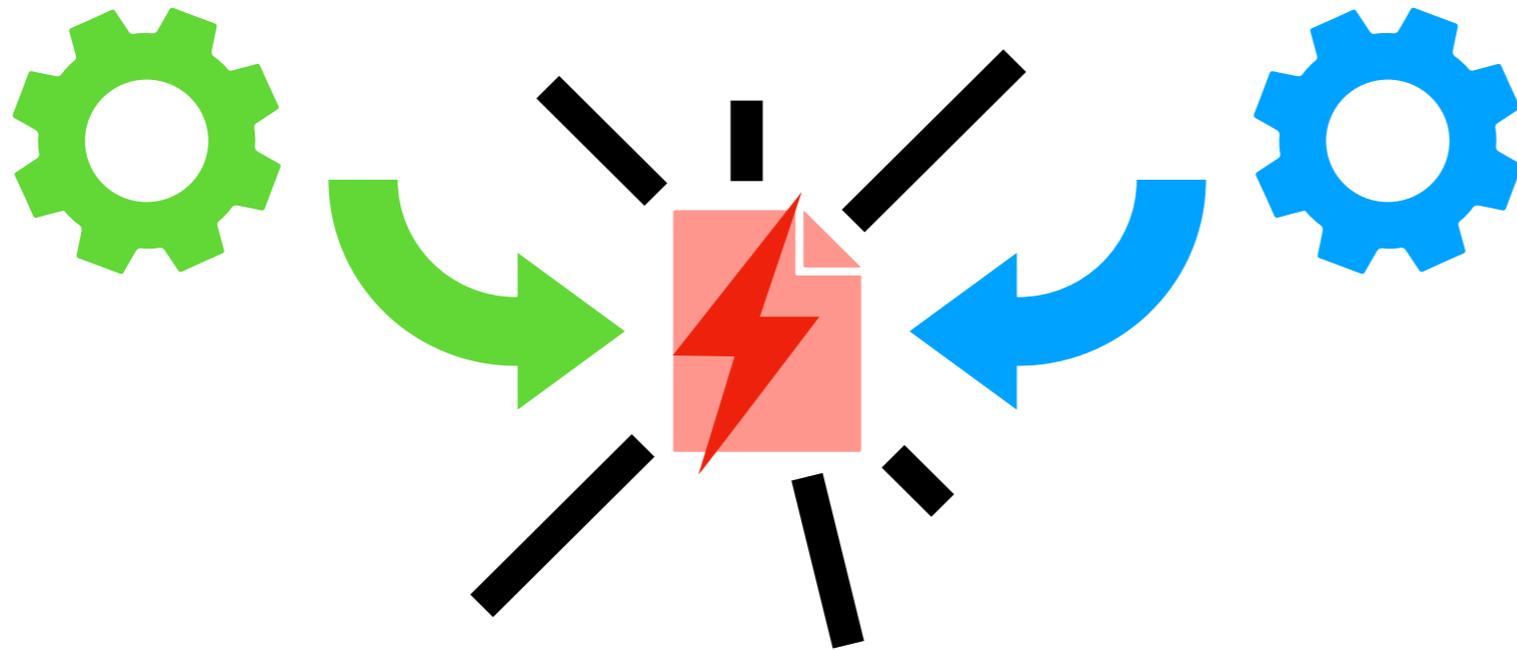


Linguaggi di Programmazione



Roberta Gori

CCS sintassi & semantica operativa-11.1,11.2,11.3

CCS

Calculus of Communicating Systems

Sequenziale vs Concorrente



Esempio: un programma con 2 thread

```
#include <stdio.h>
#include <pthread.h>
#include <unistd.h>

// Funzione eseguita dal primo thread
void* stampa_thread1(void* arg) {
    for (int i = 0; i < 10; i++) {
        printf("Thread 1: messaggio %d\n", i);
        usleep(100000); // pausa di 100ms
    }
    return NULL;
}

// Funzione eseguita dal secondo thread
void* stampa_thread2(void* arg) {
    for (int i = 0; i < 10; i++) {
        printf("    Thread 2: messaggio %d\n", i);
        usleep(100000); // pausa di 100ms
    }
    return NULL;
}

int main() {
    pthread_t t1, t2;

    // Creazione dei thread
    pthread_create(&t1, NULL, stampa_thread1, NULL);
    pthread_create(&t2, NULL, stampa_thread2, NULL);

    // Attesa che entrambi i thread finiscano
    pthread_join(t1, NULL);
    pthread_join(t2, NULL);

    printf("Fine del programma.\n");
    return 0;
}
```

per compilare
gcc main.c -lpthread

Concorrenza

IMP/HOFL (paradigmi sequenziali)

- determinatezza
- due programmi che non terminano sono equivalenti

paradigmi concorrenti

- presentano un non determinismo 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)

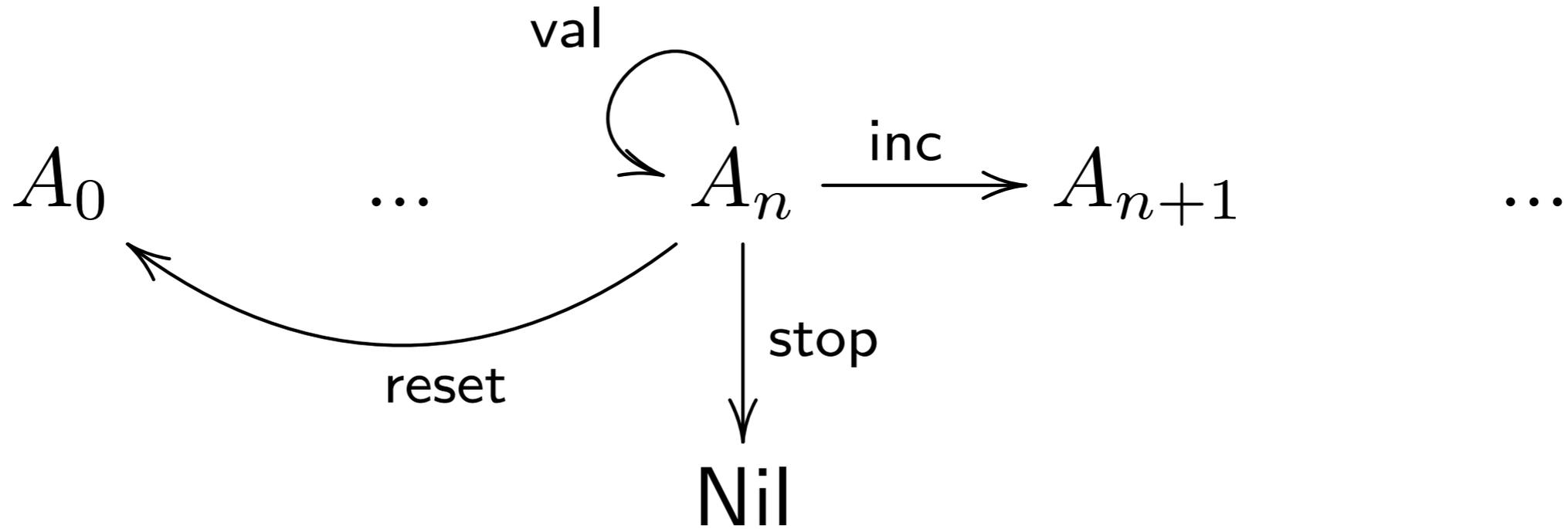
$p \xrightarrow{\mu} q$

un processo
nel suo stato attuale

il processo
dopo l'interazione

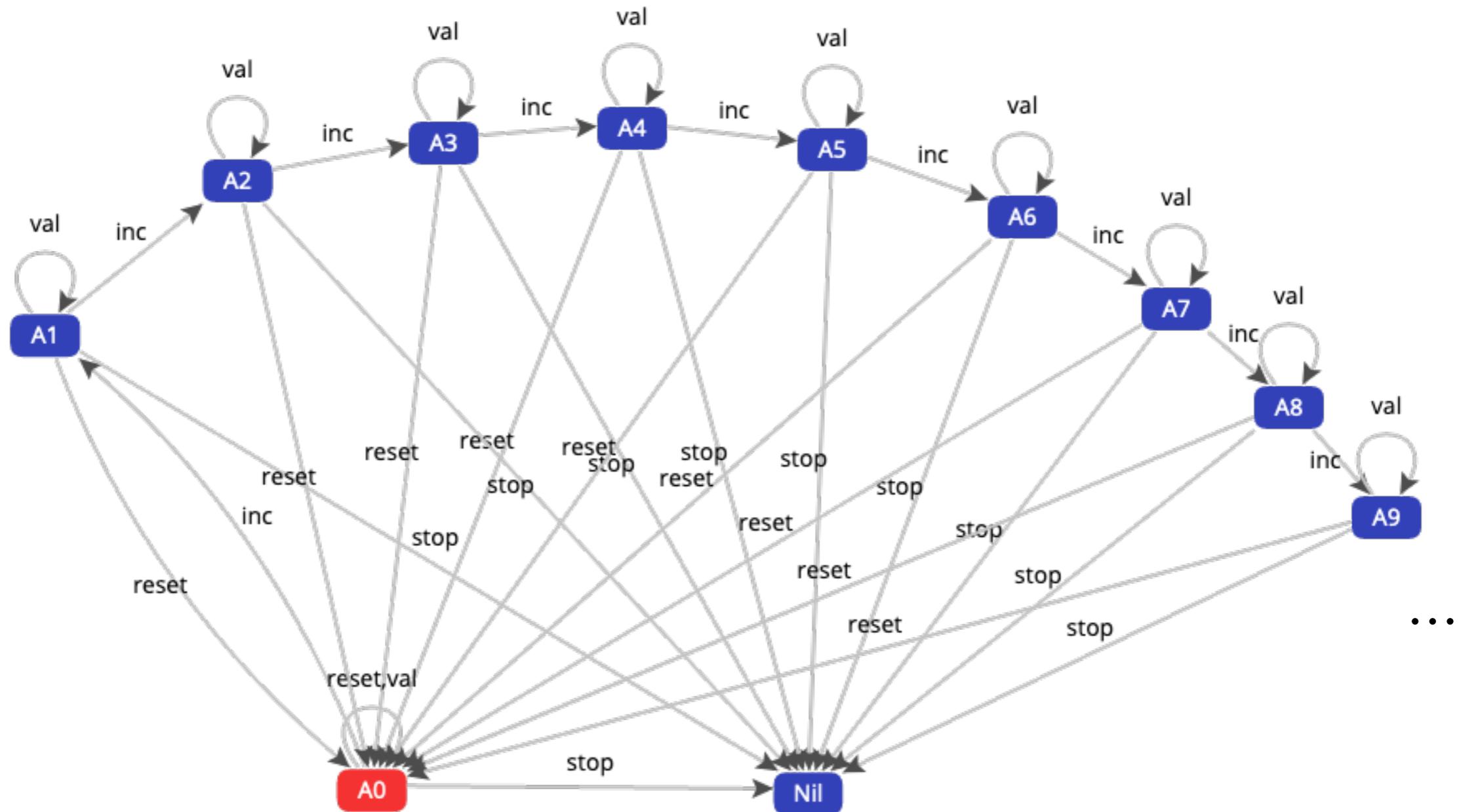
numero di stati/transizioni
può essere infinito

Esempio: contatore



LTS: Labelled Transition System

System



CCS: stati e etichette

Cos'è un **processo** p ?

un agente sequenziale

un **sistema** e' formato da molti agenti sequenziali che interagiscono

Cos'è un'etichetta μ ?

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

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

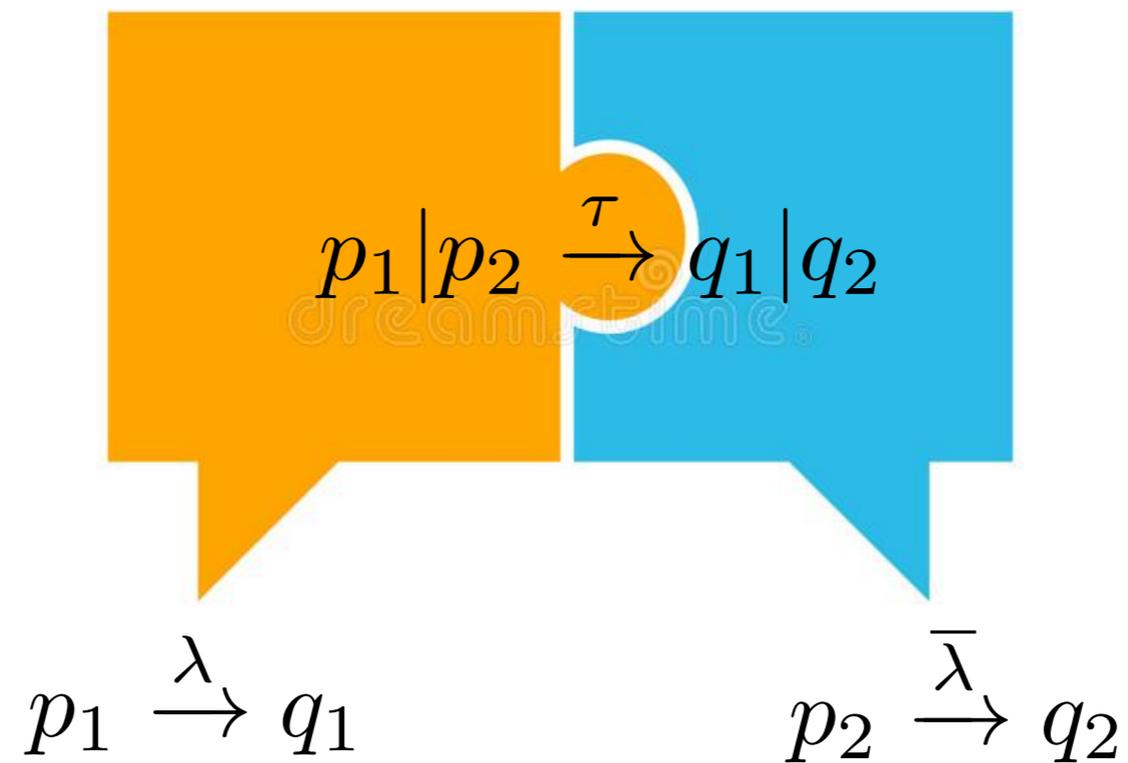
$\alpha!v$ diventa solo $\overline{\alpha}_v$ o solo $\overline{\alpha}$

$\alpha?v$ diventa solo α_v o solo α

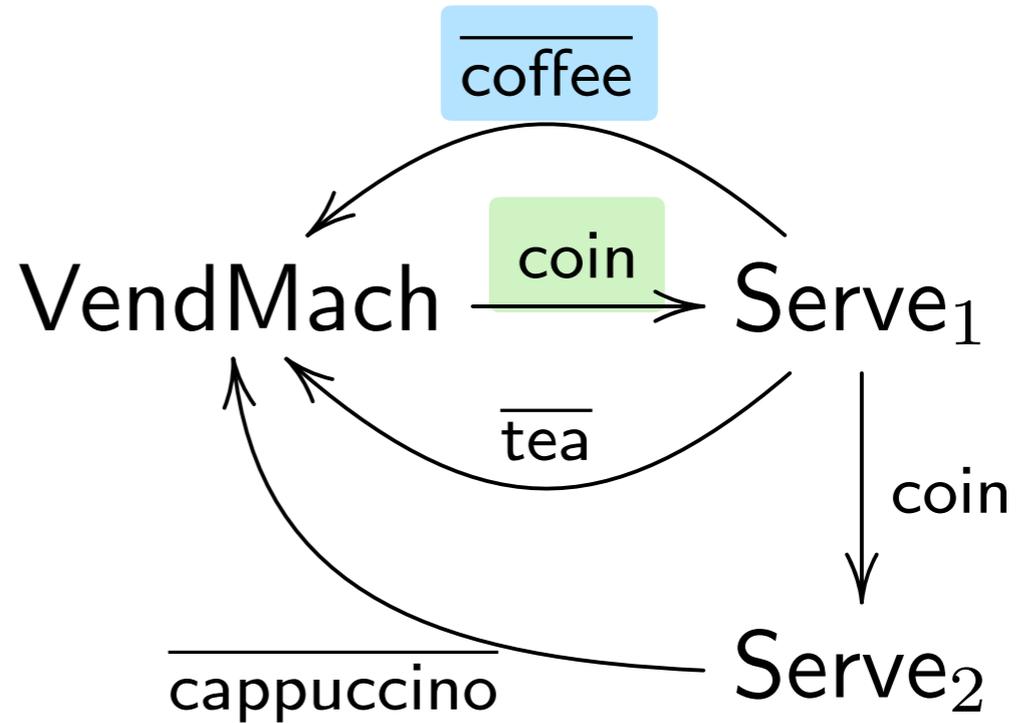
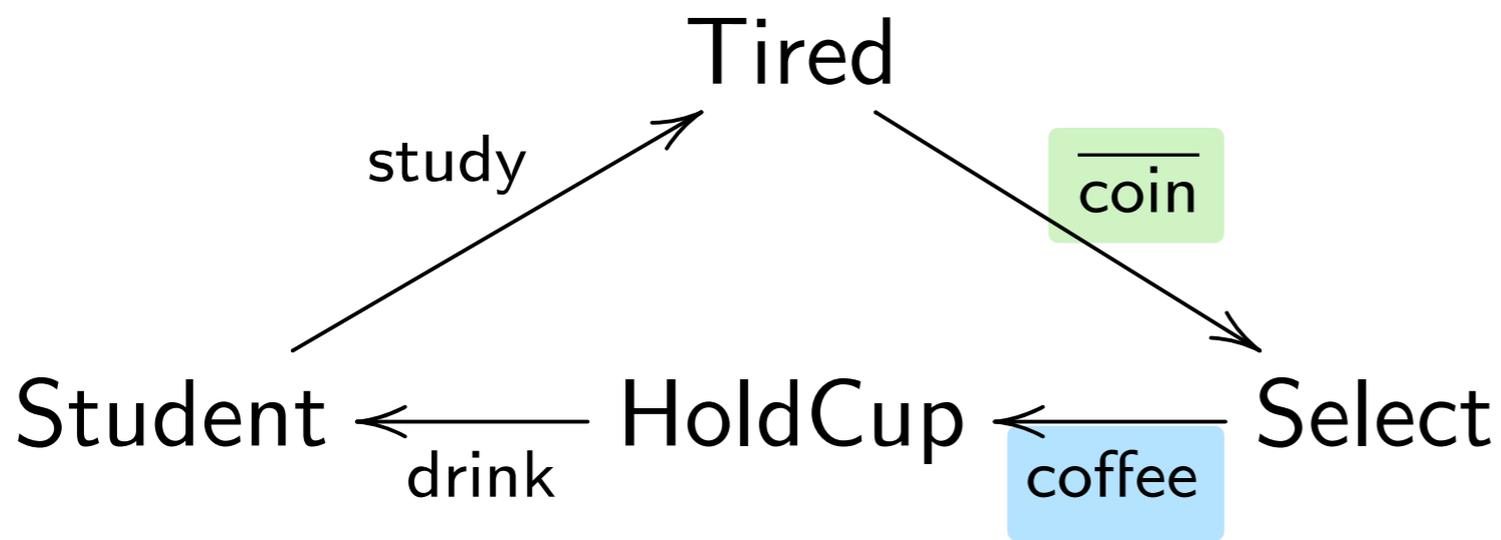
λ denota o α o $\overline{\alpha}$

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

CCS: comunicazione



Esempio: distributore automatico



Sintassi CCS

CCS: sintassi

p, q	$::=$	nil	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
		rec $x. p$	ricorsione

(gli operatori sono elencati in ordine di precedenza)

CCS: sintassi

$p, q ::=$

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

(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

$$fv(p)$$

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

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

semantica operativa del CCS

CCS: etichette

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

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

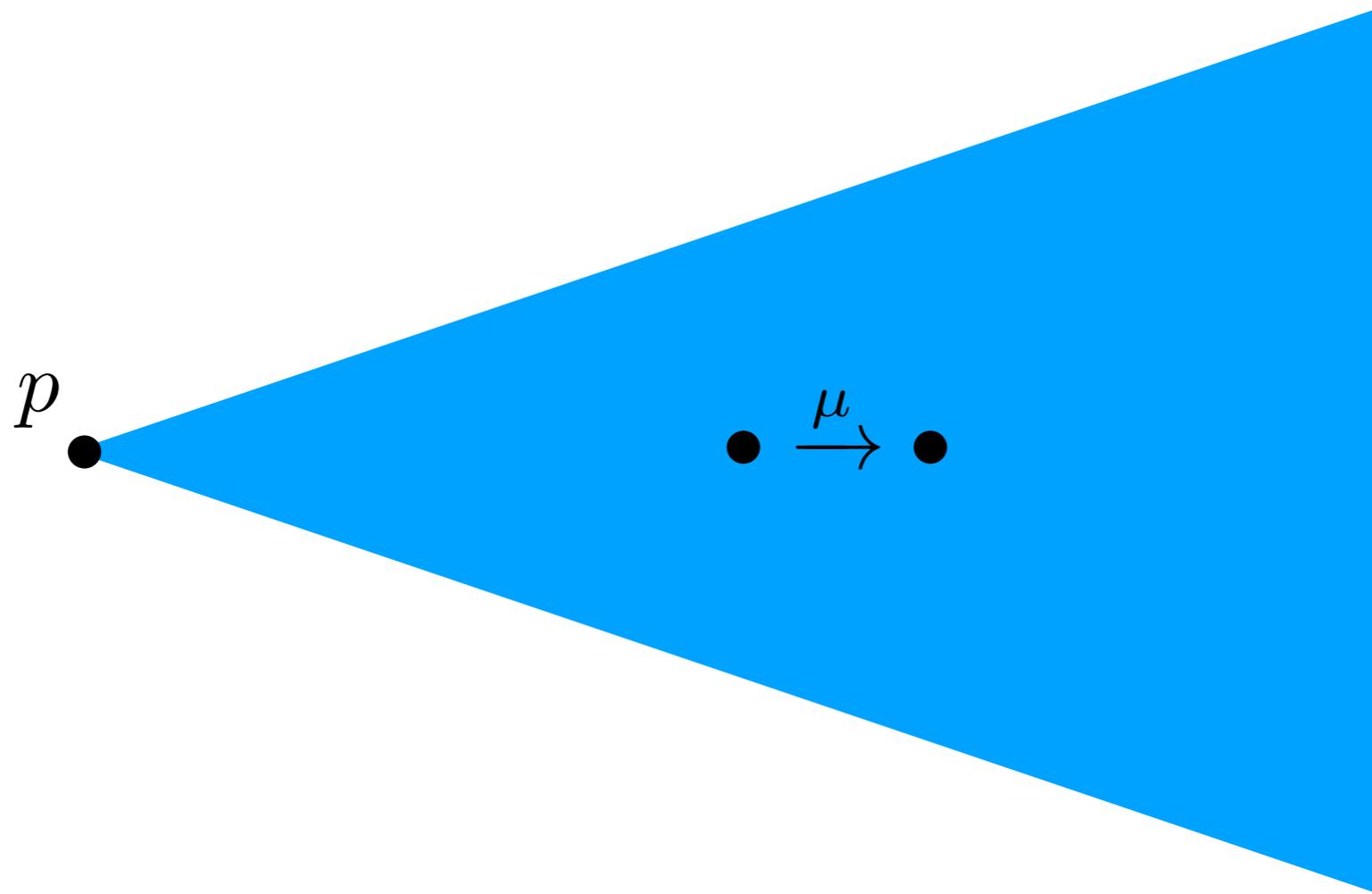
$\Lambda = \mathcal{C} \cup \bar{\mathcal{C}}$ insieme di azioni osservabili, denotate da λ $\bar{\lambda}$

$\tau \notin \Lambda$ un'azione silenziosa (dedicata)

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

LTS di un processo

L'LTS di p può essere 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

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

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

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

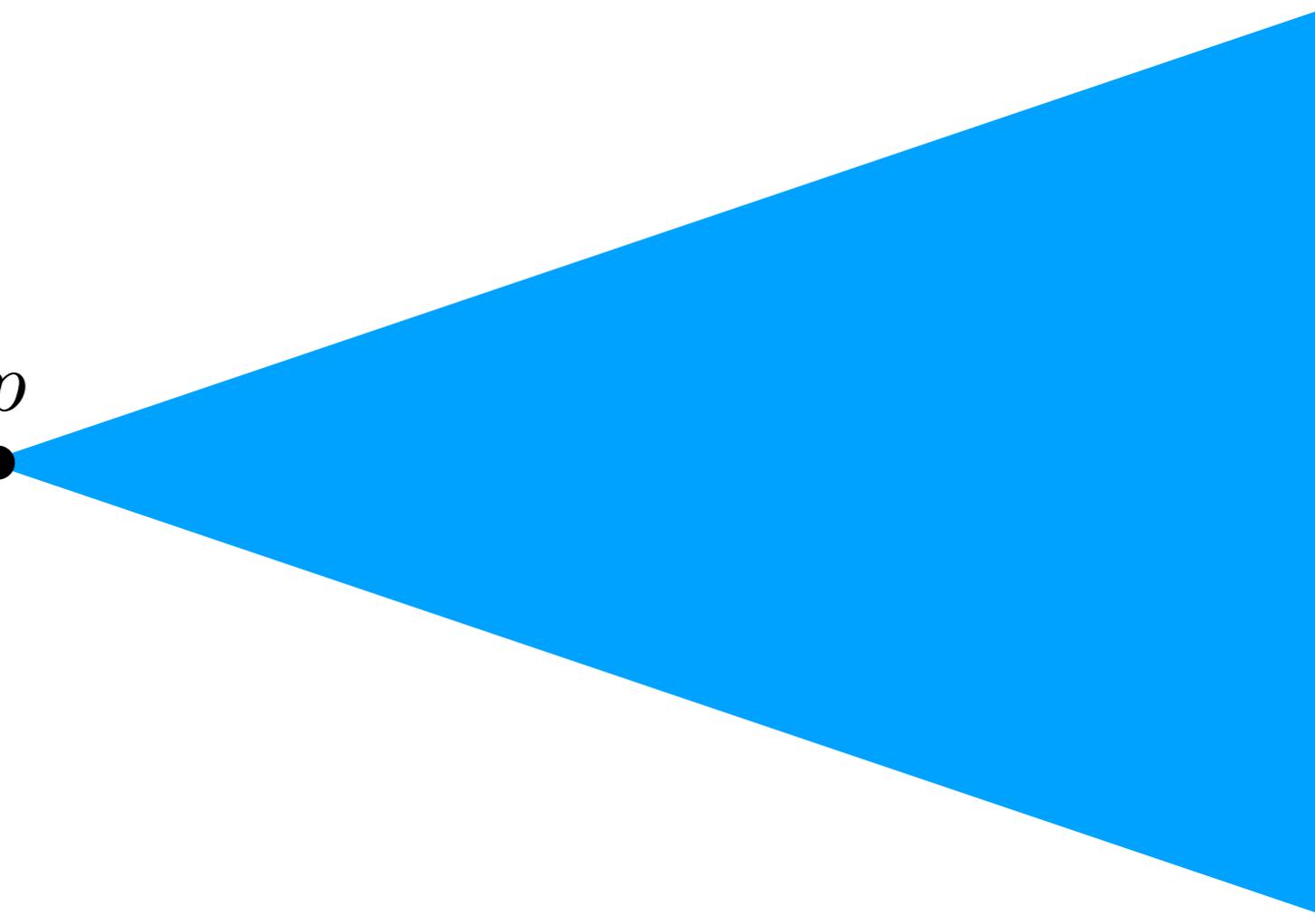
$$\textit{coin}.\overline{\textit{coffe}}.\mathbf{nil}$$

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

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

LTS del processo

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



Scelta non deterministica

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

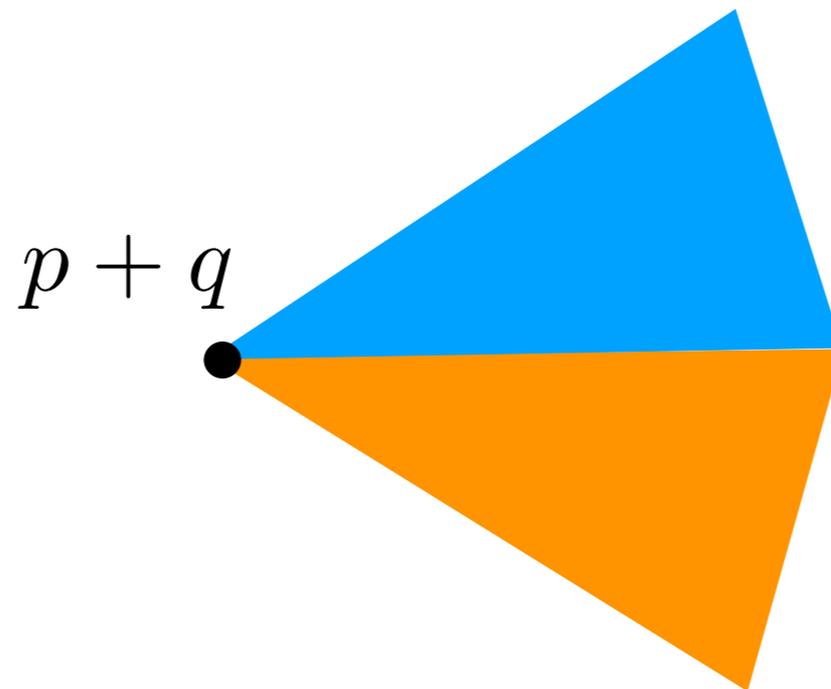
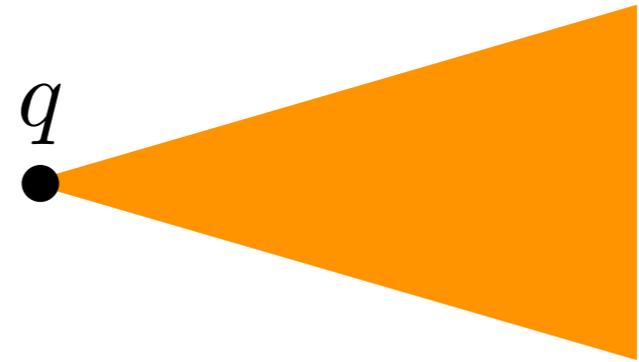
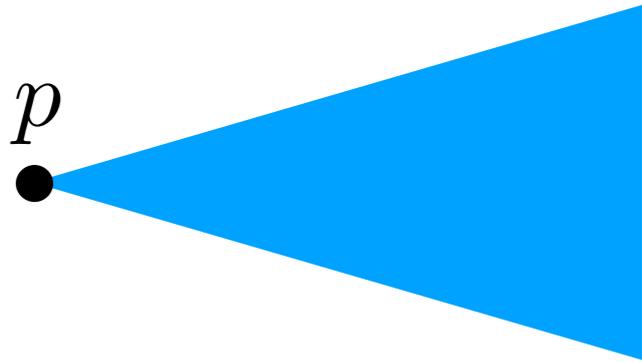
Il processo $p_1 + p_2$ può comportarsi come p_1 o come p_2

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

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

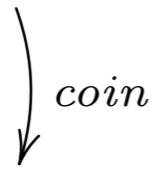
$$\text{Rec)} \frac{p[\mathbf{rec} \ x. \ p / x] \xrightarrow{\mu} q}{\mathbf{rec} \ x. \ p \xrightarrow{\mu} q}$$

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

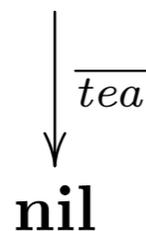
$$\mathbf{rec} \ x. \ \text{coin}.\overline{(\text{coffee}.x + \text{tea.nil})}$$

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

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

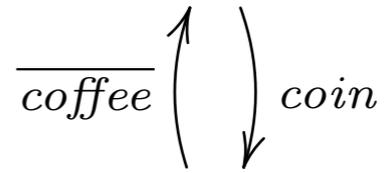


$$\overline{\text{coffee}.\mathbf{rec} \ x. \ \text{coin}.\overline{(\text{coffee}.x + \text{tea.nil})} + \text{tea.nil}}$$

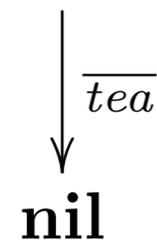


nil

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



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



nil

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

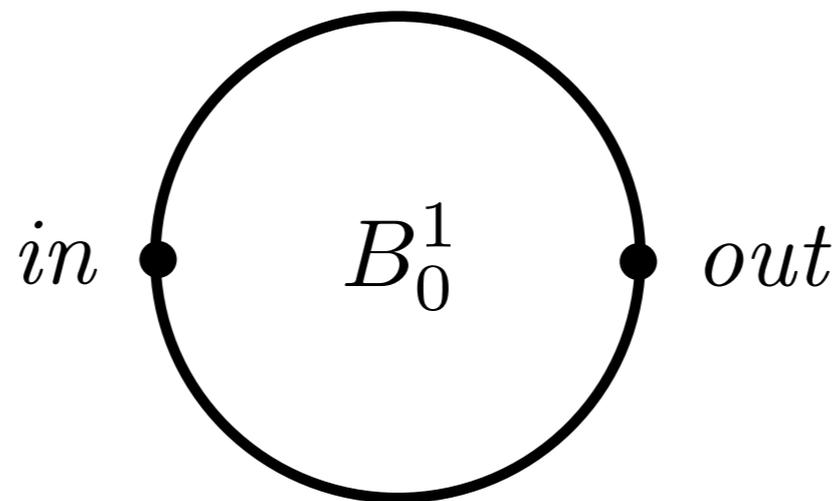
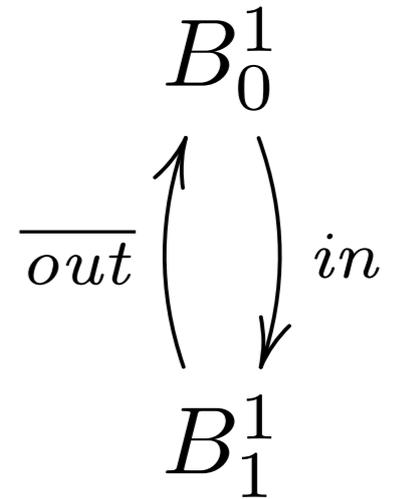
$$\text{Const) } \frac{A \triangleq p \in \Delta \quad p \xrightarrow{\mu} q}{A \xrightarrow{\mu} q}$$

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

CCS: buffer di capacita' 1

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

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

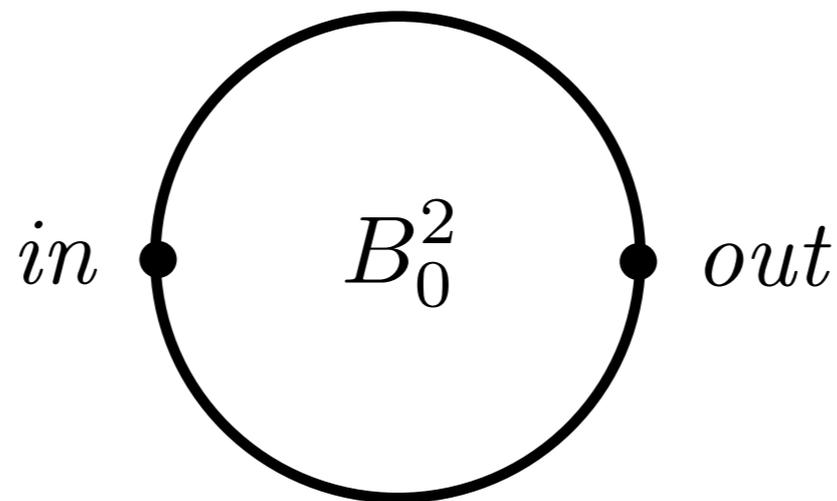
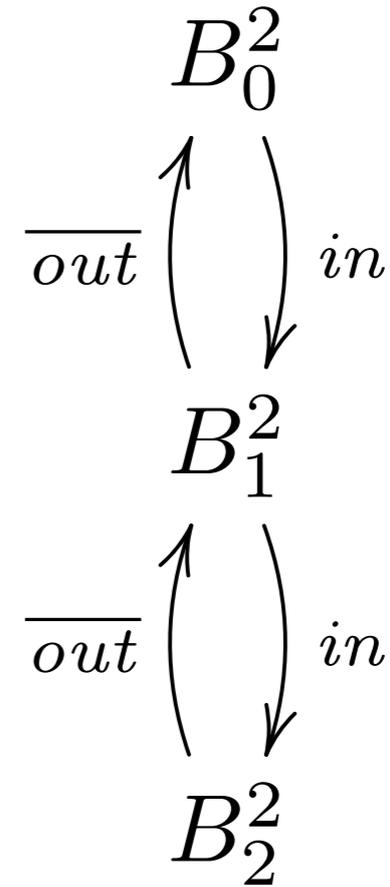


CCS: buffer di capacita' 2

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

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

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

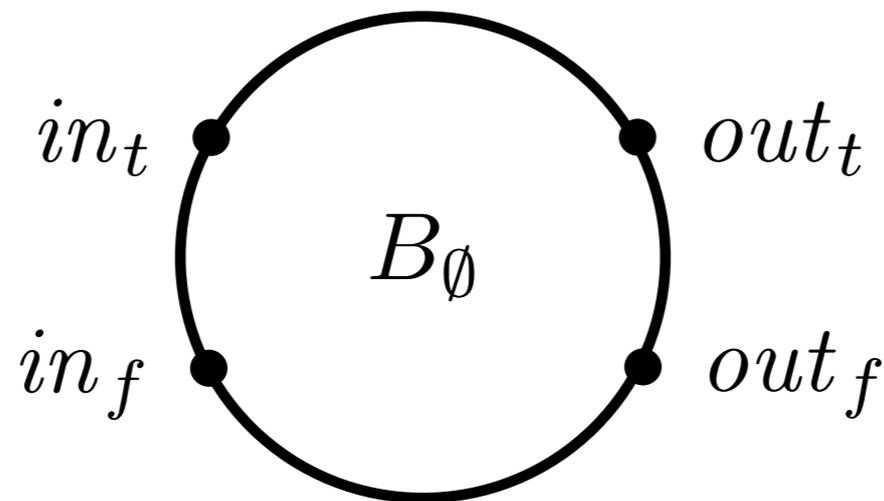
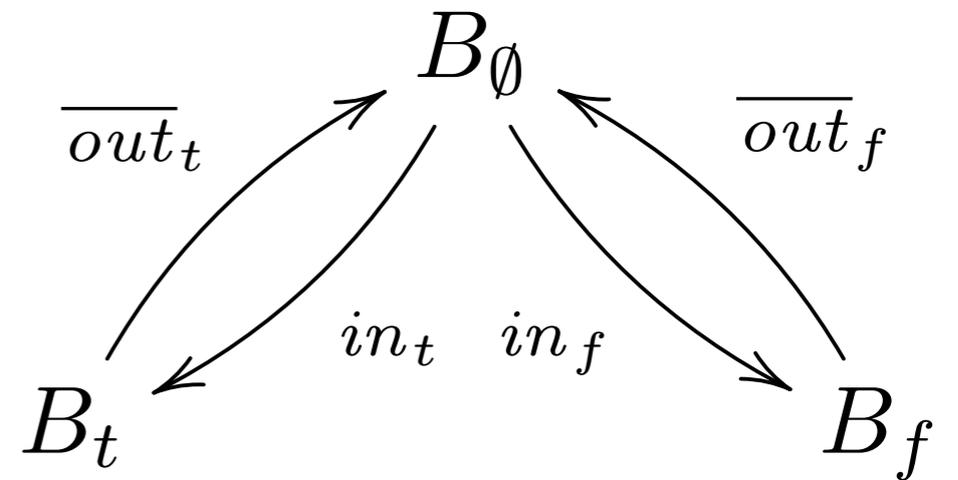


CCS: buffer booleano

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

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

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



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

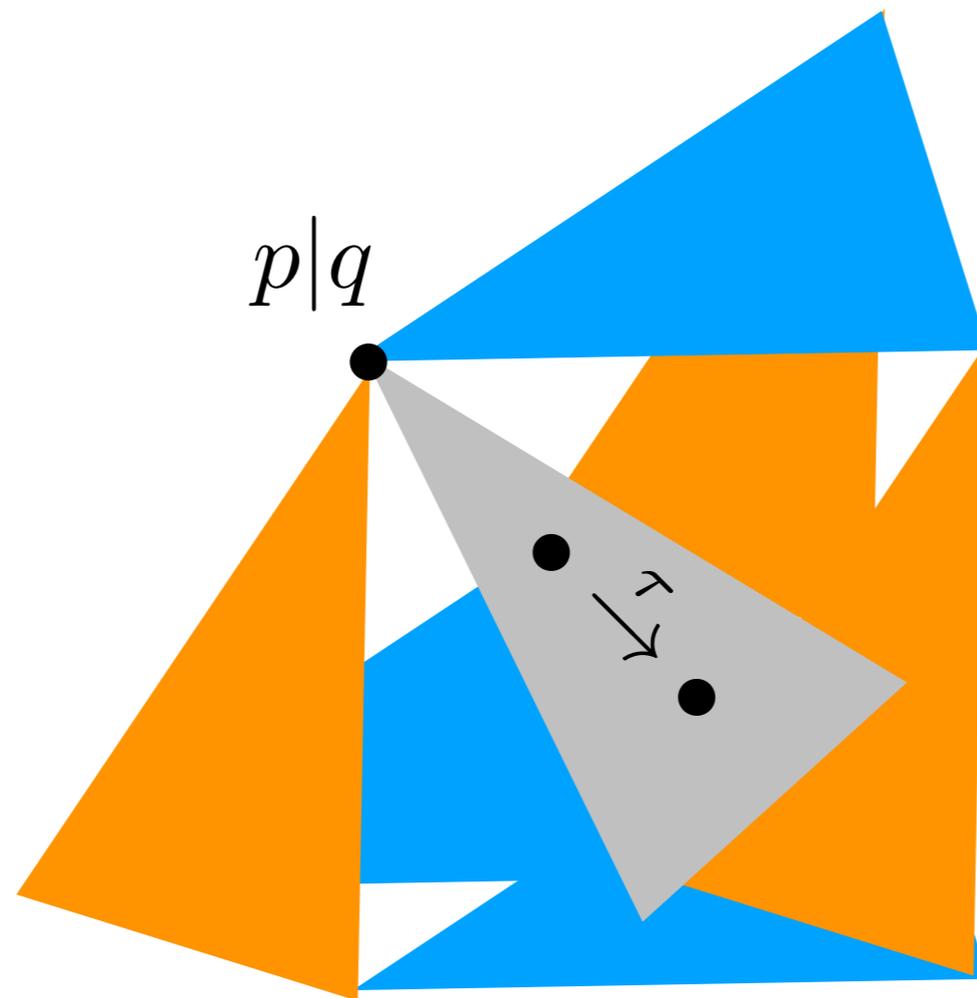
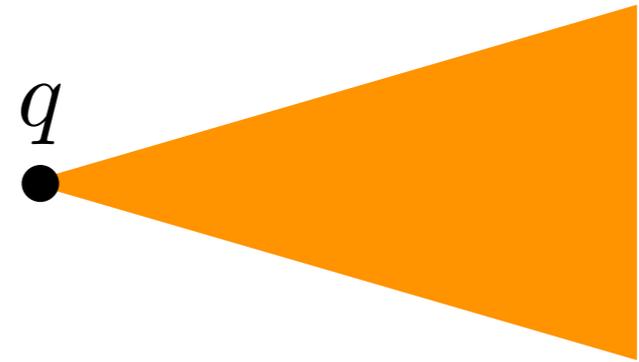
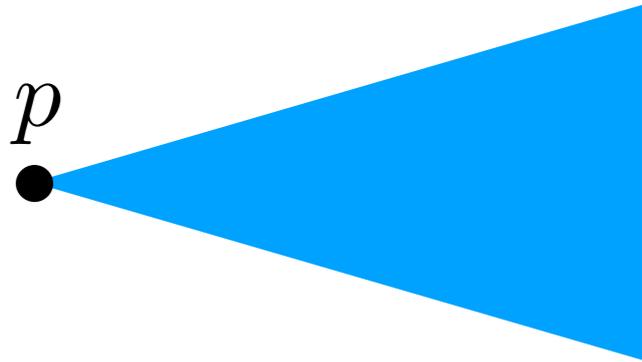
$$P \triangleq \overline{coin}.coffee.nil \quad M \triangleq coin.(\overline{coffee}.nil + \overline{tea}.nil)$$

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

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

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

LTS del processo



CCS: buffer paralleli

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

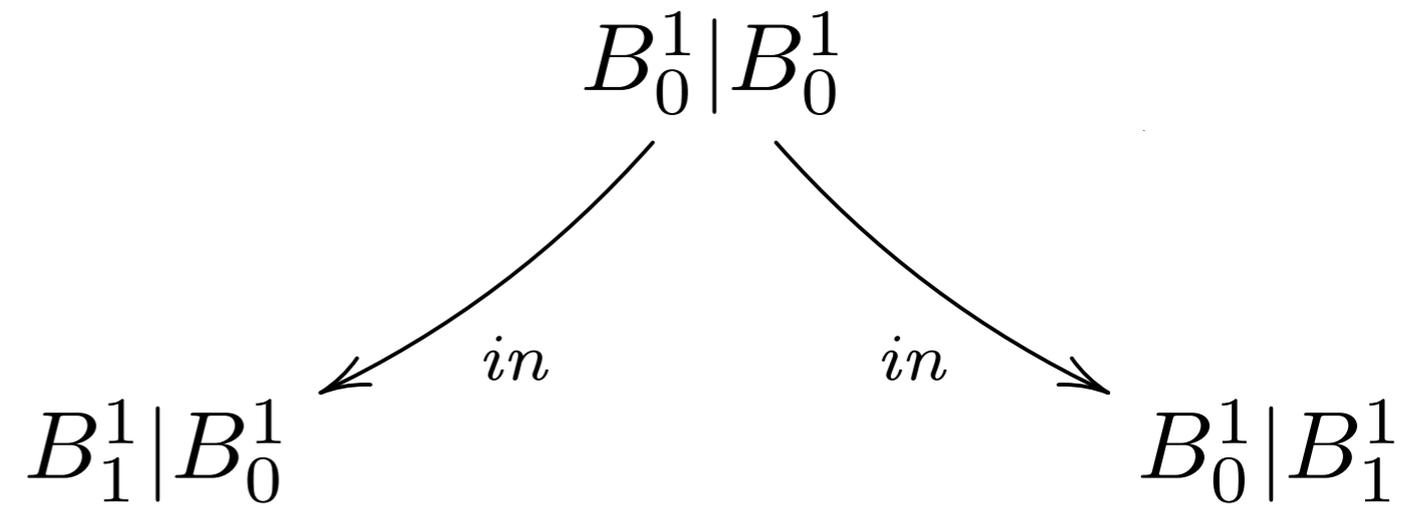
$$B_1^1 \triangleq \overline{out}.B_0^1$$

$$B_0^1 | B_0^1$$

CCS: buffer paralleli

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

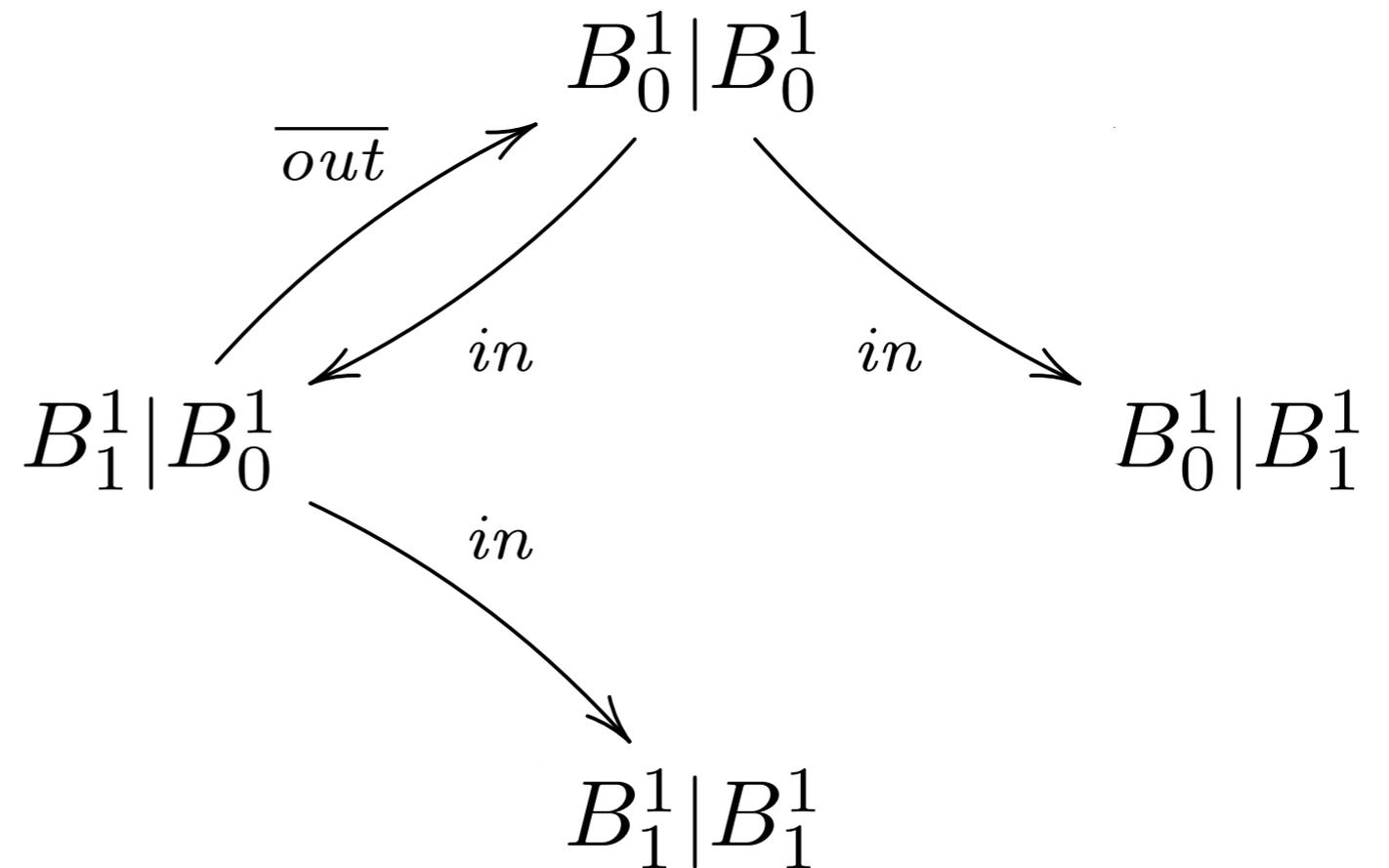
$$B_1^1 \triangleq \overline{out}.B_0^1$$



CCS: buffer paralleli

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

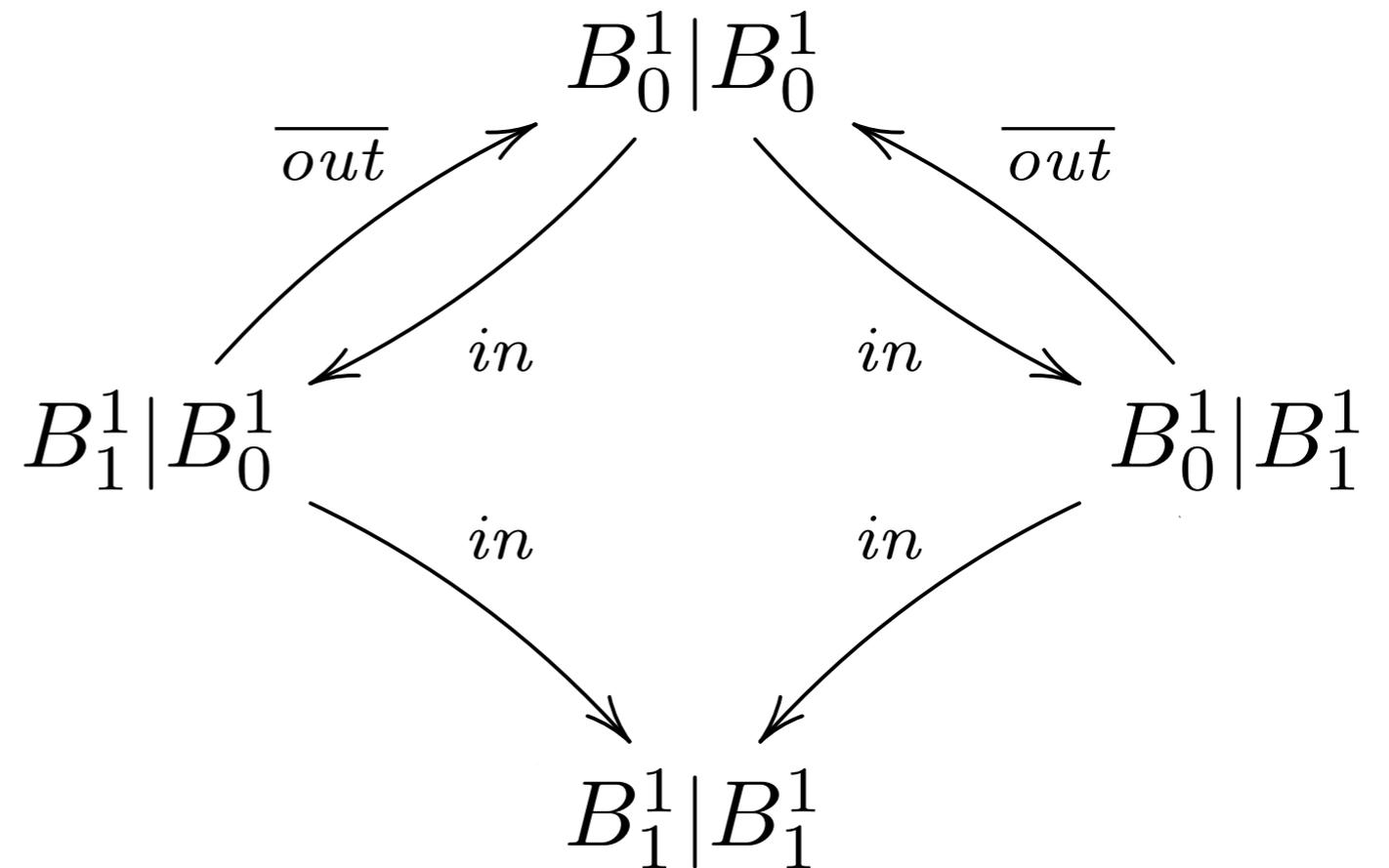
$$B_1^1 \triangleq \overline{out}.B_0^1$$



CCS: buffer paralleli

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

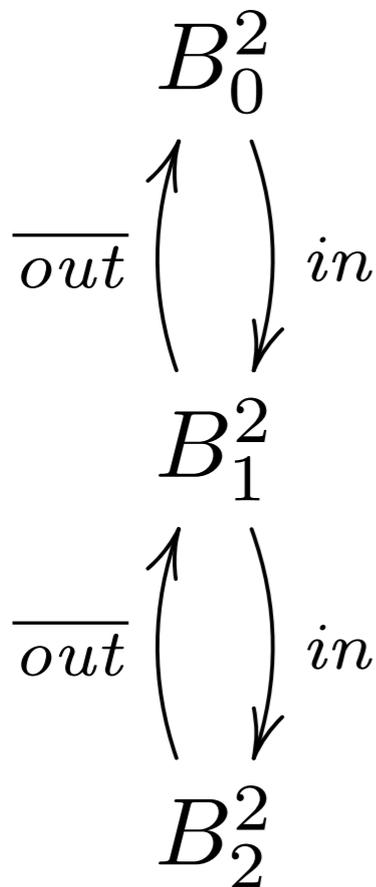
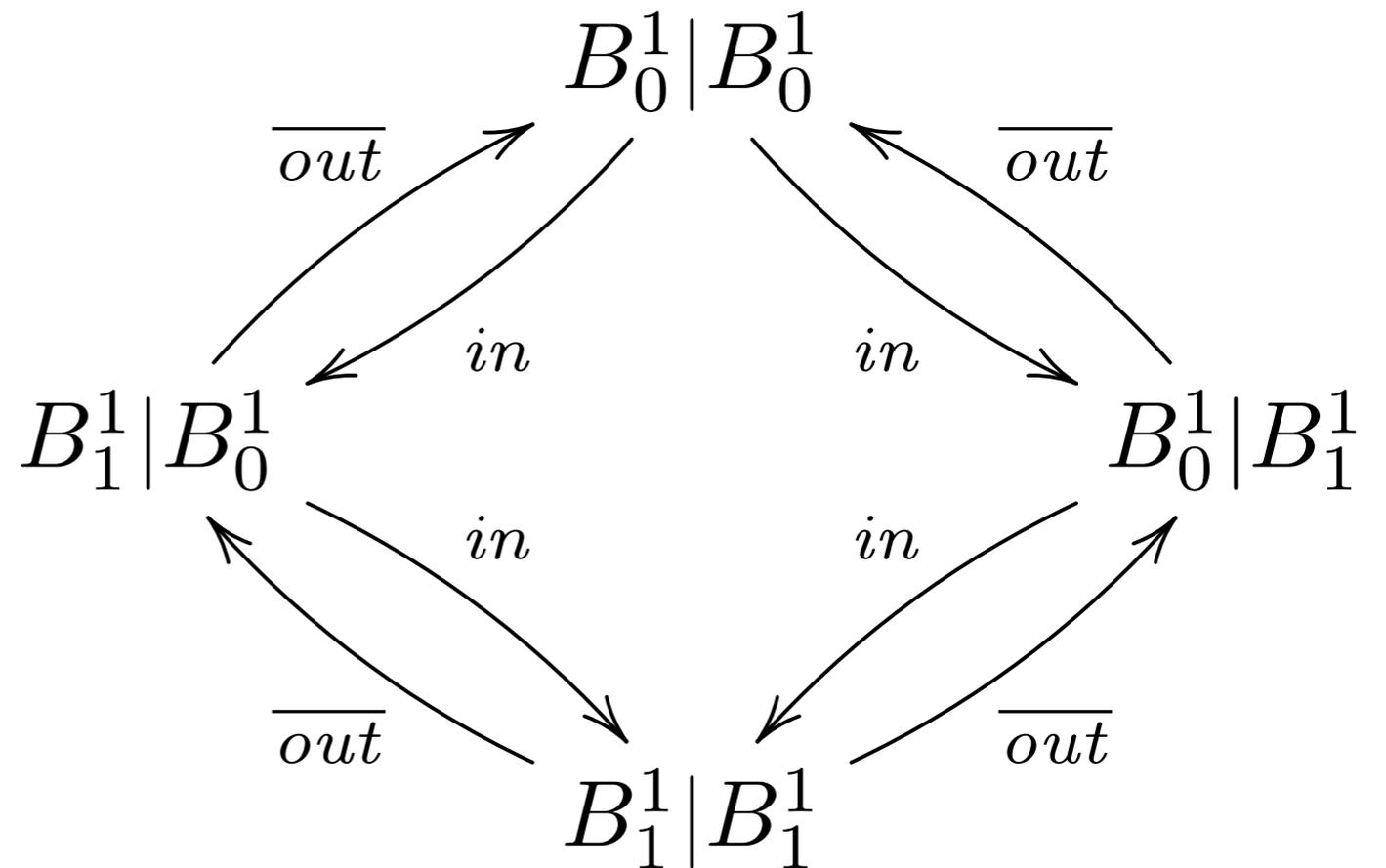
$$B_1^1 \triangleq \overline{out}.B_0^1$$



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 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

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

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

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

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**

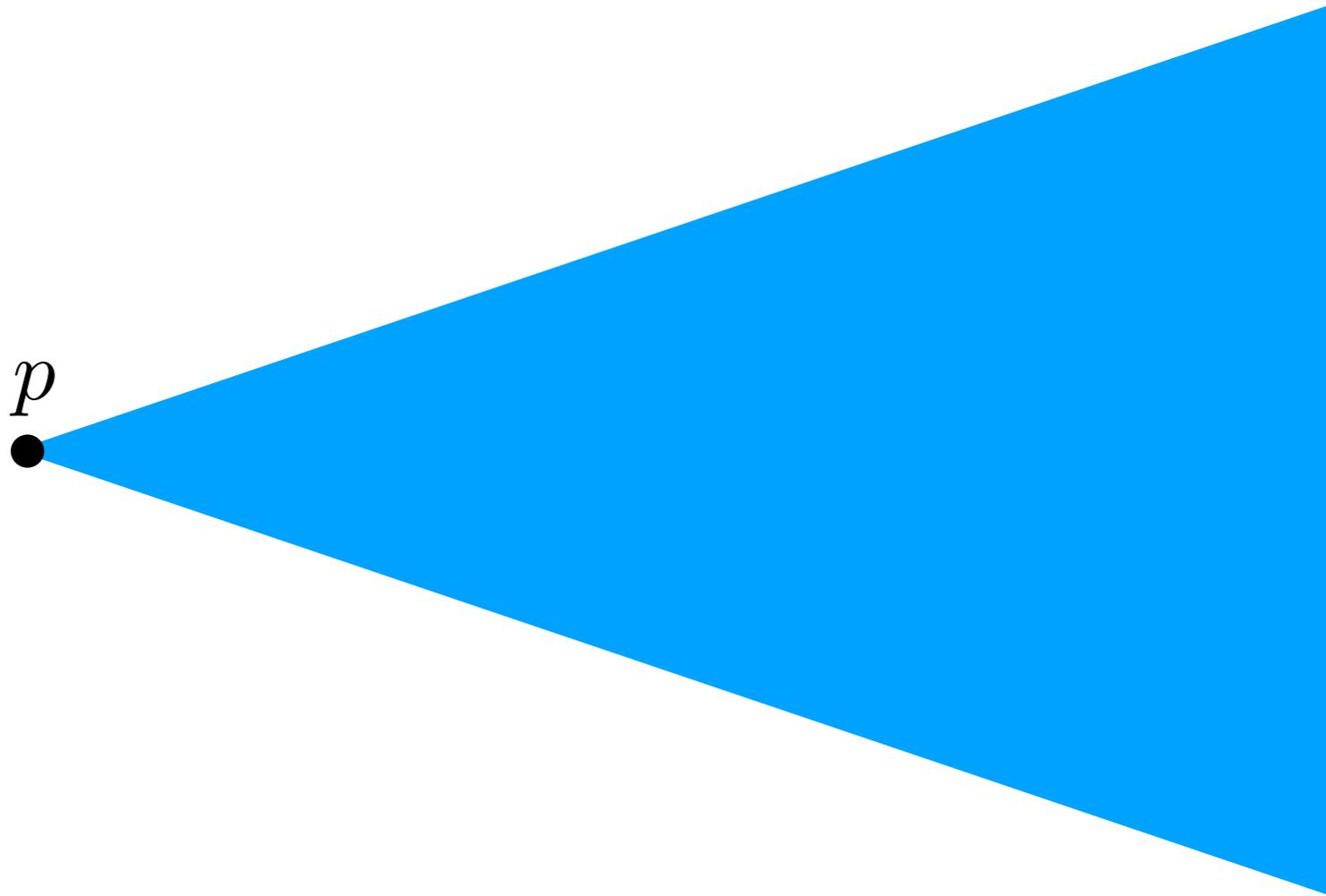
$$P \triangleq \overline{coin}.coffee$$

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

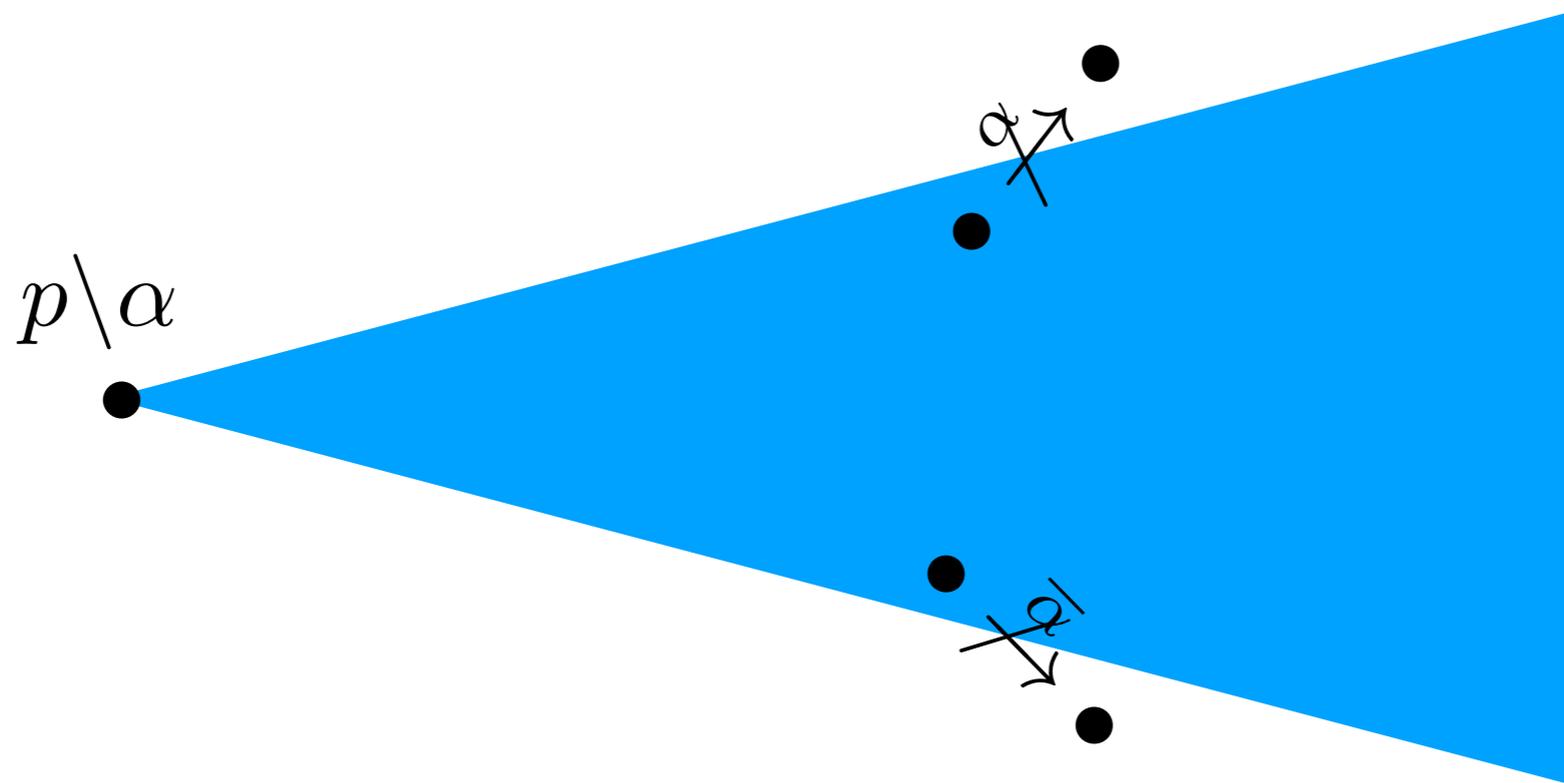
$$S \triangleq \{coin, coffee, tea\}$$

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

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 ϕ

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

ci permette di riusare i processi

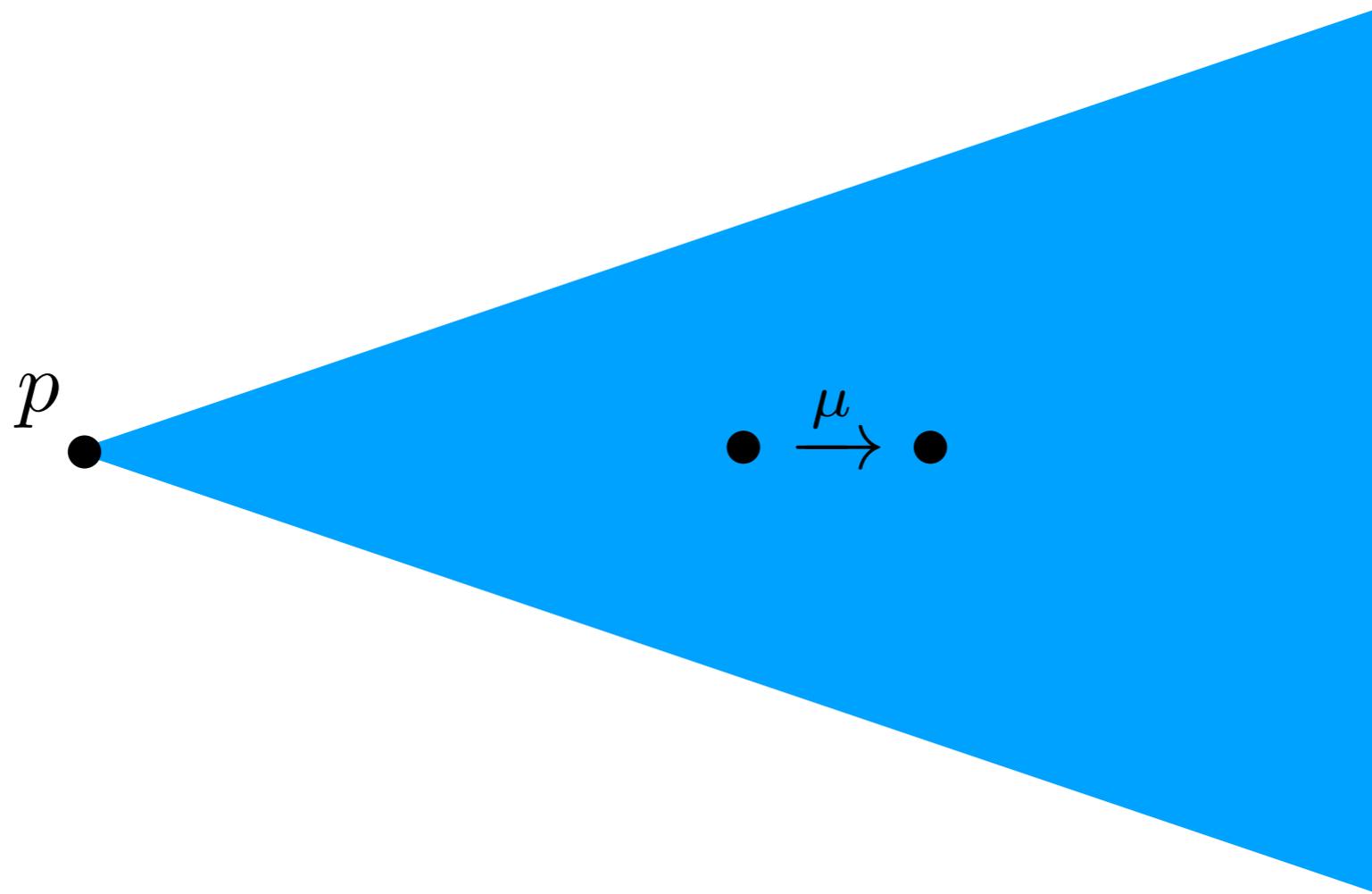
$P \triangleq \overline{coin}.coffee$

$\phi(coin) = moneta$

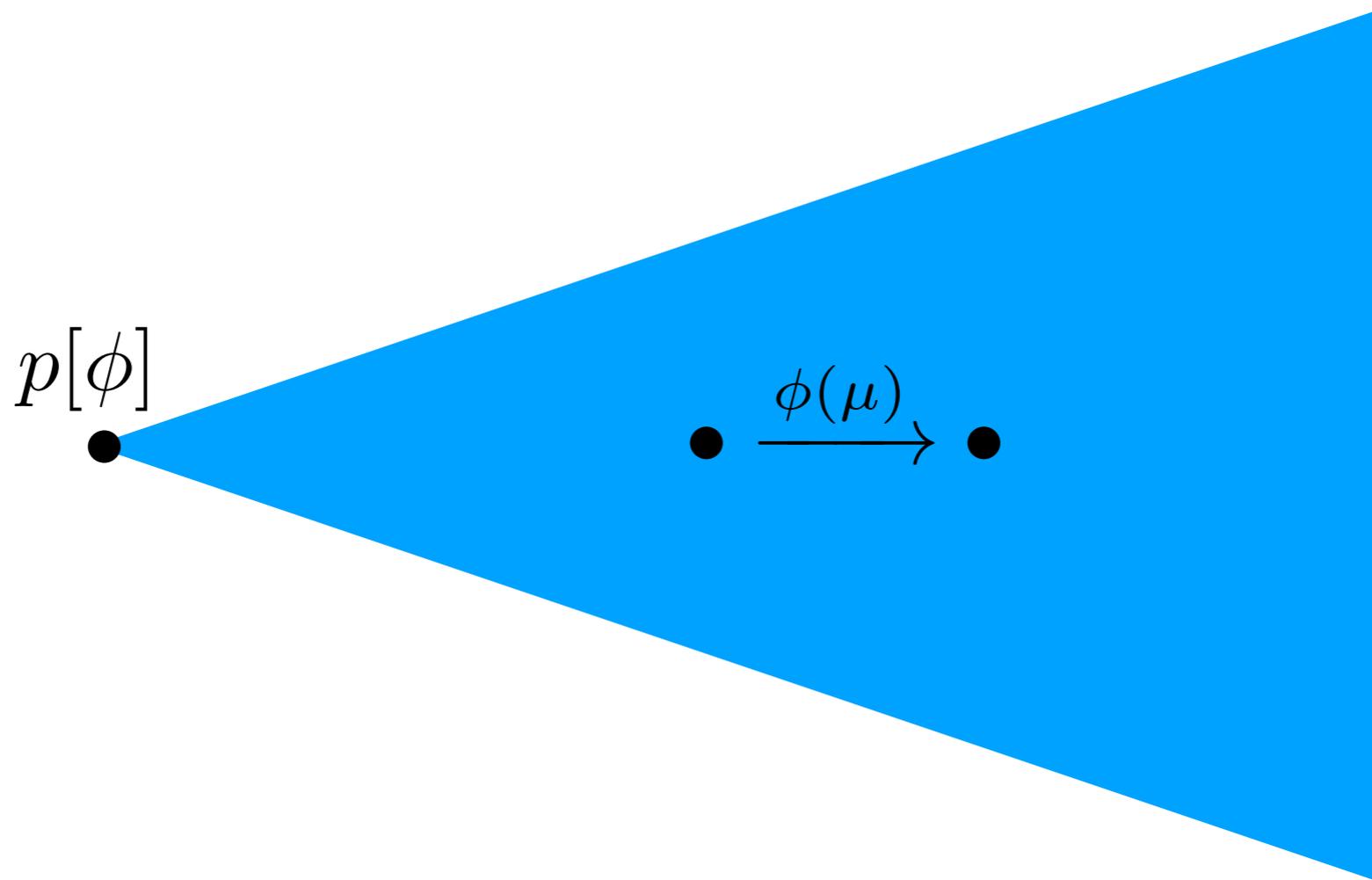
$\phi(coffee) = caffè$

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

LTS del processo



LTS del processo



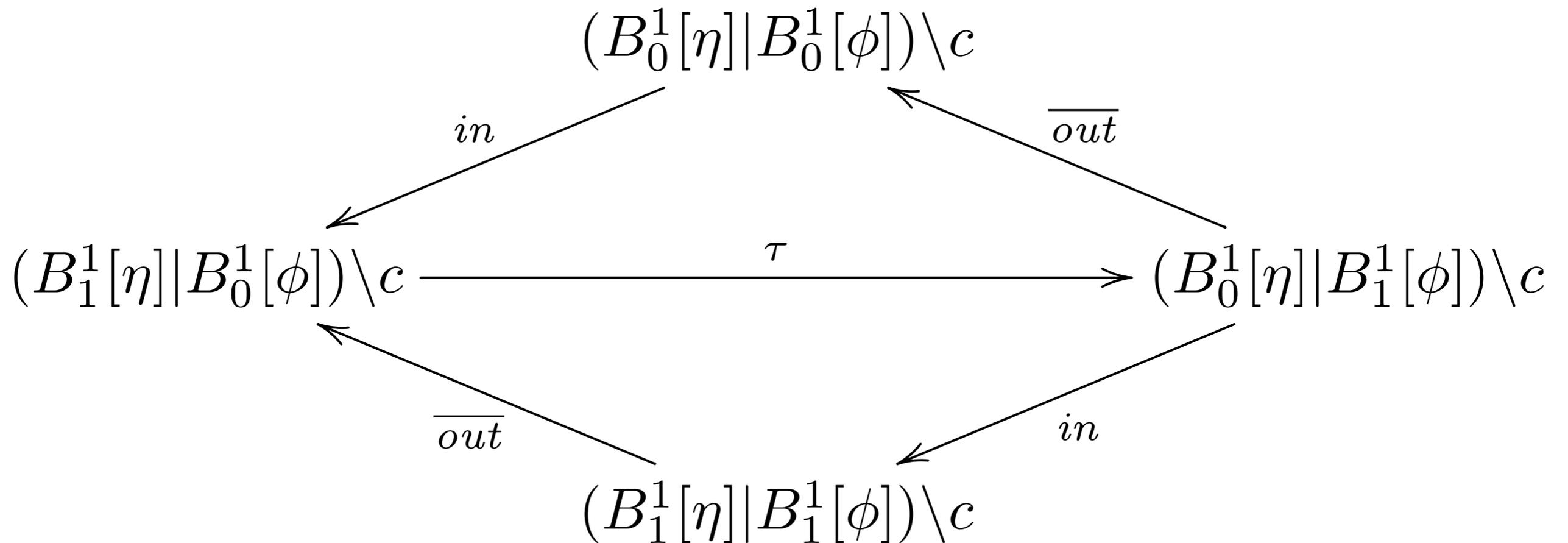
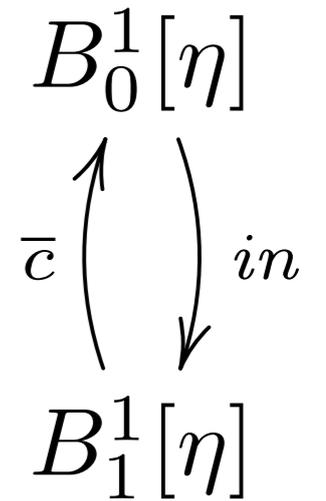
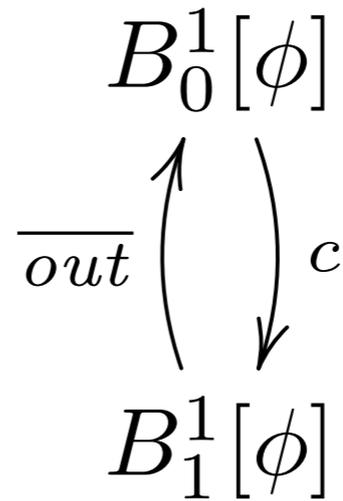
CCS semantica operativa

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

Buffer collegati

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

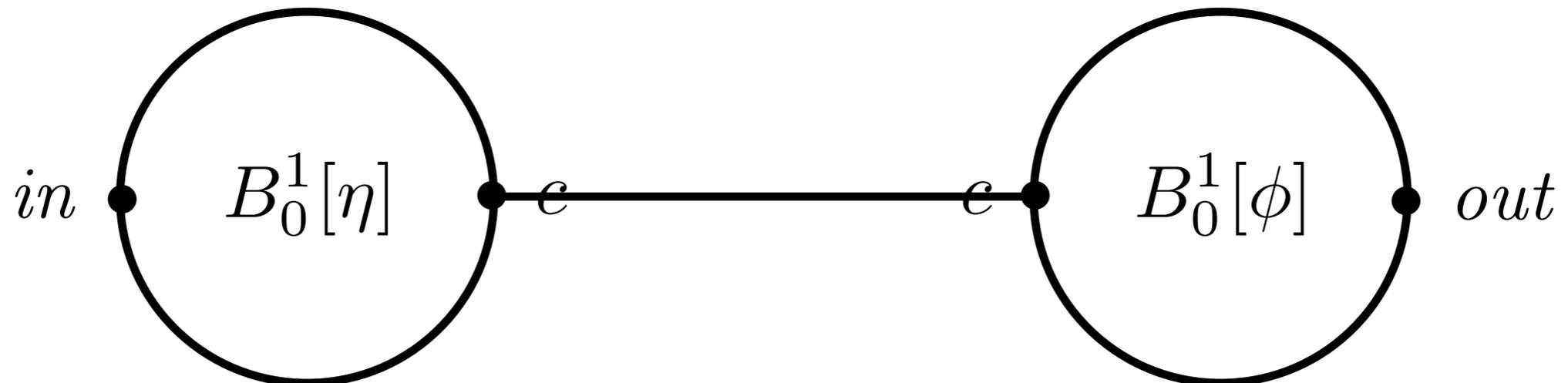
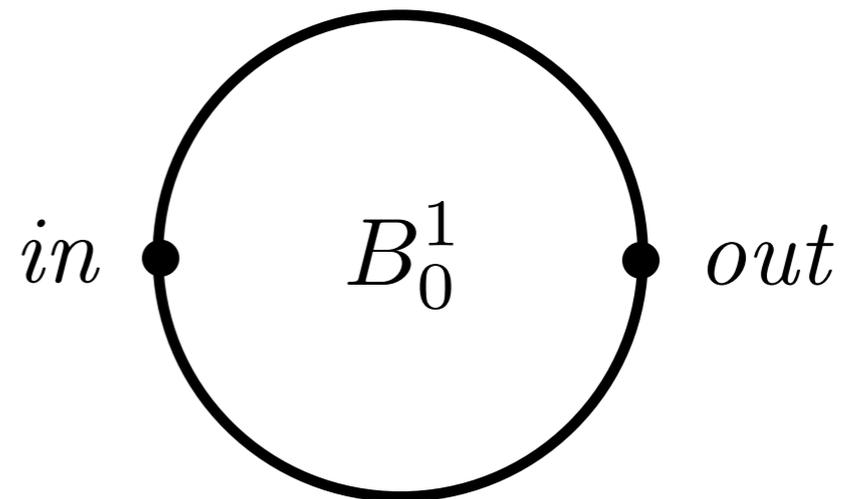
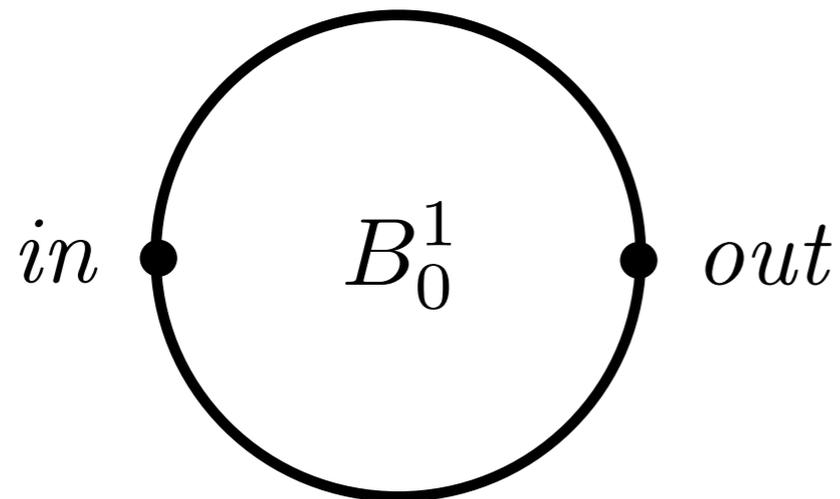
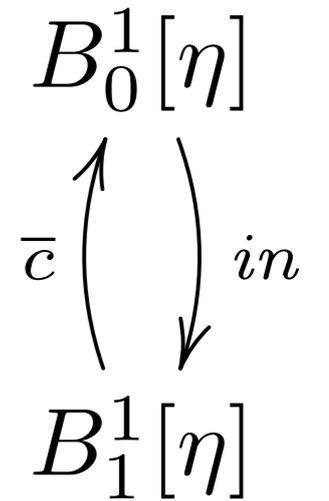
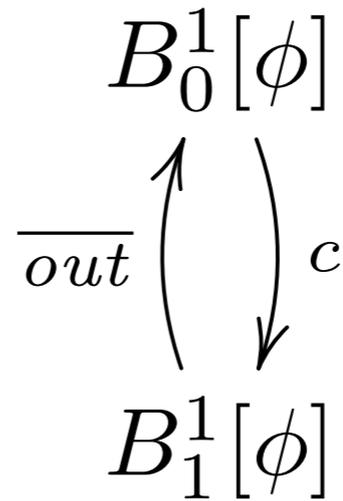
$$B_1^1 \triangleq \overline{out}.B_0^1 \quad \phi(in) = c$$



Buffer collegati

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

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

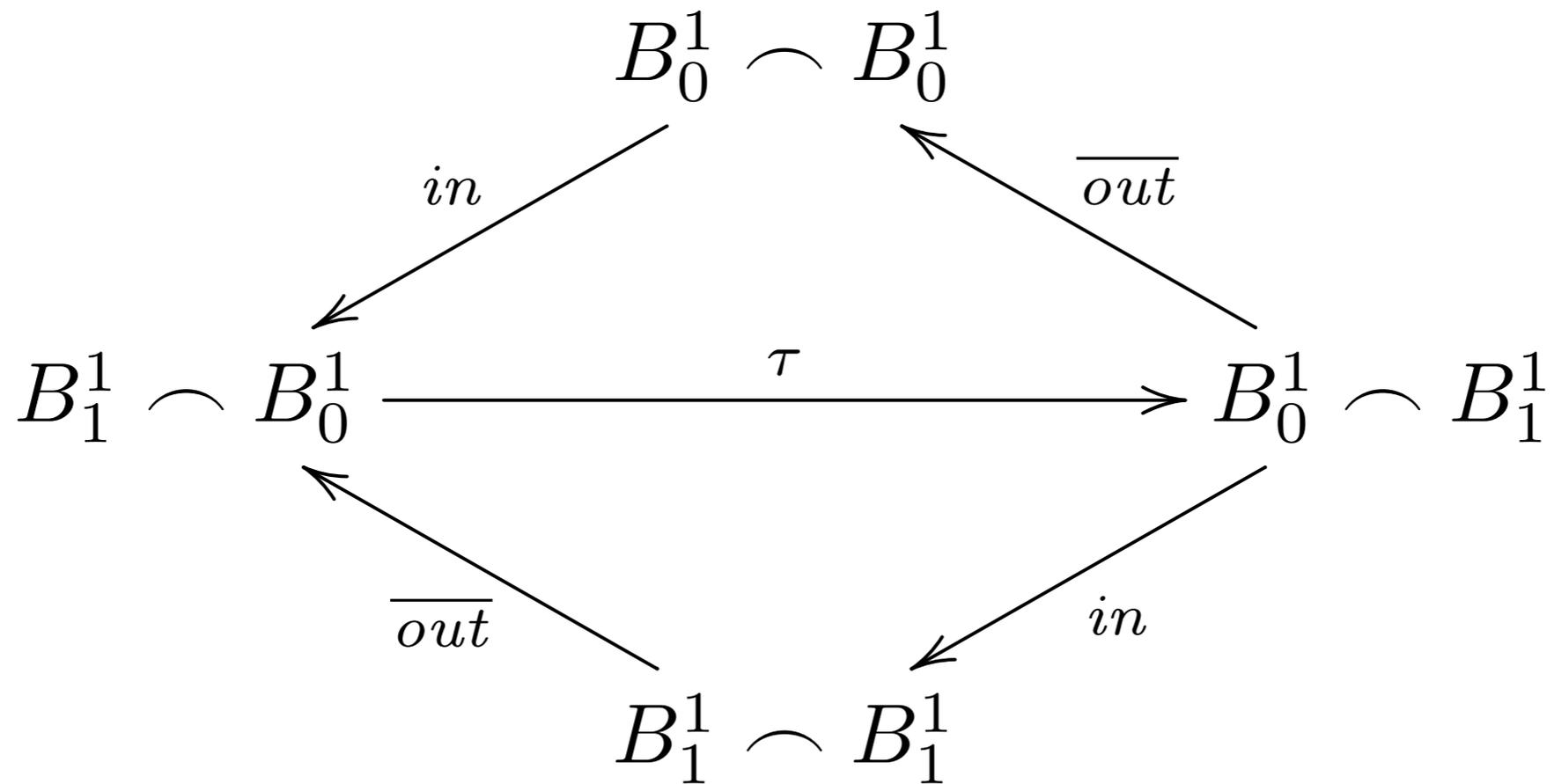


Buffer collegati

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

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

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



Buffer collegati booleani

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

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

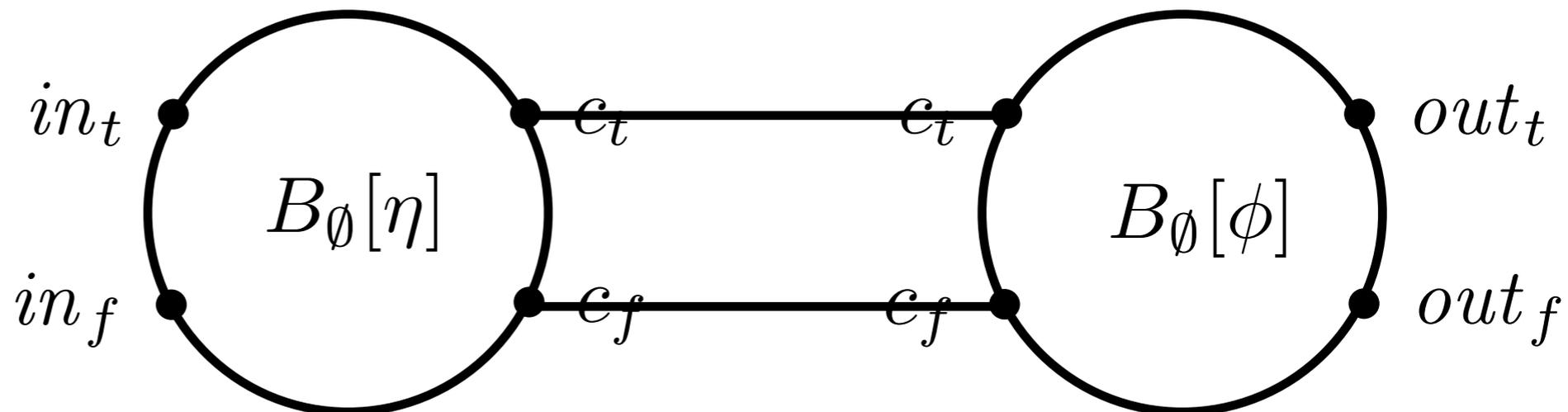
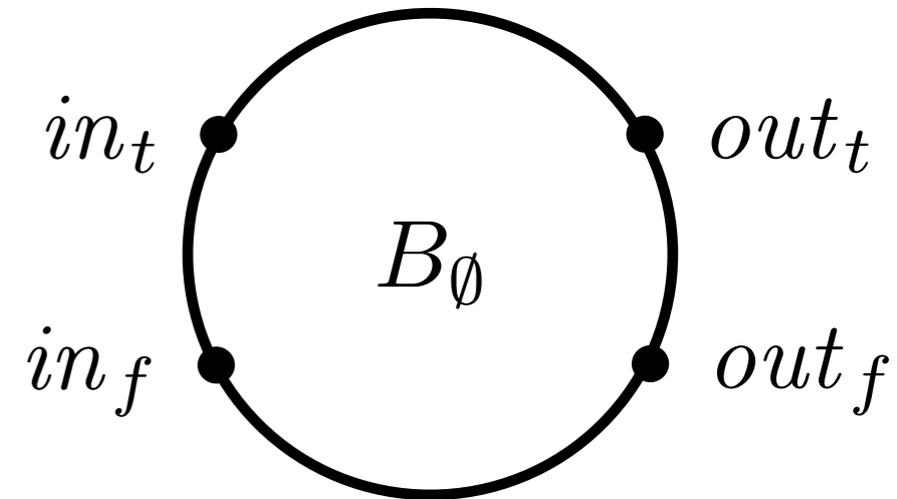
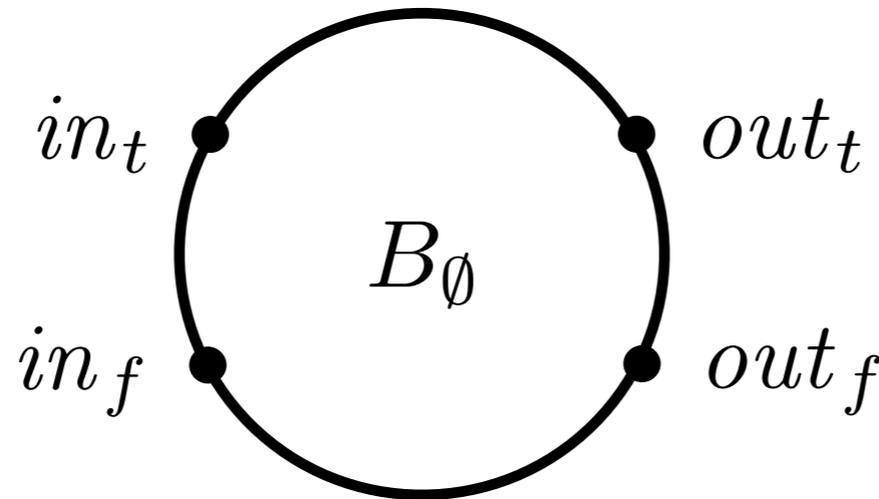
$$B_f \triangleq \overline{out_f}.B_{\emptyset} \quad p \frown q \triangleq (p[\eta] | q[\phi]) \setminus \{c_t, c_f\}$$

Buffer collegati booleani

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

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

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

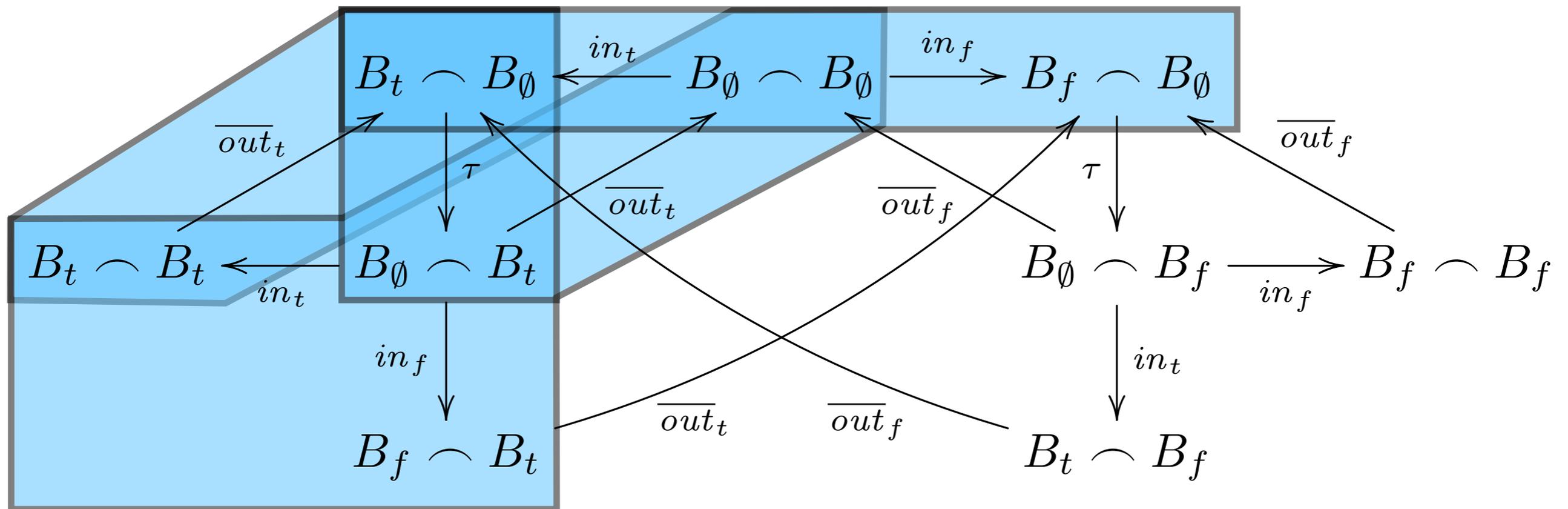


Buffer collegati booleani

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

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

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

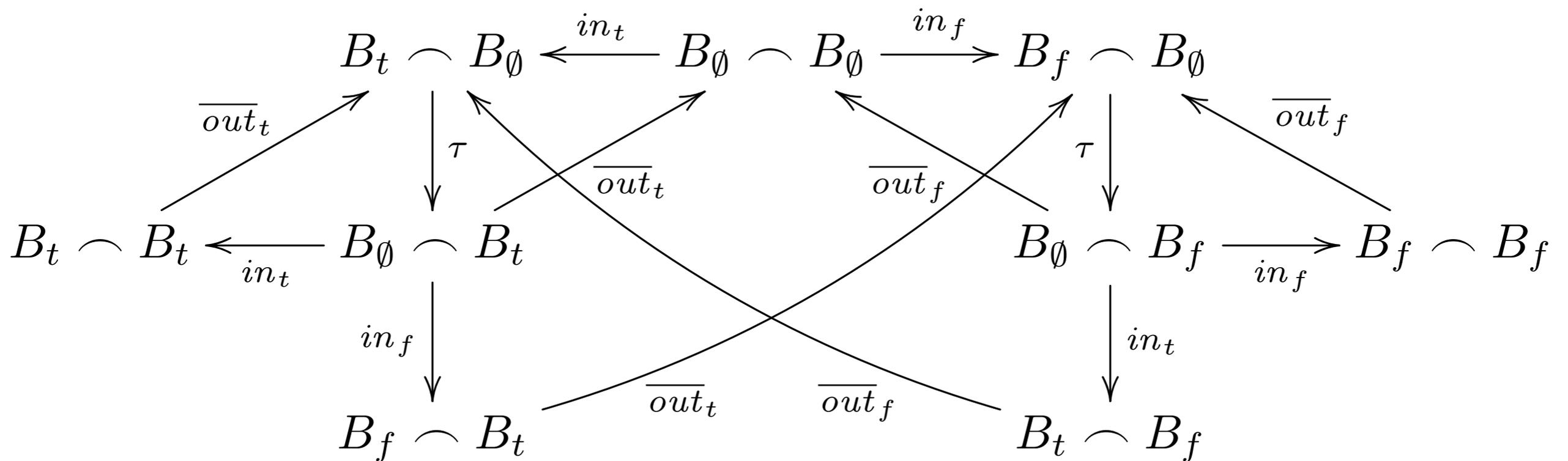


Buffer collegati booleani

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

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

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



CCS con passaggio di valore

$$\overline{\alpha_v.p} \xrightarrow{\overline{\alpha_v}} p$$

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

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

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

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

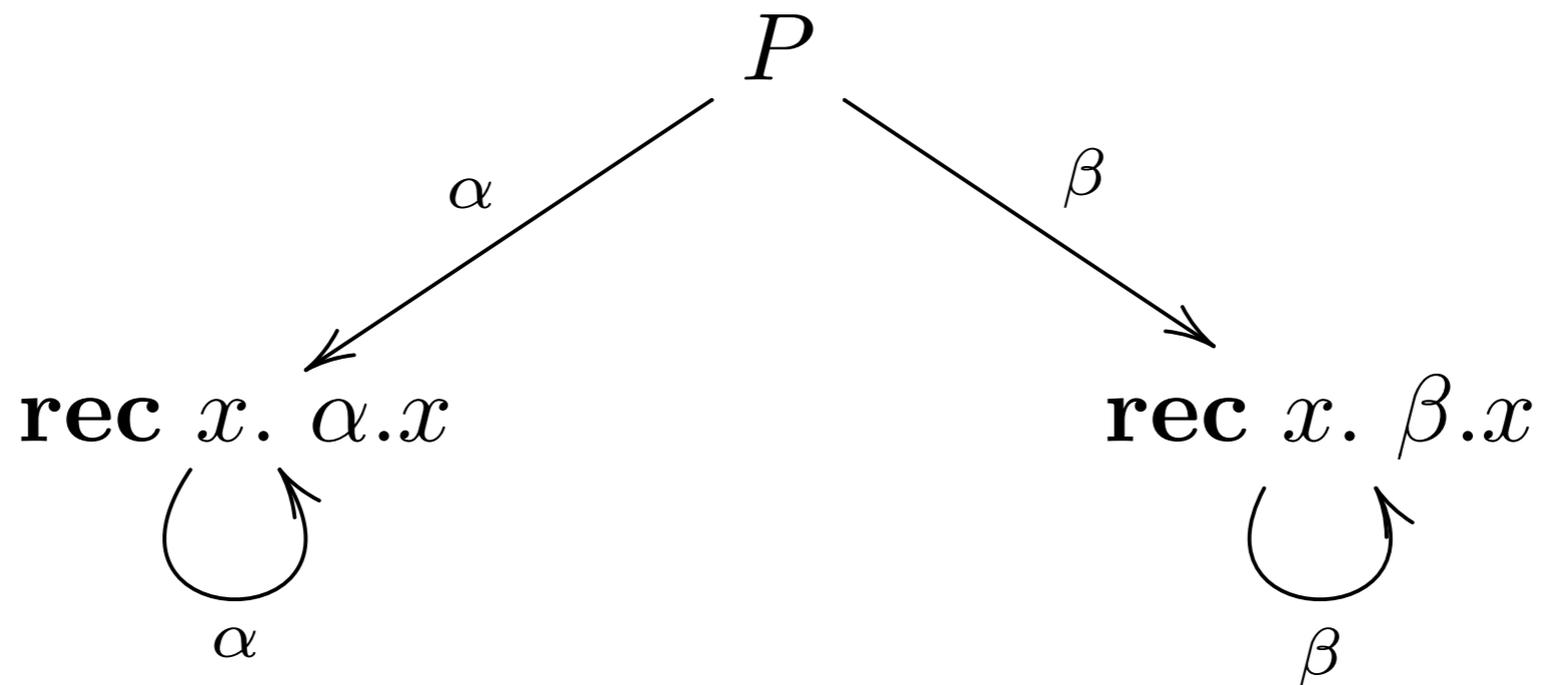
receive

$$\begin{array}{l} v \rightarrow p \\ w \rightarrow q \\ _ \rightarrow r \end{array} \equiv \alpha_v.p + \alpha_w.q + \sum_{z \neq v, w} \alpha_z.r$$

end

Esercizio: LTS?

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



Esercizio: LTS?

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

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

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

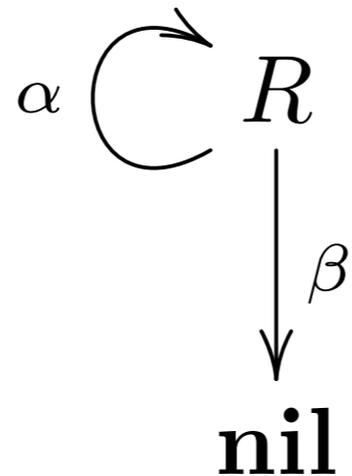


Esercizio: LTS?

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

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

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

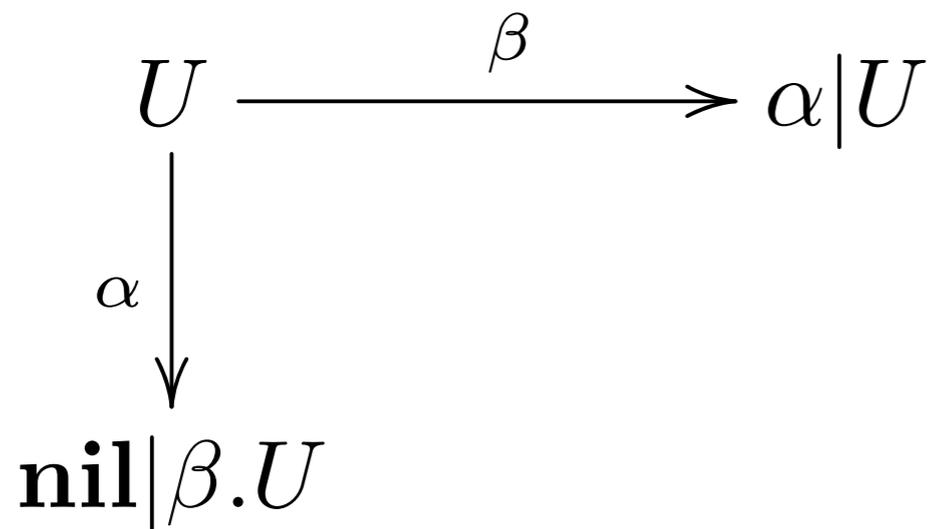


Esercizio: LTS?

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

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

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

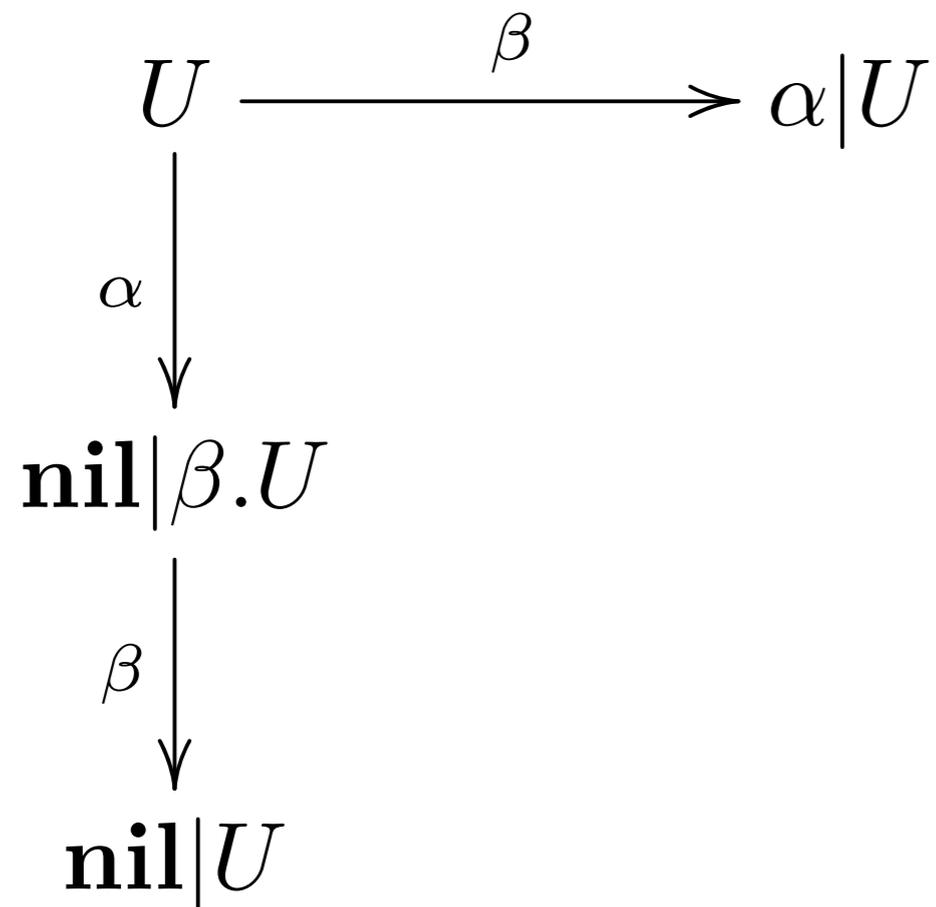


Esercizio: LTS?

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

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

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

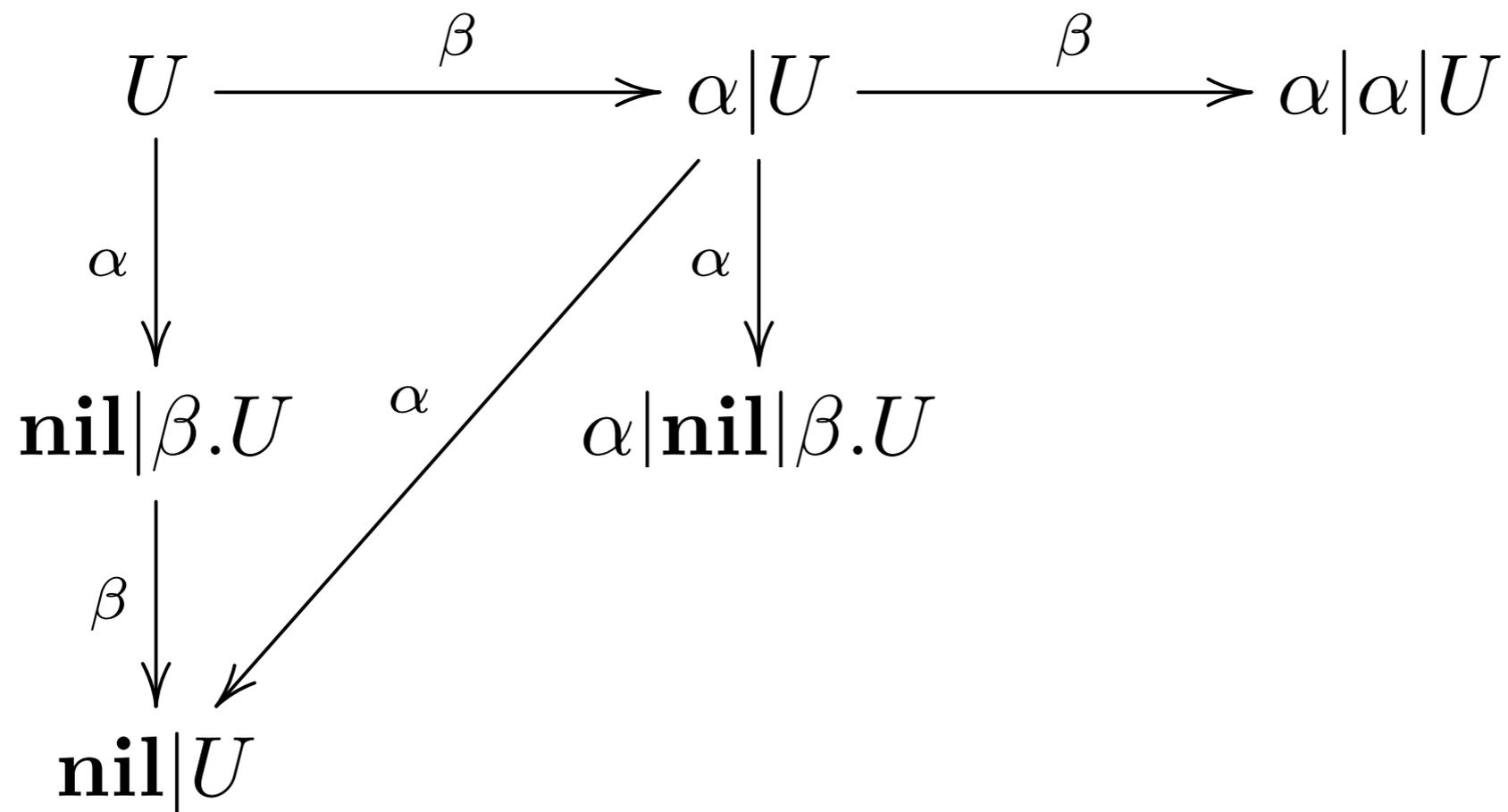


Esercizio: LTS?

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

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

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

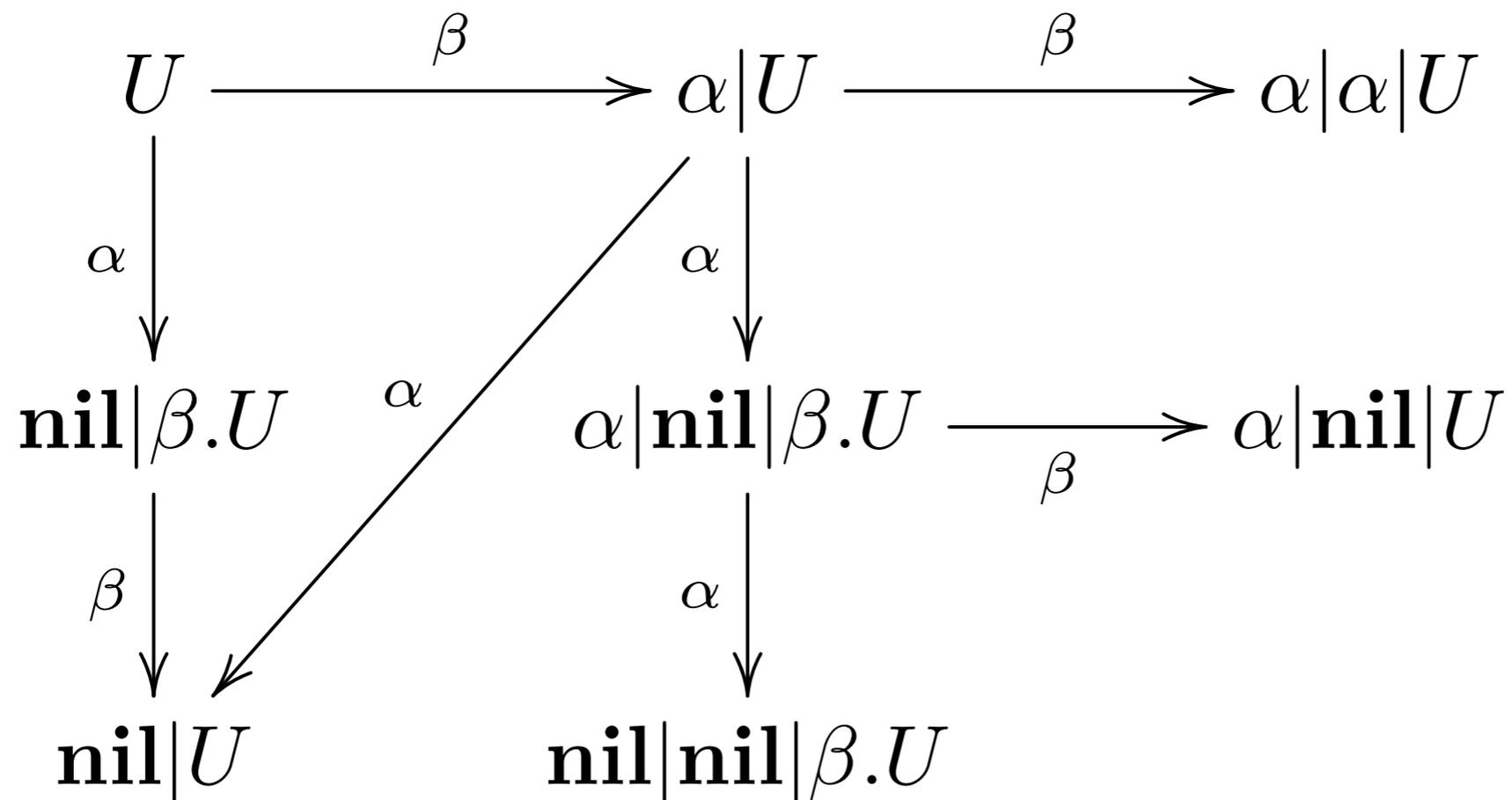


Esercizio: LTS?

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

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

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

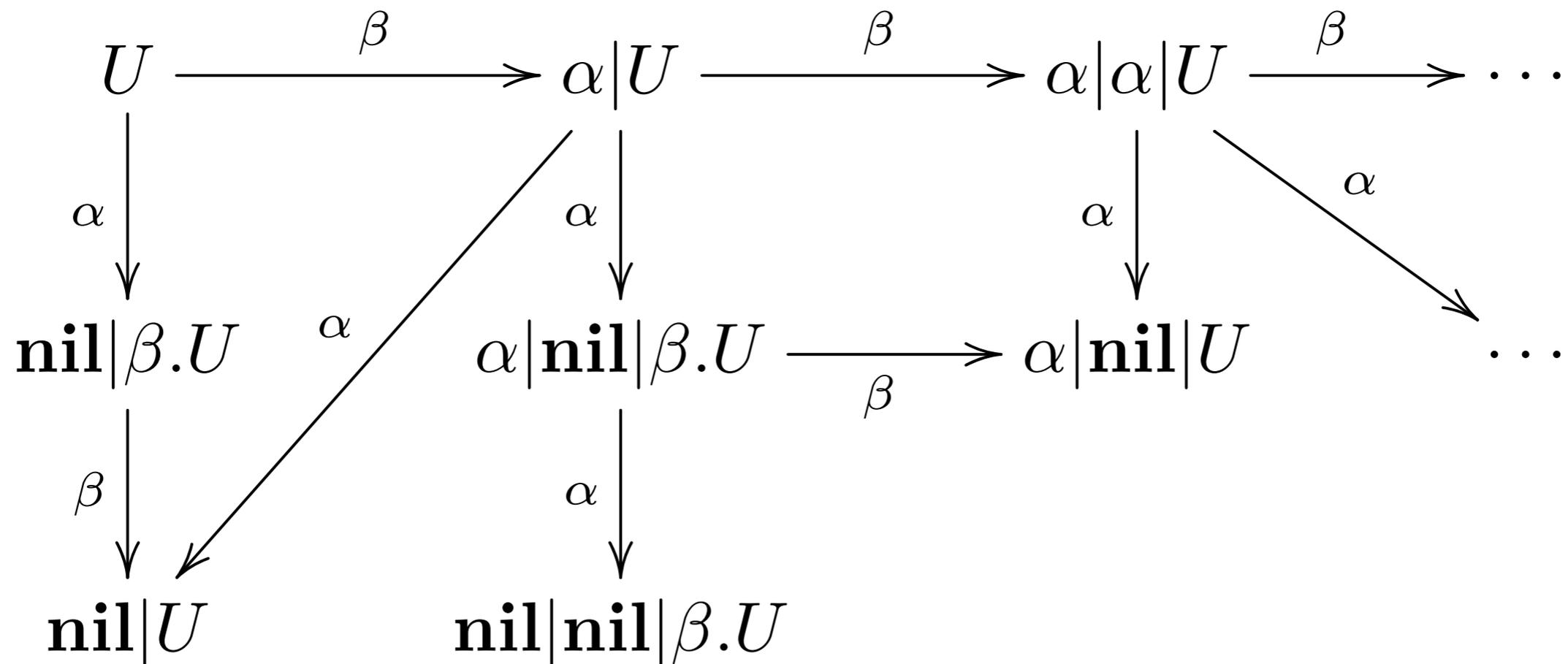


Esercizio: LTS?

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

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

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

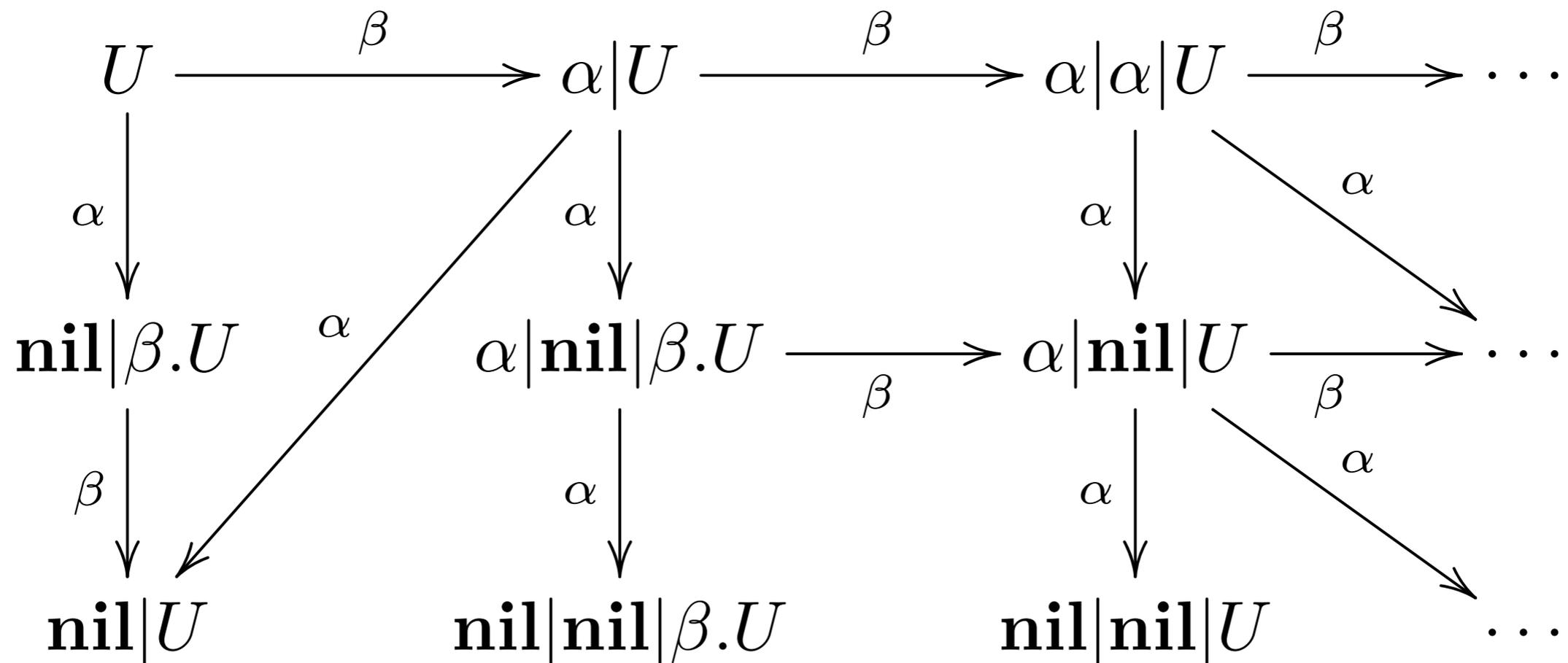


Esercizio: LTS?

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

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

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



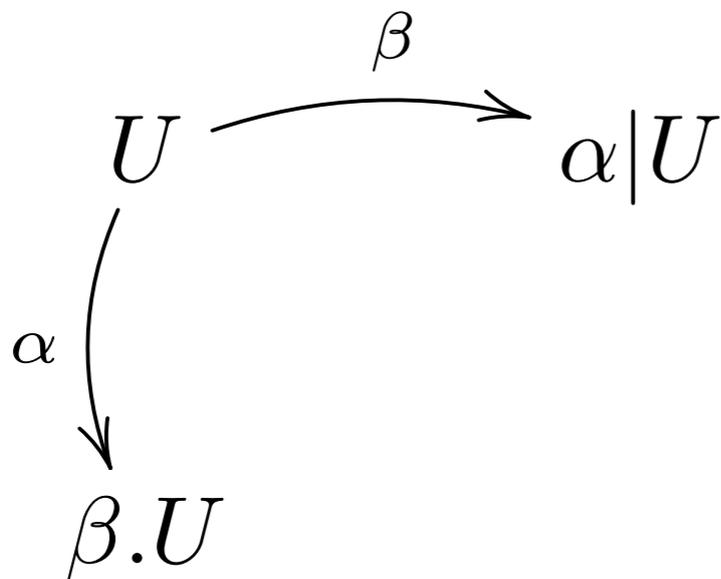
Esercizio: LTS?

ignoriamo **nil**

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

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

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



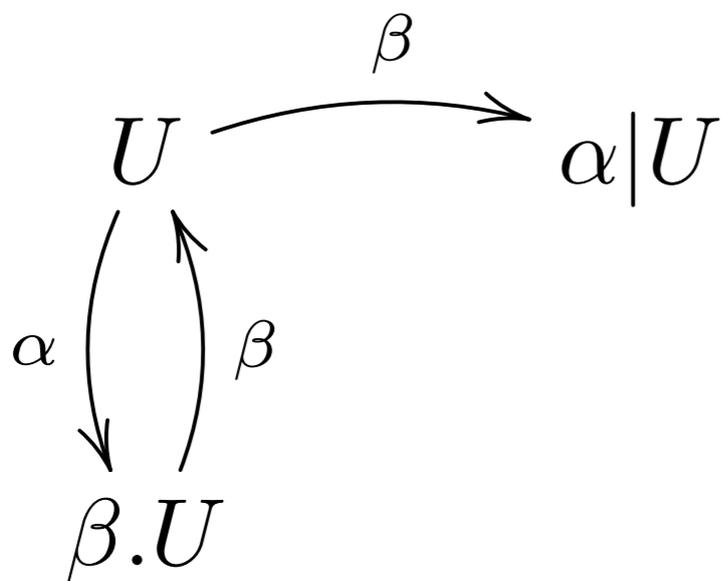
Esercizio: LTS?

ignoriamo **nil**

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

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

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



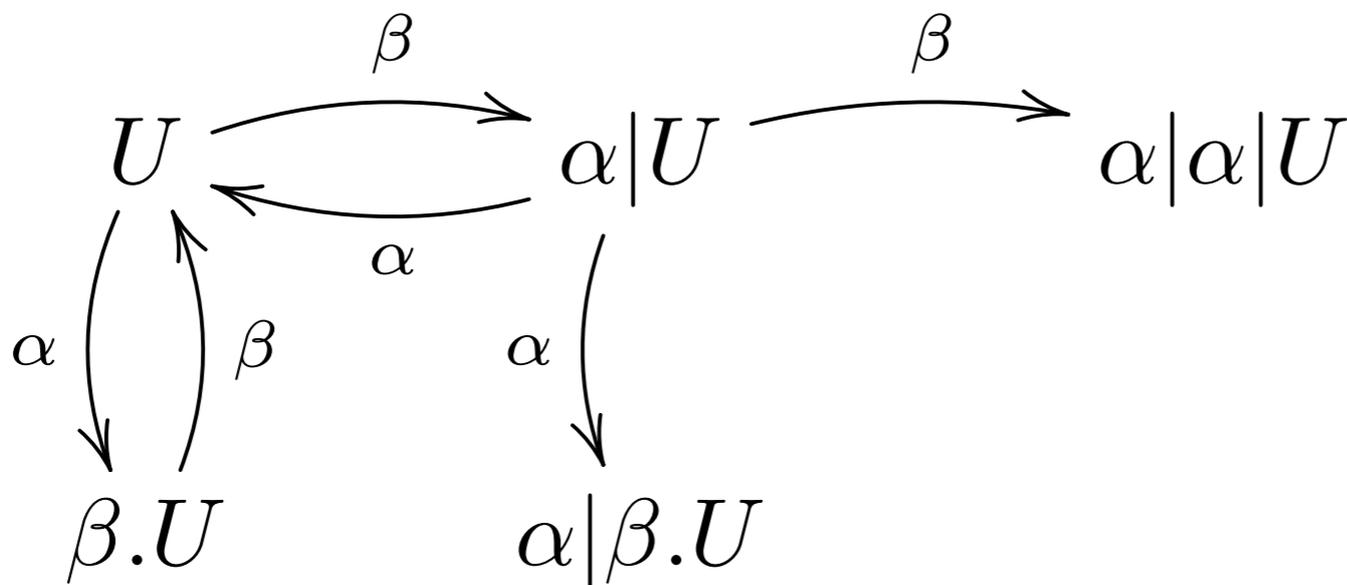
Esercizio: LTS?

ignoriamo nil

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

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

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



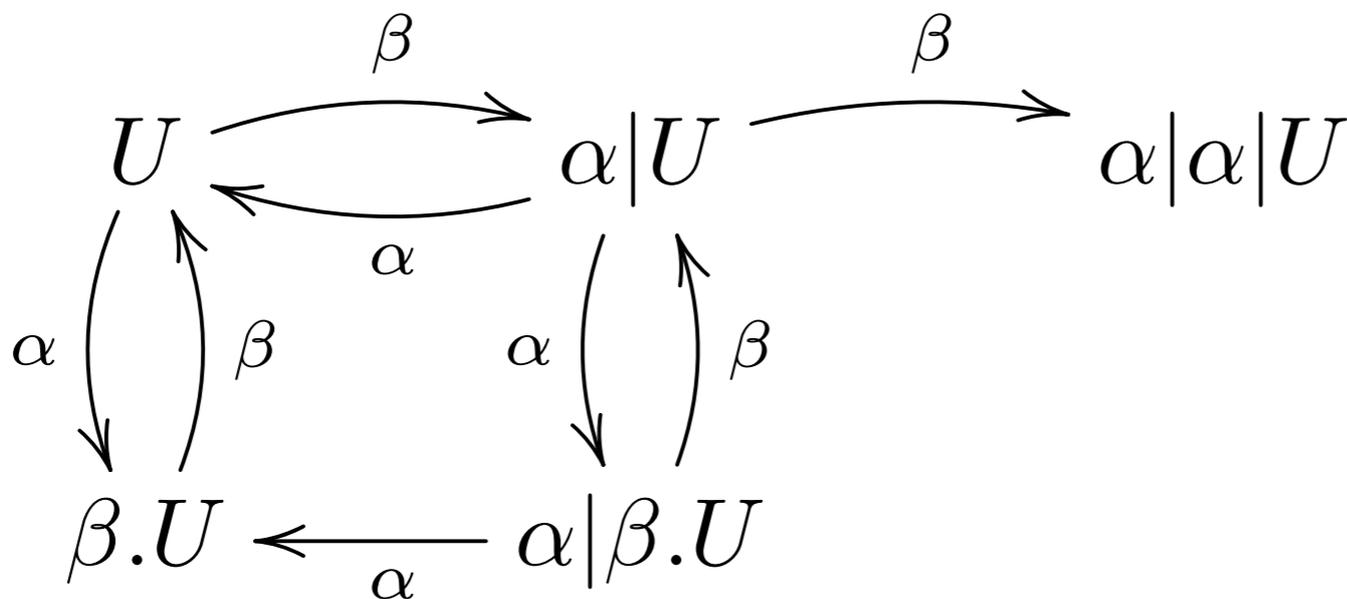
Esercizio: LTS?

ignoriamo nil

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

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

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



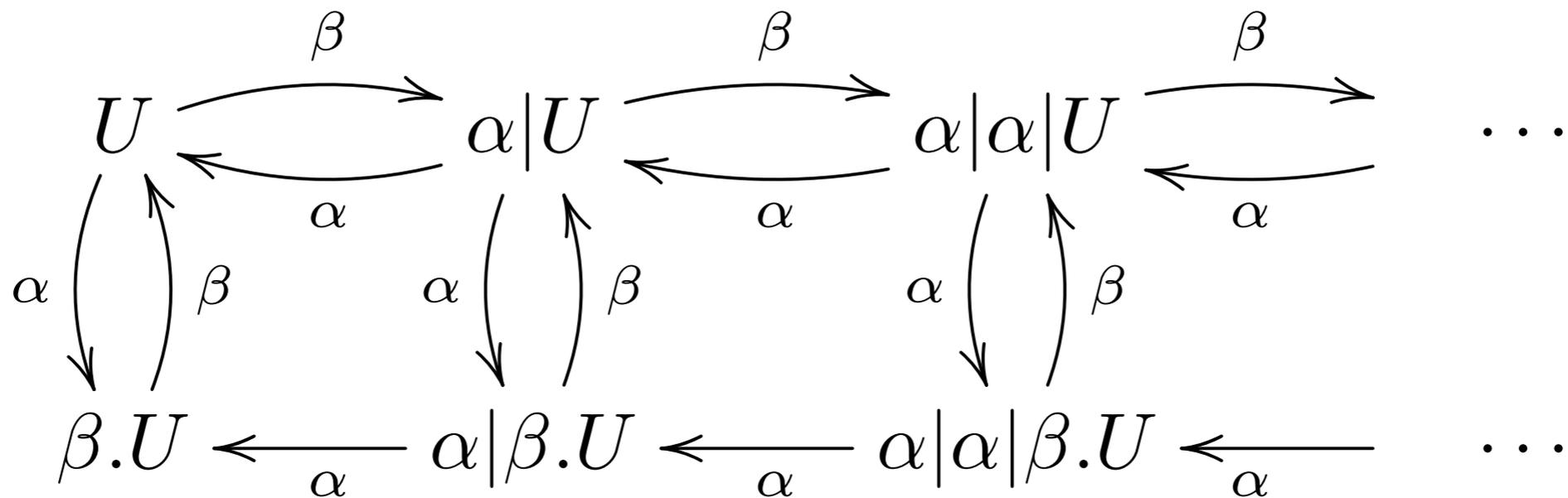
Esercizio: LTS?

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:

C0, C1, C2, C3

usato per visualizzare il valore corrente del contatore

Disegna l'LTS del processo `contatore`.