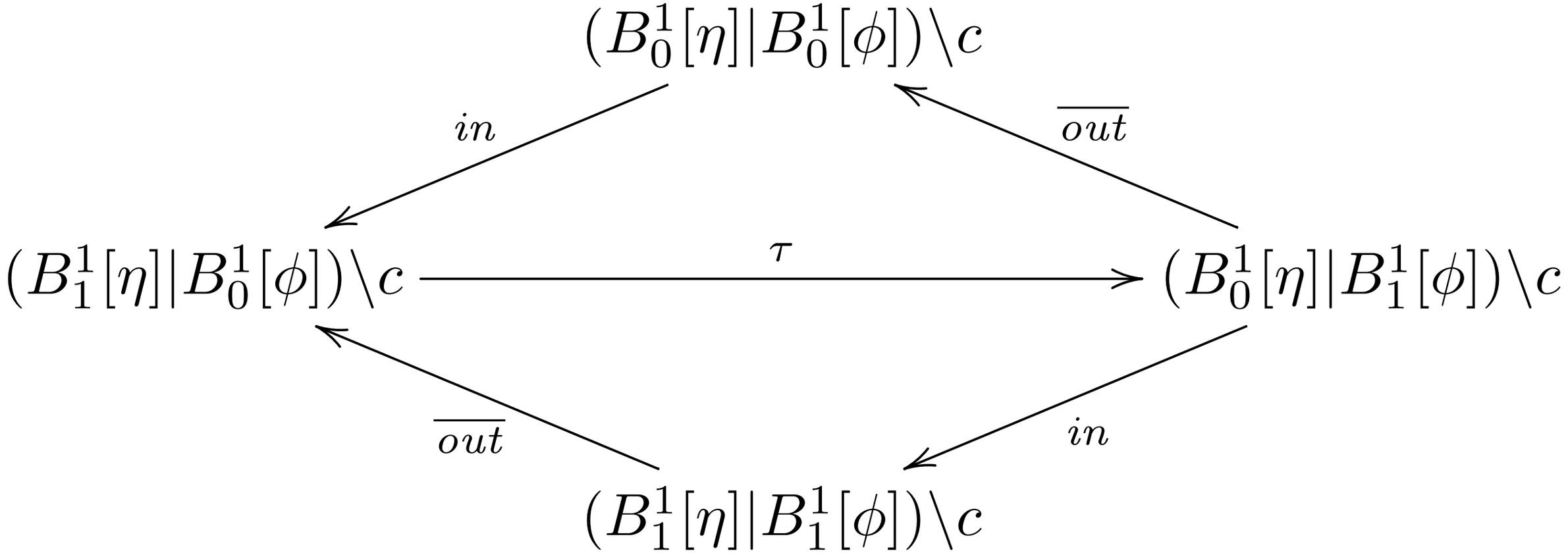
Rec) $$\frac{p\left[{}^{\mathbf{rec}\ x.\ p}/_{x}\right] \xrightarrow{\mu} q}{\mathbf{rec}\ x.\ p \xrightarrow{\mu} q}$$

Buffer collegati

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

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

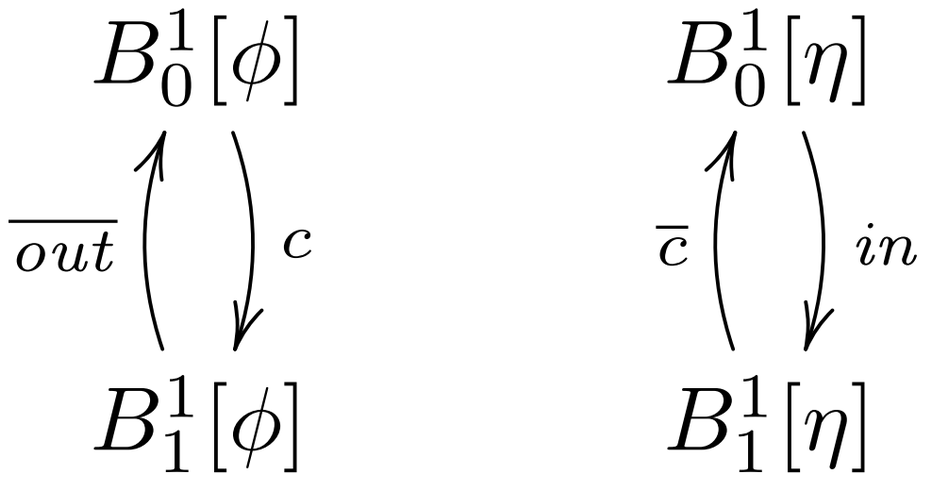


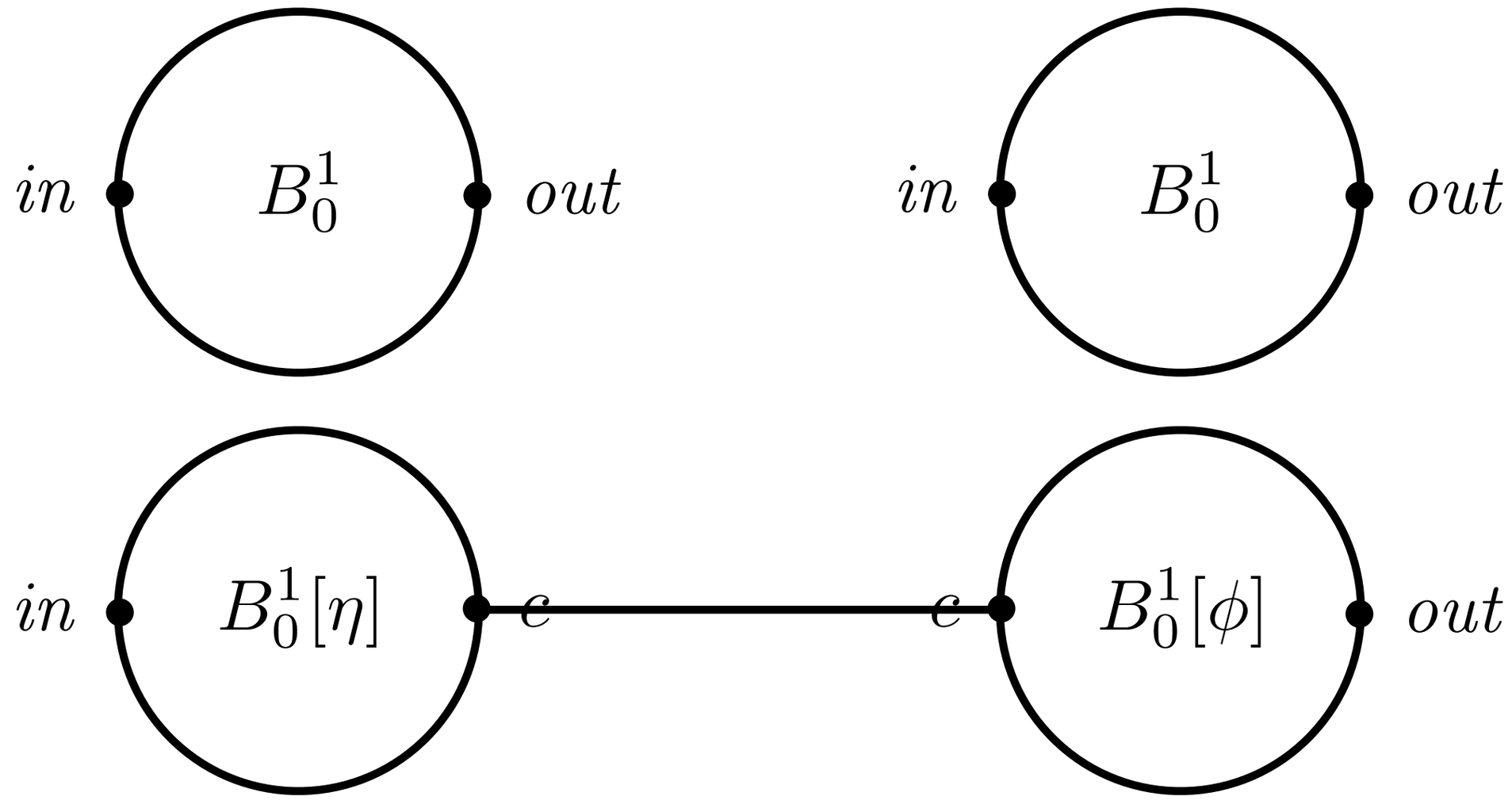


Buffer collegati

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

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



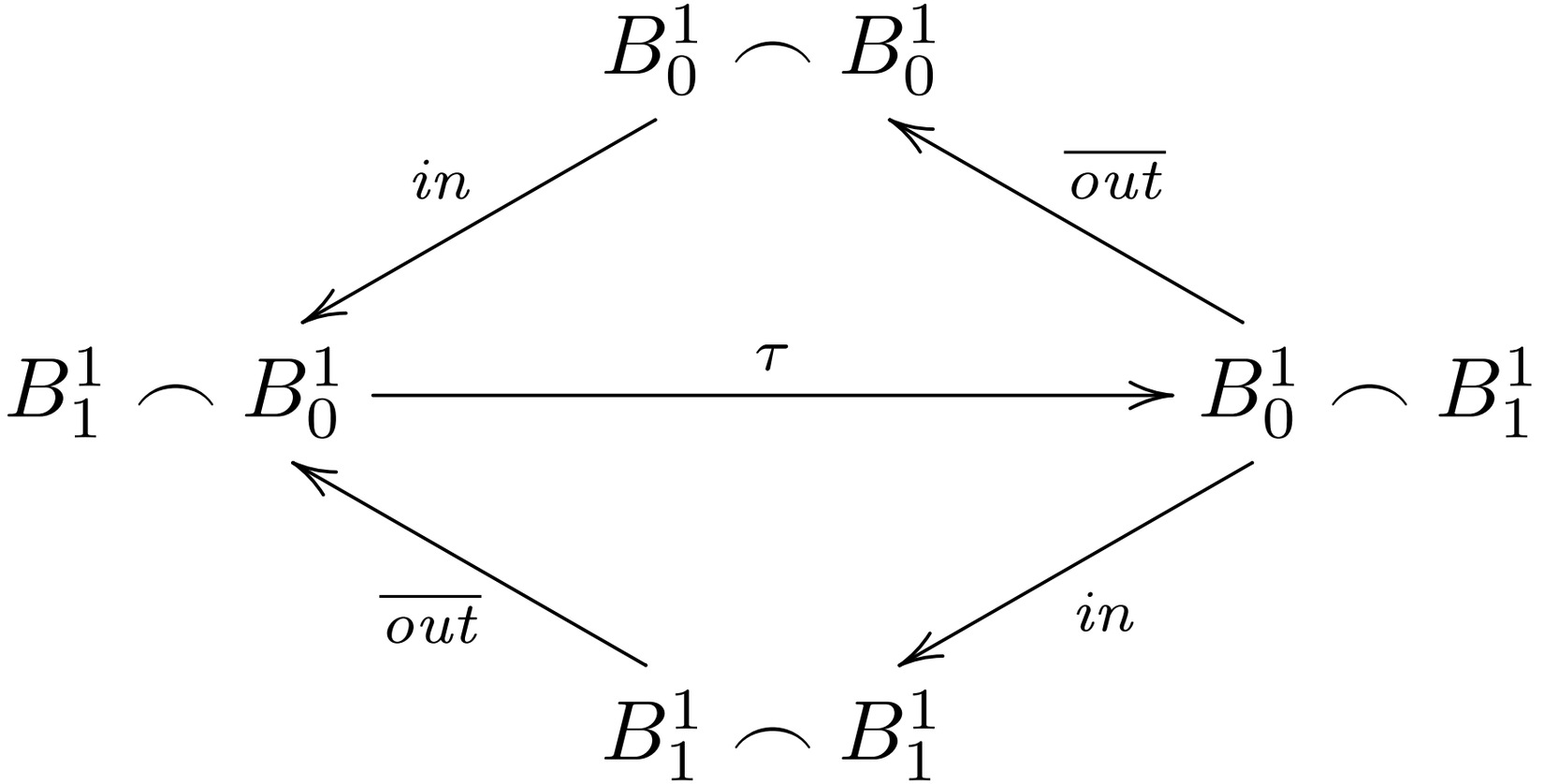


Buffer collegati

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

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

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



Buffer collegati booleani

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

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

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

$\eta(out_t) = c_t$ $\eta(out_f) = c_f$

$\phi(in_t) = c_t$ $\phi(in_f) = c_f$

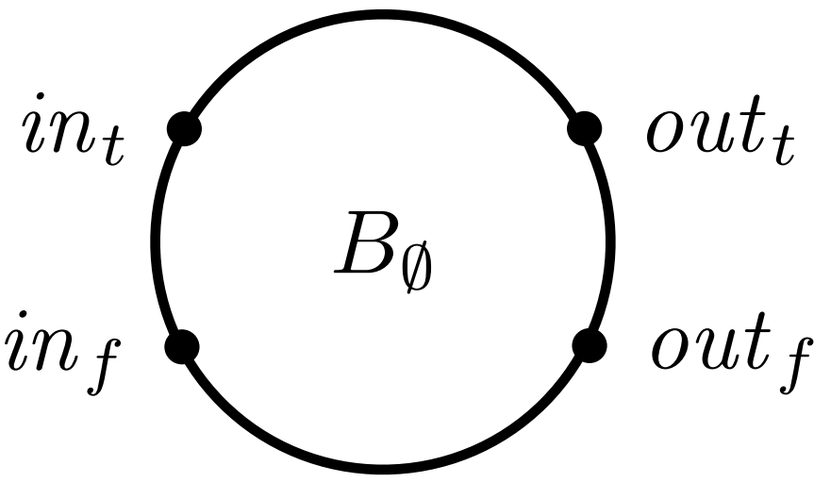
$p \frown q \triangleq (p[\eta]|q[\phi])\backslash\{c_t, c_f\}$

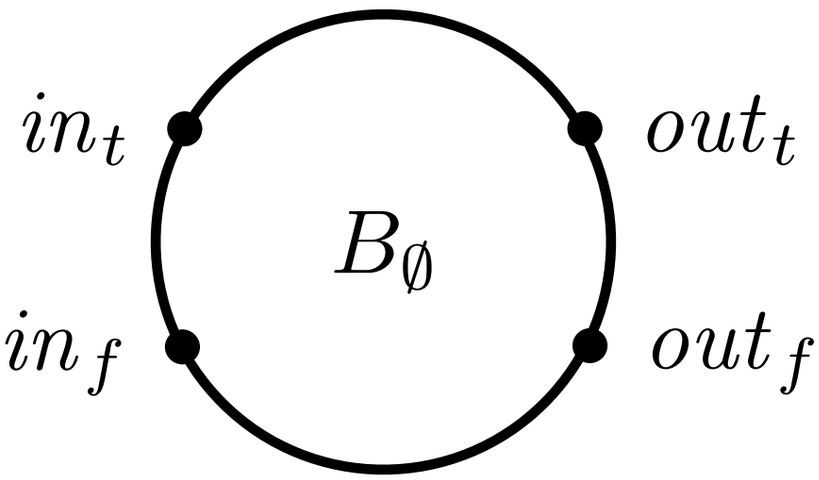
Buffer collegati booleani

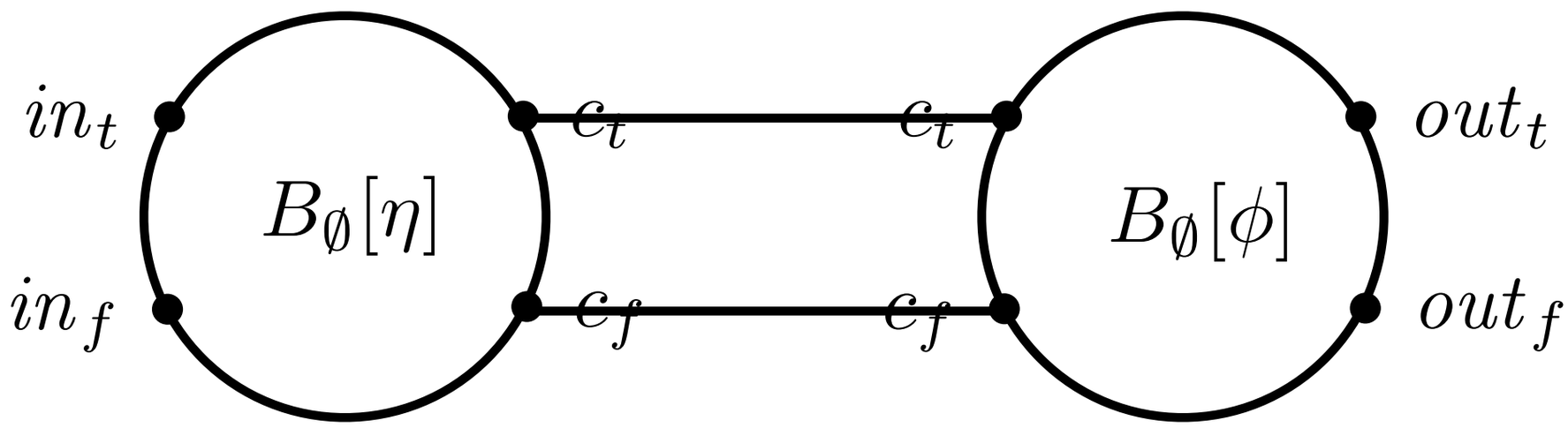
$$B_\emptyset \triangleq in_t.B_t + in_f.B_f$$

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

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





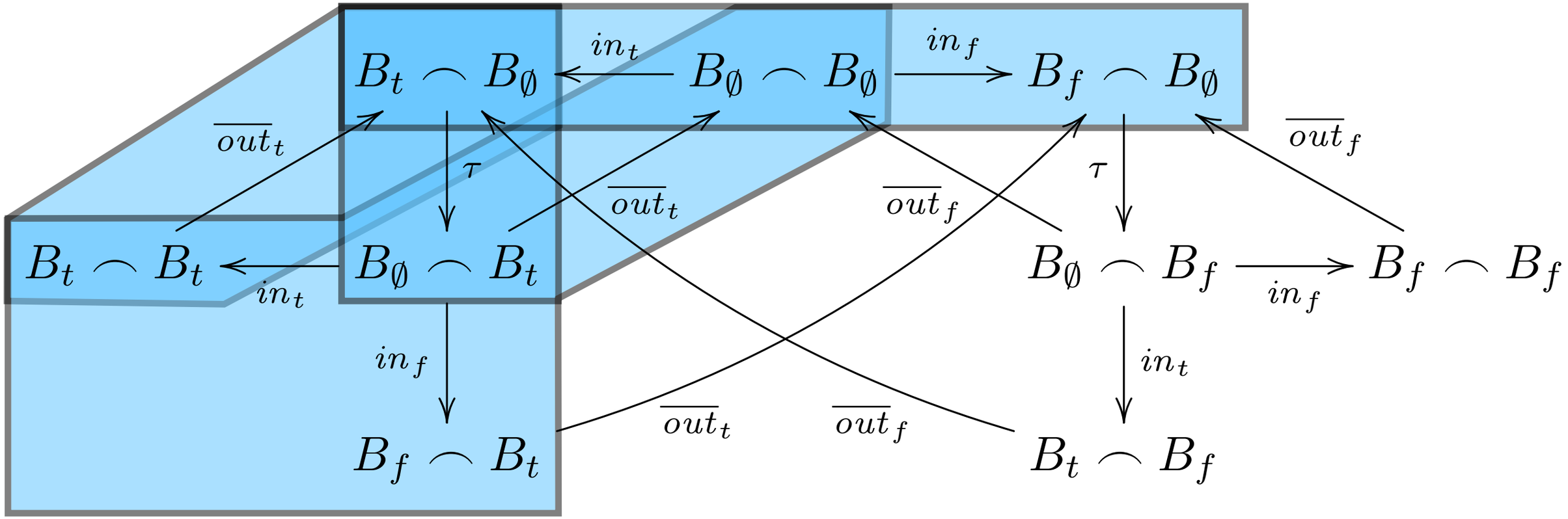


Buffer collegati booleani

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

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

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

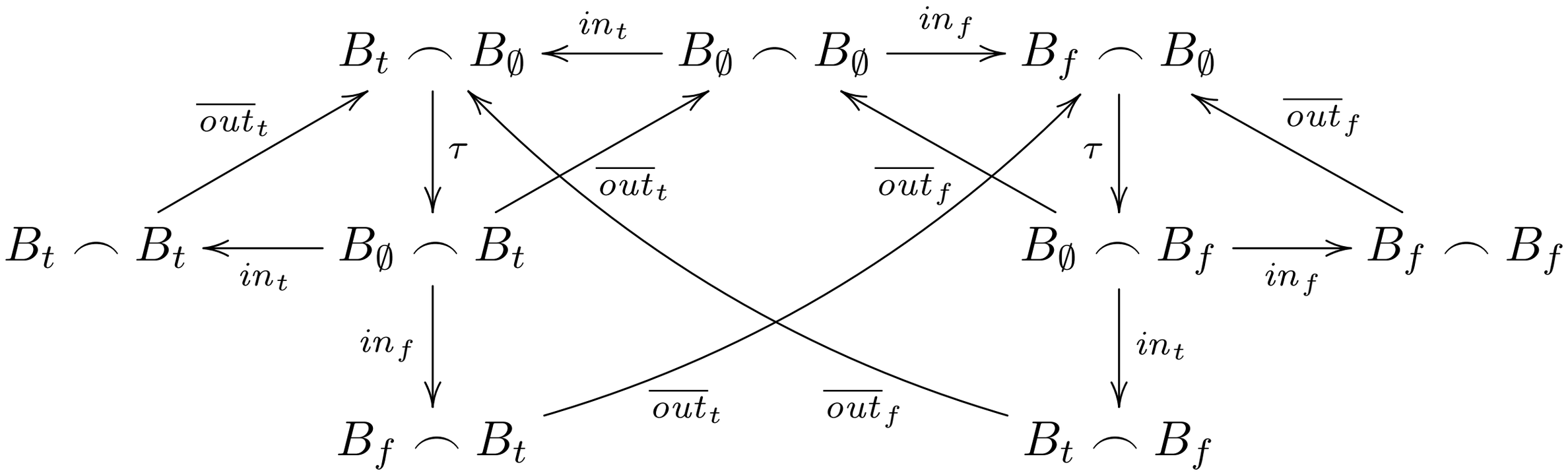


Buffer collegati booleani

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

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

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



CCS con passaggio di valore

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

quando l'insieme dei valori è finito $V \triangleq \{v_1, ..., v_n\}$

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

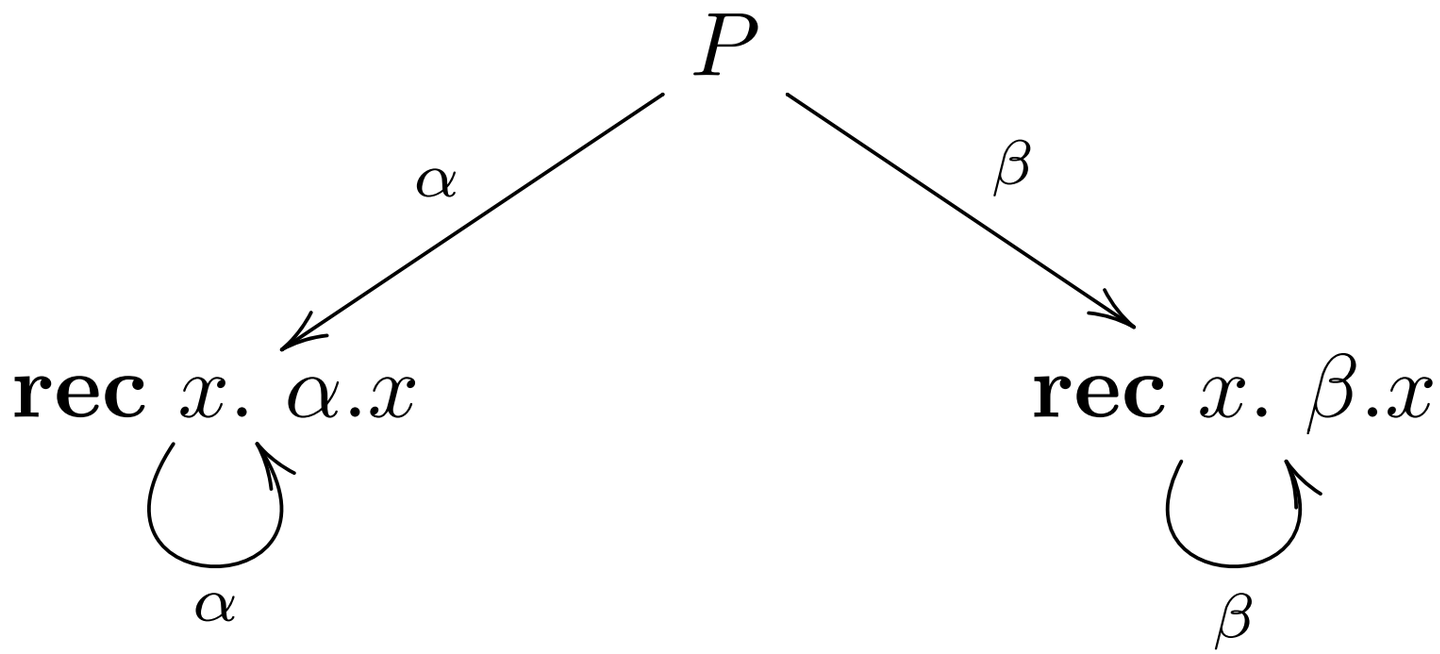
$$\alpha?x.p \equiv \alpha_{v_1}.p[^{v_1}/_x] + \cdots + \alpha_{v_n}.p[^{v_n}/_x]$$

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

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

Esercizio: LTS?

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

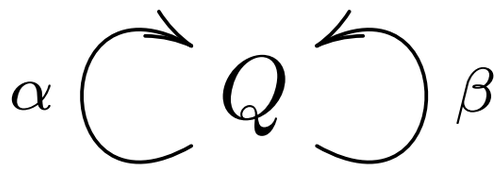


Esercizio: LTS?

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

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

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

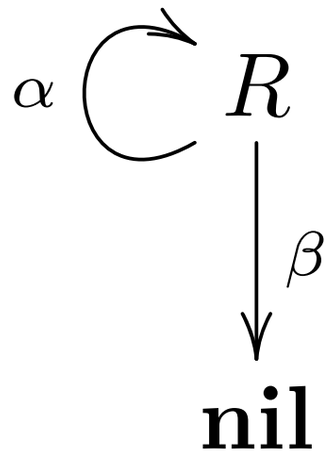


Esercizio: LTS?

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

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

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

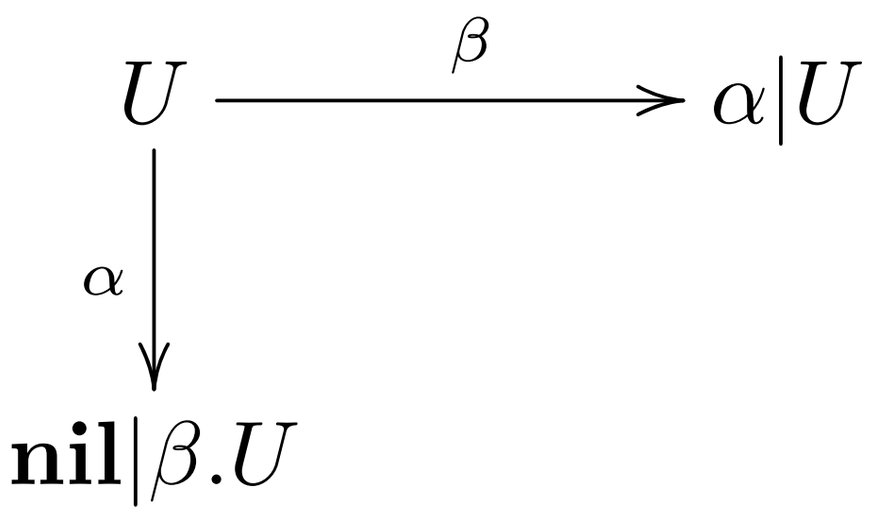


Esercizio: LTS?

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

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

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

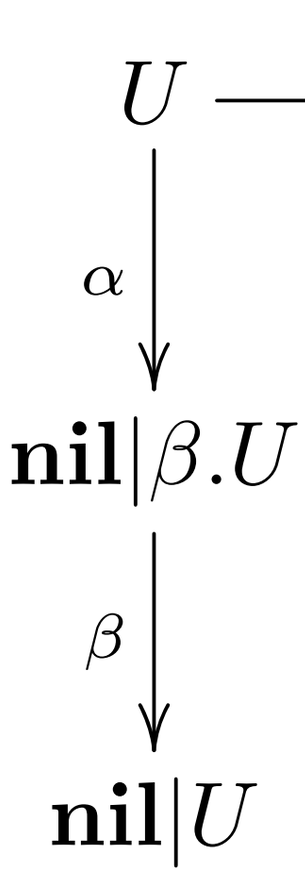


Esercizio: LTS?

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

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

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

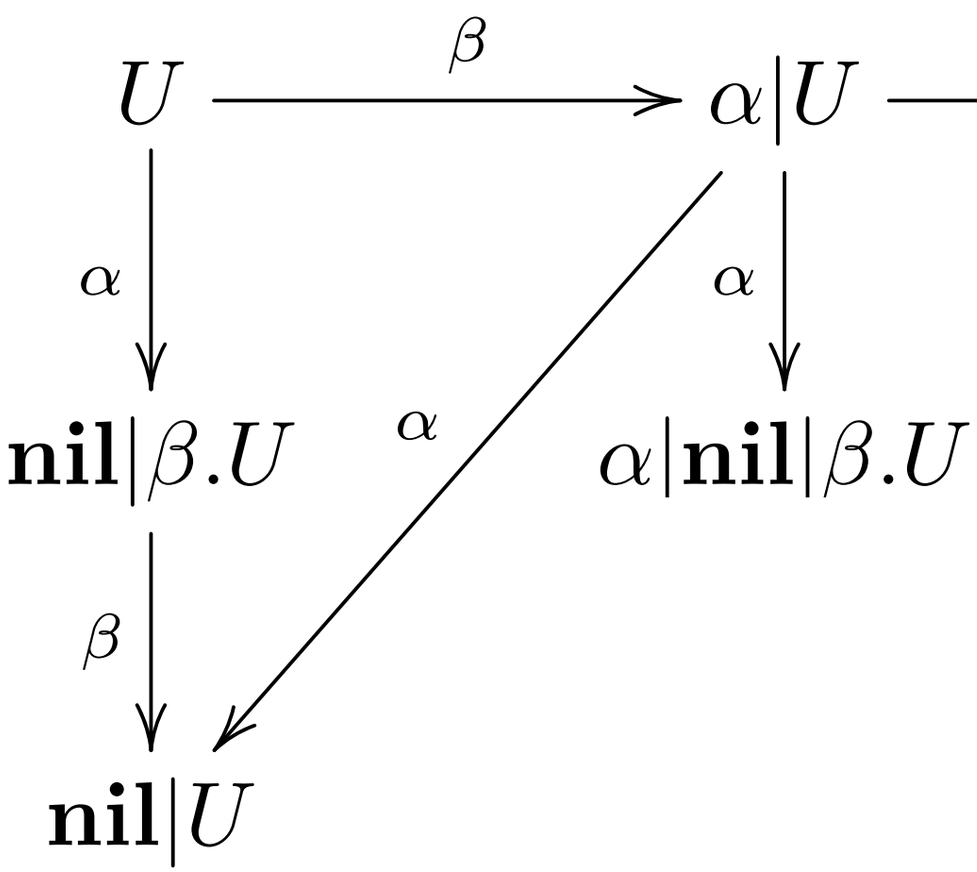


Esercizio: LTS?

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

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

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

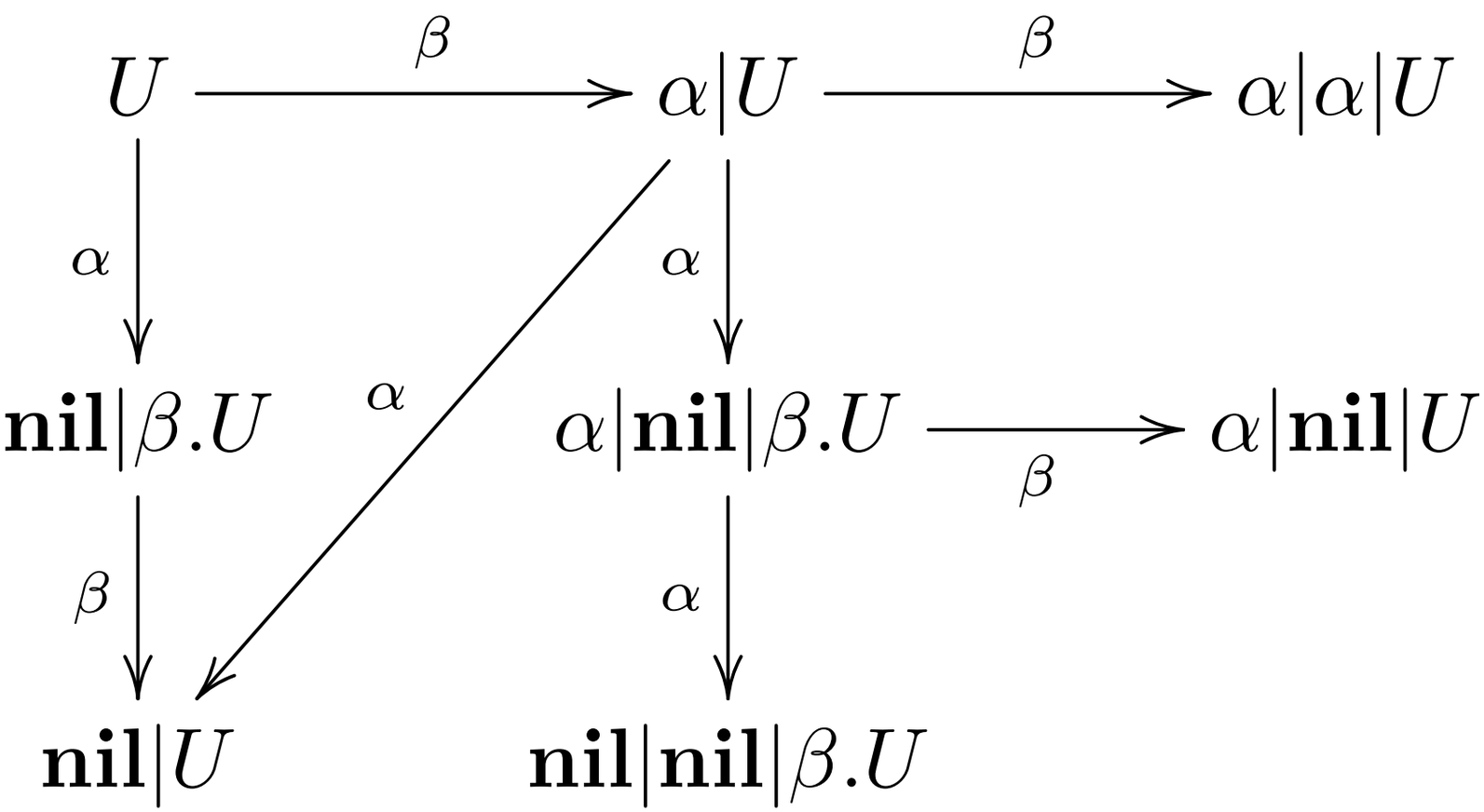


Esercizio: LTS?

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

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

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

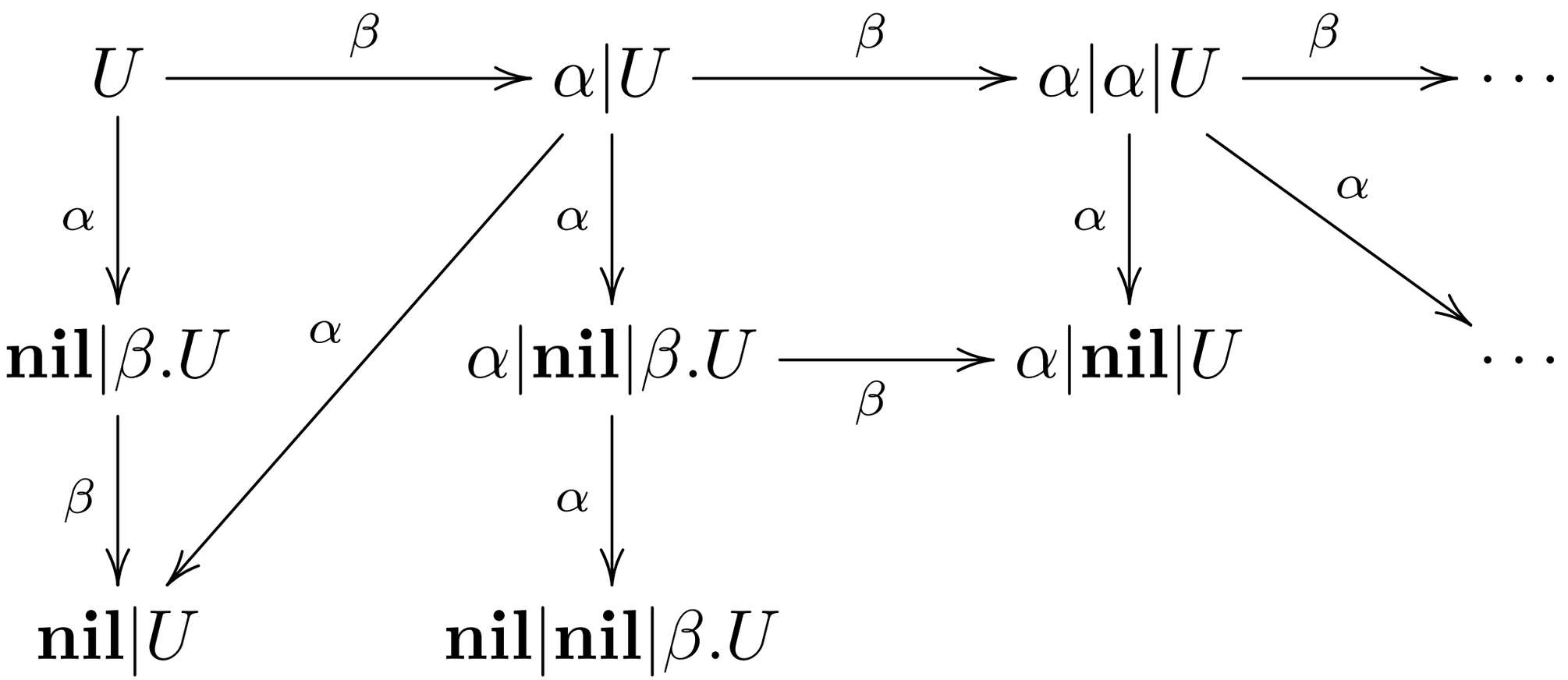


Esercizio: LTS?

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

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

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

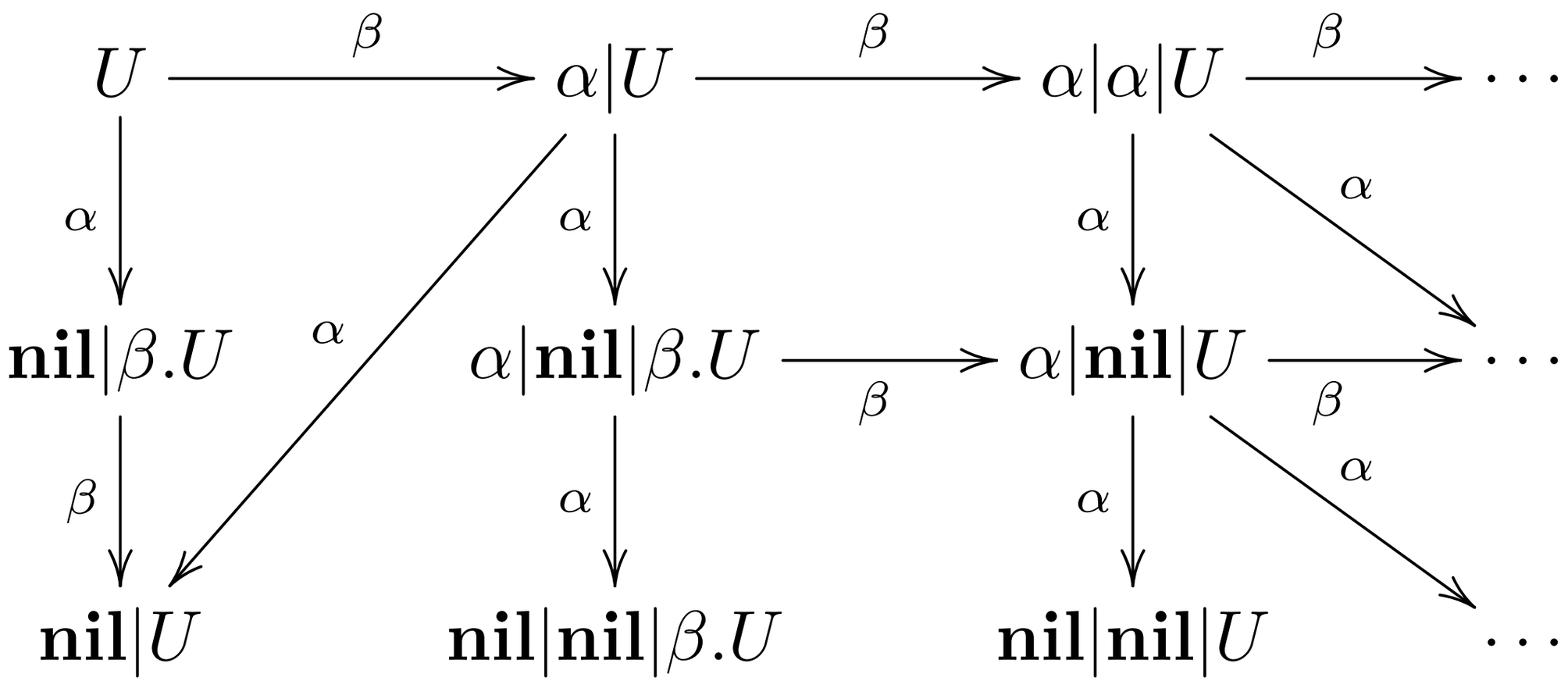


Esercizio: LTS?

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

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

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



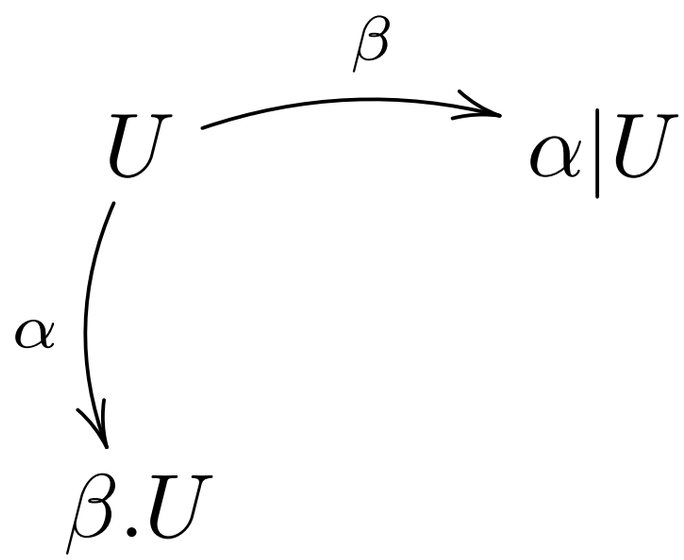
Esercizio: LTS?

ignoriamo **nil**

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

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

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



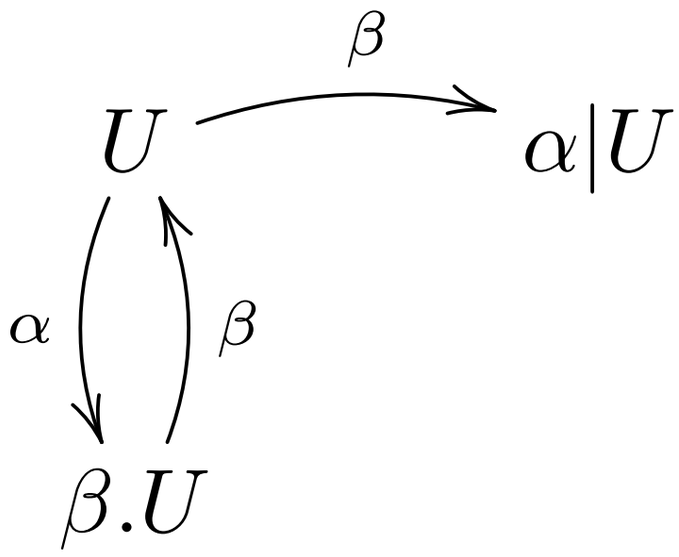
Esercizio: LTS?

ignoriamo **nil**

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

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

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



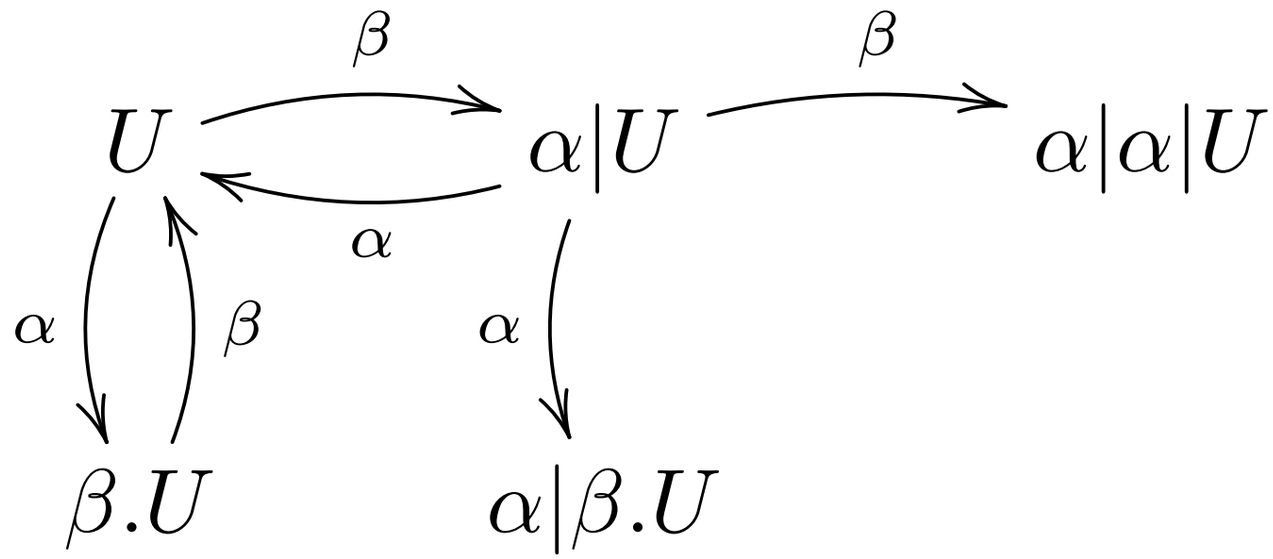
Esercizio: LTS?

ignoriamo **nil**

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

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

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



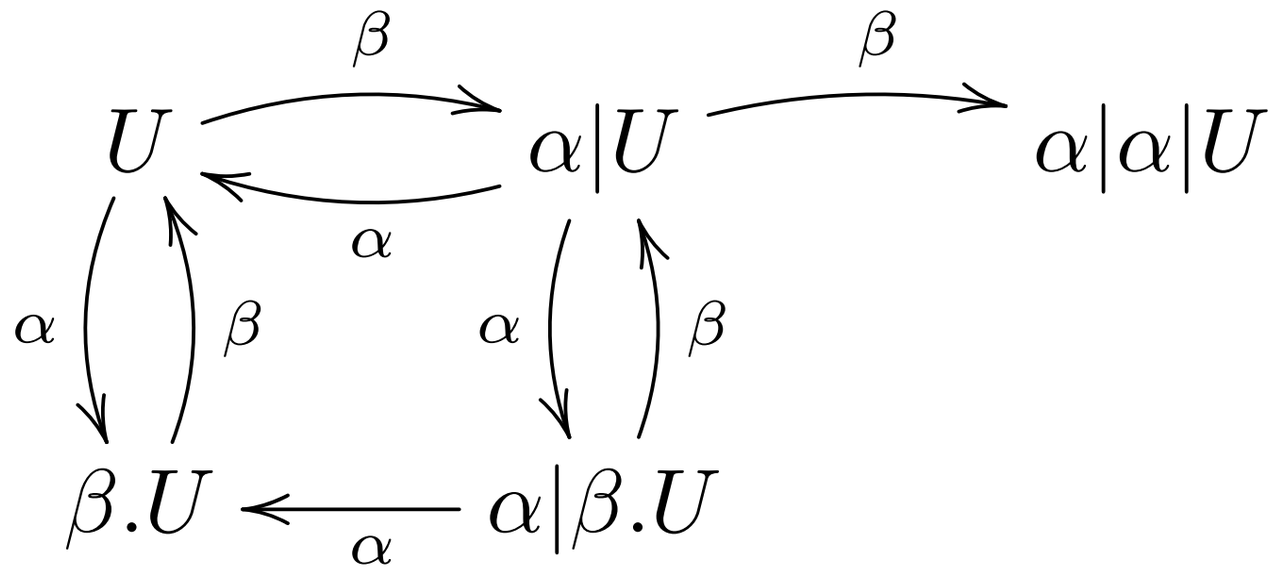
Esercizio: LTS?

ignoriamo **nil**

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

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

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



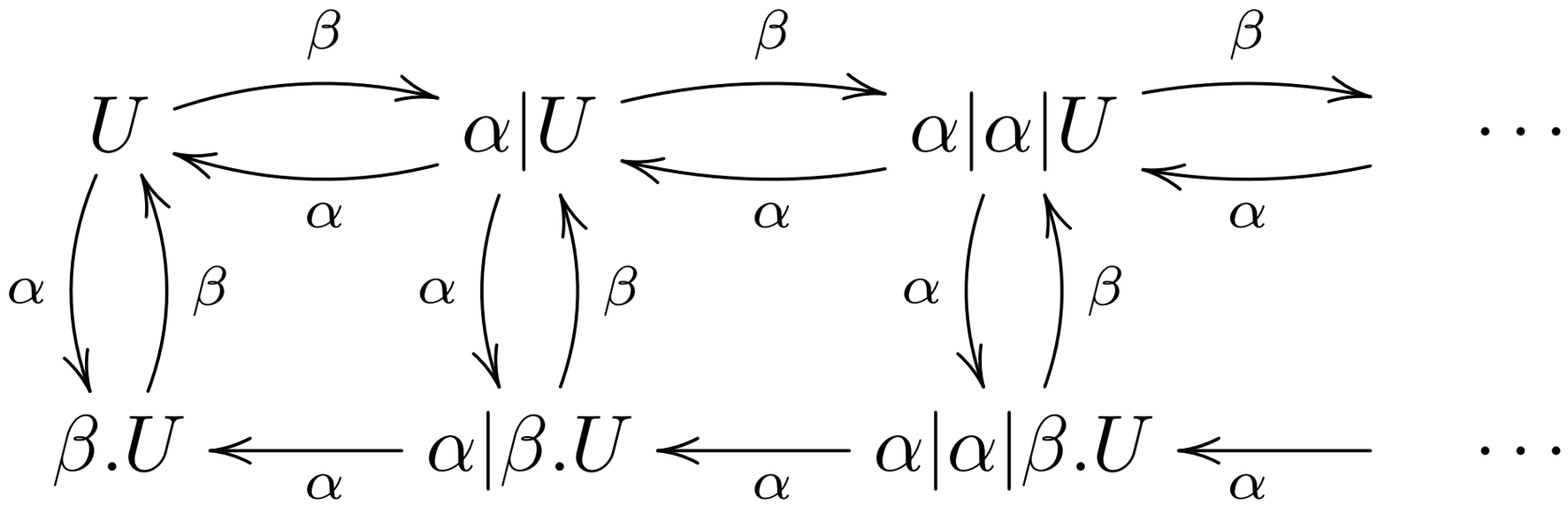
Esercizio: LTS?

ignoriamo **nil**

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

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

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



Esercizio (da consegnare)

Scrivere un contatore interattivo modulo 4 in CCS

Il processo contatore ha quattro canali di ingresso:

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

e quattro canali di uscita:

c_0, c_1, c_2, c_3

usato per visualizzare il valore corrente del contatore

Disegna l'LTS del processo contatore.