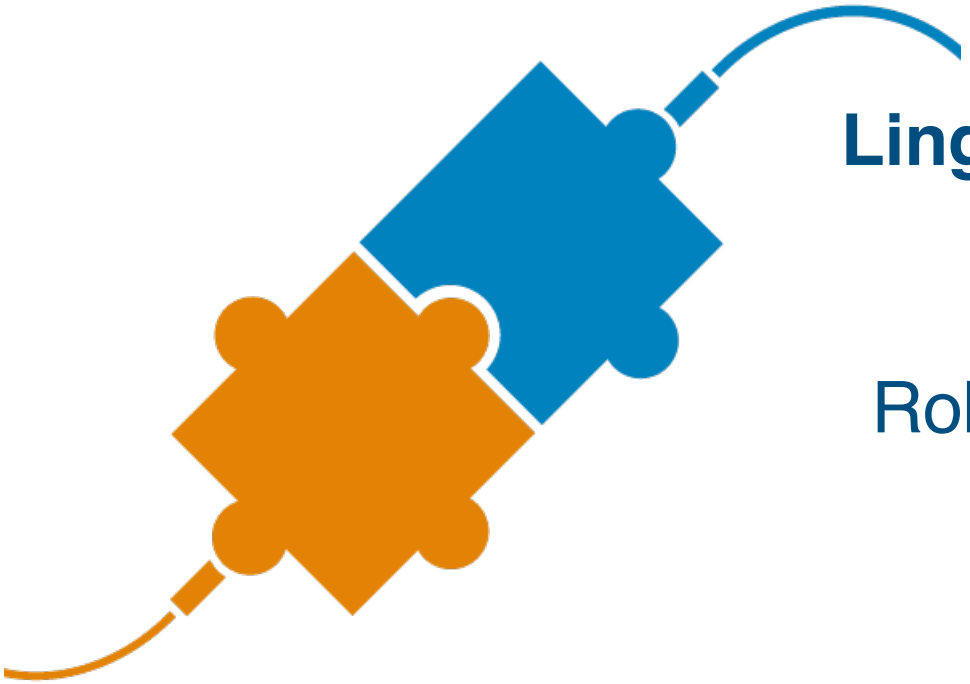


Linguaggi di Programmazione

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)

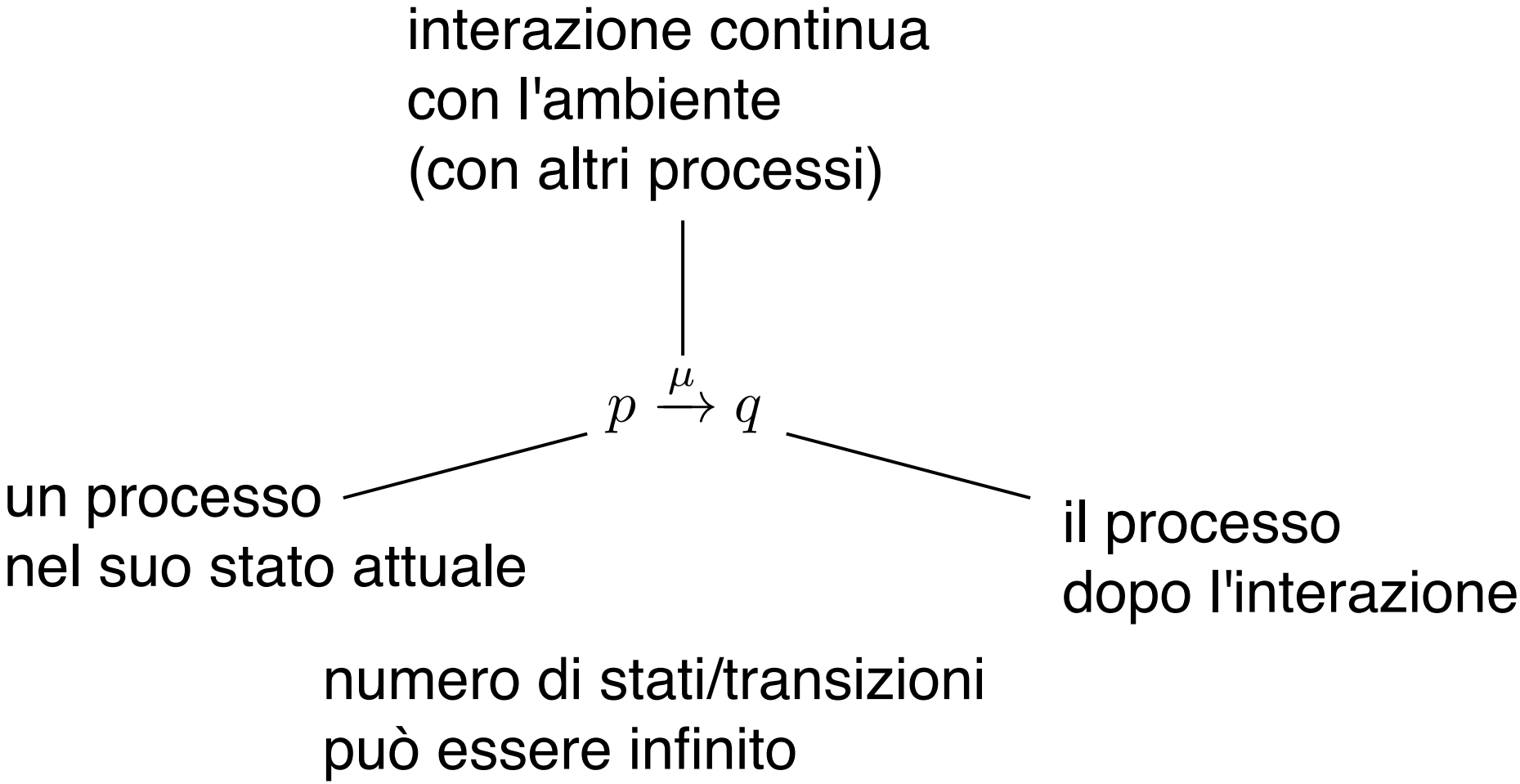
Comunicazione

- binaria, passaggio di messaggi su canali

Semantica operazionale strutturale

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

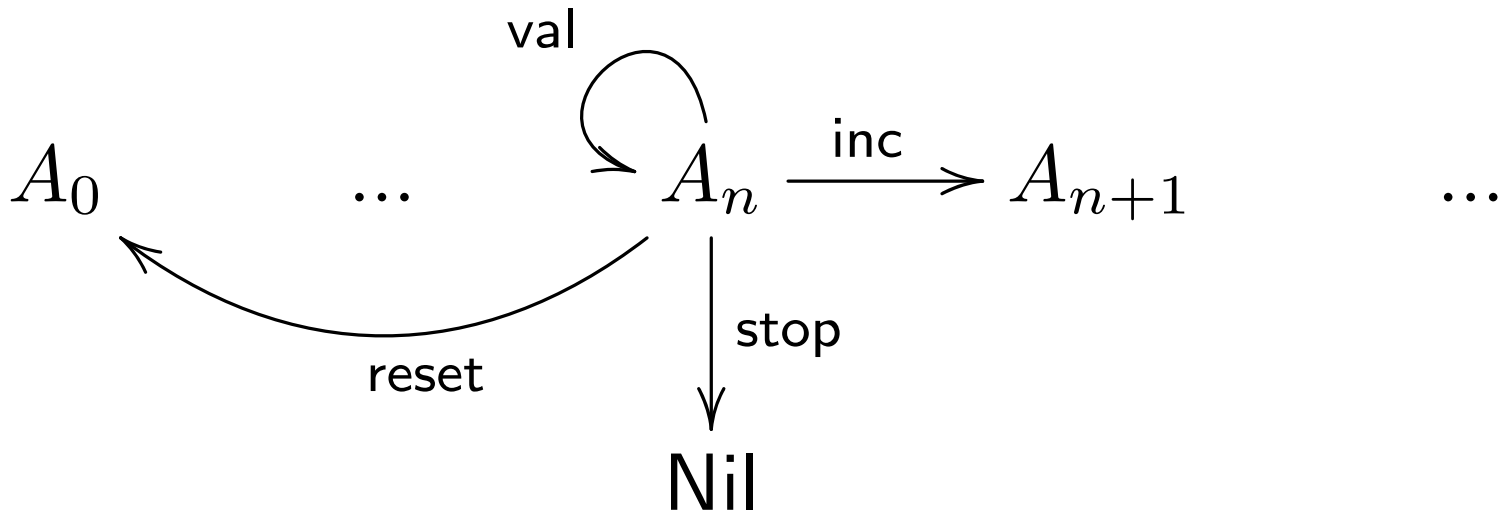
Transizioni con etichette



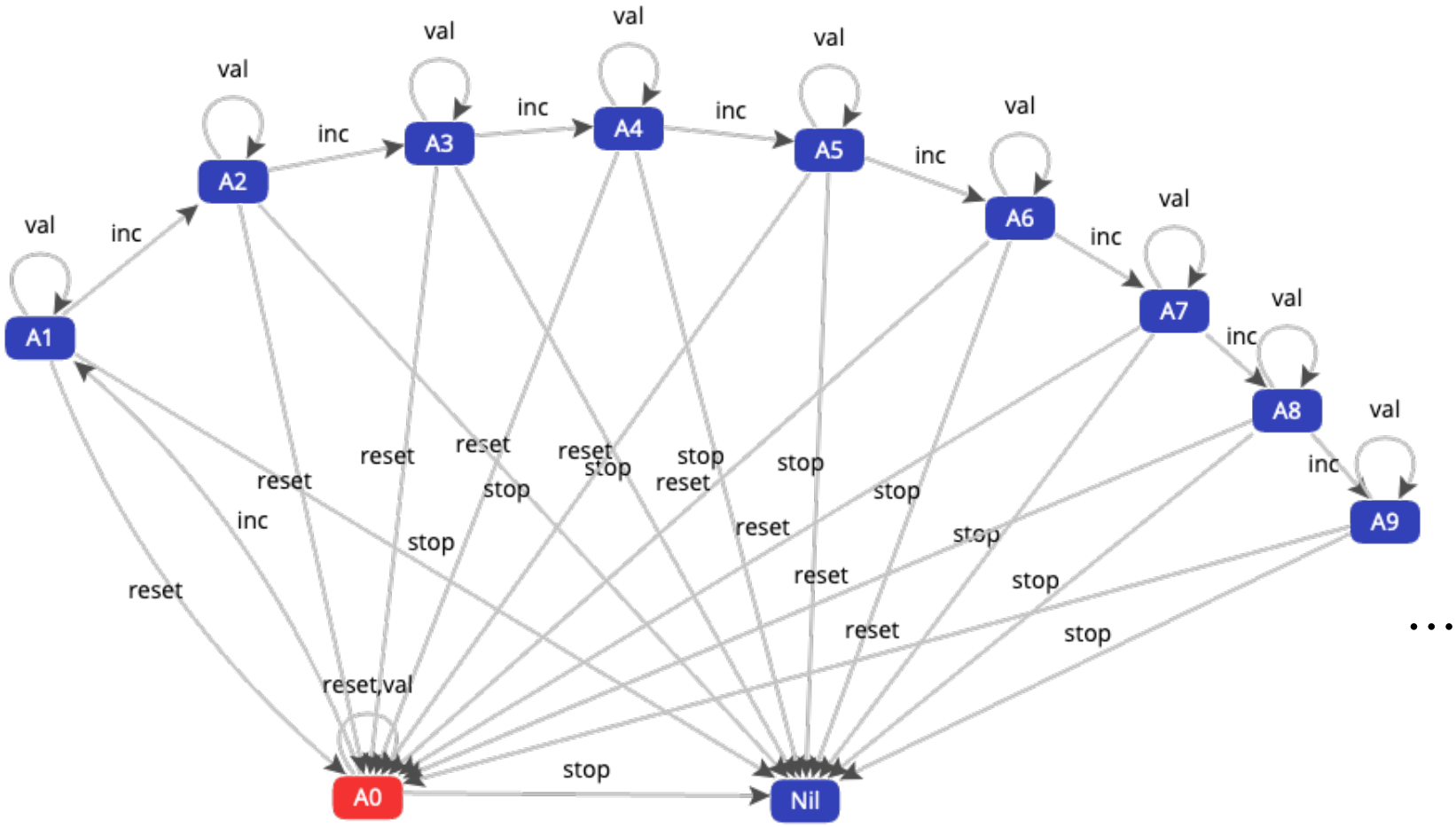
Esempio: processo contatore

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



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 (ad esempio un ingresso) $\alpha?v$ — ricevuto v sul **canale** α

un'azione interna (azione silenziosa) τ
(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 $\overline{\alpha}$

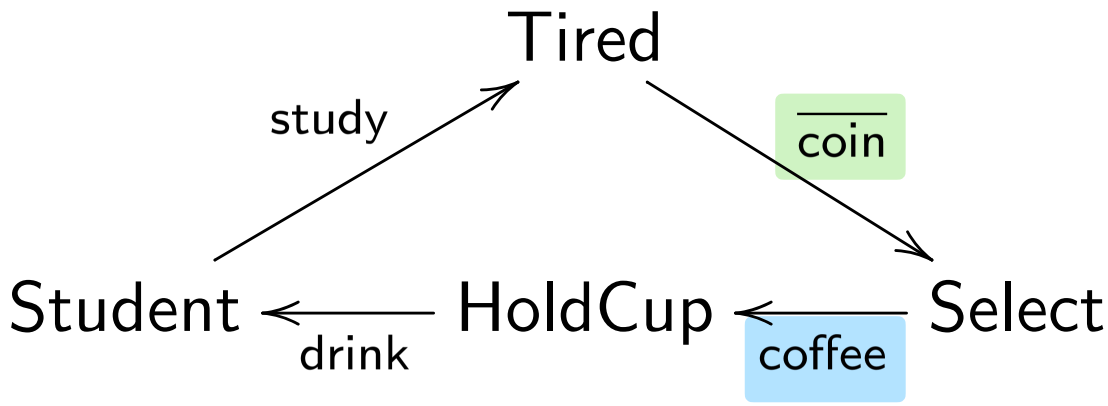
$\alpha?v$ diventa solo α_v o solo α

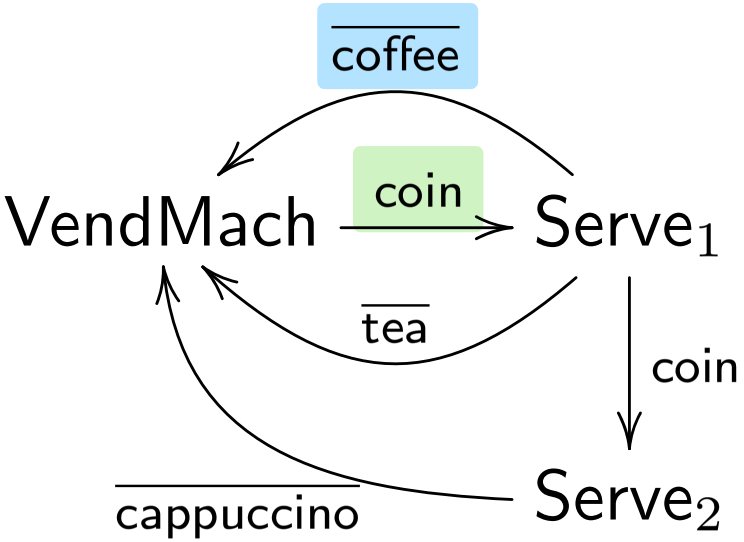
λ denota o α o $\overline{\alpha}$

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

CCS: comunicazione

Esempio: distributore automatico





Sintassi CCS

CCS: sintassi

p, q	$::=$	$\mathbf{nil}$	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)
	$\mid$	$p \mid q$	composizione parallela
	$\mid$	$\mathbf{rec}\ x.\ p$	ricorsione

(gli operatori sono elencati in ordine di precedenza)

CCS: sintassi

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

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

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 operatore che lega le variabili e' la 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-ridenominati rispetto alle variabili vincolate

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

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

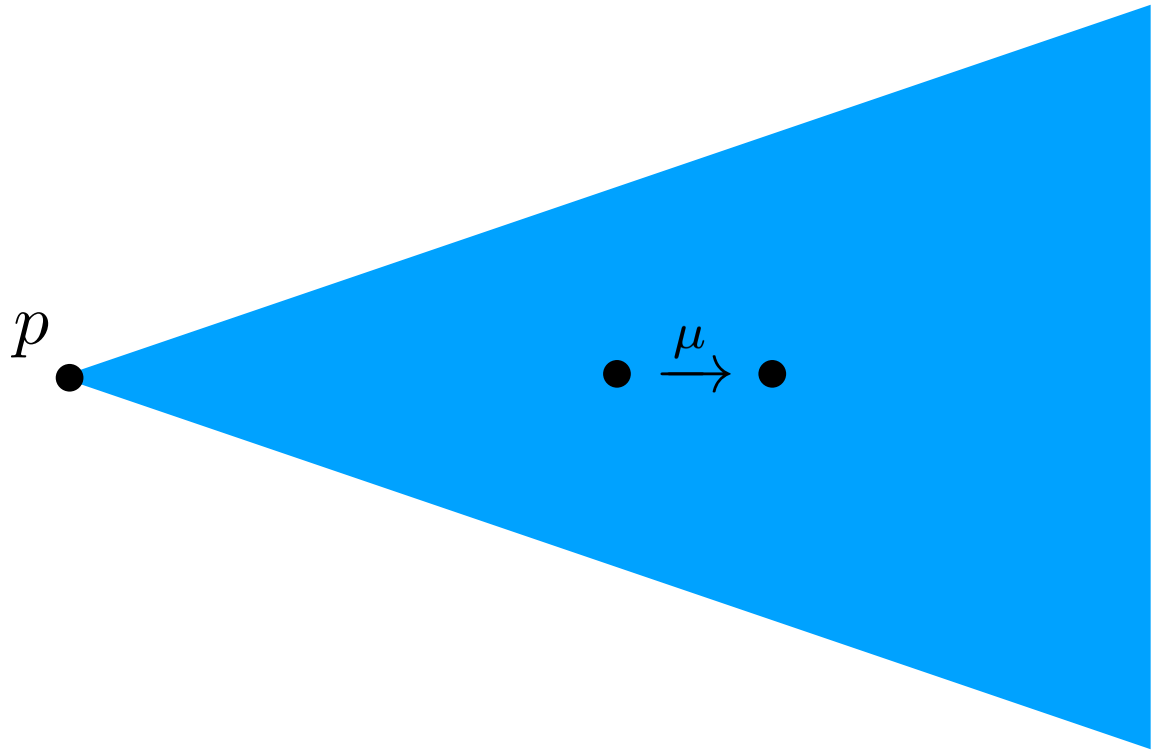
$\Lambda = \mathcal{C} \cup \overline{\mathcal{C}}$ insieme di azioni osservabili, denotate da $\lambda\ \overline{\lambda}$

$\tau \notin \Lambda$ 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

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

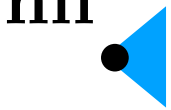
il processo inattivo non fa nulla

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

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

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

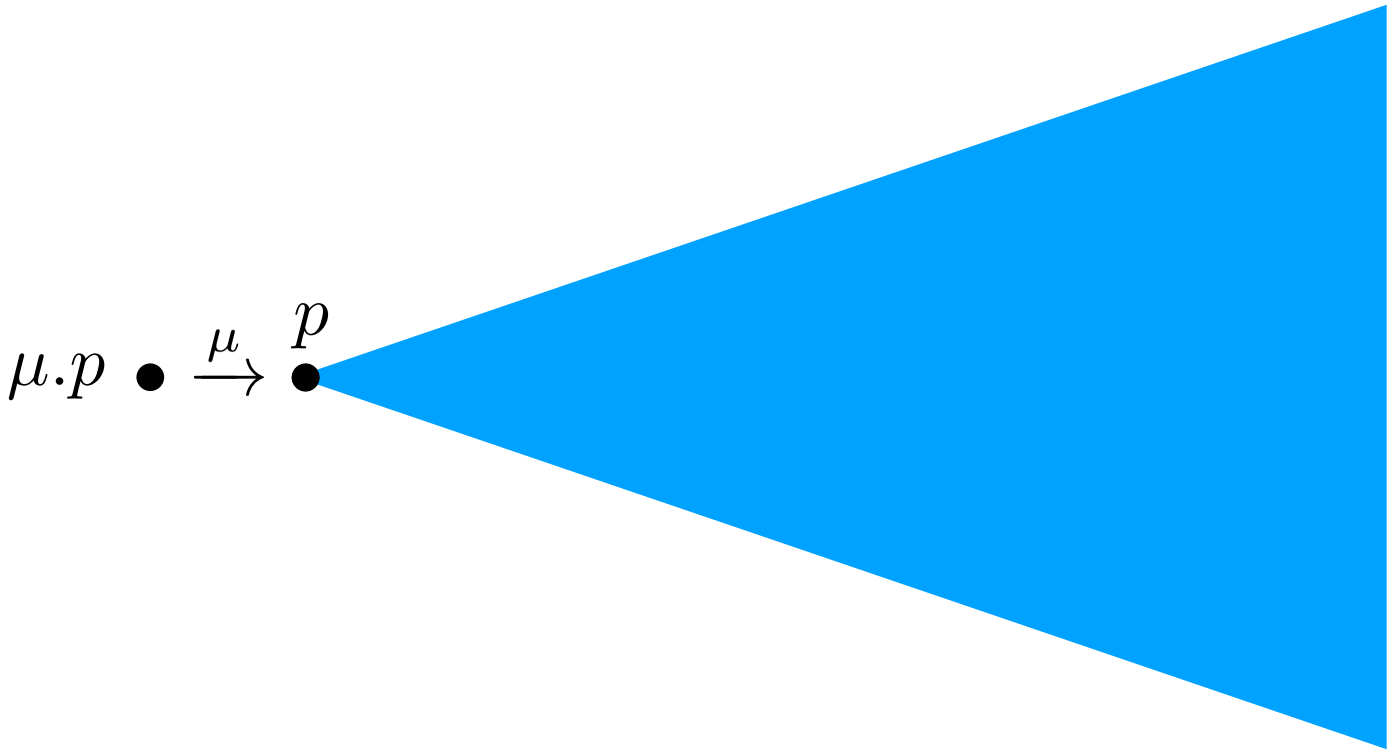
l'azione può comportare un'interazione con l'ambiente

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

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



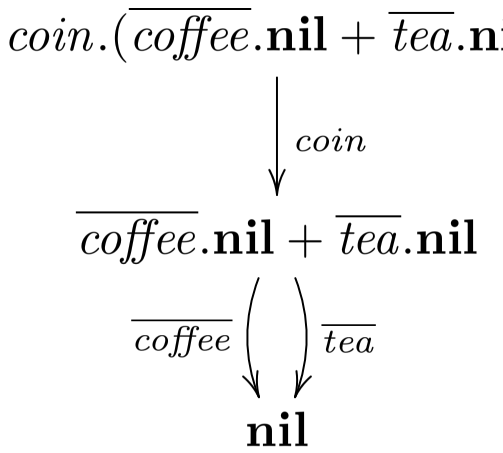
Scelta non deterministica

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

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

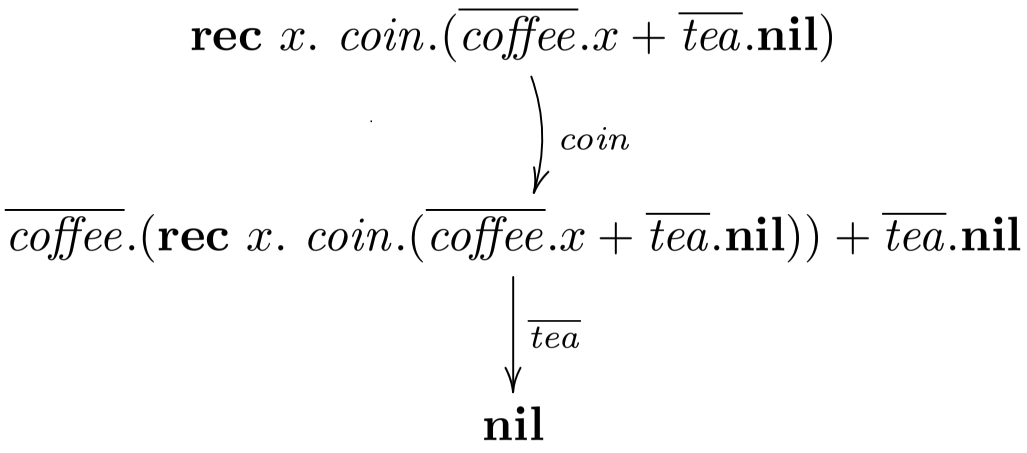
Ricorsione

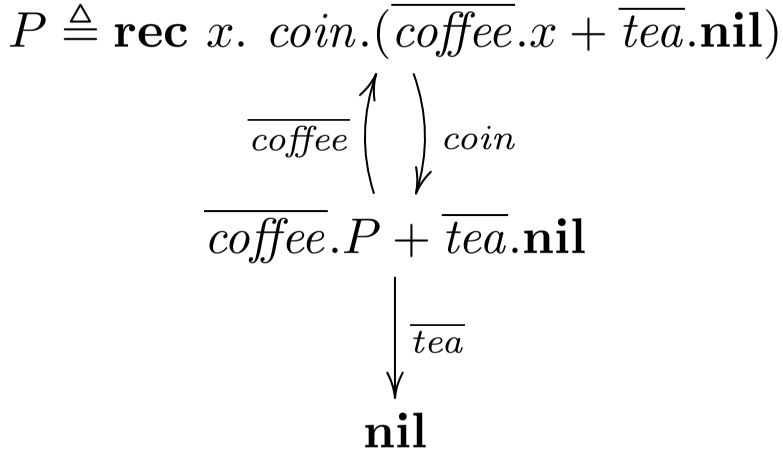
$$\text{Rec)}\ \frac{p\left[{}^{\mathbf{rec}\ x.\ p}/_{x}\right] \xrightarrow{\mu} q}{\mathbf{rec}\ x.\ p \xrightarrow{\mu} q}$$

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

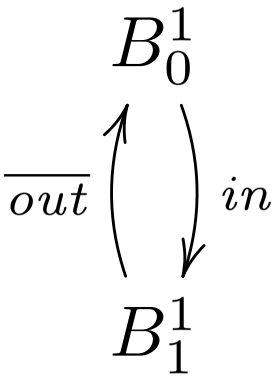
$$\text{Const)}\ \frac{A \triangleq p \in \Delta \quad p \xrightarrow{\mu} q}{A \xrightarrow{\mu} q}$$

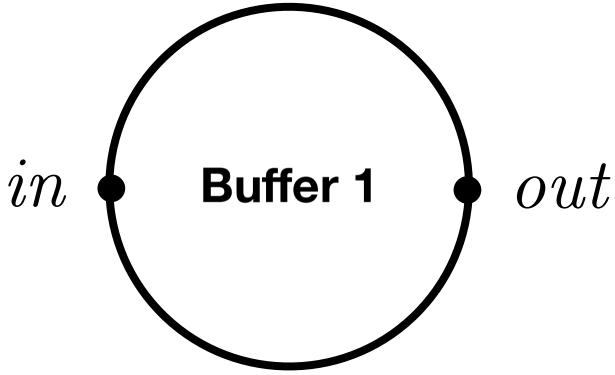
$$P \triangleq coin.(\overline{coffee}.P + \overline{tea}.\mathbf{nil})$$

CCS: buffer di capacita' 1

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

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



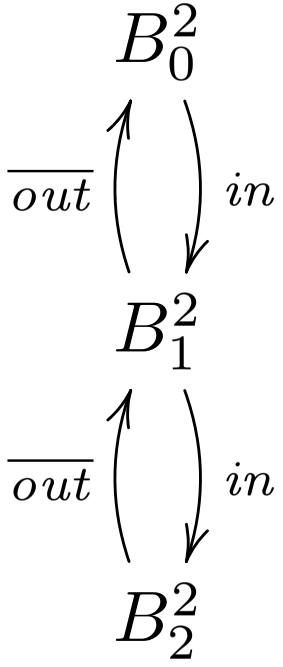


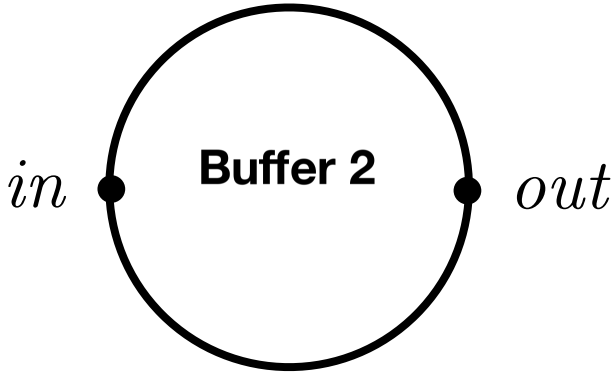
CCS: buffer di capacita' 2

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

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

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



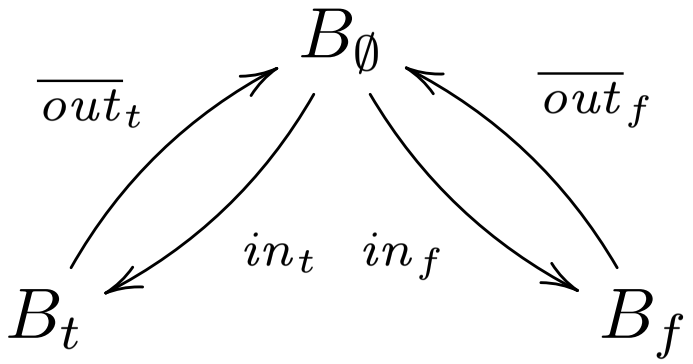


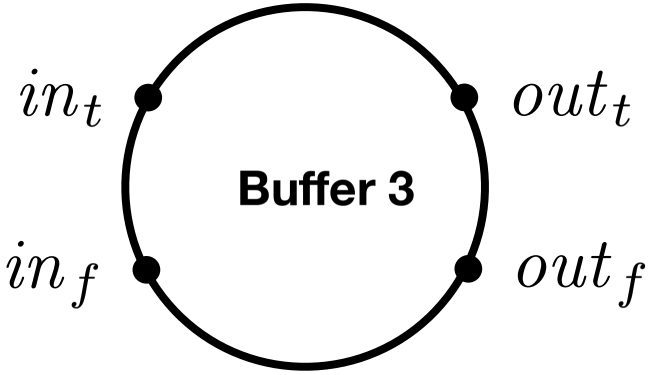
CCS: buffer booleano

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

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

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





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

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

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

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

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

LTS del processo

CCS: buffer parallelo

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

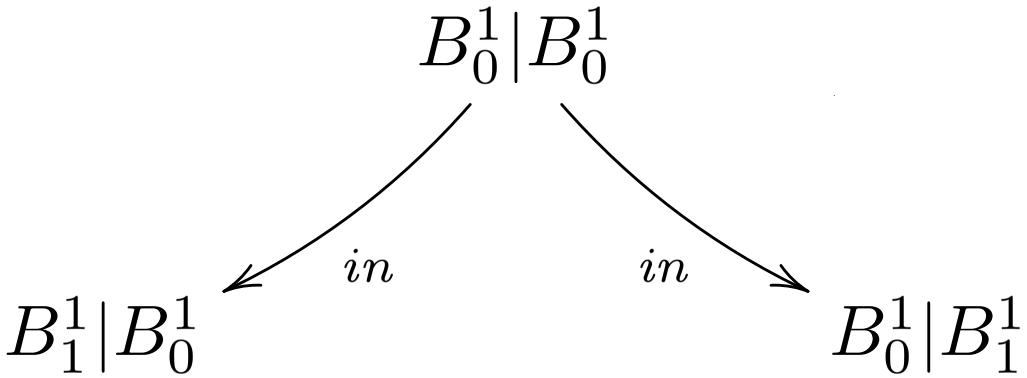
$$B_1^1 \triangleq \overline{out}.B_0^1$$

$$B_0^1 | B_0^1$$

CCS: buffer parallelo

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

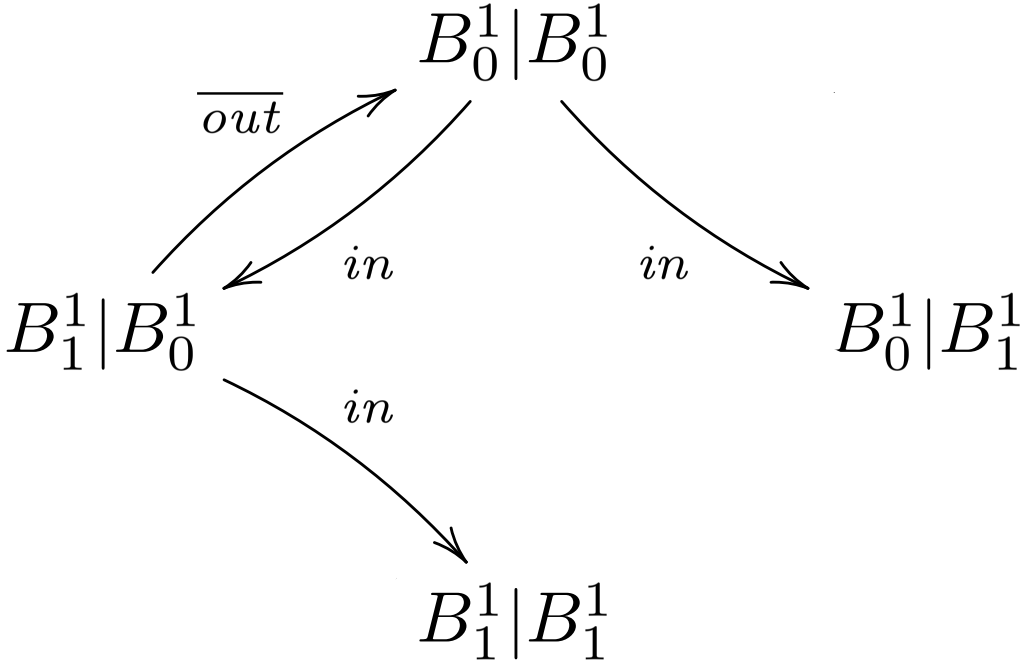
$$B_1^1 \triangleq \overline{out}.B_0^1$$



CCS: buffer parallelo

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

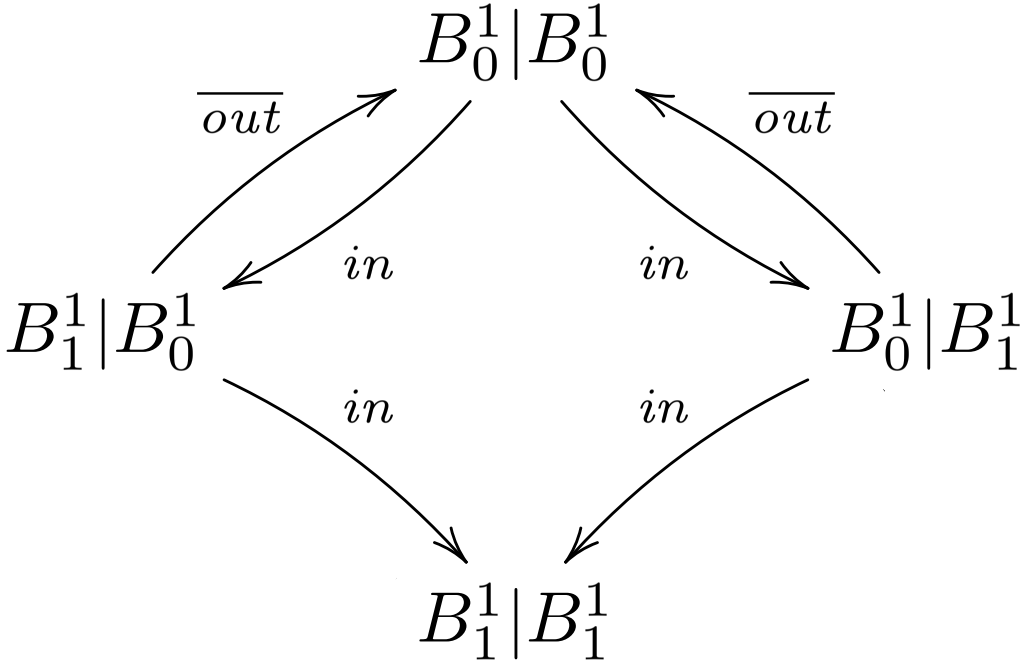
$$B_1^1 \triangleq \overline{out}.B_0^1$$



CCS: buffer parallelo

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

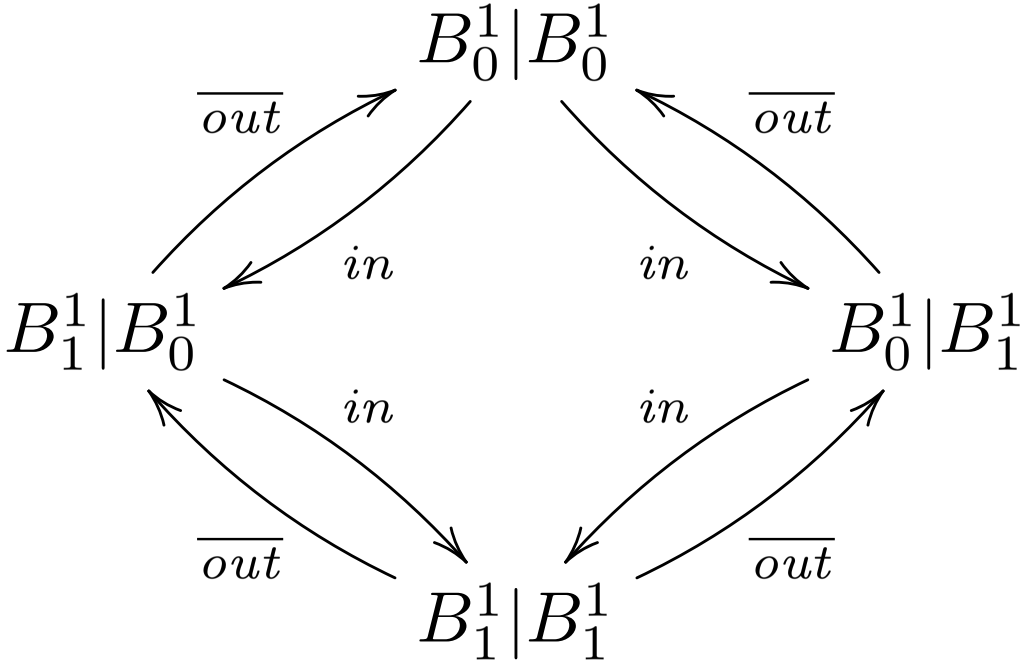
$$B_1^1 \triangleq \overline{out}.B_0^1$$

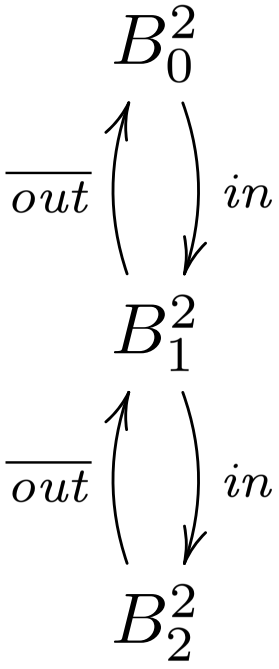


CCS: buffer paralleli

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

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





confrontare con il buffer a capacità 2