

# Linguaggi di Programmazione

Roberta Gori

CCS sintassi & semantica operazionale-11.1,11.2,11.3

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 (Turing equivalente)

## 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

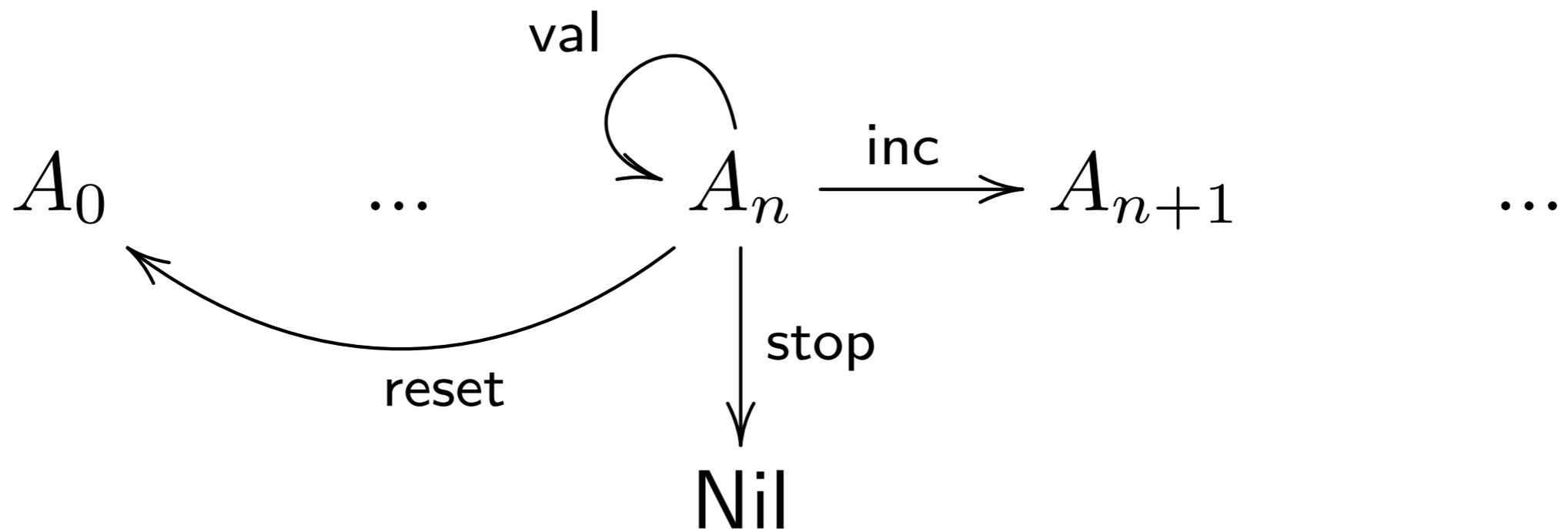
il processo  
dopo l'interazione

numero di stati/transizioni  
può essere infinito

# Esempio: processo contatore

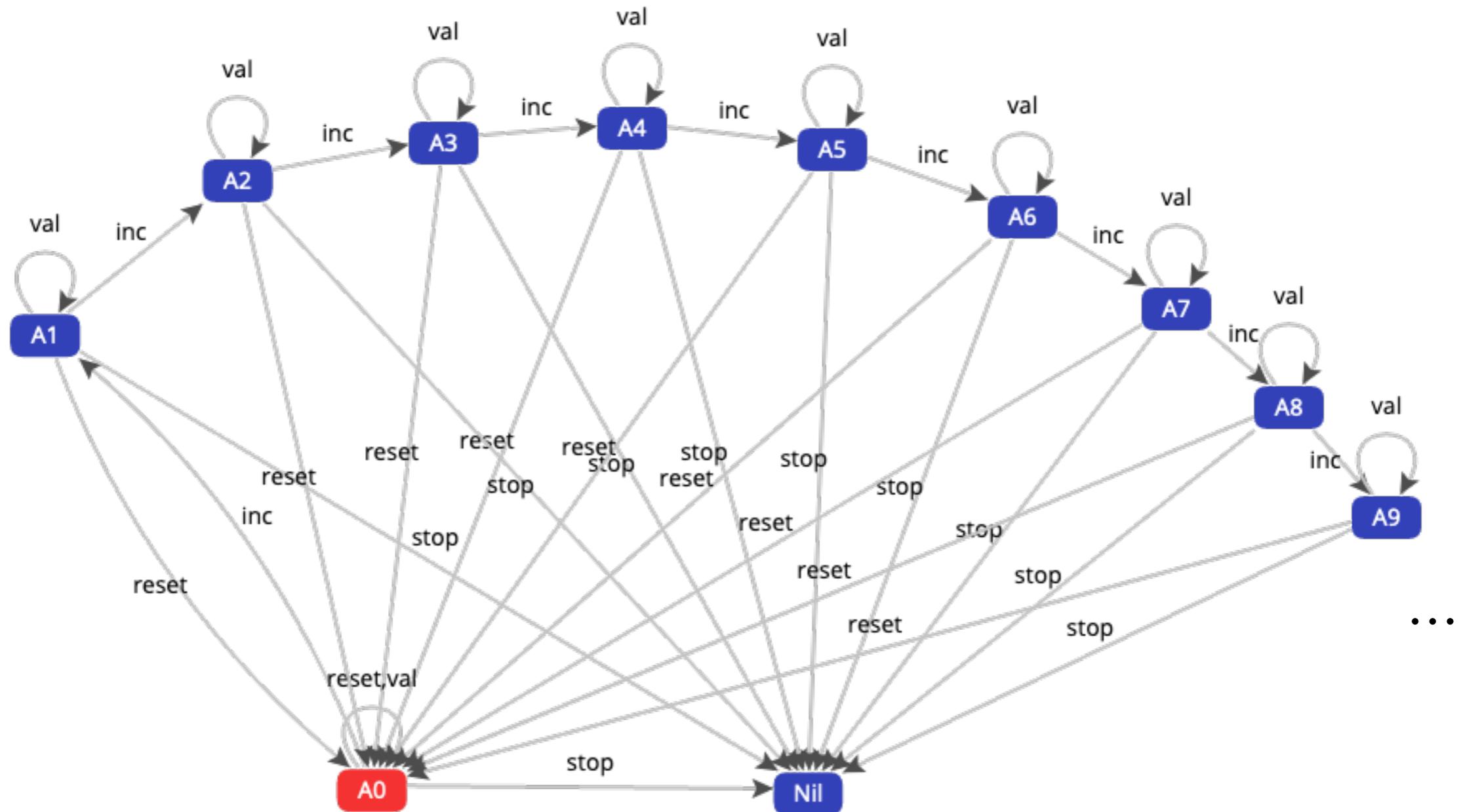
4 possibili interazioni con l'esterno:

- **reset** - resetta il contatore a 0
- **inc** - incrementa il contatore
- **stop** - ferma il contatore
- **val** - restituisce il valore del contatore



# LTS: Labelled Transition System

## 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  $\mu$ ?

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

un'azione (ad esempio un ingresso)  $\alpha?v$  ricevuto  $v$  sul **canale**  $\alpha$

un'azione interna (azione silenziosa)  $\tau$   
(nessuna interazione con l'ambiente)

# 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)

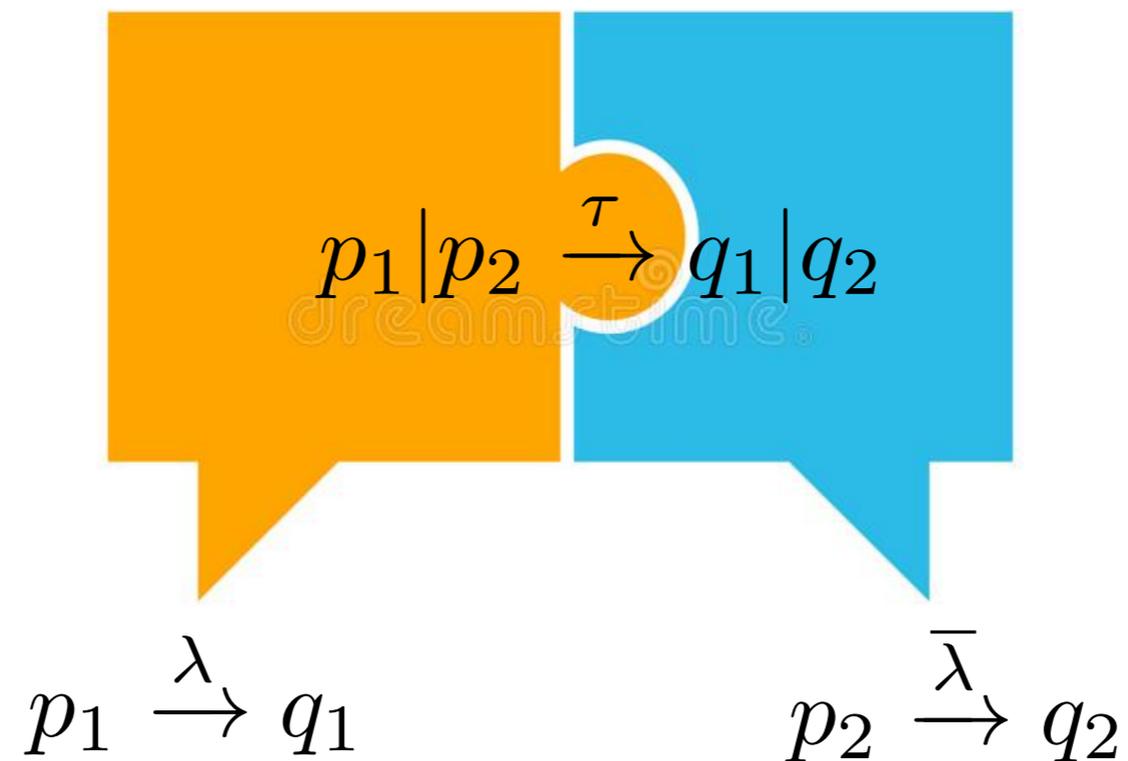
$\alpha!v$  diventa solo  $\overline{\alpha}_v$  o solo  $\overline{\alpha}$

$\alpha?v$  diventa solo  $\alpha_v$  o solo  $\alpha$

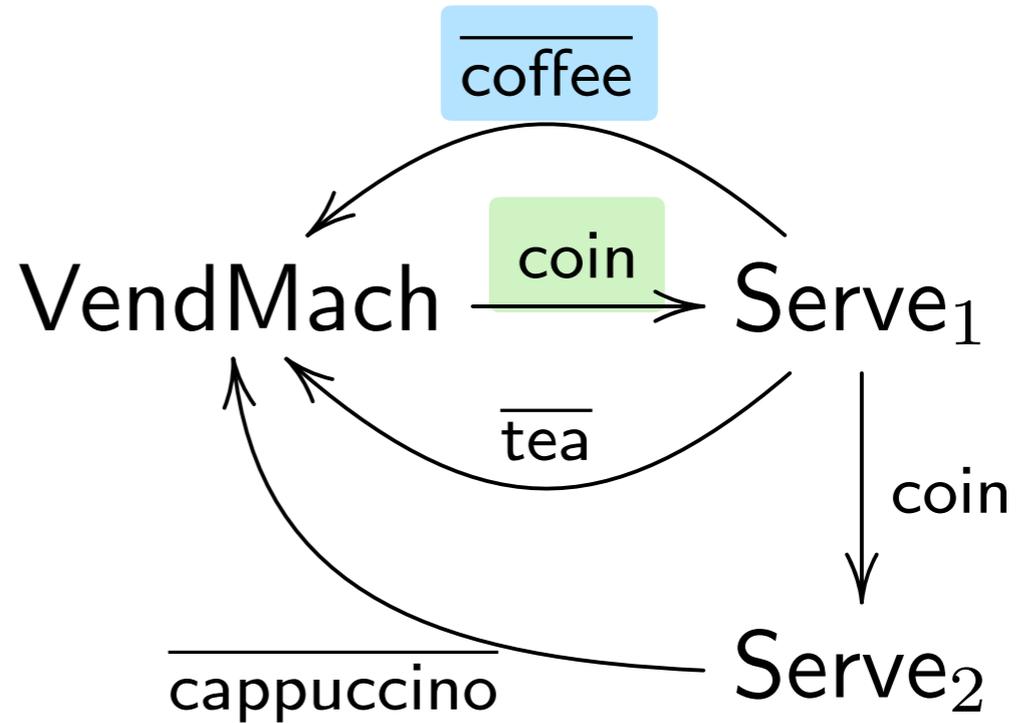
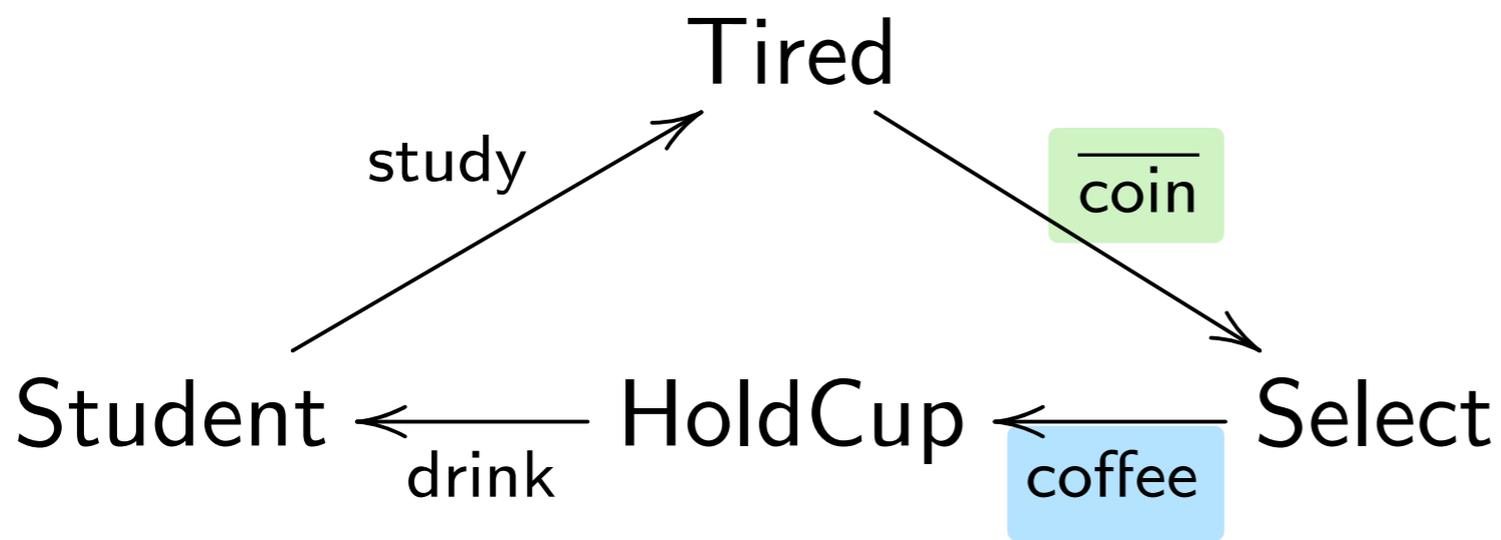
$\lambda$  denota o  $\alpha$  o  $\overline{\alpha}$

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

# CCS: comunicazione



# Esempio: distributore automatico



# Sintassi CCS

# CCS: sintassi

$p, q$	$::=$	<b>nil</b>	processo inattivo
		$x$	variabile di processo (per la ricorsione)
		$\mu.p$	prefisso azione
		$p \setminus \alpha$	canale ristretto
		$p[\phi]$	rietichettatura del canale
		$p + q$	scelta nondeterministica (somma)
		$p   q$	composizione parallela
		<b>rec</b> $x. p$	ricorsione

(gli operatori sono elencati in ordine di precedenza)

# CCS: sintassi

$p, q ::=$

- $\mathbf{nil}$
- $x$        $\mathbf{rec } x. \text{ coffee.}x + \text{ tea.nil} \mid \text{ water.nil}$
- $\mu.p$
- $p \setminus \alpha$
- $p[\phi]$       **che si legge**
- $p + q$
- $p \mid q$        $\mathbf{rec } x. (((\text{coffee.}x) + \text{tea.nil}) \mid \text{water.nil})$
- $\mathbf{rec } x. p$

(gli operatori sono elencati in ordine di precedenza)

# CCS: sintassi

l'unico operatore che lega le variabili e' la ricorsione

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

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

$$fv(p)$$

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-ridenominati rispetto alle variabili vincolate

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

semantica operativa del CCS

# CCS: etichette

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

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

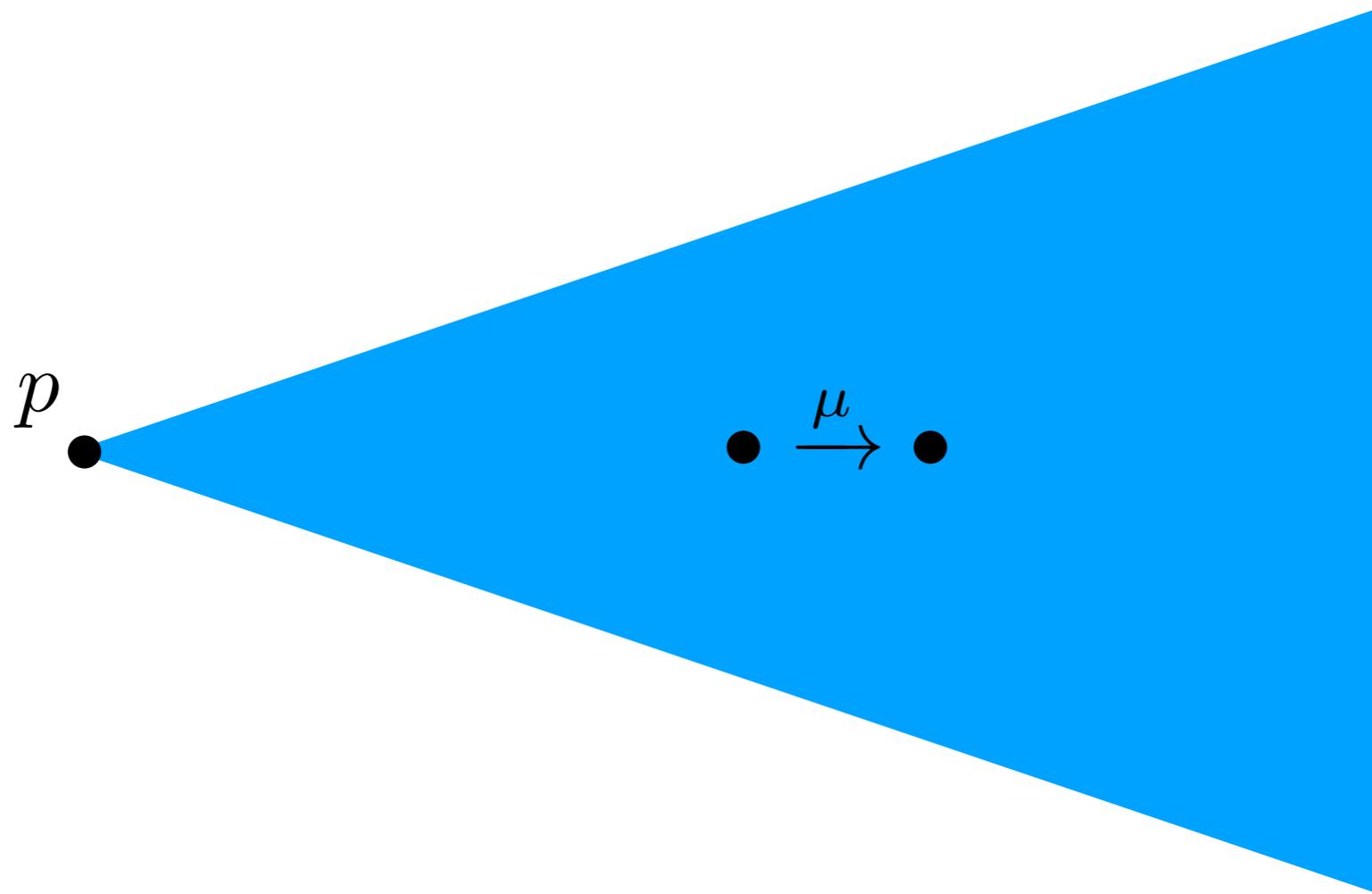
$\Lambda = \mathcal{C} \cup \bar{\mathcal{C}}$  insieme di azioni osservabili, denotate da  $\lambda$   $\bar{\lambda}$

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

$\mathcal{L} = \Lambda \cup \{\tau\}$  insieme di azioni, denotate da  $\mu$

# 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

$\text{nil} \nrightarrow$

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

nil 

# prefisso azione

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

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

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

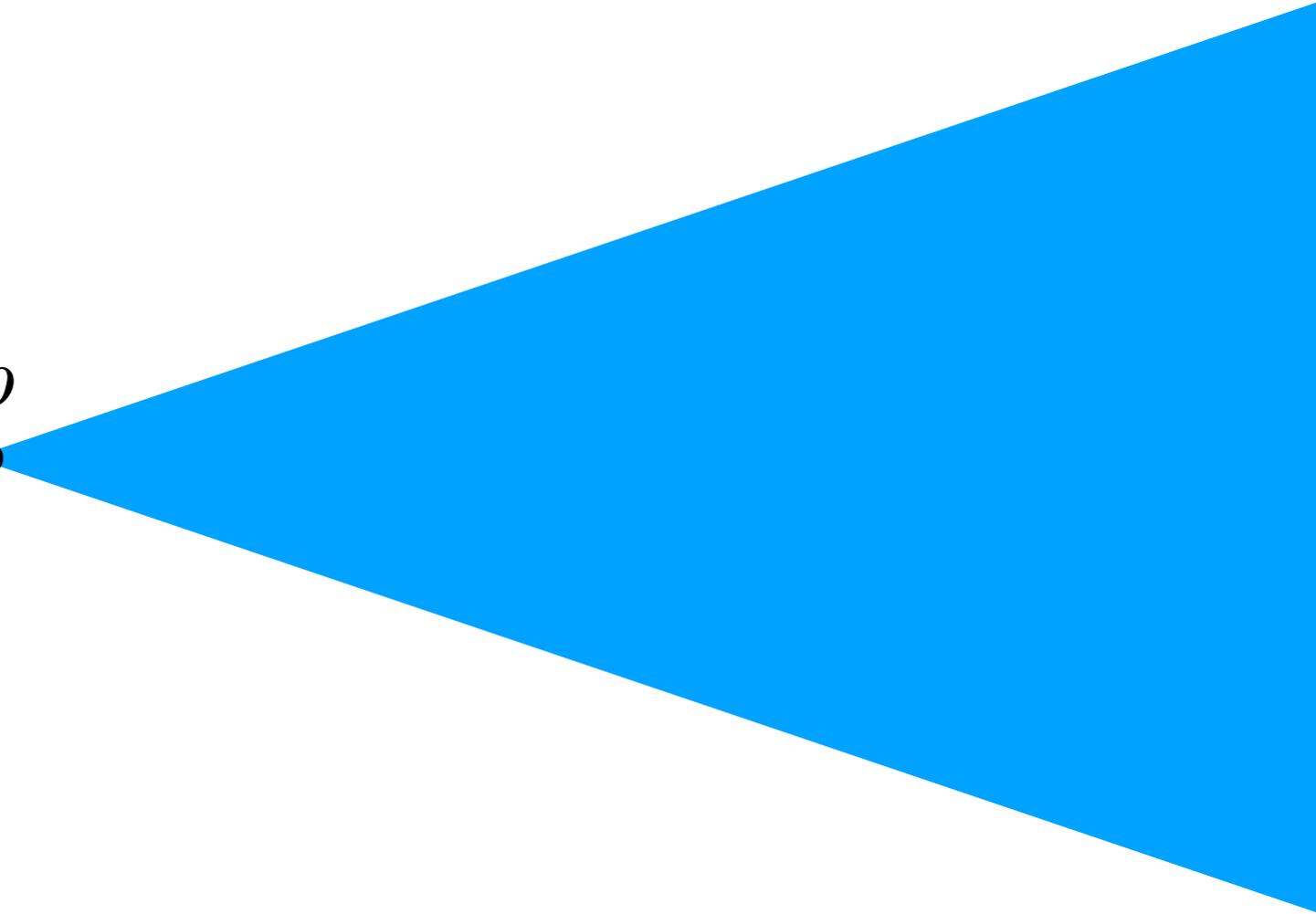
$$\overline{coin.coffee.nil}$$

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

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

# LTS del processo

$\mu.p \bullet \xrightarrow{\mu} p \bullet$



# Scelta non deterministica

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

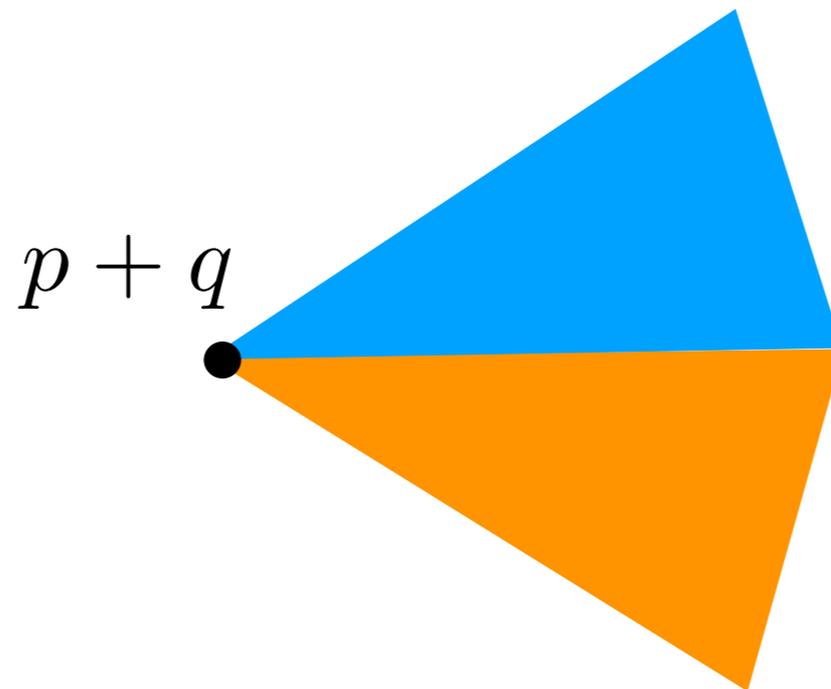
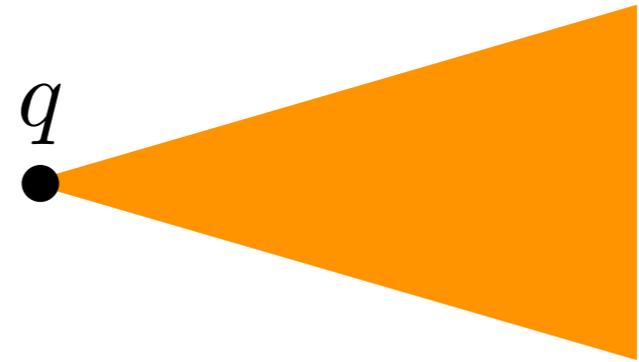
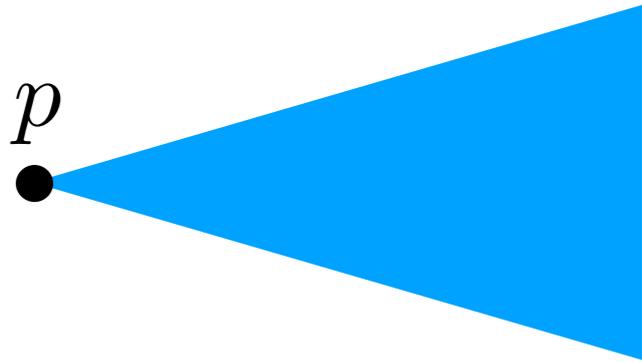
Il processo  $p_1 + p_2$  può comportarsi come  $p_1$  o come  $p_2$

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

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

$$\begin{array}{c} \text{coin.}(\overline{\text{coffee.nil}} + \overline{\text{tea.nil}}) \\ \downarrow \text{coin} \\ \overline{\text{coffee.nil}} + \overline{\text{tea.nil}} \\ \begin{array}{c} \overline{\text{coffee}} \left( \right) \overline{\text{tea}} \\ \downarrow \quad \downarrow \\ \text{nil} \end{array} \end{array}$$

# LTS del processo



# Ricorsione

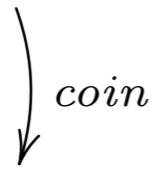
$$\text{Rec)} \frac{p[\mathbf{rec} \ x. \ p / x] \xrightarrow{\mu} q}{\mathbf{rec} \ x. \ p \xrightarrow{\mu} q}$$

come una definizione ricorsiva  $\text{let } x = p \text{ in } x$

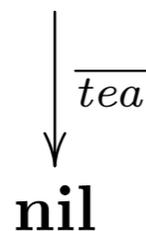
$$\mathbf{rec} \ x. \ \text{coin}.\overline{(\text{coffee}.x + \text{tea.nil})}$$

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

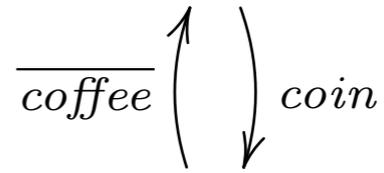
$$\mathbf{rec} \ x. \ \text{coin}.\overline{(\text{coffee}.x + \text{tea.nil})}$$



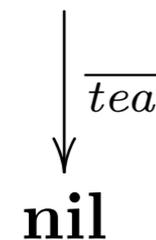
$$\overline{\text{coffee}.\mathbf{rec} \ x. \ \text{coin}.\overline{(\text{coffee}.x + \text{tea.nil})}} + \overline{\text{tea.nil}}$$



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



$$\overline{\text{coffee}.P} + \overline{\text{tea.nil}}$$



# Ricorsione tramite costanti di processo

immaginate che alcune costanti  $A$  di processo siano disponibili insieme ad  $\Delta$  un insieme di dichiarazioni della forma

$$A \triangleq p$$

per ogni costante

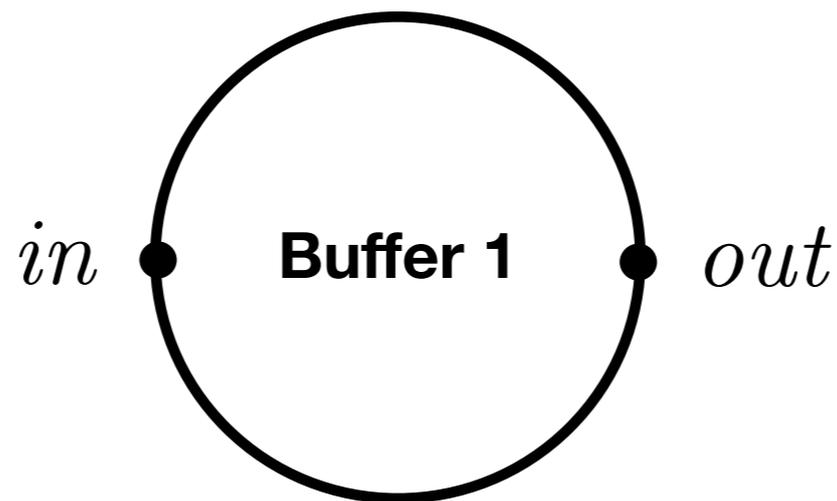
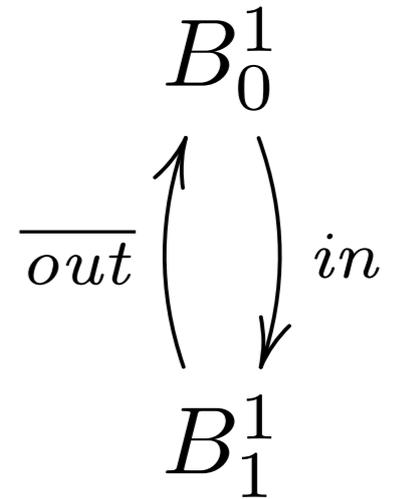
$$\text{Const) } \frac{A \triangleq p \in \Delta \quad p \xrightarrow{\mu} q}{A \xrightarrow{\mu} q}$$

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

# CCS: buffer di capacita' 1

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

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

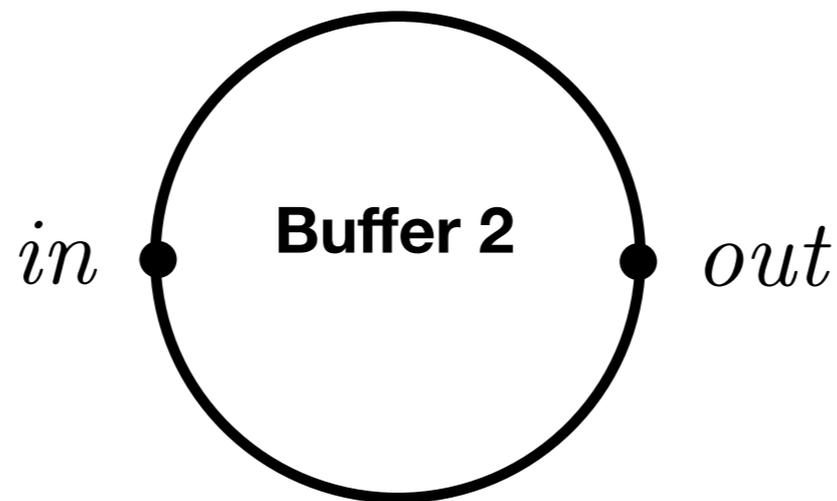
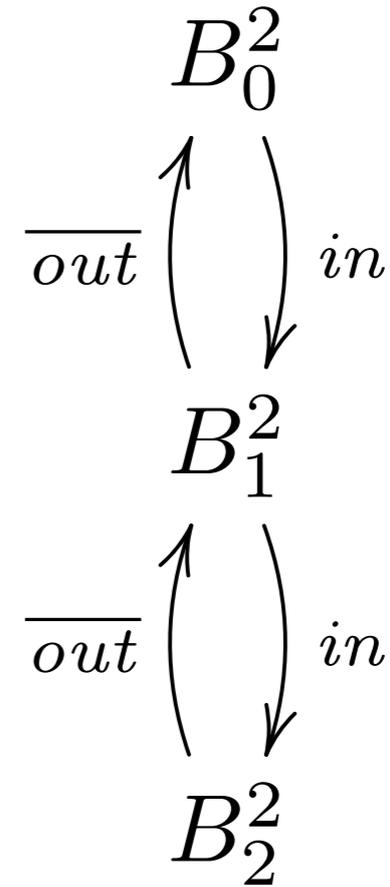


# CCS: buffer di capacita' 2

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

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

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

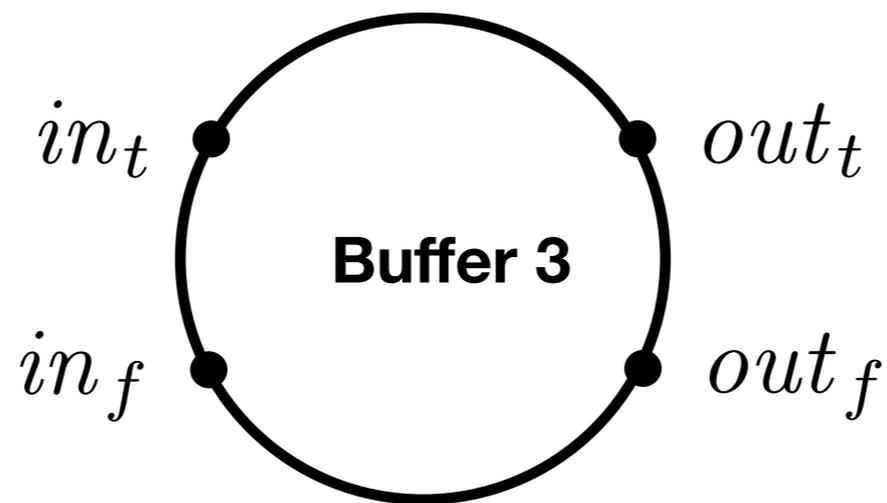
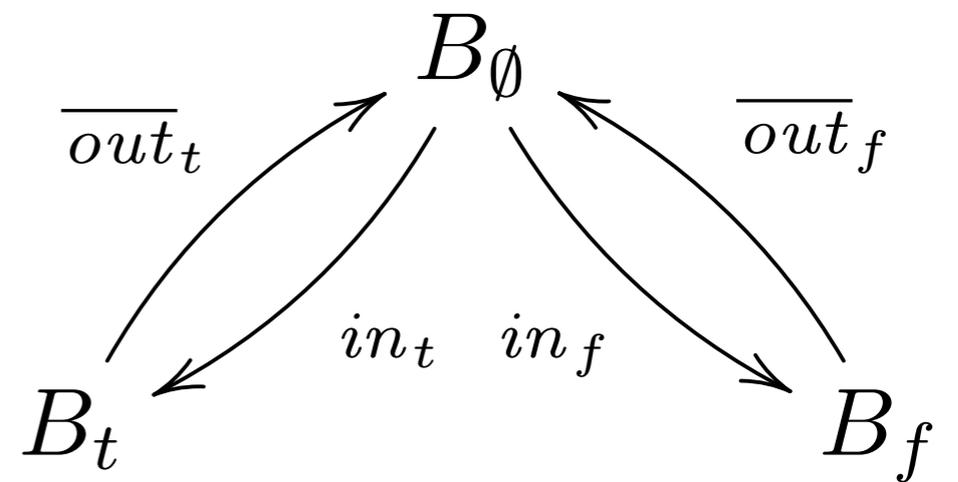


# CCS: buffer booleano

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

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

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



# Composizione parallela

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

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

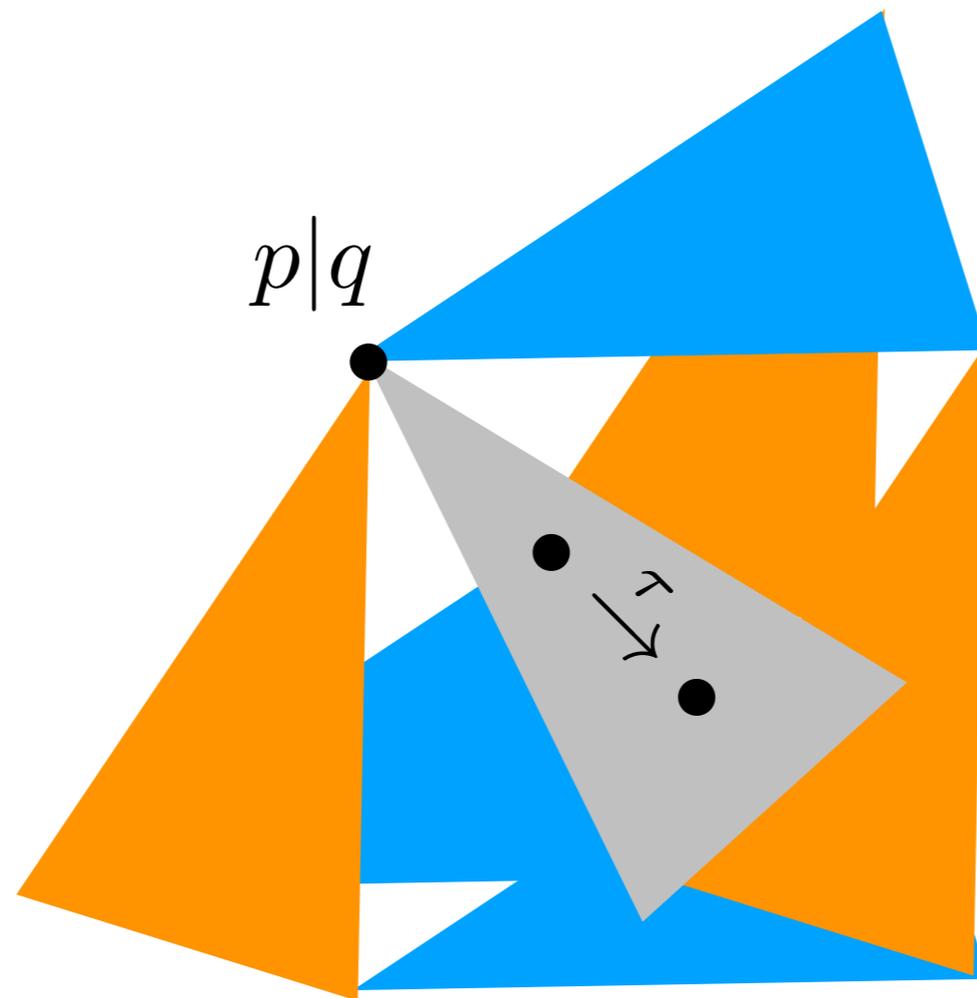
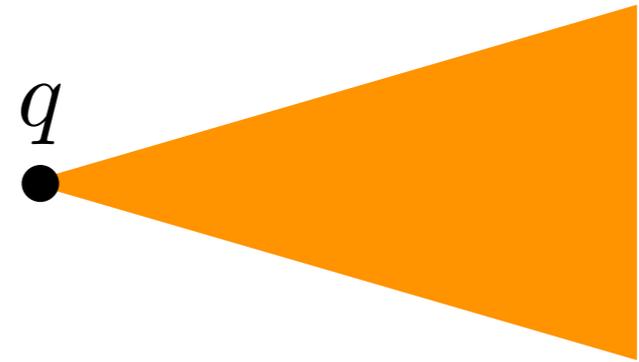
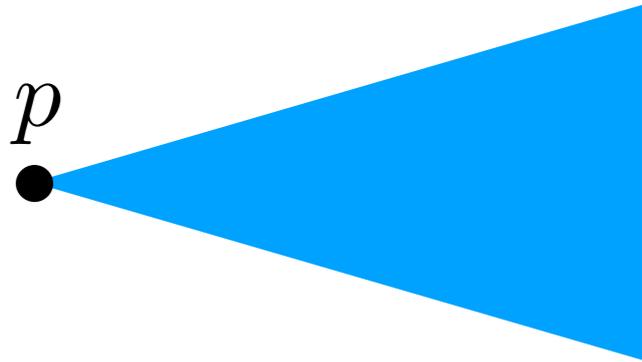
$$P \triangleq \overline{coin}.coffee.nil \quad M \triangleq coin.(\overline{coffee}.nil + \overline{tea}.nil)$$

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

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

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

# LTS del processo



# CCS: buffer paralleli

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

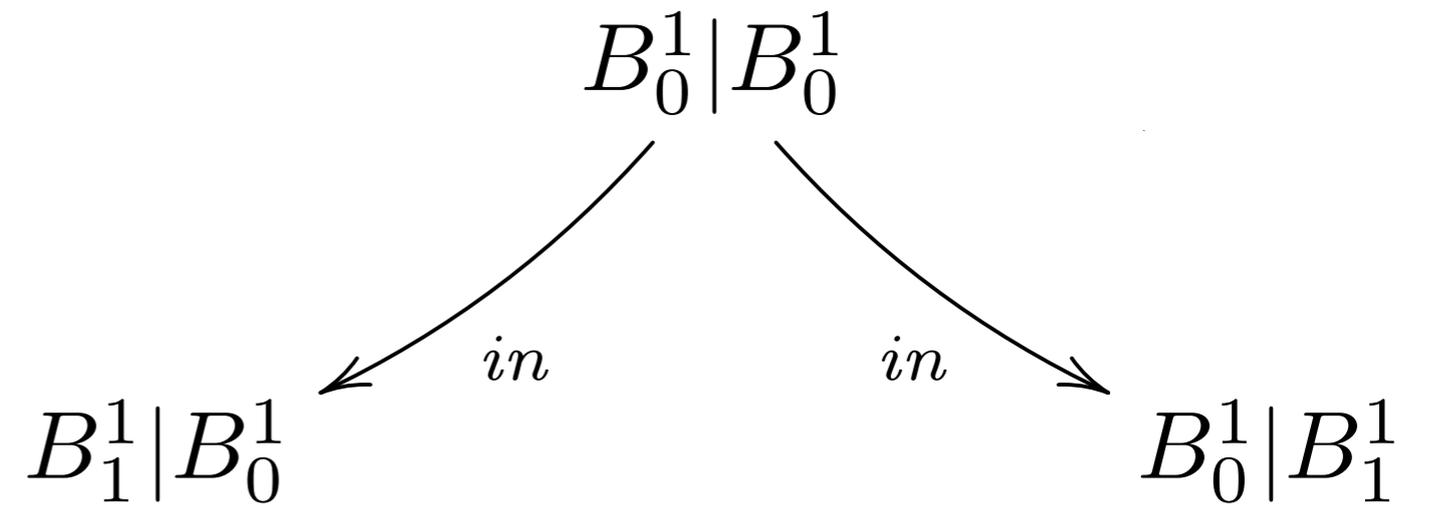
$$B_1^1 \triangleq \overline{out}.B_0^1$$

$$B_0^1 | B_0^1$$

# CCS: buffer paralleli

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

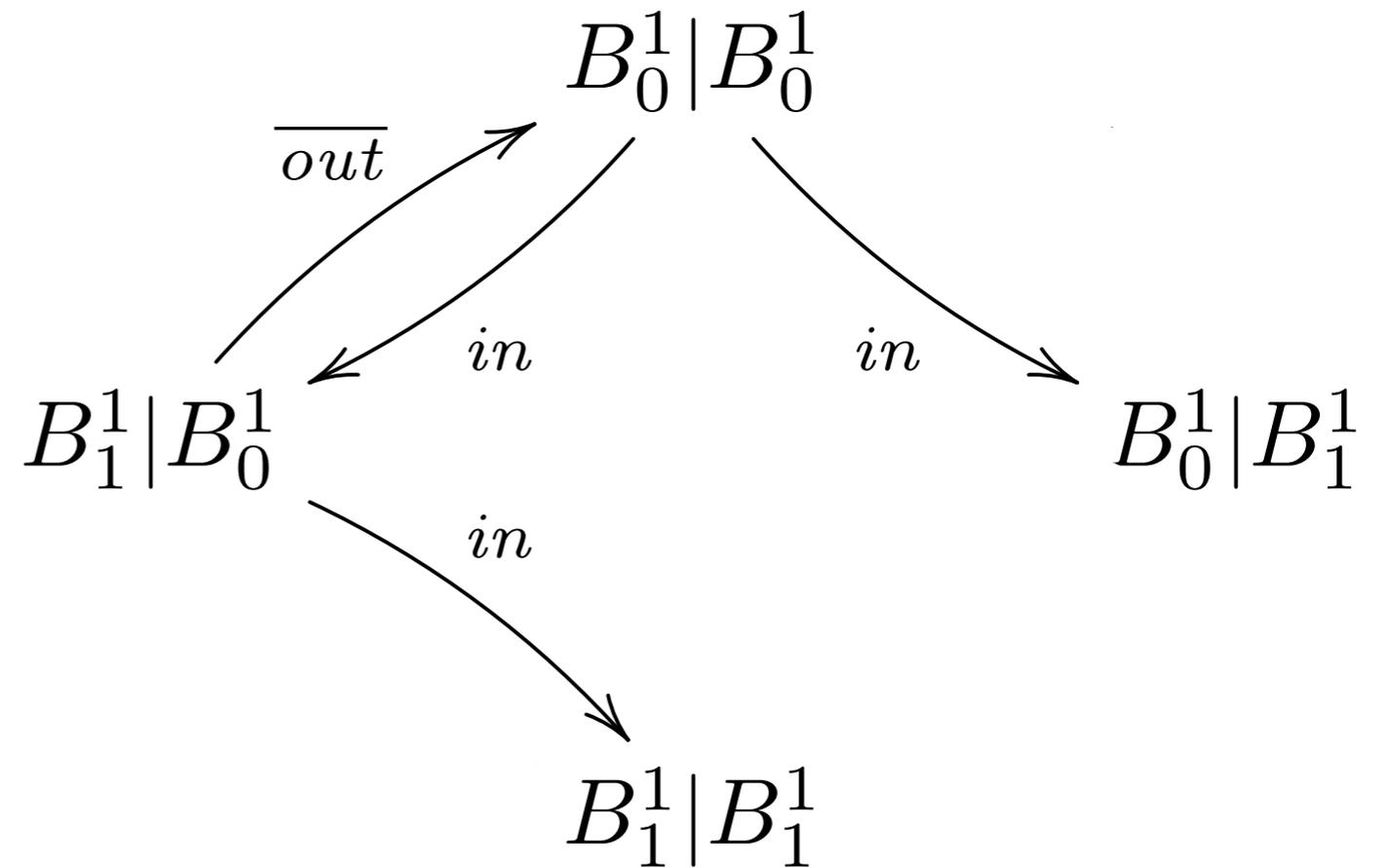
$$B_1^1 \triangleq \overline{out}.B_0^1$$



# CCS: buffer paralleli

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

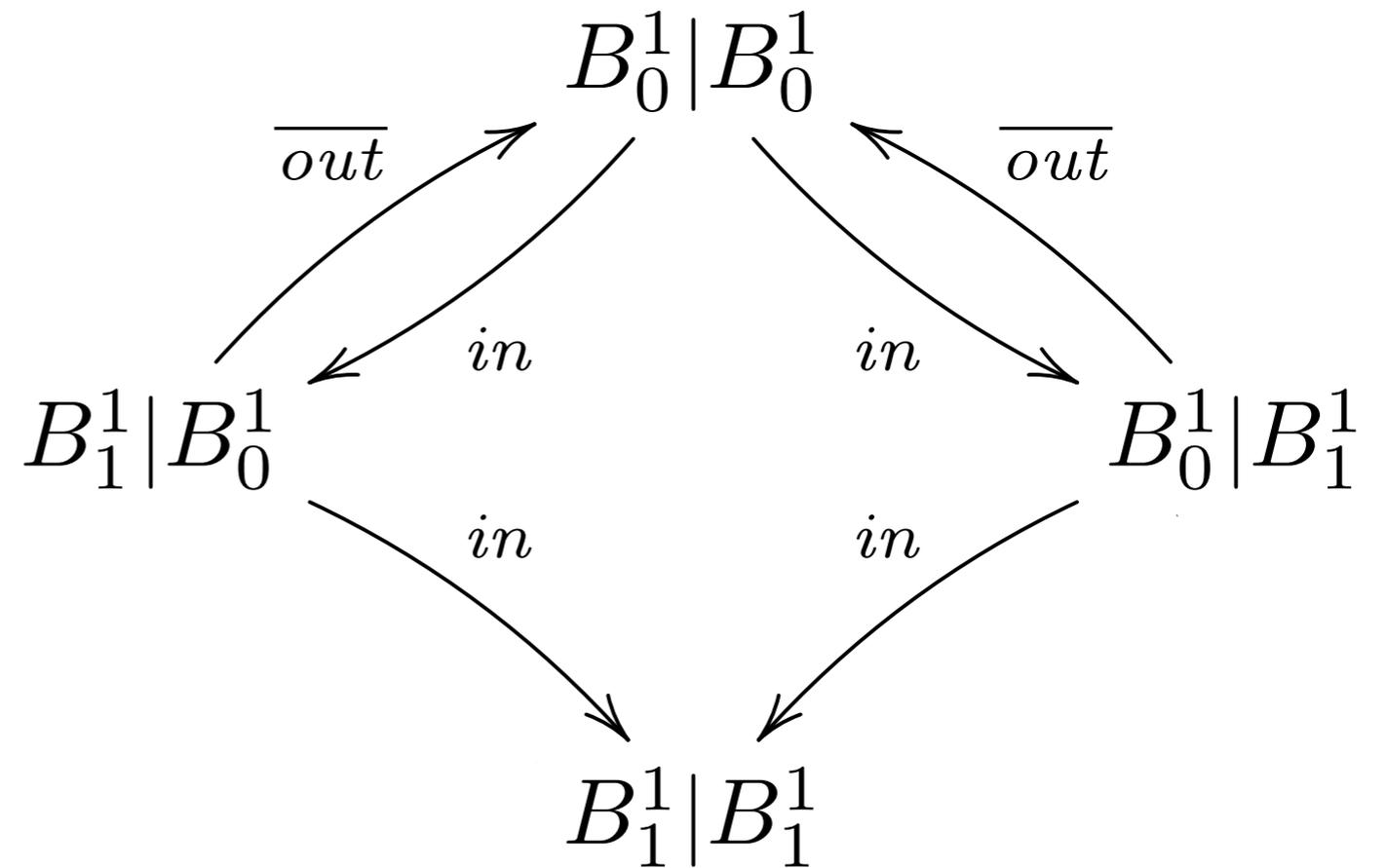
$$B_1^1 \triangleq \overline{out}.B_0^1$$



# CCS: buffer paralleli

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

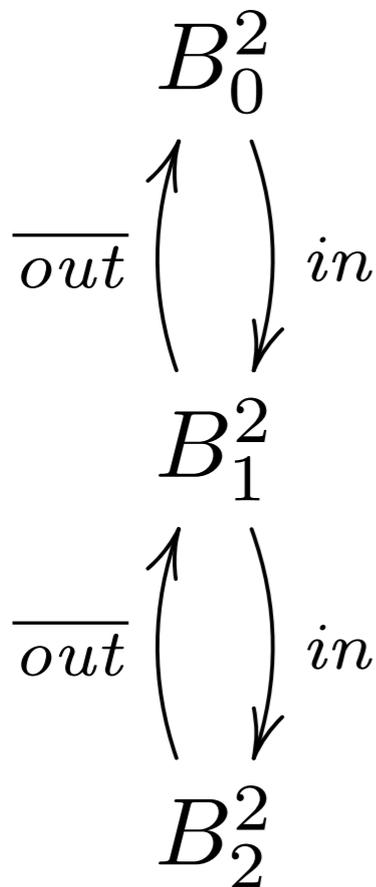
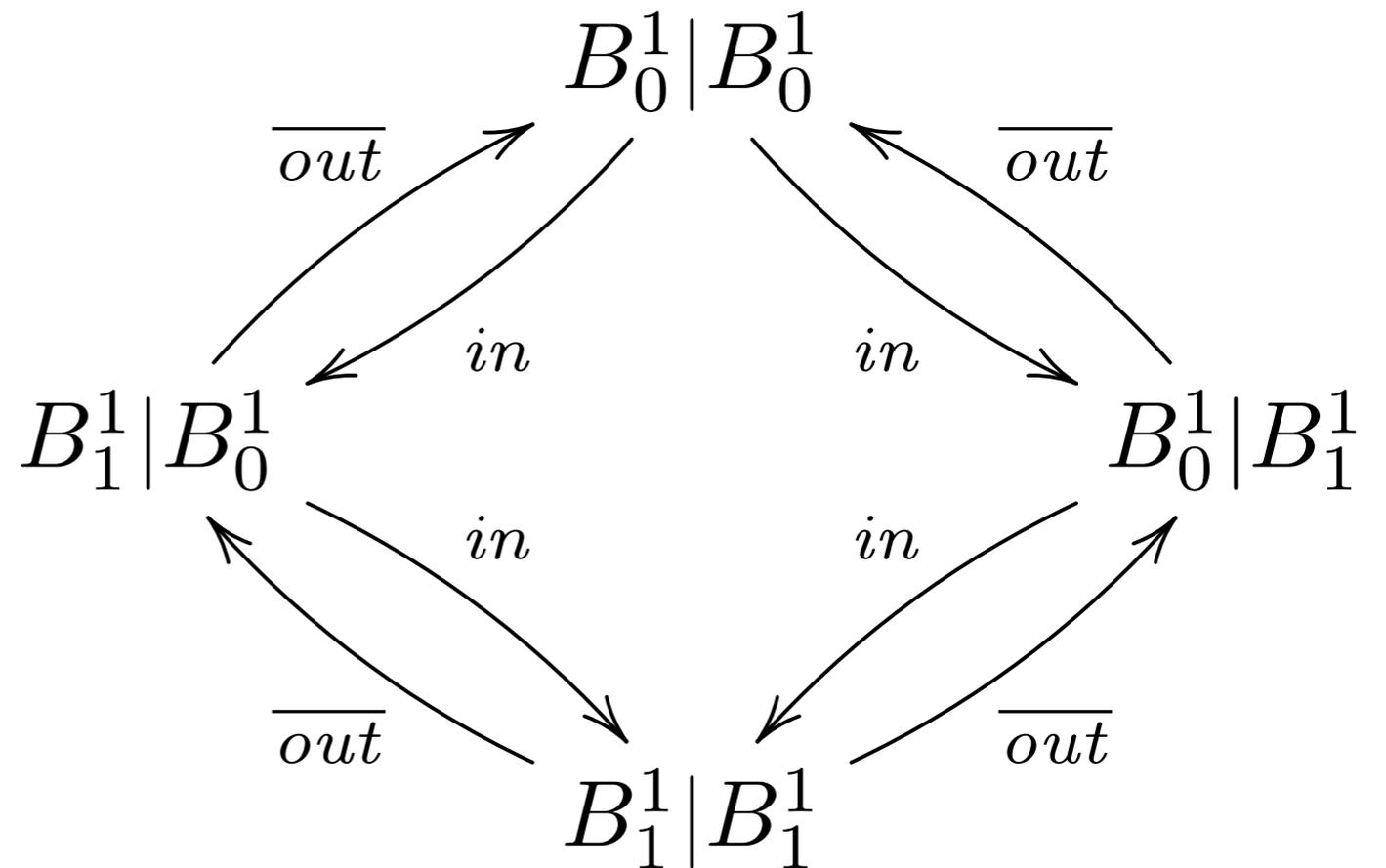
$$B_1^1 \triangleq \overline{out}.B_0^1$$



# 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

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

rende il canale  $\alpha$  privato a  $p$

nessuna interazione sul canale  $\alpha$  con l'ambiente

se  $p$  è la composizione parallela dei processi, allora possono sincronizzarsi su  $\alpha$

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

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

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

# Restrizione: shorthand

dato  $S = \{\alpha_1, \dots, \alpha_n\}$  scriviamo  $p \setminus S$

invece di  $p \setminus \alpha_1 \dots \setminus \alpha_n$

omettiamo **nil**

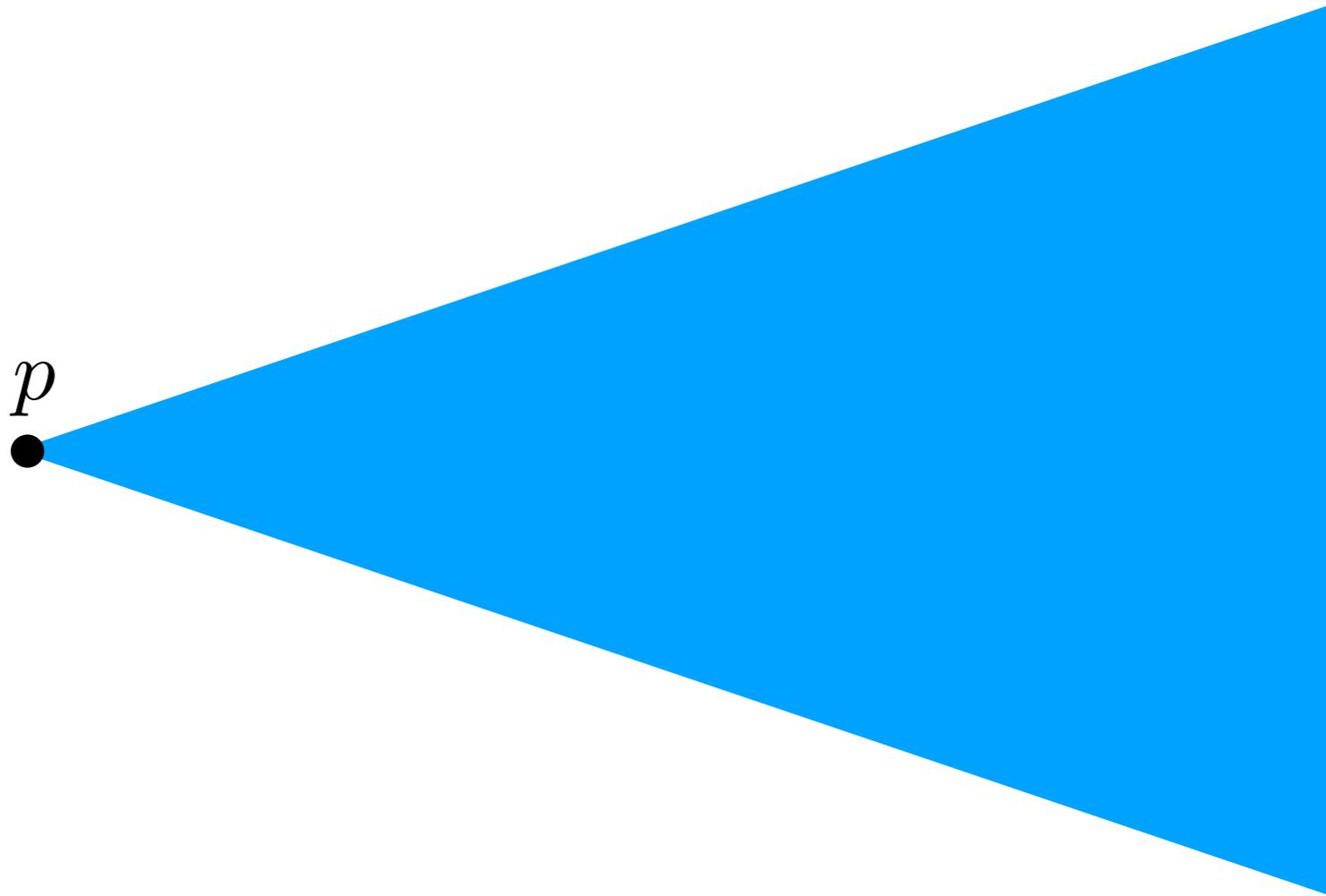
$$P \triangleq \overline{coin}.coffee$$

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

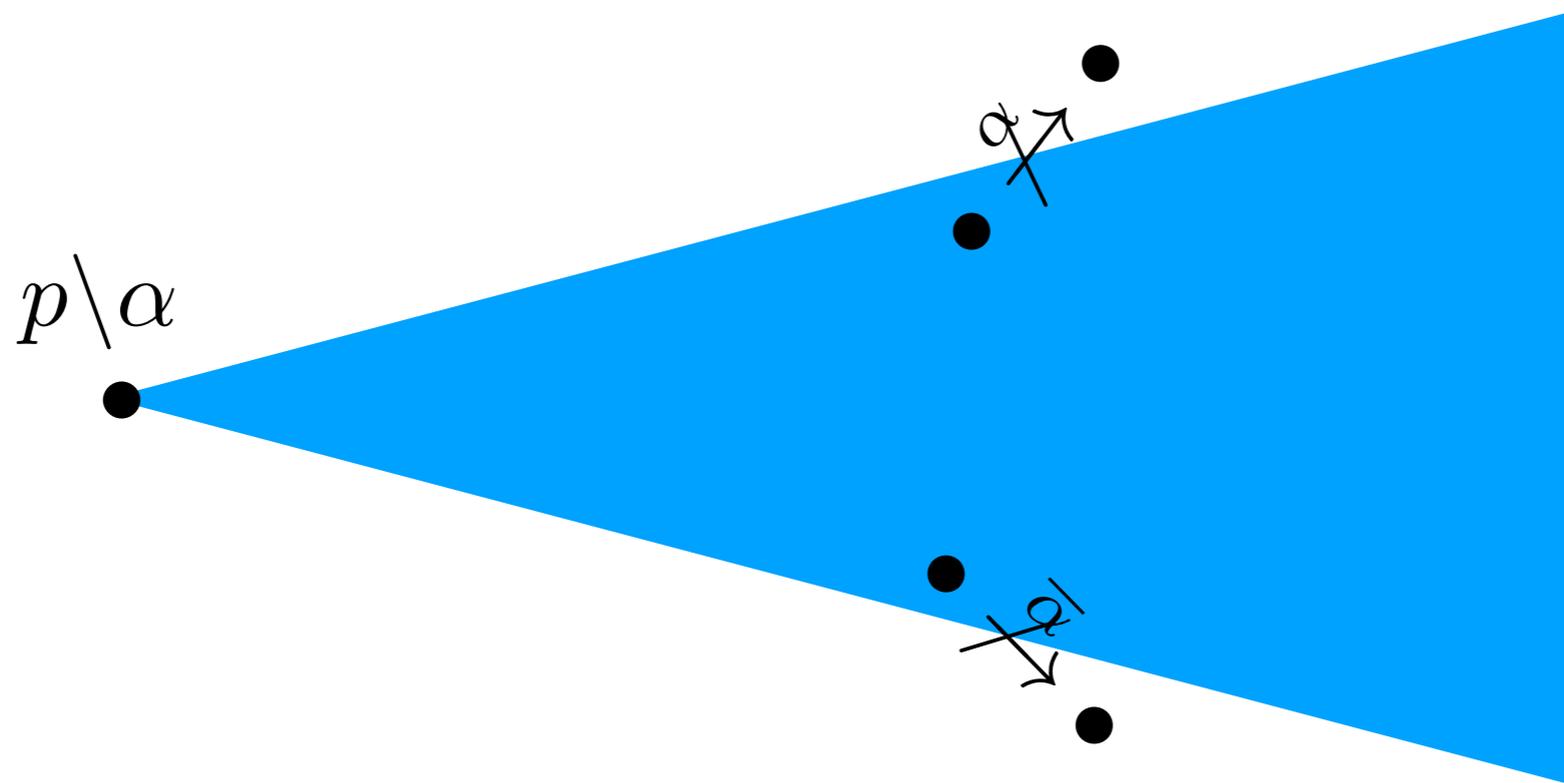
$$S \triangleq \{coin, coffee, tea\}$$

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

# 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  $\phi$

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

ci permette di riusare i processi

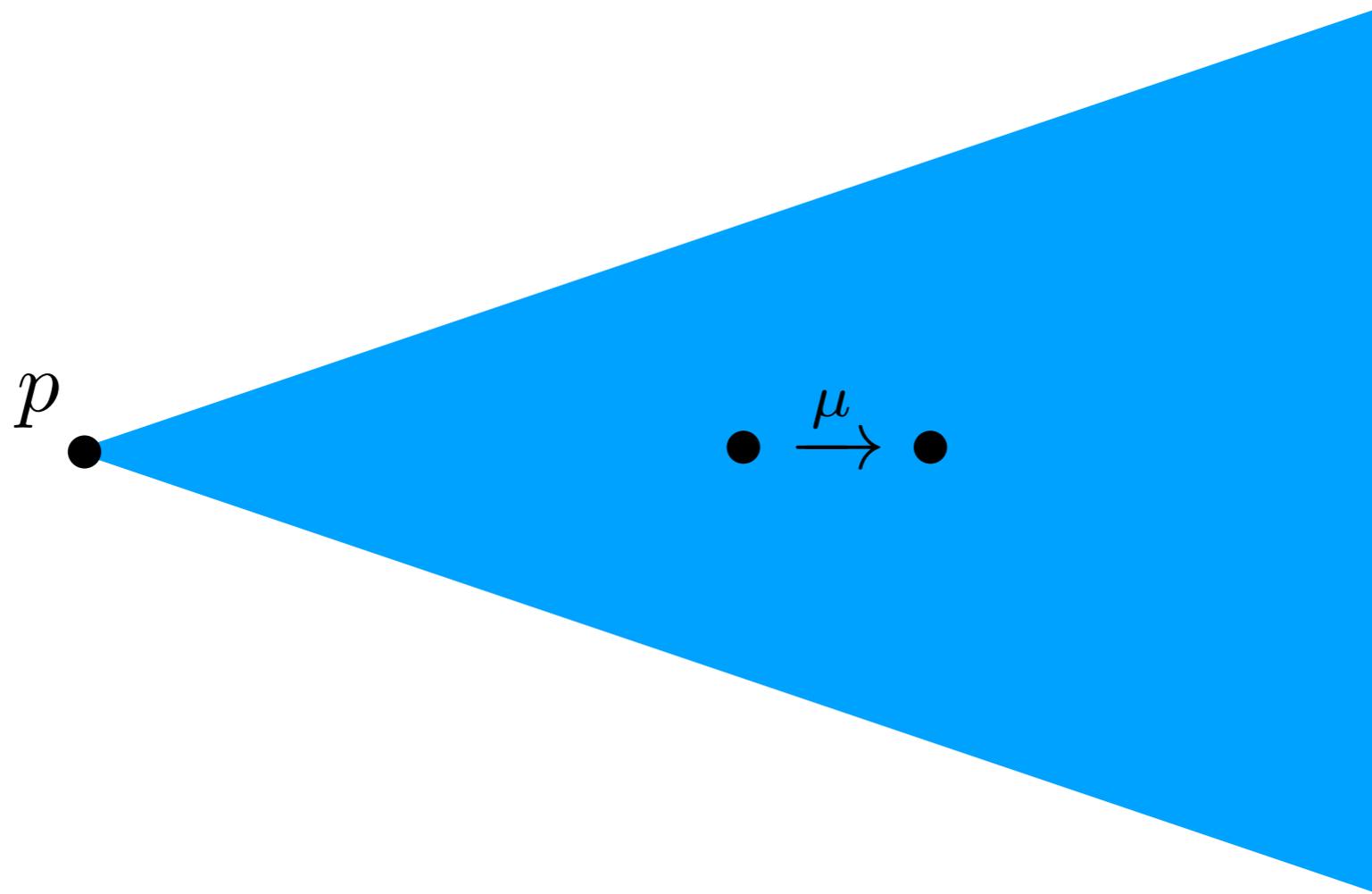
$P \triangleq \overline{coin}.coffee$

$\phi(coin) = moneta$

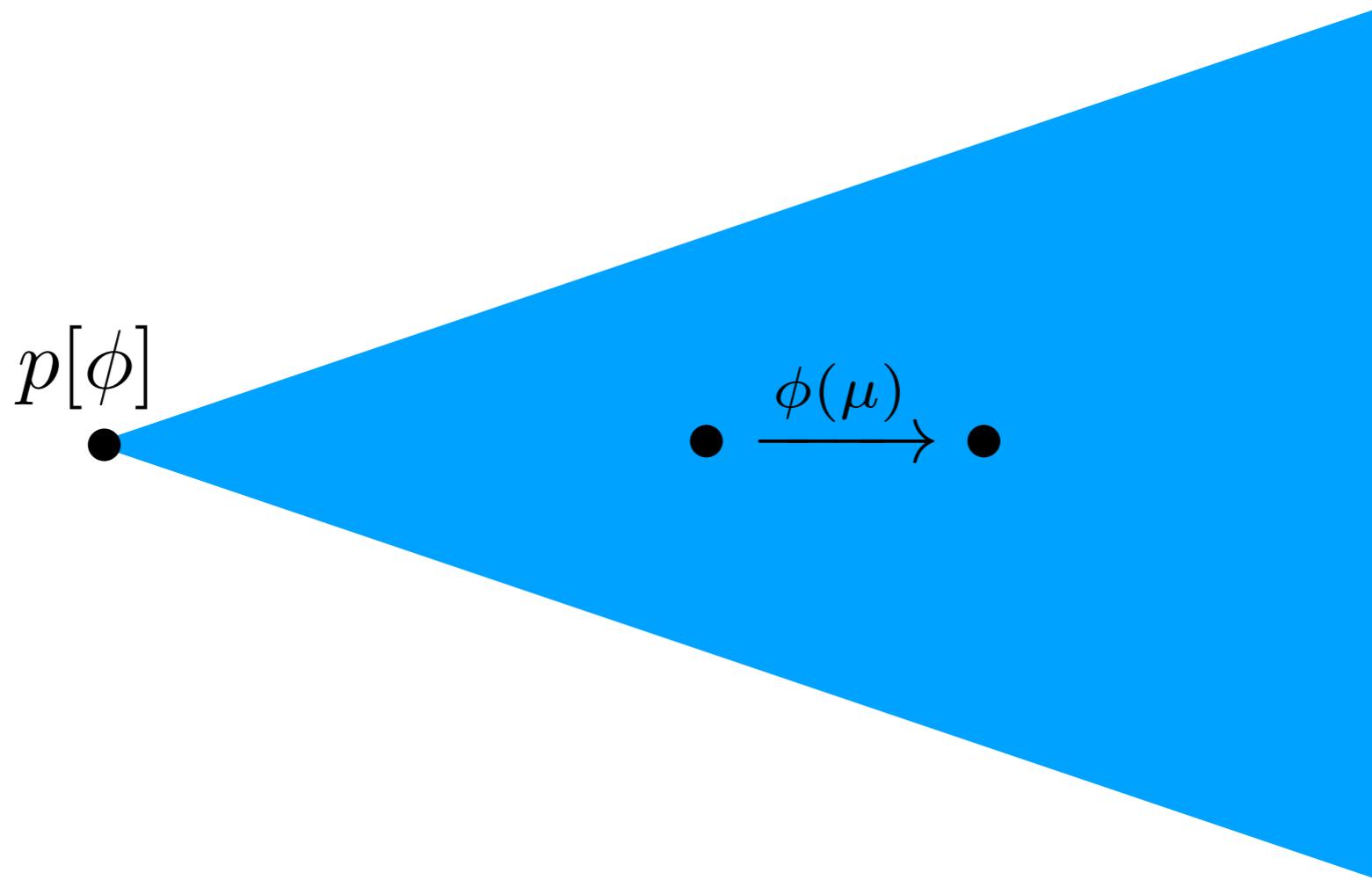
$\phi(coffee) = caffè$

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

# LTS del processo



# LTS del processo



# CCS semantica operativa

$$\text{Act)} \frac{}{\mu.p \xrightarrow{\mu} p} \quad \text{Res)} \frac{p \xrightarrow{\mu} q \quad \mu \notin \{\alpha, \bar{\alpha}\}}{p \setminus \alpha \xrightarrow{\mu} q \setminus \alpha} \quad \text{Rel)} \frac{p \xrightarrow{\mu} q}{p[\phi] \xrightarrow{\phi(\mu)} q[\phi]}$$

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

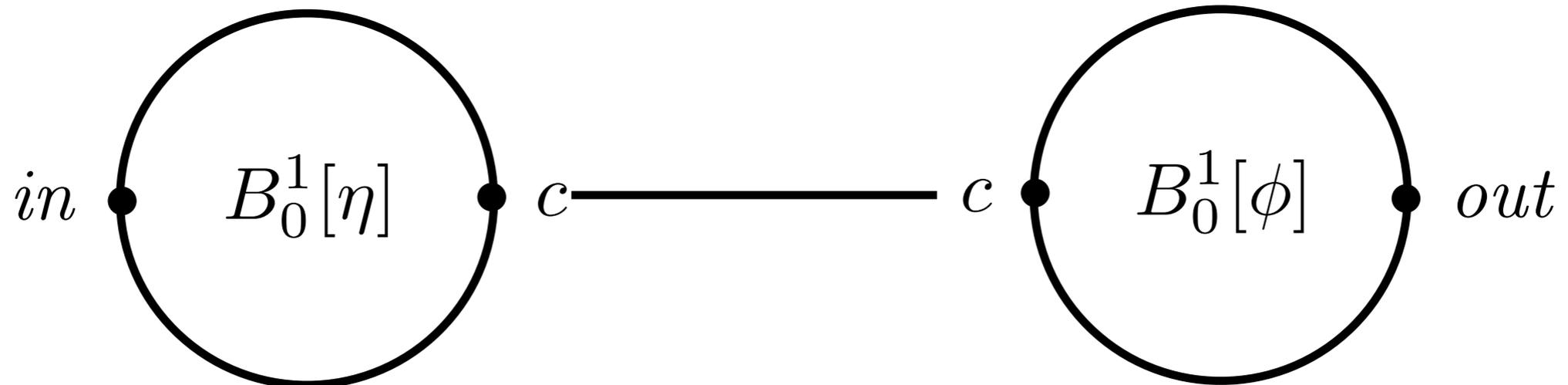
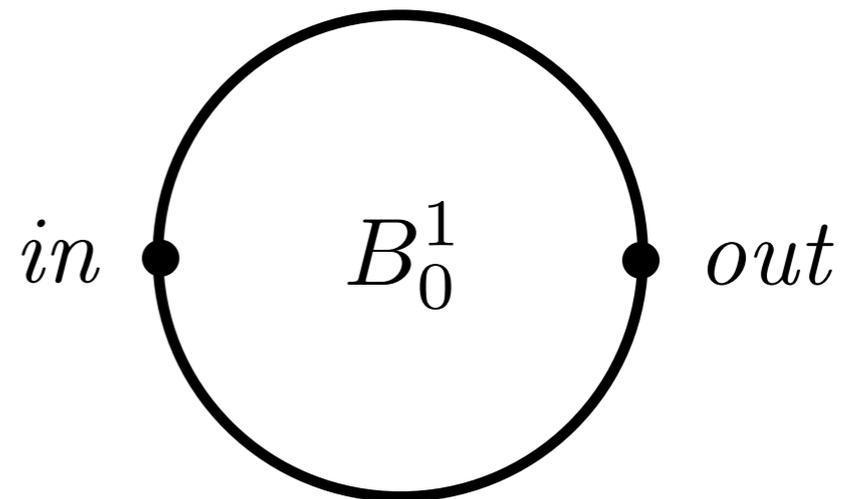
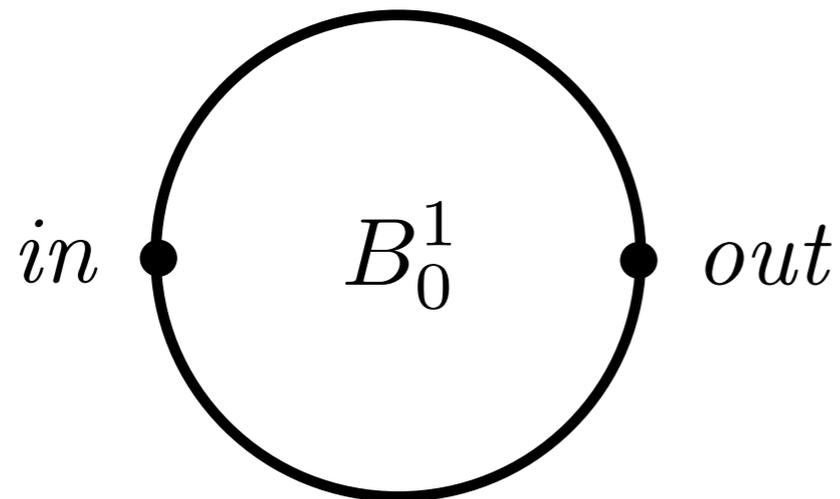
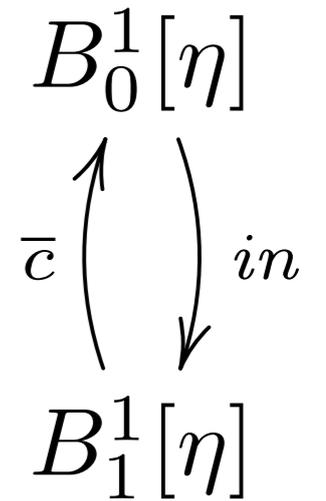
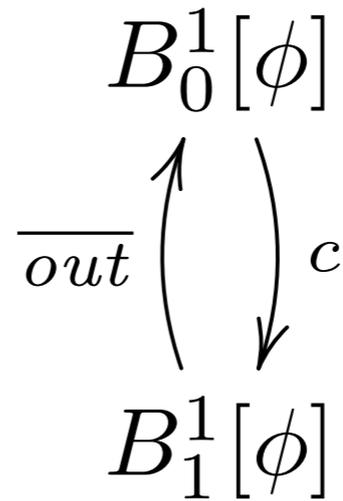
$$\text{ParL)} \frac{p_1 \xrightarrow{\mu} q_1}{p_1 | p_2 \xrightarrow{\mu} q_1 | p_2} \quad \text{Com)} \frac{p_1 \xrightarrow{\lambda} q_1 \quad p_2 \xrightarrow{\bar{\lambda}} q_2}{p_1 | p_2 \xrightarrow{\tau} q_1 | q_2} \quad \text{ParR)} \frac{p_2 \xrightarrow{\mu} q_2}{p_1 | p_2 \xrightarrow{\mu}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

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

# Buffer collegati

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

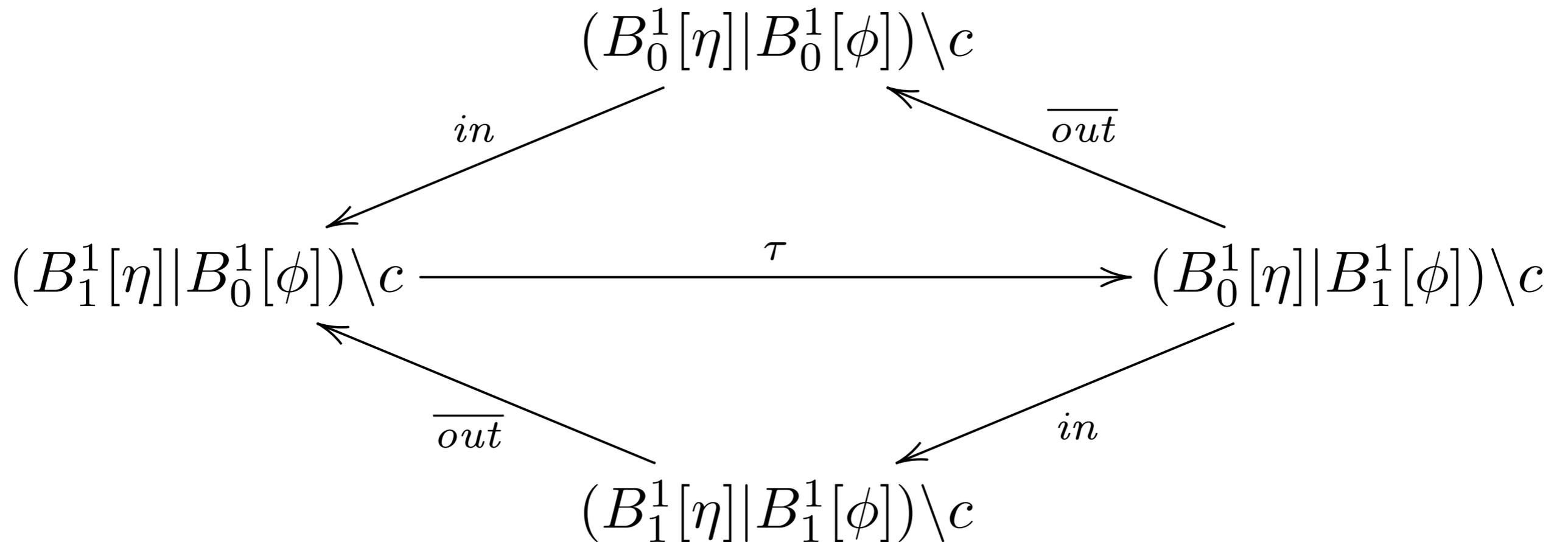
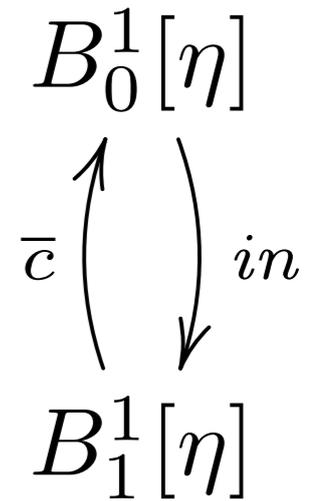
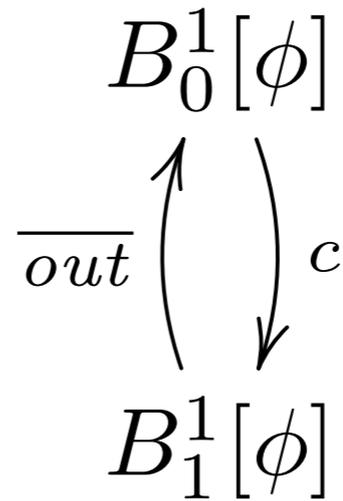
$$B_1^1 \triangleq \overline{out}.B_0^1 \quad \phi(in) = c$$



# Buffer collegati

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

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

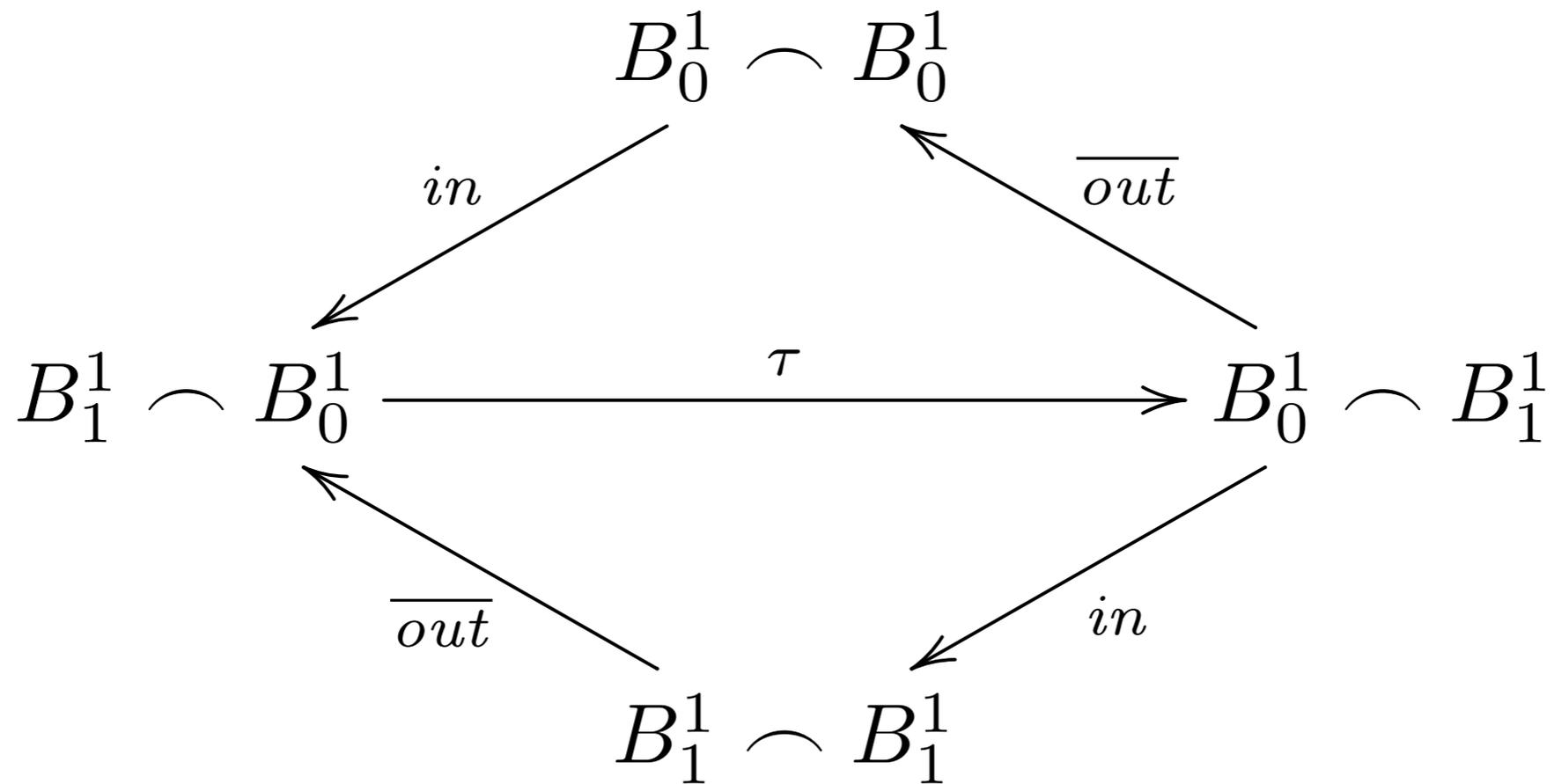


# Buffer collegati

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

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

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



# Buffer collegati booleani

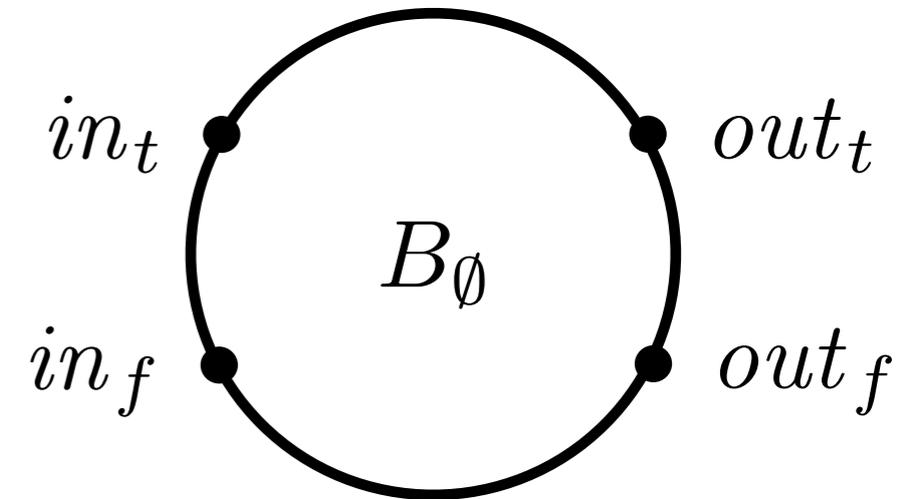
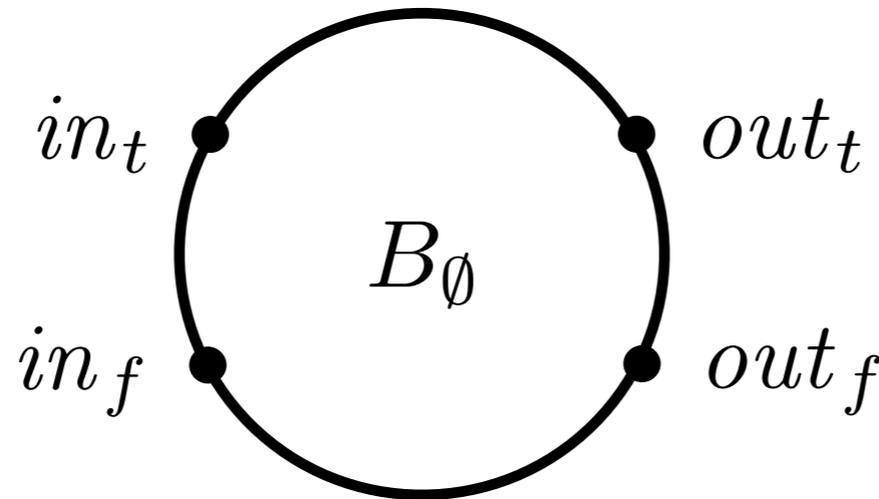
$$\begin{array}{lll} B_{\emptyset} \triangleq in_t.B_t + in_f.B_f & \eta(out_t) = c_t & \phi(in_t) = c_t \\ B_t \triangleq \overline{out_t}.B_{\emptyset} & \eta(out_f) = c_f & \phi(in_f) = c_f \\ B_f \triangleq \overline{out_f}.B_{\emptyset} & p \curvearrowright q \triangleq (p[\eta]|q[\phi]) \setminus \{c_t, c_f\} & \end{array}$$

# Buffer collegati booleani

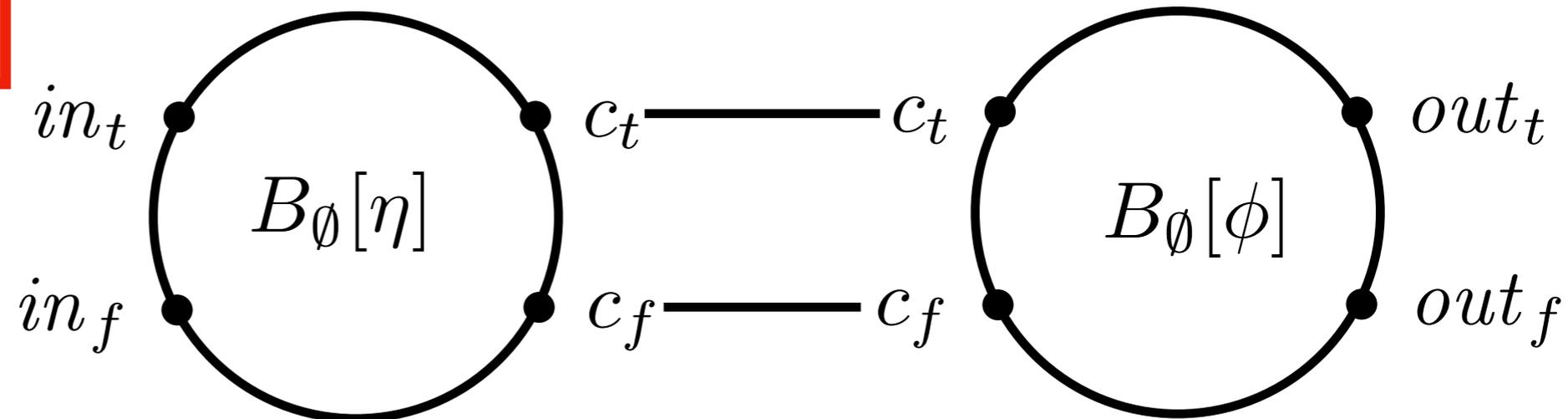
$$B_{\emptyset} \triangleq in_t.B_t + in_f.B_f$$

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

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



$$\eta(out_t) = c_t \quad \phi(in_t) = c_t$$
$$\eta(out_f) = c_f \quad \phi(in_f) = c_f$$

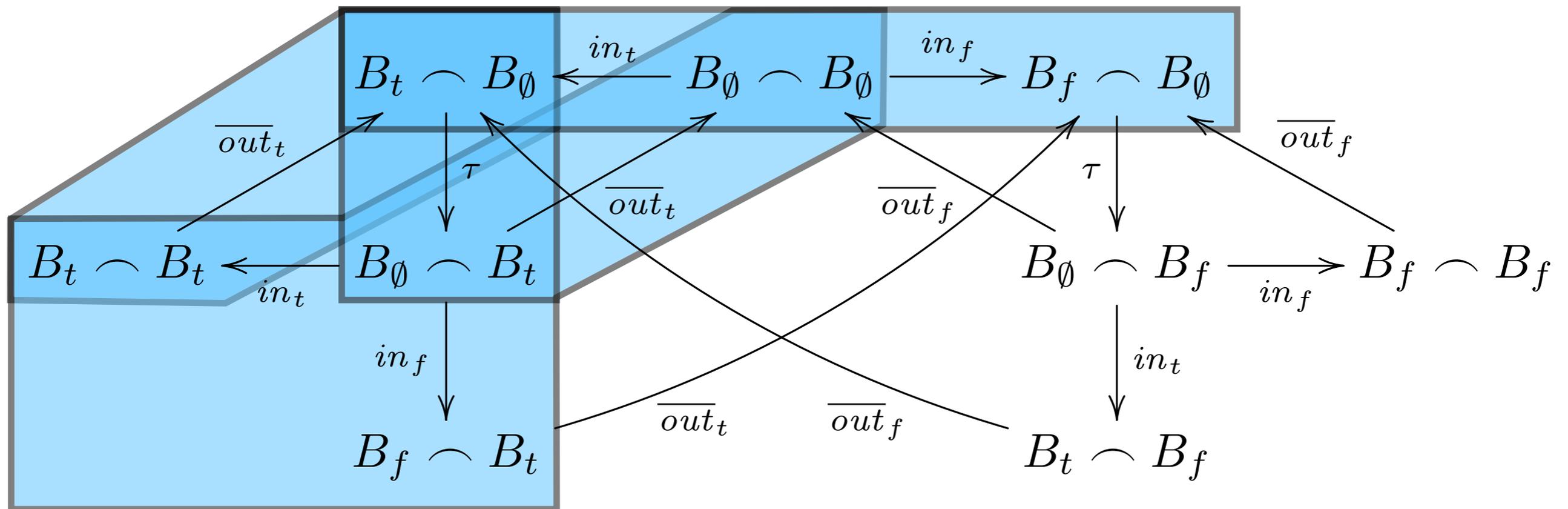


# Buffer collegati booleani

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

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

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

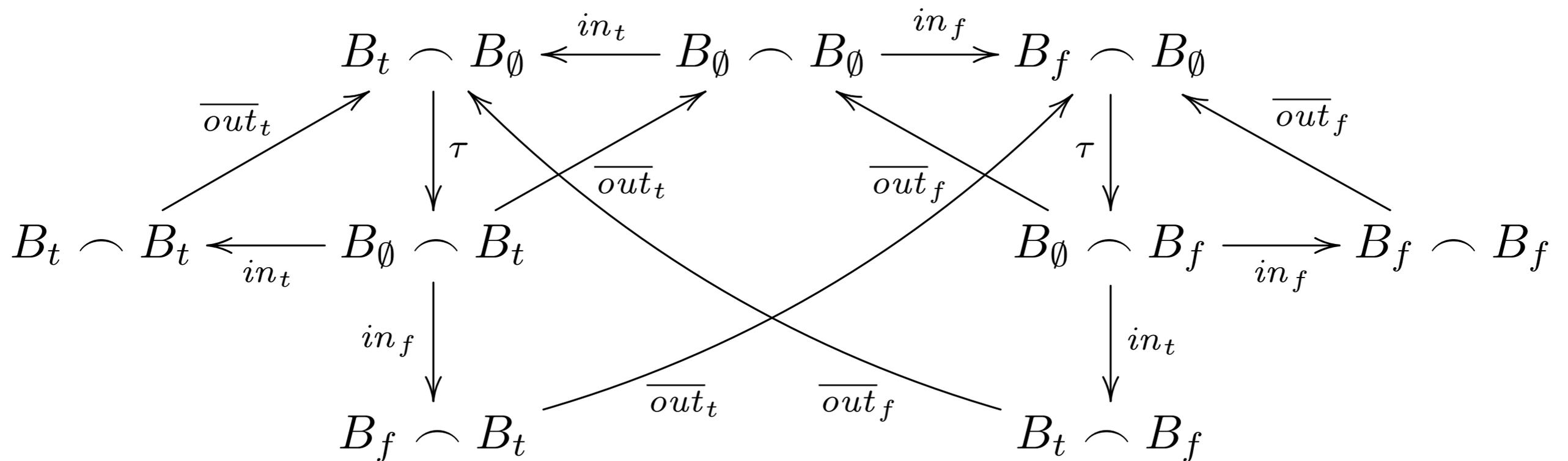


# Buffer collegati booleani

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

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

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



# Sono uguali questi processi?

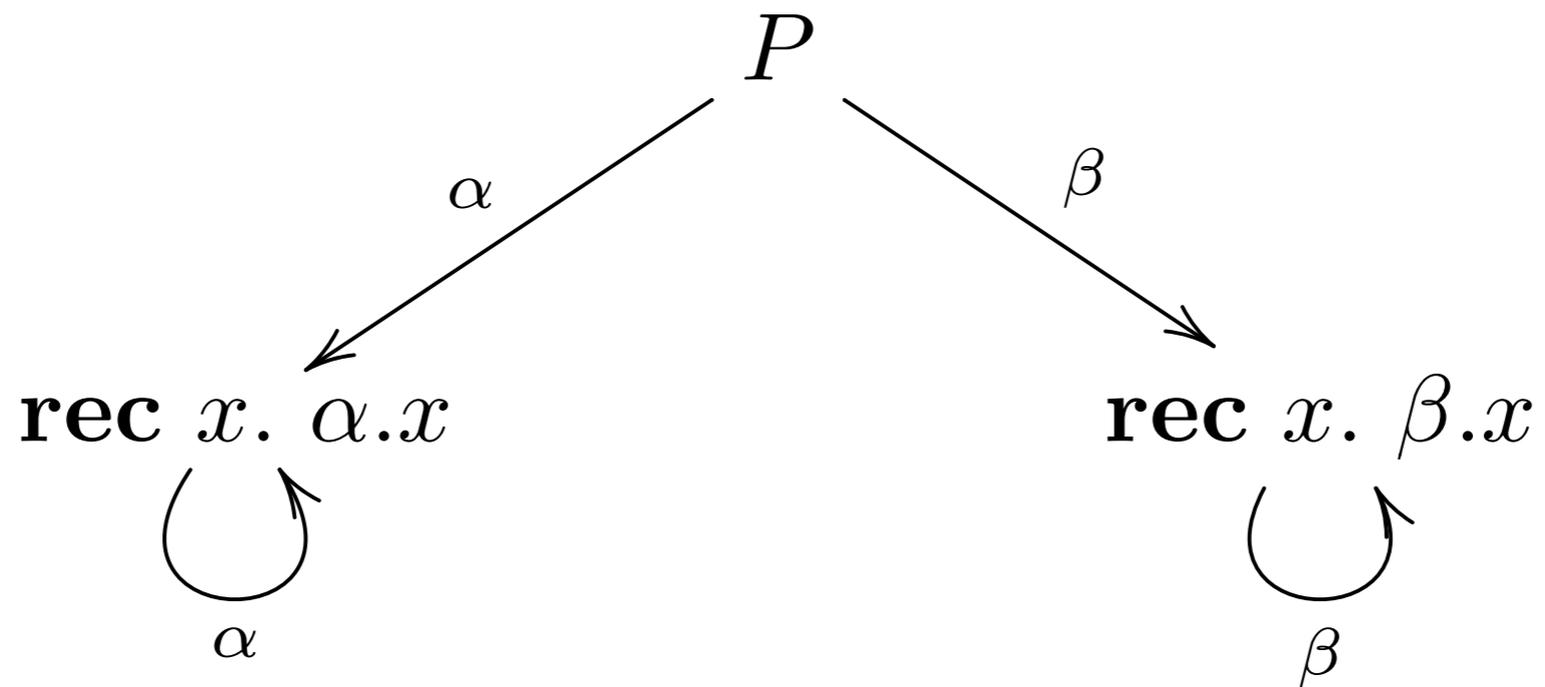
$$P \triangleq (\mathbf{rec} \ x. \ \alpha.x) + (\mathbf{rec} \ x. \ \beta.x)$$

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

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

# Esercizio: LTS?

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



# Esercizio: LTS?

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

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

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

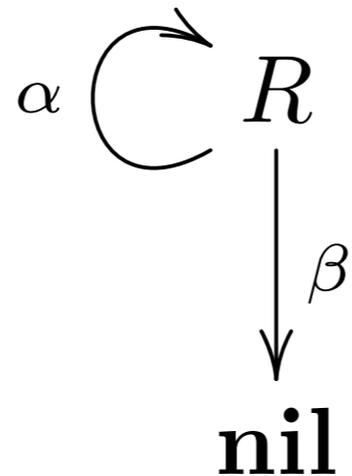


# Esercizio: LTS?

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

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

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



# Somma vs parallelismo

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

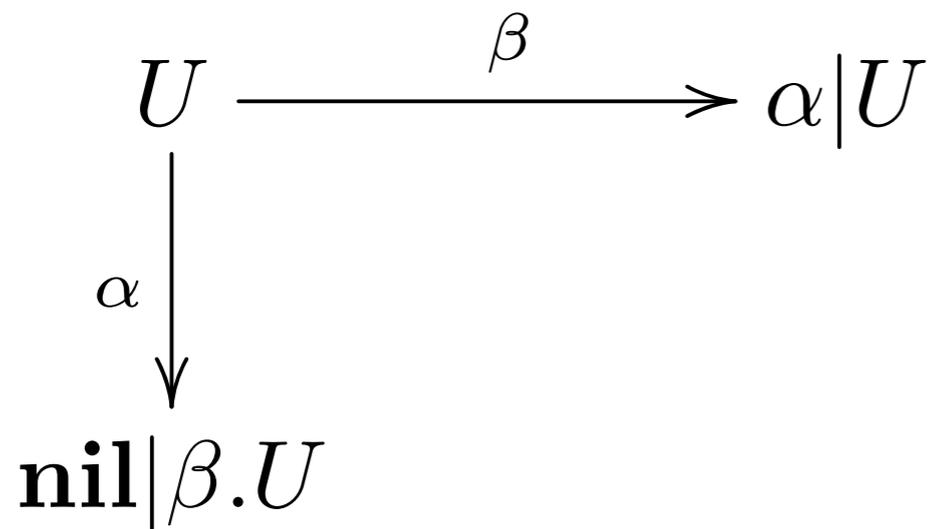
$$U \triangleq \mathbf{rec} \ x. ((\alpha.\mathbf{nil})|\beta.x)$$

# Esercizio: LTS?

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

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

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

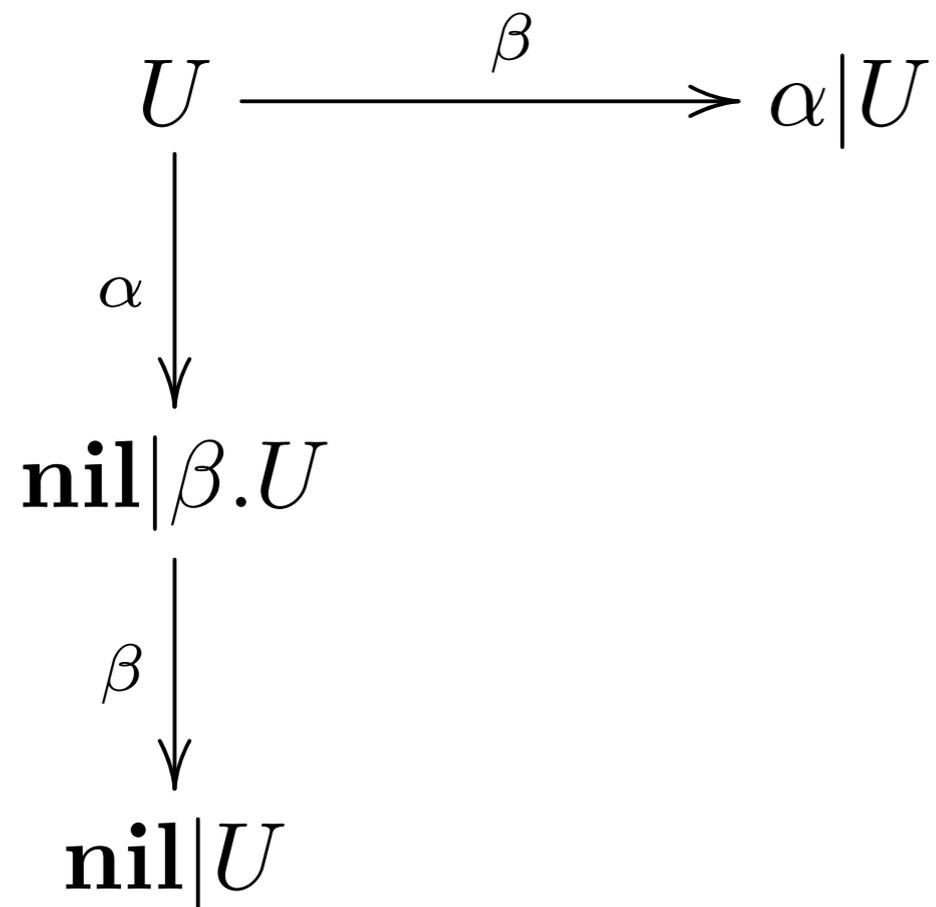


# Esercizio: LTS?

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

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

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

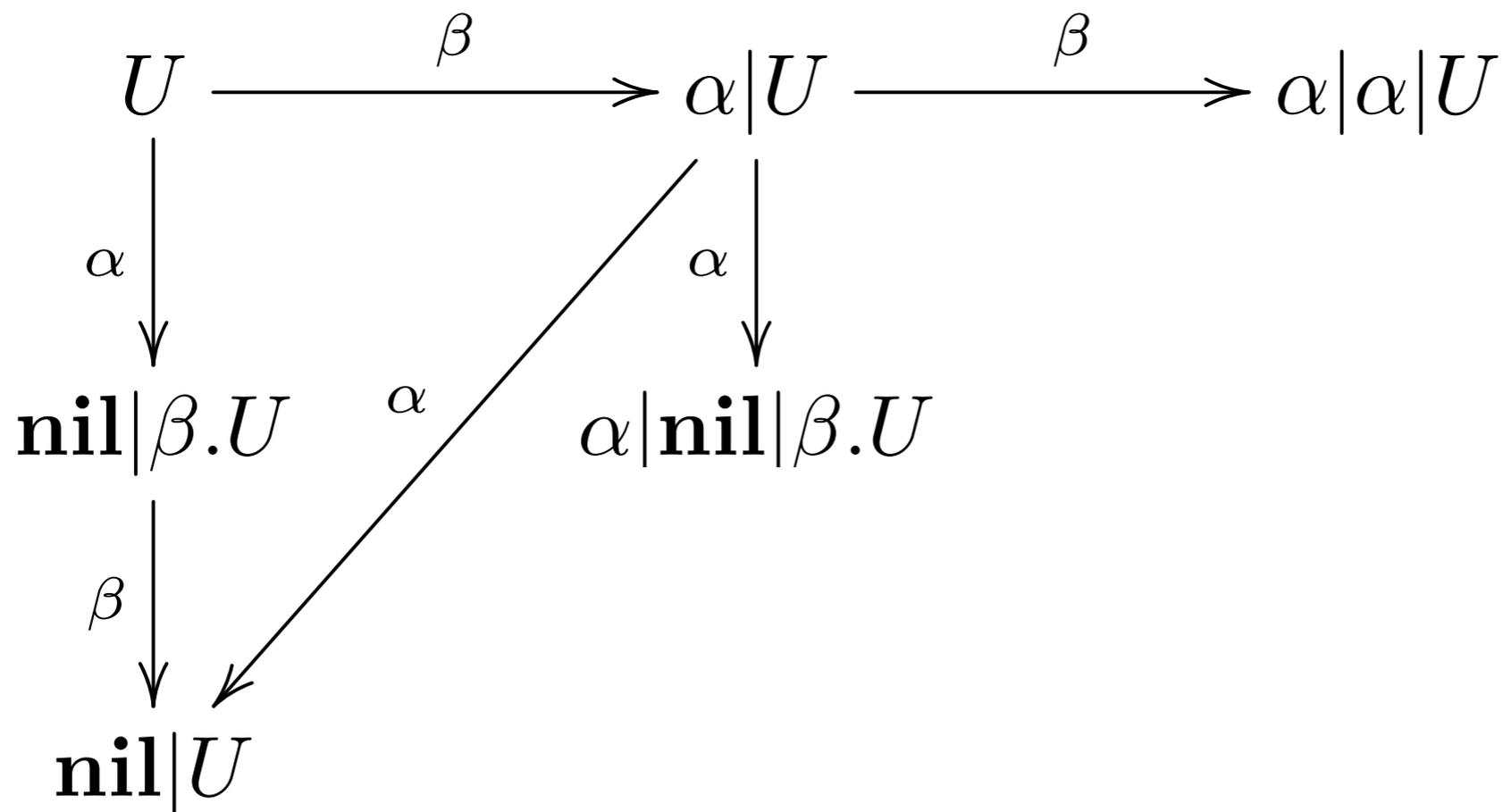


# Esercizio: LTS?

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

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

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

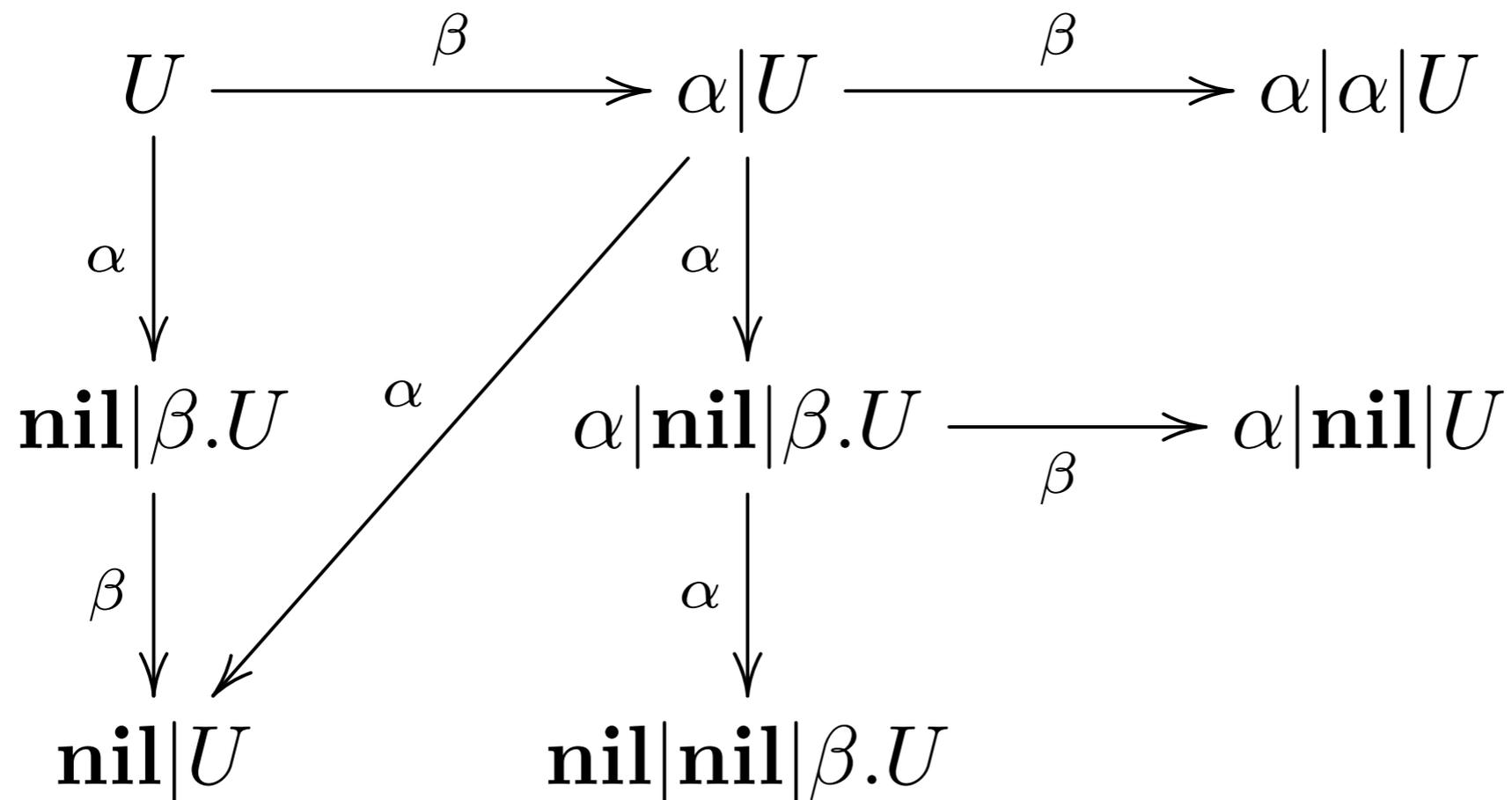


# Esercizio: LTS?

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

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

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

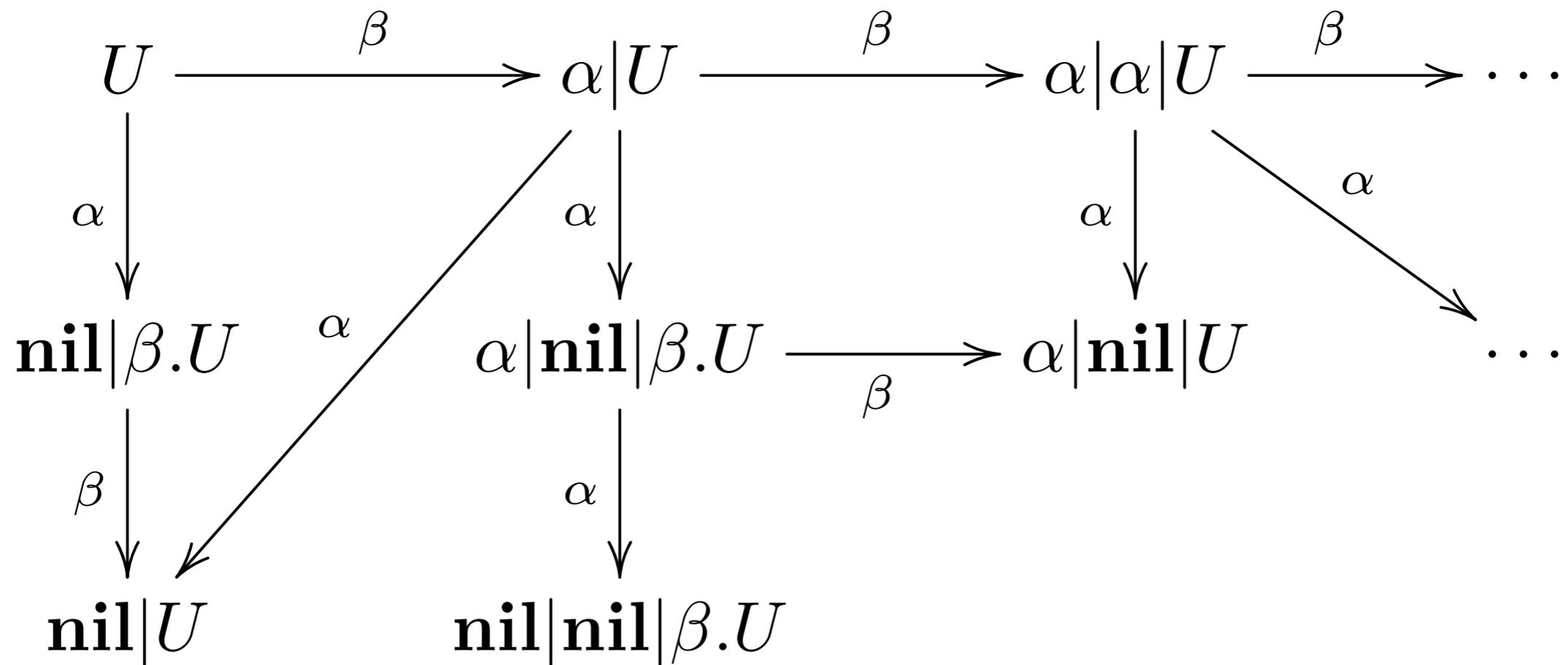


# Esercizio: LTS?

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

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

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

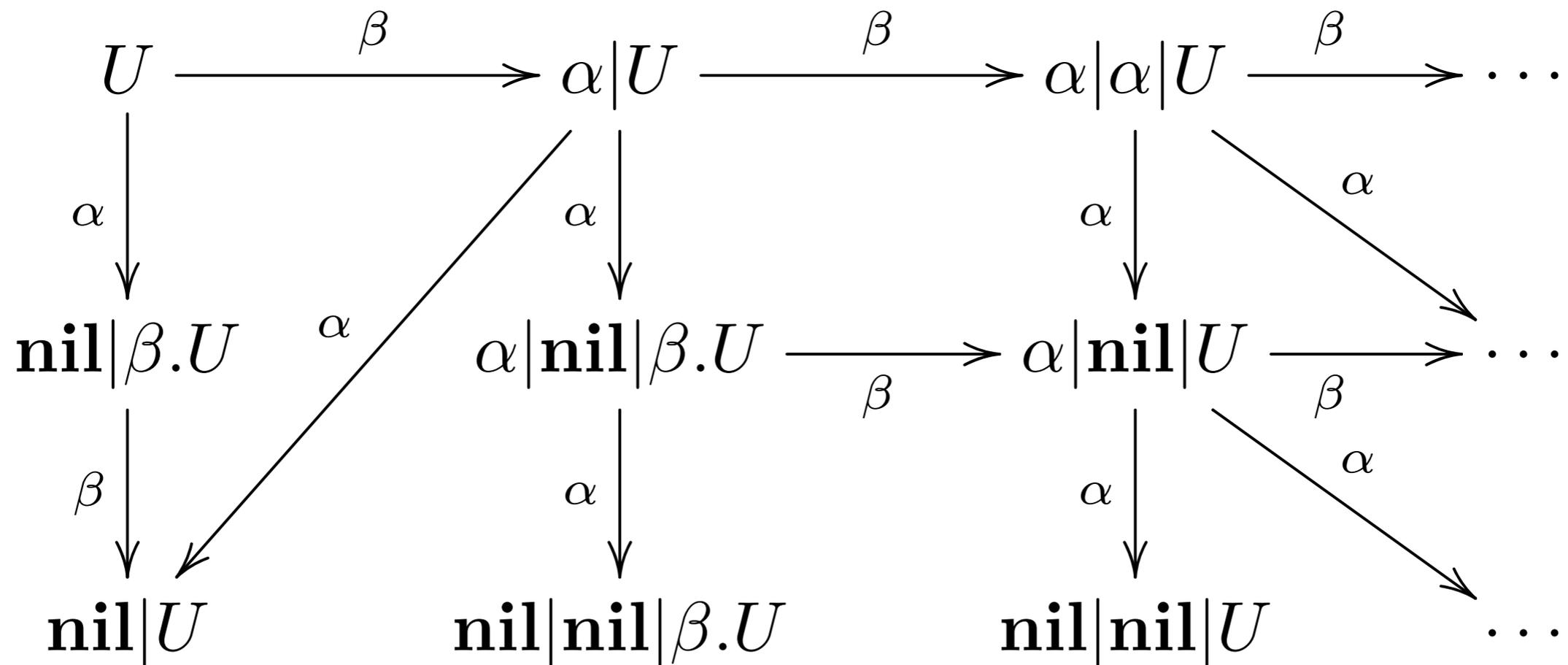


# Esercizio: LTS?

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

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

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



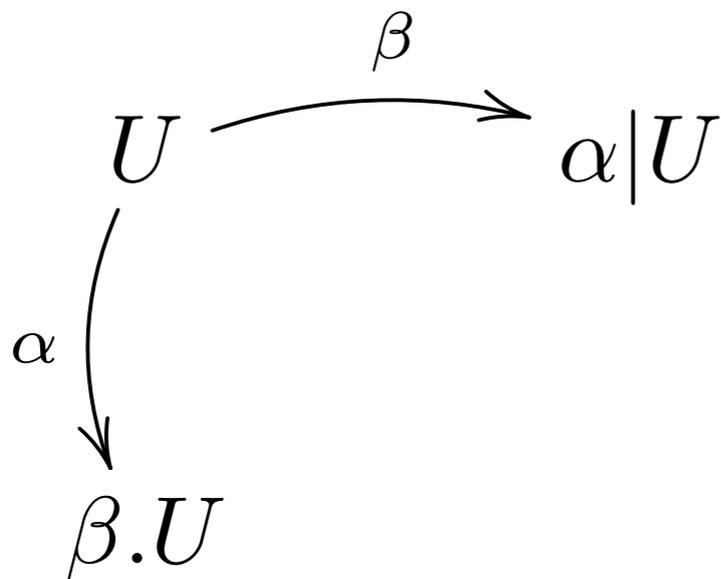
# Esercizio: LTS?

ignoriamo **nil**

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

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

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



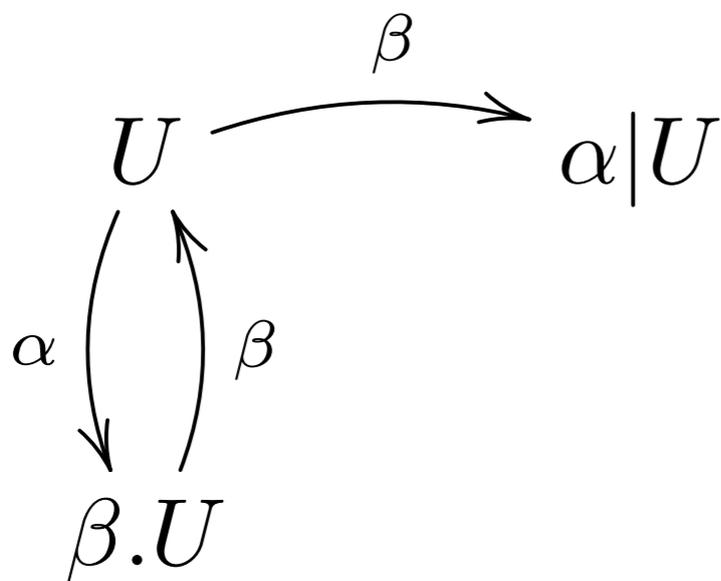
# Esercizio: LTS?

ignoriamo **nil**

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

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

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



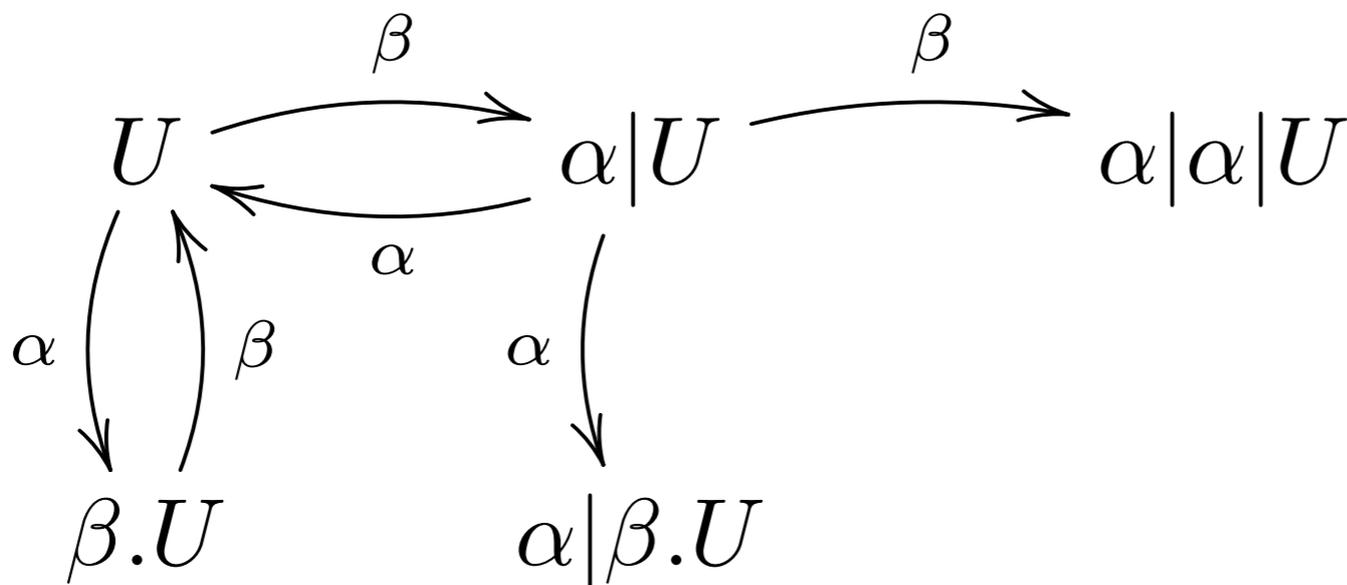
# Esercizio: LTS?

ignoriamo nil

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

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

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



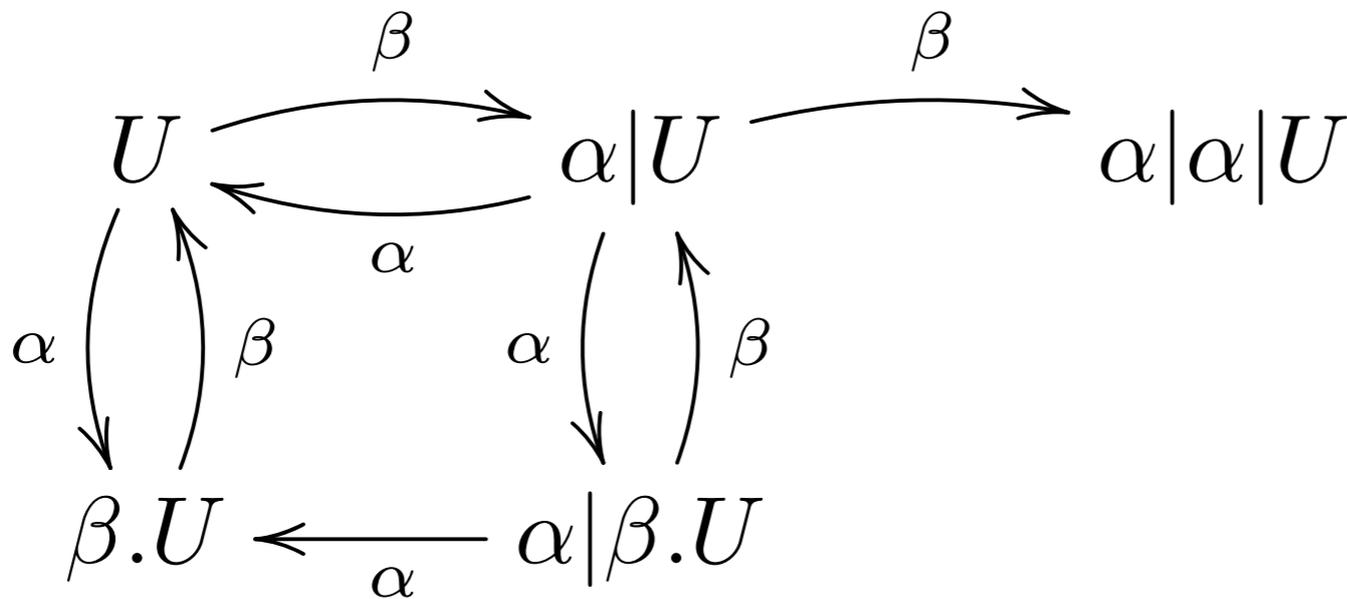
# Esercizio: LTS?

ignoriamo nil

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

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

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



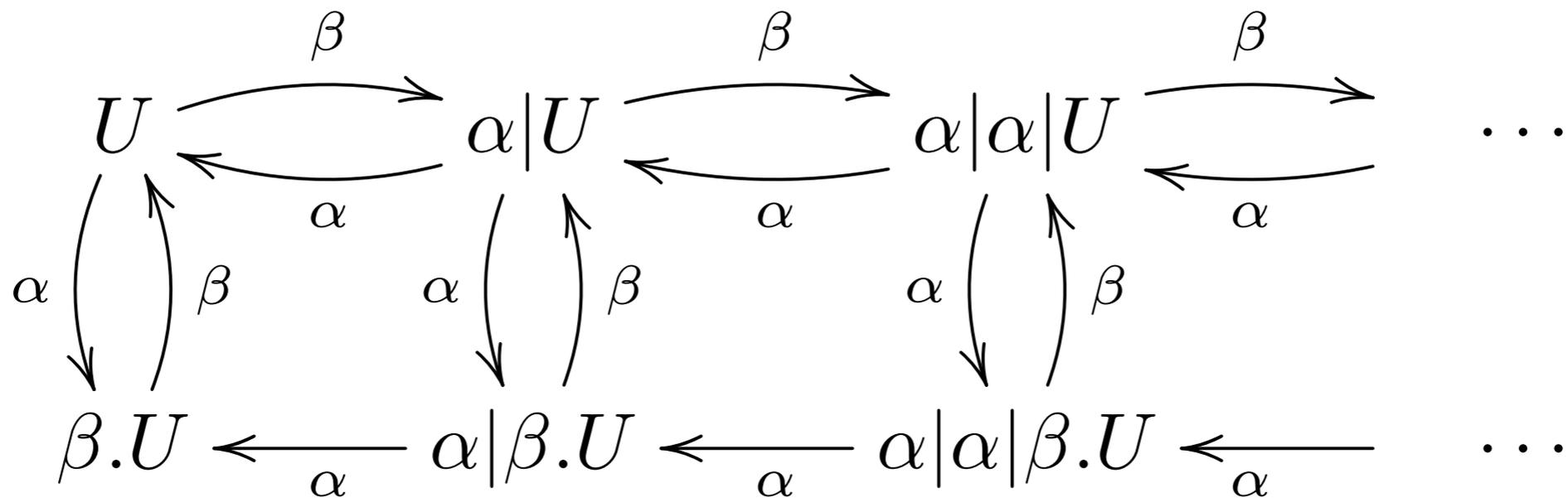
# Esercizio: LTS?

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:

*C0, C1, C2, C3*

usato per visualizzare il valore corrente del contatore

Disegna l'LTS del processo `contatore`.