Un modello fully abstract del PCF

Grilletti Gianluca Barbarino Giovanni

Università di Pisa

December 3, 2014

I tipi di PCF

I tipi di PCF sono definiti ricorsivamente a partire dalle seguenti clausole:

- *Nat* e *Bool* sono tipi (*i tipi base*)
- Se S e T sono tipi, $S \times T$ è un tipo
- Se S e T sono tipi, $S \to T$ è un tipo

Esempio

- $Nat \times Nat$
- $(Nat \times Bool) \to Bool$
- $Nat \to Nat \to Nat$ (da intendere $Nat \to (Nat \to Nat)$)
- $(Nat \to Nat) \to Nat \to Nat$

Grammatica per generare i termini di PCF

$$< nat\_exp > ::= \underline{0}|\underline{1}|\underline{2}| \dots | < nat\_exp > + < nat\_exp >$$

$$< bool\_exp > ::= true|false|Eq? < nat\_exp >< nat\_exp >$$

$$< \sigma \to \tau\_exp > ::= \lambda(x : \sigma). < \tau\_exp >$$

$$< \sigma \times \tau\_exp > ::=<< \sigma\_exp >, < \tau\_exp >>$$

$$\begin{aligned} < \sigma\_exp > ::=&< \sigma\_var > | \\ & if < bool\_exp > then < \sigma\_exp > else < \sigma\_exp > | \\ & < \sigma\_application > | < \sigma\_projection > | < \sigma\_fixed\_point > \\ < \sigma\_application > ::=&< \tau \to \sigma\_exp >< \tau\_exp > \\ < \sigma\_projection > ::=& Proj_1 < \sigma \times \tau\_exp > |Proj_2 < \tau \times \sigma\_exp > \\ < \sigma\_fixed\_point > ::=& Y_\sigma < \sigma \to \sigma\_exp > \end{aligned}$$

Con $t : T$ indichiamo che il termine t è di tipo T

Esempio

- $(\underline{n} + \underline{m}) + \underline{n} : Nat$
- $Eq?(\underline{n})(\underline{m}) : Bool$
- $< true, \underline{n} >: Bool \times Nat$
- $Proj_1 < true, \underline{n} >: Bool$
- $\lambda(x : Nat).x + 1 : Nat \to Nat$ (indichiamolo con $Succ$)
- $Succ(\underline{n}) : Nat$
- $if[Eq?(\underline{n})(\underline{m})] \quad then[\underline{n}] \quad else[Succ]$ non è ben formato
- $if[Eq?(\underline{n})(\underline{m})] \quad then[\underline{n}] \quad else[Succ(\underline{n})] : Nat$
- $\lambda(x : Nat).if[Eq?(\underline{0})(x)] \quad then[true] \quad else[false] : Nat \to Bool$ (Indichiamolo con $IsZero$)
- $Y[Succ] : Nat$
- $Y[IsZero]$ non è ben formato

Programmi

Un programma di PCF è un termine:

- ben formato
- chiuso
- di tipo *Nat* o *Bool* (tipi *osservabili*)

Esempio

- $Eq?(\underline{n})(\underline{m}) : Bool$ è un programma
- $Y[Succ] : Nat$ è un programma
- $Succ : Nat \to Nat$ *non* è un programma (tipo non osservabile)
- $x + \underline{n} : Nat$ *non* è un programma (non è chiuso)

Semantica operazionale

Diamo le seguenti regole di riduzione:

add $\underline{n} + \underline{m} \to \underline{n+m}$

Eq? $Eq?(\underline{n})(\underline{n}) \to true$

$Eq?(\underline{n})(\underline{m}) \to false$ (per n ed m distinti)

cond $if[true] \quad then[M] \quad else[N] \to M$

$if[false] \quad then[M] \quad else[N] \to N$

proj $Proj_1 < M, N > \to M$

$Proj_2 < M, N > \to N$

α $\lambda(x : \sigma).M \to \lambda(y : \sigma).[y/x]M$ (con y non libera in M)

β $[\lambda(x : \sigma).M](N) \to [N/x]M$

Y $Y_\sigma \to \lambda(f : \sigma \to \sigma).f(Y_\sigma f)$

- Indichiamo con $\twoheadrightarrow$ la chiusura transitiva della relazione $\rightarrow$
- Diciamo che un termine N è in forma normale se non può essere ridotto tramite le regole sopra introdotte
- Dato un termine M, diciamo che la sua valutazione rispetto alla semantica operazionale è N se
  - N è in forma normale
  - $M \twoheadrightarrow N$

  E lo indichiamo con $Eval(M) = N$

Proprietà di Church-Rosser

Se $M \twoheadrightarrow N_1$ e $M \twoheadrightarrow N_2$, allora esiste P tale che $N_1 \twoheadrightarrow P$ e $N_2 \twoheadrightarrow P$

Questo risultato assicura l'unicità della valutazione Non sempre però un termine ha una forma normale, in questo caso scriviamo $Eval(M) = undef$

Esempio

$$\begin{aligned} Y(\lambda x.5) &\twoheadrightarrow [\lambda x.5](Y(\lambda x.5)) \\ &\twoheadrightarrow 5 \end{aligned}$$

Quindi $Eval(Y(\lambda x.5)) = 5$

Esempio

Posto $F \equiv \lambda f.\lambda x.f(x+1)$

$$\begin{aligned} Y(F) &\twoheadrightarrow F(Y(F)) \\ &\twoheadrightarrow \lambda x.Y(F)(x+1) \\ &\twoheadrightarrow \lambda x.[\lambda y.Y(F)(y+1)](x+1) \\ &\twoheadrightarrow \lambda x.Y(F)(x+2) \\ &\twoheadrightarrow \ldots \end{aligned}$$

Si può mostrare che non esistono riduzioni ad una forma normale; quindi $Eval(Y(F)) = undef$

Equivalenza osservazionale

Definiamo un *contesto* come un termine in cui compare un "buco" indicato con $[\,]$

Esempio

$C[\,] \equiv \lambda(x : Nat).x + [\,]$
Porre il termine $\underline{n}$ nel contesto $C[\,]$ significa considerare il termine $C[\underline{n}] \equiv \lambda(x : Nat).x + \underline{n}$

Diciamo che due termini M ed N sono osservazionalmente equivalenti se per ogni contesto $C[\,]$ si ha $Eval(C[M]) = Eval(C[N])$ e lo indichiamo con $M \stackrel{obs}{=} N$

Espressività di PCF

Diciamo che una funzione parziale $f : \mathbb{N} \to \mathbb{N}$ è *calcolabile* se esiste un programma per computer[1] P tale che:

- Se $f(n) = m$, allora il programma P con input n termina con output m
- Se $f(n)$ non è definita, allora il programma P con input n non termina

Teorema della Fermata

Non esiste un algoritmo per capire se un generico programma termini o meno

[1] Con computer si intende una macchina a registri (URM); idealmente, un computer con infinita memoria

Fatto

Data una funzione parziale calcolabile f, esiste un termine di PCF t tale che

- Se $f(n) = m$, allora $Eval(t(\underline{n})) = \underline{m}$
- Se $f(n)$ non è definito, allora $Eval(t(\underline{n})) = undef$

Fatto

Non esiste un algoritmo per capire se un generico termine di PCF ammetta una forma normale

Fatto

Non esiste un algoritmo per capire se due termini di PCF siano osservazionalmente equivalenti

Full Abstraction

Diciamo che un modello per PCF è Fully Abstract se e solo se per ogni coppia di termini M e N:

$$M \stackrel{\text{obs}}{=} N \Leftrightarrow [\![M]\!] = [\![N]\!]$$

Diciamo che un modello per PCF è intensionally fully abstract se:

- È algebrico
- Gli elementi compatti sono definibili in PCF

Teorema

Dato un modello $\mathcal{I}$ intensionally fully abstract, esiste una relazione di equivalenza $\approx$ tale che $\mathcal{E} = \mathcal{I}/\approx$ sia un modello fully abstract

A questo punto vorremmo un modello per PCF tale che:

1. Sia fully abstract
2. Il modello sia *definibile* (cioè ogni elemento del modello sia interpretazione di un termine di PCF)
3. Il modello sia *minimale* (cioè esista una "immersione" in ogni altro modello fully abstract)

I giochi

Il modello che andremo a considerare si basa sulla *teoria dei giochi*.
Un **gioco** è una 4-upla $A = (M_A, \lambda_A, P_A, \approx_A)$ dove:

I giochi

Il modello che andremo a considerare si basa sulla *teoria dei giochi*.
Un **gioco** è una 4-upla $A = (M_A, \lambda_A, P_A, \approx_A)$ dove:

- M_A è l'insieme delle mosse

I giochi

Il modello che andremo a considerare si basa sulla *teoria dei giochi*.
Un **gioco** è una 4-upla $A = (M_A, \lambda_A, P_A, \approx_A)$ dove:

- M_A è l'insieme delle mosse
- λ_A è una funzione da M_A all'insieme $\{O, P\} \times \{Q, A\}$; in particolare:
    - O indica il giocatore "opponent" e P il giocatore "player"
    - Q indica una domanda e A una risposta

I giochi

Il modello che andremo a considerare si basa sulla *teoria dei giochi*.
Un **gioco** è una 4-upla $A = (M_A, \lambda_A, P_A, \approx_A)$ dove:

- M_A è l'insieme delle mosse
- λ_A è una funzione da M_A all'insieme $\{O, P\} \times \{Q, A\}$; in particolare:
  - O indica il giocatore "opponent" e P il giocatore "player"
  - Q indica una domanda e A una risposta
- Una *partita* è una stringa finita di mosse tale che:
  1. La prima mossa è di O
  2. P e O si alternano
  3. In ogni sottostringa iniziale, il numero di risposte deve essere al più uguale al numero di domande (*bracketing condition*)

I giochi

Il modello che andremo a considerare si basa sulla *teoria dei giochi*.
Un **gioco** è una 4-upla $A = (M_A, \lambda_A, P_A, \approx_A)$ dove:

- M_A è l'insieme delle mosse
- λ_A è una funzione da M_A all'insieme $\{O, P\} \times \{Q, A\}$; in particolare:
  - O indica il giocatore "opponent" e P il giocatore "player"
  - Q indica una domanda e A una risposta
- Una *partita* è una stringa finita di mosse tale che:
  1. La prima mossa è di O
  2. P e O si alternano
  3. In ogni sottostringa iniziale, il numero di risposte deve essere al più uguale al numero di domande (*bracketing condition*)
- P_A è un sottoinsieme prefix-closed di partite; chiameremo P_A l'insieme delle *partite valide*

I giochi

Il modello che andremo a considerare si basa sulla *teoria dei giochi*.
Un **gioco** è una 4-upla $A = (M_A, \lambda_A, P_A, \approx_A)$ dove:

- M_A è l'insieme delle mosse
- λ_A è una funzione da M_A all'insieme $\{O, P\} \times \{Q, A\}$; in particolare:
  - O indica il giocatore "opponent" e P il giocatore "player"
  - Q indica una domanda e A una risposta
- Una *partita* è una stringa finita di mosse tale che:
  1. La prima mossa è di O
  2. P e O si alternano
  3. In ogni sottostringa iniziale, il numero di risposte deve essere al più uguale al numero di domande (*bracketing condition*)
- P_A è un sottoinsieme prefix-closed di partite; chiameremo P_A l'insieme delle *partite valide*
- $\approx_A$ è una relazione di equivalenza sulle partite valide

Tavolo di Gioco

Possiamo rappresentare partite accettabili tramite il *Tavolo di Gioco*

Tavolo di Gioco

Possiamo rappresentare partite accettabili tramite il *Tavolo di Gioco*

Gioco $A = (M_A, \lambda_A, P_A, \approx_A)$

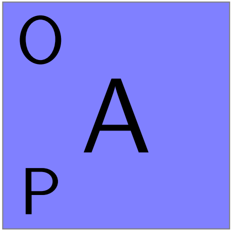
- $M_A = \{*_1, *_2, n_1, n_2\}$
- $\lambda_A = \{(*_1, OQ);$
  $(*_2, PQ); (n_1, PA); (n_2, OA)\}$
- $P_A = \{\epsilon,\ *_1,\ *_1 n_1,$
  $*_1 *_2,\ *_1 *_2\ n_2,\ *_1 *_2\ n_2 n_1\}$
- $\approx_A = id_A$

Tavolo di Gioco

Possiamo rappresentare partite accettabili tramite il *Tavolo di Gioco*

Gioco $A = (M_A, \lambda_A, P_A, \approx_A)$

- $M_A = \{*_1, *_2, n_1, n_2\}$
- $\lambda_A = \{(*_1, OQ);$
  $(*_2, PQ); (n_1, PA); (n_2, OA)\}$
- $P_A = \{\epsilon,\ *_1,\ *_1 n_1,$
  $*_1 *_2,\ *_1 *_2\ n_2,\ *_1 *_2\ n_2 n_1\}$
- $\approx_A = id_A$

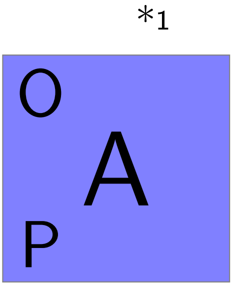


Tavolo di Gioco

Possiamo rappresentare partite accettabili tramite il *Tavolo di Gioco*

Gioco $A = (M_A, \lambda_A, P_A, \approx_A)$

- $M_A = \{*_1, *_2, n_1, n_2\}$
- $\lambda_A = \{(*_1, OQ);$
  $(*_2, PQ); (n_1, PA); (n_2, OA)\}$
- $P_A = \{\epsilon, *_1, *_1 n_1,$
  $*_1 *_2, *_1 *_2 n_2, *_1 *_2 n_2 n_1\}$
- $\approx_A = id_A$

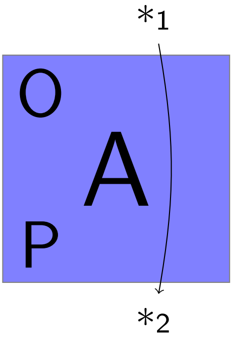


Tavolo di Gioco

Possiamo rappresentare partite accettabili tramite il *Tavolo di Gioco*

Gioco $A = (M_A, \lambda_A, P_A, \approx_A)$

- $M_A = \{*_1, *_2, n_1, n_2\}$
- $\lambda_A = \{(*_1, OQ);$
  $(*_2, PQ); (n_1, PA); (n_2, OA)\}$
- $P_A = \{\epsilon,\ *_1,\ *_1 n_1,$
  $\color{red}{*_1 *_2}$$,\ *_1 *_2\ n_2,\ *_1 *_2\ n_2 n_1\}$
- $\approx_A = id_A$

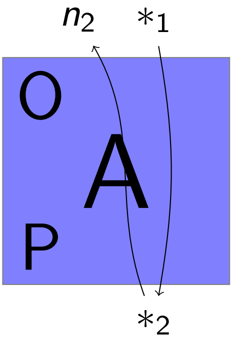


Tavolo di Gioco

Possiamo rappresentare partite accettabili tramite il *Tavolo di Gioco*

Gioco $A = (M_A, \lambda_A, P_A, \approx_A)$

- $M_A = \{*_1, *_2, n_1, n_2\}$
- $\lambda_A = \{(*_1, OQ);$
  $(*_2, PQ); (n_1, PA); (n_2, OA)\}$
- $P_A = \{\epsilon,\ *_1,\ *_1 n_1,$
  $*_1 *_2,\ *_1\ *_2\ n_2,\ *_1\ *_2\ n_2 n_1\}$
- $\approx_A = id_A$

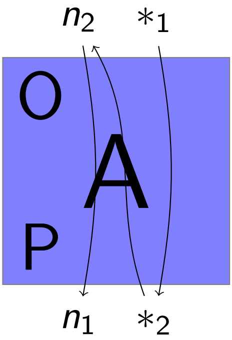


Tavolo di Gioco

Possiamo rappresentare partite accettabili tramite il *Tavolo di Gioco*

Gioco $A = (M_A, \lambda_A, P_A, \approx_A)$

- $M_A = \{*_1, *_2, n_1, n_2\}$
- $\lambda_A = \{(*_1, OQ);$ $(*_2, PQ); (n_1, PA); (n_2, OA)\}$
- $P_A = \{\epsilon,\ *_1,\ *_1 n_1,$ $*_1 *_2,\ *_1\ *_2\ n_2,\ *_1\ *_2\ n_2 n_1\}$
- $\approx_A = id_A$

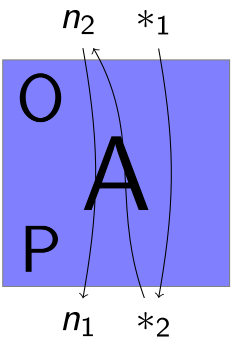


Tavolo di Gioco

Possiamo rappresentare partite accettabili tramite il *Tavolo di Gioco*

Gioco $A = (M_A, \lambda_A, P_A, \approx_A)$

- $M_A = \{*_1, *_2, n_1, n_2\}$
- $\lambda_A = \{(*_1, OQ);$
  $(*_2, PQ); (n_1, PA); (n_2, OA)\}$
- $P_A = \{\epsilon,\ *_1,\ *_1 n_1,$
  $*_1 *_2,\ *_1\ *_2\ n_2,\ *_1\ *_2\ n_2 n_1\}$
- $\approx_A = id_A$



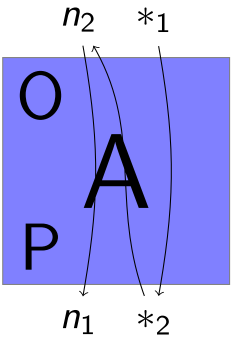
Bracketing condition:
$$\begin{array}{cccc} *_1 & *_2 & n_2 & n_1 \\ \downarrow & \downarrow & \downarrow & \downarrow \\ (& (&) &) \end{array}$$

Tavolo di Gioco

Possiamo rappresentare partite accettabili tramite il *Tavolo di Gioco*

Gioco $A = (M_A, \lambda_A, P_A, \approx_A)$

- $M_A = \{*_1, *_2, n_1, n_2\}$
- $\lambda_A = \{(*_1, OQ); (*_2, PQ); (n_1, PA); (n_2, OA)\}$
- $P_A = \{\epsilon,\ *_1,\ *_1 n_1,\ *_1 *_2,\ *_1\ *_2\ n_2,\ {\color{red}*_1\ *_2\ n_2 n_1}\}$
- $\approx_A = id_A$



Bracketing condition: $\begin{array}{cccc} *_1 & *_2 & n_2 & n_1 \\ \downarrow & \downarrow & \downarrow & \downarrow \\ (& (&) &) \end{array}$

Ad ogni risposta è associata naturalmente una domanda

Strategie

Una **strategia** σ è un insieme di partite di lunghezza pari (l'ultima mossa è di P) tali che:

Strategie

Una **strategia** σ è un insieme di partite di lunghezza pari (l'ultima mossa è di P) tali che:

- σ sia prefix-closed sulle partite di lunghezza pari

Strategie

Una **strategia** σ è un insieme di partite di lunghezza pari (l'ultima mossa è di P) tali che:

- σ sia prefix-closed sulle partite di lunghezza pari
- σ sia *history free*, cioè
  - $sab, tac \in \sigma \Rightarrow b = c$
  - $sab, t \in \sigma, ta \in P_A \Rightarrow tab \in \sigma$

Strategie

Una **strategia** σ è un insieme di partite di lunghezza pari (l'ultima mossa è di P) tali che:

- σ sia prefix-closed sulle partite di lunghezza pari
- σ sia *history free*, cioè
  - $sab, tac \in \sigma \Rightarrow b = c$
  - $sab, t \in \sigma, ta \in P_A \Rightarrow tab \in \sigma$

La condizione di history free, rende le strategie esprimibili tramite una *funzione parziale* a loro associata:

$$f : M^O \rightharpoonup M^P$$

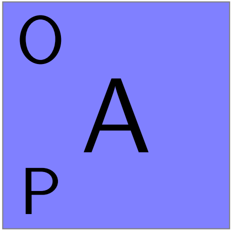
$$sab \in \sigma \Rightarrow f_\sigma(a) = b$$

$$X = \{ab \mid f(a) = b\} \rightarrow \sigma_f = < X >$$

Strategie

Gioco $A = (M_A, \lambda_A, P_A, \approx_A)$

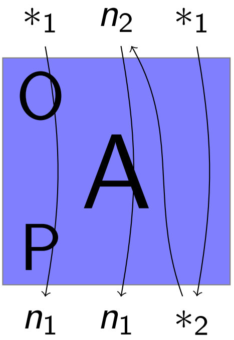
- $M_A = \{*_1, *_2, n_1, n_2\}$
- $\lambda_A = \{(*_1, OQ);$
  $(*_2, PQ); (n_1, PA); (n_2, OA)\}$
- $P_A = \{\epsilon,\ *_1,\ *_1 n_1,$
  $*_1 *_2,\ *_1 *_2\ n_2,\ *_1 *_2\ n_2 n_1\}$
- $\approx_A = id_A$



Strategie

Gioco $A = (M_A, \lambda_A, P_A, \approx_A)$

- $M_A = \{*_1, *_2, n_1, n_2\}$
- $\lambda_A = \{(*_1, OQ);$
  $(*_2, PQ); (n_1, PA); (n_2, OA)\}$
- $P_A = \{\epsilon,\ *_1,\ *_1 n_1,$
  $*_1 *_2,\ *_1 *_2\ n_2,\ *_1 *_2\ n_2 n_1\}$
- $\approx_A = id_A$



Esempi di strategie:

$$\sigma = \{\epsilon, *_1 n_1\} \leftrightarrow f_\sigma(*_1) = n_1$$
$$\tau = \{\epsilon,\ *_1 *_2,\ *_1 *_2\ n_2 n_1\} \leftrightarrow f_\tau(*_1) = *_2,\ f_\tau(n_2) = n_1$$

Ordine fra strategie

Estendiamo la relazione $\approx$ alle strategie, ponendo:

Ordine fra strategie

Estendiamo la relazione $\approx$ alle strategie, ponendo:

- $\underline{\sigma \precsim_s \tau}$ se per ogni $sab \in \sigma$ e $s' \in \tau$ t.c. $sa \approx s'a'$, esiste b' tale che $s'a'b' \in \tau$ e $sab \approx s'a'b'$
- $\underline{\sigma \approx_s \tau}$ sse $\sigma \precsim_s \tau \wedge \tau \precsim_s \sigma$

Ordine fra strategie

Estendiamo la relazione $\approx$ alle strategie, ponendo:

- $\underline{\sigma \precsim_s \tau}$ se per ogni $sab \in \sigma$ e $s' \in \tau$ t.c. $sa \approx s'a'$, esiste b' tale che $s'a'b' \in \tau$ e $sab \approx s'a'b'$
- $\underline{\sigma \approx_s \tau}$ sse $\sigma \precsim_s \tau \wedge \tau \precsim_s \sigma$

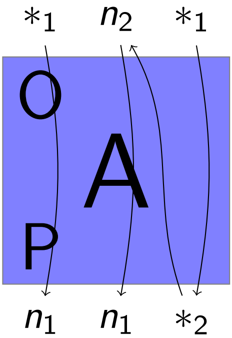
In particolare, la definizione porta alcune conseguenze:

- $\precsim_s$ è un preordine sulle strategie; di conseguenza $\approx_s$ è una relazione di equivalenza parziale
- Nel caso l'equivalenza $\approx$ del gioco sia l'identità, l'ordine tra strategie si può vedere come inclusione di insiemi o tra le funzioni parziali. Intuitivamente, $\sigma \precsim_s \tau$ se τ può fare più mosse di σ

Ordine fra strategie

Gioco $A = (M_A, \lambda_A, P_A, \approx_A)$

- $M_A = \{*_1, *_2, n_1, n_2\}$
- $\lambda_A = \{(*_1, OQ);$
  $(*_2, PQ); (n_1, PA); (n_2, OA)\}$
- $P_A = \{\epsilon,\ *_1,\ *_1 n_1,$
  $*_1 *_2,\ *_1\ *_2\ n_2,\ *_1\ *_2\ n_2 n_1\}$
- $\approx_A = id_A$



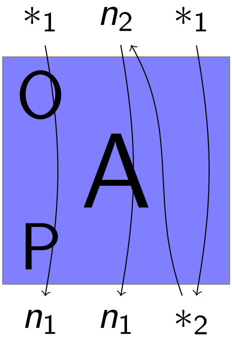
$$\sigma = \{\epsilon, *_1 n_1\} \leftrightarrow f_\sigma(*_1) = n_1$$

$$\tau = \{\epsilon,\ *_1 *_2,\ *_1\ *_2\ n_2 n_1\} \leftrightarrow f_\tau(*_1) = *_2,\ f_\tau(n_2) = n_1$$

Ordine fra strategie

Gioco $A = (M_A, \lambda_A, P_A, \approx_A)$

- $M_A = \{*_1, *_2, n_1, n_2\}$
- $\lambda_A = \{(*_1, OQ);$
  $(*_2, PQ); (n_1, PA); (n_2, OA)\}$
- $P_A = \{\epsilon,\ *_1,\ *_1 n_1,$
  $*_1 *_2,\ *_1\ *_2\ n_2,\ *_1\ *_2\ n_2 n_1\}$
- $\approx_A = id_A$



$$\sigma = \{\epsilon, *_1 n_1\} \leftrightarrow f_\sigma(*_1) = n_1$$

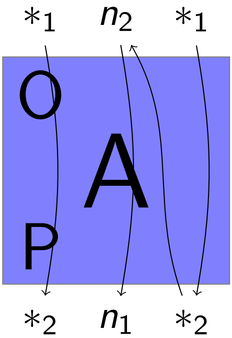
$$\tau = \{\epsilon,\ *_1 *_2,\ *_1\ *_2\ n_2 n_1\} \leftrightarrow f_\tau(*_1) = *_2,\ f_\tau(n_2) = n_1$$

$$\sigma \not\preccurlyeq_s \tau,\ \sigma \not\preccurlyeq_s \tau$$

Ordine fra strategie

Gioco $A = (M_A, \lambda_A, P_A, \approx_A)$

- $M_A = \{*_1, *_2, n_1, n_2\}$
- $\lambda_A = \{(*_1, OQ);$
  $(*_2, PQ); (n_1, PA); (n_2, OA)\}$
- $P_A = \{\epsilon,\ *_1,\ *_1 n_1,$
  $*_1 *_2,\ *_1\ *_2\ n_2,\ *_1\ *_2\ n_2 n_1\}$
- $\approx_A = id_A$

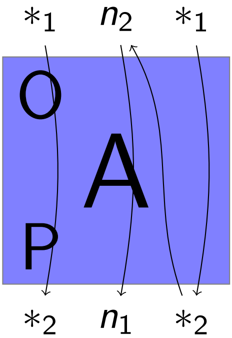


$$\sigma = \{\epsilon,\ *_1 *_2\} \leftrightarrow f_\sigma(*_1) = *_2$$
$$\tau = \{\epsilon,\ *_1 *_2,\ *_1\ *_2\ n_2 n_1\} \leftrightarrow f_\tau(*_1) = *_2,\ f_\tau(n_2) = n_1$$

Ordine fra strategie

Gioco $A = (M_A, \lambda_A, P_A, \approx_A)$

- $M_A = \{*_1, *_2, n_1, n_2\}$
- $\lambda_A = \{(*_1, OQ);$
  $(*_2, PQ); (n_1, PA); (n_2, OA)\}$
- $P_A = \{\epsilon,\ *_1,\ *_1 n_1,$
  $*_1 *_2,\ *_1 *_2\ n_2,\ *_1 *_2\ n_2 n_1\}$
- $\approx_A = id_A$



$$\sigma = \{\epsilon,\ *_1 *_2\} \leftrightarrow f_\sigma(*_1) = *_2$$

$$\tau = \{\epsilon,\ *_1 *_2,\ *_1 *_2\ n_2 n_1\} \leftrightarrow f_\tau(*_1) = *_2,\ f_\tau(n_2) = n_1$$

$$\sigma \preccurlyeq_s \tau$$

Il gioco $A \otimes B$

Dati due giochi A e B definiamo il gioco $A \otimes B$ come:

Il gioco $A \otimes B$

Dati due giochi A e B definiamo il gioco $A \otimes B$ come:

- $M_{A \otimes B} = M_A \coprod M_B$

Il gioco $A \otimes B$

Dati due giochi A e B definiamo il gioco $A \otimes B$ come:

- $M_{A \otimes B} = M_A \coprod M_B$
- $\lambda_{A \otimes B} = \lambda_A \coprod \lambda_B$

Il gioco $A \otimes B$

Dati due giochi A e B definiamo il gioco $A \otimes B$ come:

- $M_{A \otimes B} = M_A \coprod M_B$
- $\lambda_{A \otimes B} = \lambda_A \coprod \lambda_B$
- $P_{A \otimes B}$ sono tutte le partite s tali che $s|_{M_A} \in P_A \wedge s|_{M_B} \in P_B$
  - $s|_{M_A} \in P_A \wedge s|_{M_B} \in P_B$
  - Per ogni domanda in A, la risposta deve essere in A; lo stesso con B

Il gioco $A \otimes B$

Dati due giochi A e B definiamo il gioco $A \otimes B$ come:

- $M_{A\otimes B} = M_A \coprod M_B$
- $\lambda_{A\otimes B} = \lambda_A \coprod \lambda_B$
- $P_{A\otimes B}$ sono tutte le partite s tali che $s|_{M_A} \in P_A \wedge s|_{M_B} \in P_B$
    - $s|_{M_A} \in P_A \wedge s|_{M_B} \in P_B$
    - Per ogni domanda in A, la risposta deve essere in A; lo stesso con B
- $s \approx_{A\otimes B} t \Leftrightarrow s|_A \approx_A t|_A \wedge s|_B \approx_B t|_B \wedge fst(s) = fst(t)$

Il gioco $A \otimes B$

Dati due giochi A e B definiamo il gioco $A \otimes B$ come:

- $M_{A\otimes B} = M_A \coprod M_B$
- $\lambda_{A\otimes B} = \lambda_A \coprod \lambda_B$
- $P_{A\otimes B}$ sono tutte le partite s tali che $s|_{M_A} \in P_A \wedge s|_{M_B} \in P_B$
  - $s|_{M_A} \in P_A \wedge s|_{M_B} \in P_B$
  - Per ogni domanda in A, la risposta deve essere in A; lo stesso con B
- $s \approx_{A\otimes B} t \Leftrightarrow s|_A \approx_A t|_A \wedge s|_B \approx_B t|_B \wedge fst(s) = fst(t)$

Proprietà

Il gioco $A \otimes B$

Dati due giochi A e B definiamo il gioco $A \otimes B$ come:

- $M_{A\otimes B} = M_A \coprod M_B$
- $\lambda_{A\otimes B} = \lambda_A \coprod \lambda_B$
- $P_{A\otimes B}$ sono tutte le partite s tali che $s|_{M_A} \in P_A \wedge s|_{M_B} \in P_B$
    - $s|_{M_A} \in P_A \wedge s|_{M_B} \in P_B$
    - Per ogni domanda in A, la risposta deve essere in A; lo stesso con B
- $s \approx_{A\otimes B} t \Leftrightarrow s|_A \approx_A t|_A \wedge s|_B \approx_B t|_B \wedge fst(s) = fst(t)$

Proprietà

- Il prodotto tensore è associativo

Il gioco $A \otimes B$

Dati due giochi A e B definiamo il gioco $A \otimes B$ come:

- $M_{A\otimes B} = M_A \coprod M_B$
- $\lambda_{A\otimes B} = \lambda_A \coprod \lambda_B$
- $P_{A\otimes B}$ sono tutte le partite s tali che $s|_{M_A} \in P_A \wedge s|_{M_B} \in P_B$
    - $s|_{M_A} \in P_A \wedge s|_{M_B} \in P_B$
    - Per ogni domanda in A, la risposta deve essere in A; lo stesso con B
- $s \approx_{A\otimes B} t \Leftrightarrow s|_A \approx_A t|_A \wedge s|_B \approx_B t|_B \wedge fst(s) = fst(t)$

Proprietà

- Il prodotto tensore è associativo
- Esiste un elemento neutro I, ossia il gioco vuoto

Il gioco $A \otimes B$

Dati due giochi A e B definiamo il gioco $A \otimes B$ come:

- $M_{A\otimes B} = M_A \coprod M_B$
- $\lambda_{A\otimes B} = \lambda_A \coprod \lambda_B$
- $P_{A\otimes B}$ sono tutte le partite s tali che $s|_{M_A} \in P_A \wedge s|_{M_B} \in P_B$
  - $s|_{M_A} \in P_A \wedge s|_{M_B} \in P_B$
  - Per ogni domanda in A, la risposta deve essere in A; lo stesso con B
- $s \approx_{A\otimes B} t \Leftrightarrow s|_A \approx_A t|_A \wedge s|_B \approx_B t|_B \wedge fst(s) = fst(t)$

Proprietà

- Il prodotto tensore è associativo
- Esiste un elemento neutro I, ossia il gioco vuoto
- Solamente il giocatore O può cambiare componente di gioco

Il gioco $A \otimes B$

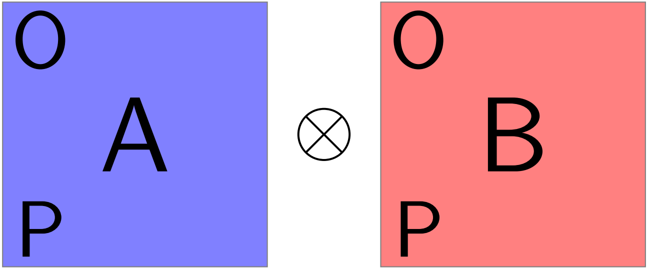
Il gioco $A \otimes B$

- A con $P_A = \{\epsilon, \ *_O, \ \ *_O \checkmark_P, \ *_O \times_P\}$
- B con $P_B = \{\epsilon, \ *_O, \ *_O 0_P, \ *_O 1_P, \ \ *_O 2_P, \ *_O 3_P, \ \ldots\}$

Il gioco $A \otimes B$

- A con $P_A = \{\epsilon,\ *_O,\ *_O \checkmark_P,\ *_O \times_P\}$
- B con $P_B = \{\epsilon,\ *_O,\ *_O 0_P,\ *_O 1_P,\ *_O 2_P,\ *_O 3_P,\ \dots\}$

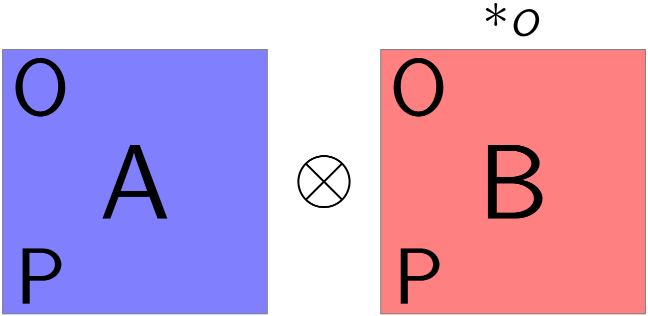
Descriviamo $A \otimes B$ tramite il tavolo di gioco



Il gioco $A \otimes B$

- A con $P_A = \{\epsilon, \ *_O, \ *_O \checkmark_P, \ *_O \times_P\}$
- B con $P_B = \{\epsilon, \ *_O, \ *_O 0_P, \ *_O 1_P, \ *_O 2_P, \ *_O 3_P, \ \ldots\}$

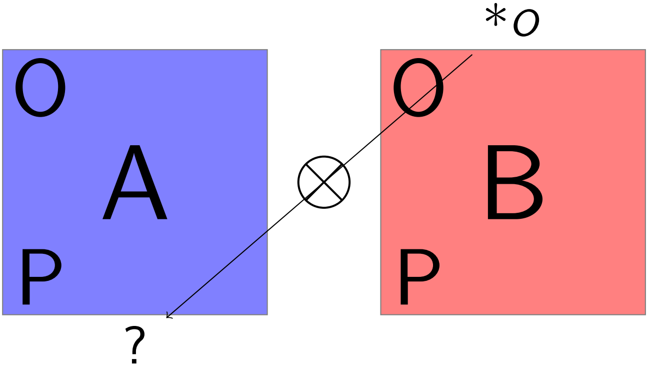
Descriviamo $A \otimes B$ tramite il tavolo di gioco



Il gioco $A \otimes B$

- A con $P_A = \{\epsilon,\ *_O,\ *_O \checkmark_P,\ *_O \times_P\}$
- B con $P_B = \{\epsilon,\ *_O,\ *_O 0_P,\ *_O 1_P,\ *_O 2_P,\ *_O 3_P,\ \ldots\}$

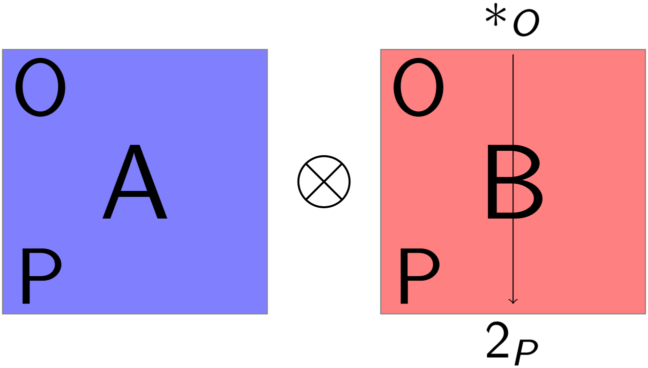
Descriviamo $A \otimes B$ tramite il tavolo di gioco



Il gioco $A \otimes B$

- A con $P_A = \{\epsilon, *_O, *_O \checkmark_P, *_O \times_P\}$
- B con $P_B = \{\epsilon, *_O, *_O 0_P, *_O 1_P, *_O 2_P, *_O 3_P, \ldots\}$

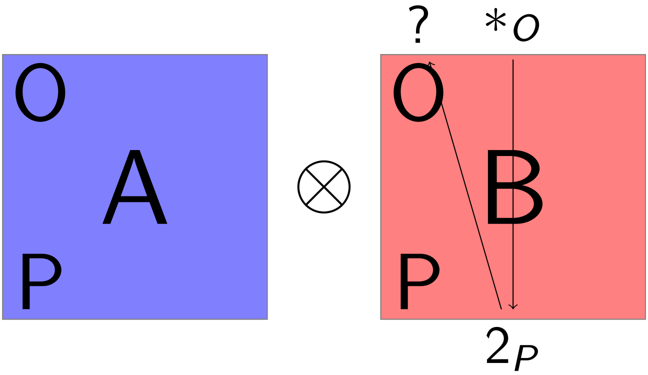
Descriviamo $A \otimes B$ tramite il tavolo di gioco



Il gioco $A \otimes B$

- A con $P_A = \{\epsilon, \ *_O, \ \ *_O \checkmark_P, \ *_O \times_P\}$
- B con $P_B = \{\epsilon, \ *_O, \ *_O 0_P, \ *_O 1_P, \ *_O 2_P, \ *_O 3_P, \ \ldots\}$

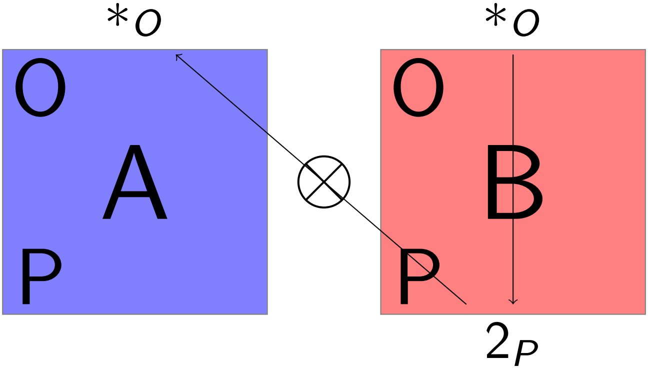
Descriviamo $A \otimes B$ tramite il tavolo di gioco



Il gioco $A \otimes B$

- A con $P_A = \{\epsilon,\ *_O,\ *_O \checkmark_P,\ *_O \times_P\}$
- B con $P_B = \{\epsilon,\ *_O,\ *_O 0_P,\ *_O 1_P,\ *_O 2_P,\ *_O 3_P,\ \ldots\}$

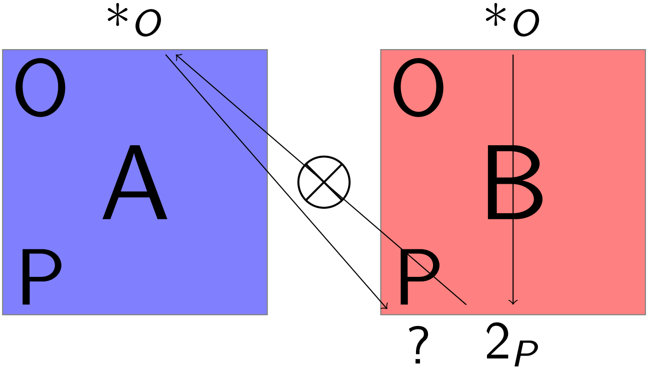
Descriviamo $A \otimes B$ tramite il tavolo di gioco



Il gioco $A \otimes B$

- A con $P_A = \{\epsilon,\ *_O,\ *_O \checkmark_P,\ *_O \times_P\}$
- B con $P_B = \{\epsilon,\ *_O,\ *_O 0_P,\ *_O 1_P,\ *_O 2_P,\ *_O 3_P,\ \dots\}$

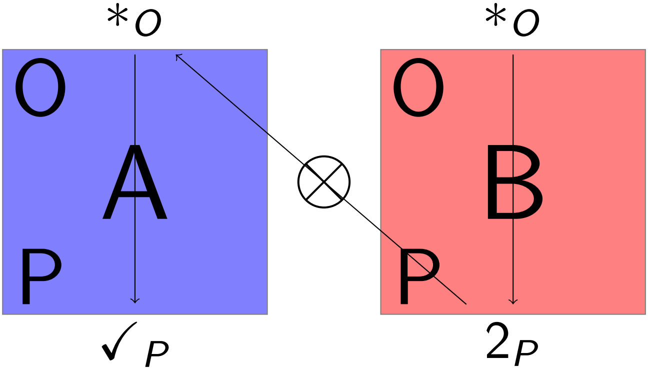
Descriviamo $A \otimes B$ tramite il tavolo di gioco



Il gioco $A \otimes B$

- A con $P_A = \{\epsilon,\ *_O,\ *_O\checkmark_P,\ *_O\times_P\}$
- B con $P_B = \{\epsilon,\ *_O,\ *_O0_P,\ *_O1_P,\ *_O2_P,\ *_O3_P,\ \dots\}$

Descriviamo $A \otimes B$ tramite il tavolo di gioco



Il gioco $A \multimap B$

Dati due giochi A e B definiamo il gioco $A \multimap B$ come:

Il gioco $A \multimap B$

Dati due giochi A e B definiamo il gioco $A \multimap B$ come:

- $M_{A \multimap B} = M_A \coprod M_B$

Il gioco $A \multimap B$

Dati due giochi A e B definiamo il gioco $A \multimap B$ come:

- $M_{A \multimap B} = M_A \coprod M_B$
- $\lambda^{QA}_{A \multimap B} = \lambda^{QA}_A \coprod \lambda^{QA}_B$

  $\lambda^{OP}_{A \multimap B} = \overline{\lambda^{OP}_A} \coprod \lambda^{OP}_B$

Il gioco $A \multimap B$

Dati due giochi A e B definiamo il gioco $A \multimap B$ come:

- $M_{A \multimap B} = M_A \coprod M_B$
- $\lambda^{QA}_{A \multimap B} = \lambda^{QA}_A \coprod \lambda^{QA}_B$

  $\lambda^{OP}_{A \multimap B} = \overline{\lambda^{OP}_A} \coprod \lambda^{OP}_B$
- $P_{A \otimes B}$ sono tutte le partite s tali che:
  - $s|_{M_A} \in P_A \wedge s|_{M_B} \in P_B$
  - Per ogni domanda in A, la risposta deve essere in A; lo stesso con B

Il gioco $A \multimap B$

Dati due giochi A e B definiamo il gioco $A \multimap B$ come:

- $M_{A \multimap B} = M_A \coprod M_B$
- $\lambda^{QA}_{A \multimap B} = \lambda^{QA}_A \coprod \lambda^{QA}_B$

  $\lambda^{OP}_{A \multimap B} = \overline{\lambda^{OP}_A} \coprod \lambda^{OP}_B$
- $P_{A \otimes B}$ sono tutte le partite s tali che:
  - $s|_{M_A} \in P_A \wedge s|_{M_B} \in P_B$
  - Per ogni domanda in A, la risposta deve essere in A; lo stesso con B
- $s \approx_{A \otimes B} t \Leftrightarrow s|_A \approx_A t|_A \wedge s|_B \approx_B t|_B \wedge fst(s) = fst(t)$

Il gioco $A \multimap B$

Dati due giochi A e B definiamo il gioco $A \multimap B$ come:

- $M_{A \multimap B} = M_A \coprod M_B$
- $\lambda^{QA}_{A \multimap B} = \lambda^{QA}_A \coprod \lambda^{QA}_B$

  $\lambda^{OP}_{A \multimap B} = \overline{\lambda^{OP}_A} \coprod \lambda^{OP}_B$
- $P_{A \otimes B}$ sono tutte le partite s tali che:
  - $s|_{M_A} \in P_A \wedge s|_{M_B} \in P_B$
  - Per ogni domanda in A, la risposta deve essere in A; lo stesso con B
- $s \approx_{A \otimes B} t \Leftrightarrow s|_A \approx_A t|_A \wedge s|_B \approx_B t|_B \wedge fst(s) = fst(t)$

Proprietà

Solamente il giocatore P può cambiare componente di gioco

Il gioco $A \multimap B$

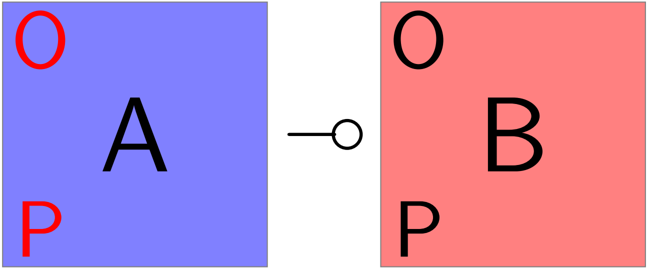
Il gioco $A \multimap B$

- A con $P_A = \{\epsilon,\ *_O,\ \ *_O \checkmark_P,\ *_O \times_P\}$
- B con $P_B = \{\epsilon,\ *_O,\ *_O 0_P,\ *_O 1_P,\ \ *_O\, 2_P,\ *_O 3_P,\ \dots\}$

Il gioco $A \multimap B$

- A con $P_A = \{\epsilon,\ *_O,\ *_O \checkmark_P,\ *_O \times_P\}$
- B con $P_B = \{\epsilon,\ *_O,\ *_O 0_P,\ *_O 1_P,\ *_O 2_P,\ *_O 3_P,\ \ldots\}$

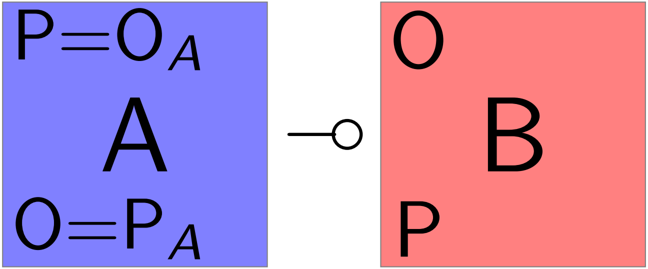
Descriviamo $A \multimap B$ tramite il tavolo di gioco



Il gioco $A \multimap B$

- A con $P_A = \{\epsilon, \ *_O, \ \ *_O \checkmark_P, \ *_O \times_P\}$
- B con $P_B = \{\epsilon, \ *_O, \ *_O 0_P, \ *_O 1_P, \ \ *_O 2_P, \ *_O 3_P, \ \ldots\}$

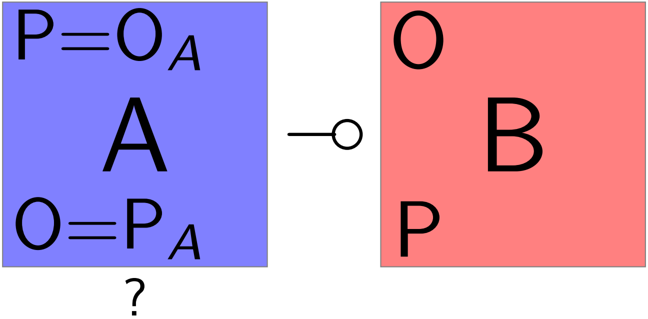
Descriviamo $A \multimap B$ tramite il tavolo di gioco



Il gioco $A \multimap B$

- A con $P_A = \{\epsilon,\ *_O,\ \ *_O \checkmark_P,\ *_O \times_P\}$
- B con $P_B = \{\epsilon,\ *_O,\ *_O 0_P,\ *_O 1_P,\ \ *_O 2_P,\ *_O 3_P,\ \ldots\}$

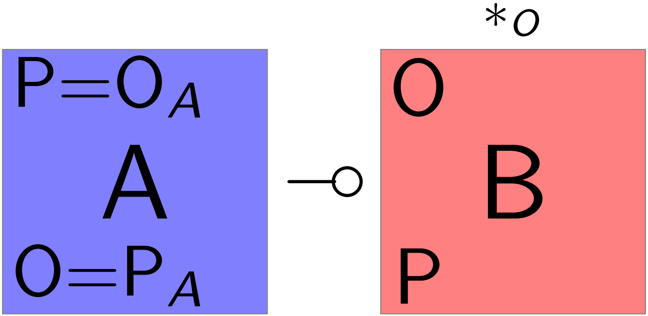
Descriviamo $A \multimap B$ tramite il tavolo di gioco



Il gioco $A \multimap B$

- A con $P_A = \{\epsilon,\ *_O,\ *_O \checkmark_P,\ *_O \times_P\}$
- B con $P_B = \{\epsilon,\ *_O,\ *_O 0_P,\ *_O 1_P,\ *_O 2_P,\ *_O 3_P,\ \dots\}$

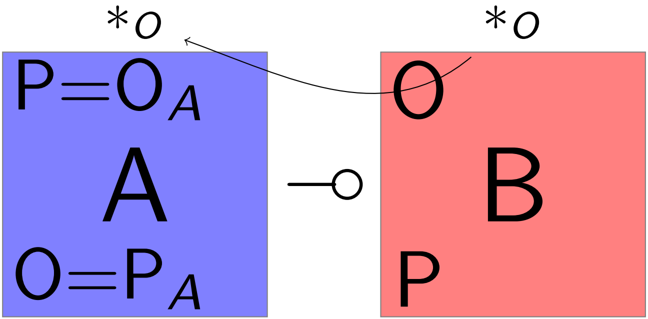
Descriviamo $A \multimap B$ tramite il tavolo di gioco



Il gioco $A \multimap B$

- A con $P_A = \{\epsilon,\ *_O,\ \ *_O \checkmark_P,\ *_O \times_P\}$
- B con $P_B = \{\epsilon,\ *_O,\ *_O 0_P,\ *_O 1_P,\ \ *_O\ 2_P,\ *_O 3_P,\ \ldots\}$

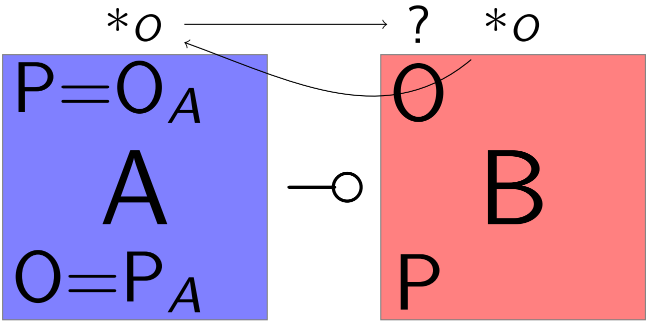
Descriviamo $A \multimap B$ tramite il tavolo di gioco



Il gioco $A \multimap B$

- A con $P_A = \{\epsilon,\ *_O,\ \ *_O \checkmark_P,\ *_O \times_P\}$
- B con $P_B = \{\epsilon,\ *_O,\ *_O 0_P,\ *_O 1_P,\ \ *_O\, 2_P,\ *_O 3_P,\ \ldots\}$

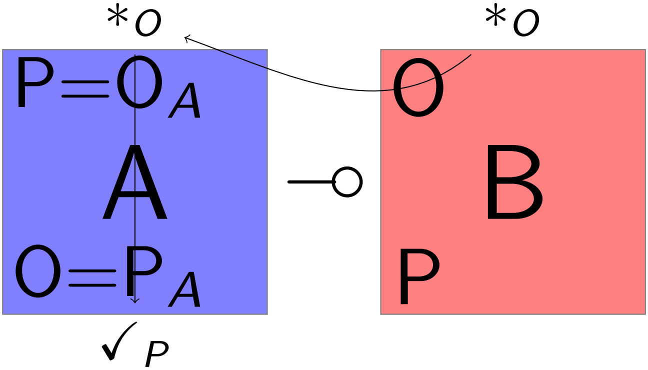
Descriviamo $A \multimap B$ tramite il tavolo di gioco



Il gioco $A \multimap B$

- A con $P_A = \{\epsilon, *_O, *_O\checkmark_P, *_O\times_P\}$
- B con $P_B = \{\epsilon, *_O, *_O 0_P, *_O 1_P, *_O 2_P, *_O 3_P, \ldots\}$

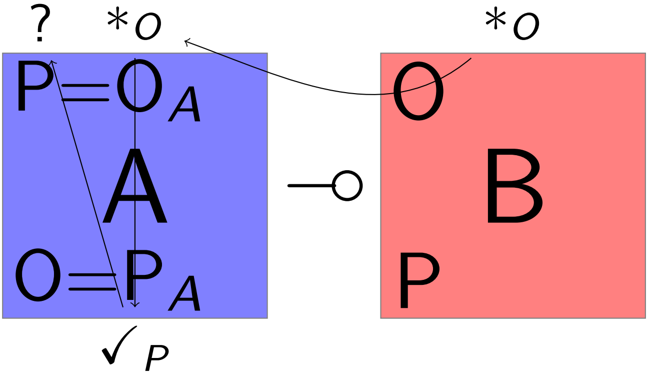
Descriviamo $A \multimap B$ tramite il tavolo di gioco



Il gioco $A \multimap B$

- A con $P_A = \{\epsilon,\ *_O,\ *_O\checkmark_P,\ *_O\times_P\}$
- B con $P_B = \{\epsilon,\ *_O,\ *_O0_P,\ *_O1_P,\ *_O2_P,\ *_O3_P,\ \dots\}$

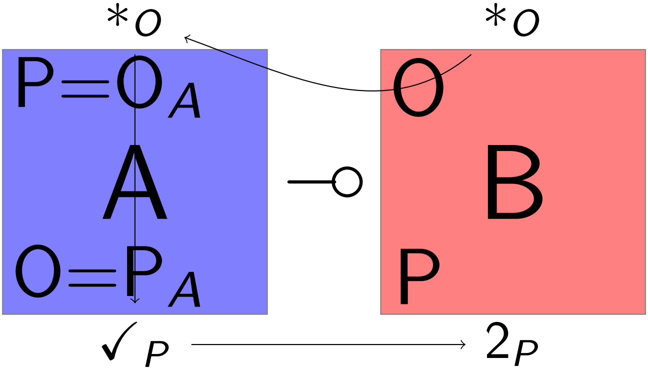
Descriviamo $A \multimap B$ tramite il tavolo di gioco



Il gioco $A \multimap B$

- A con $P_A = \{\epsilon,\ *_O,\ *_O\checkmark_P,\ *_O\times_P\}$
- B con $P_B = \{\epsilon,\ *_O,\ *_O0_P,\ *_O1_P,\ *_O2_P,\ *_O3_P,\ \ldots\}$

Descriviamo $A \multimap B$ tramite il tavolo di gioco



Il gioco $!A$

- $M_{!A} = \omega \times M_A$
- $\lambda_{!A}(i, a) = \lambda_A(a)$
- s è una partita di $P_{!A}$ se e solo se:
  - $\forall i \in \omega, s|_i \in P_A$
  - Se una domanda è nella componente i, la sua risposta deve essere nella componente i (*indexed bracketing condition*)
- $s \approx_{!A} t$ sse esiste $\pi : \omega \to \omega$ permutazione tale che $s|_i \approx_A t|_{\pi(i)} \wedge (\pi \circ fst)(s) = fst(t)$

Proprietà

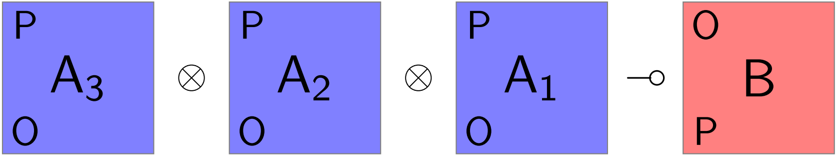
- Solamente il giocatore O può cambiare componente di gioco

Nota: concettualmente il gioco $!A$ si comporta come se avessimo infinite copie di A tensorizzate $A \otimes A \otimes A \otimes A \otimes \ldots$ con la relazione $\approx_{!A}$

Il gioco $A \Rightarrow B$

Il gioco $A \Rightarrow B$

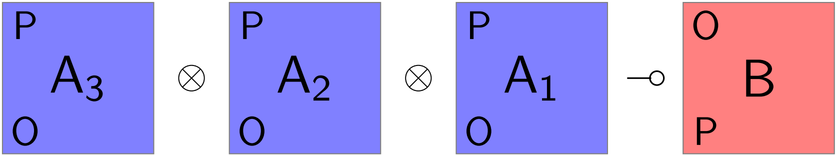
$$A \Rightarrow B \quad \equiv \quad !A \multimap B$$



Il gioco $A \Rightarrow B$

$$A \Rightarrow B \quad \equiv \quad !A \multimap B$$

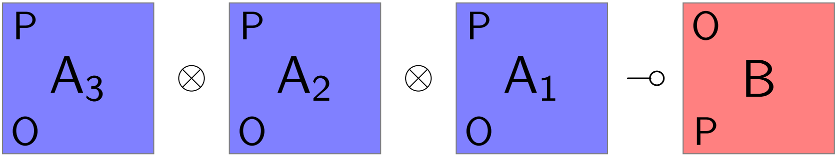
- $\mathit{Gun} = A$ con $P_A = \{\epsilon,\ \underline{\mathit{pull}},\ \underline{\mathit{pull}}\ \mathit{click},\ \underline{\mathit{pull}}\ \mathit{bang}\}$
- $\mathit{Life} = B$ con $P_B = \{\epsilon,\ *_O,\ *_O\checkmark_P,\ *_O 1_P,\ *_O 2_P,\ *_O 3_P,\ \ldots\}$



Il gioco $A \Rightarrow B$

$$A \Rightarrow B \quad \equiv \quad !A \multimap B$$

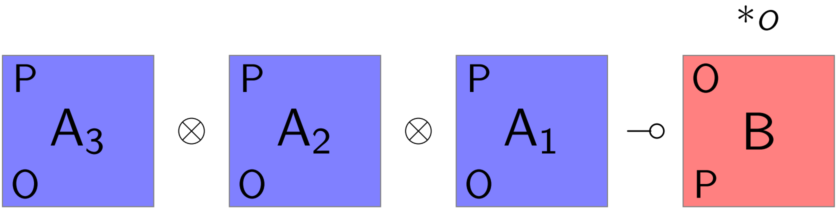
- $Gun = A$ con $P_A = \{\epsilon,\ \underline{pull},\ \underline{pull}\ click,\ \underline{pull}\ bang\}$
- $Life = B$ con $P_B = \{\epsilon,\ *_O,\ *_O\checkmark_P,\ *_O1_P,\ *_O\,2_P,\ *_O3_P,\ \ldots\}$
- Strategia *Roulette Russa*:
  $f(*_O) = \underline{pull}_1,\quad f(click_n) = \underline{pull}_{n+1}\forall n,\quad f(bang_n) = n_P$



Il gioco $A \Rightarrow B$

$$A \Rightarrow B \quad \equiv \quad !A \multimap B$$

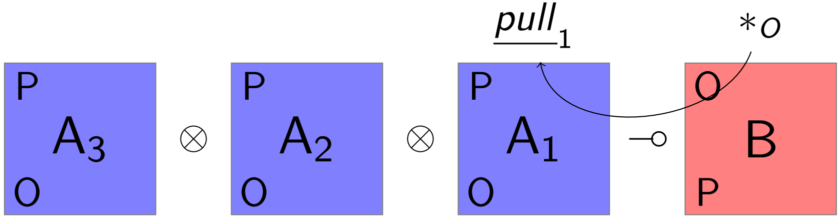
- $Gun = A$ con $P_A = \{\epsilon, \ \underline{pull}, \ \underline{pull}\ click, \ \underline{pull}\ bang\}$
- $Life = B$ con $P_B = \{\epsilon, \ *_O, \ *_O\checkmark_P, \ *_O 1_P, \ *_O 2_P, \ *_O 3_P, \ \dots\}$
- Strategia *Roulette Russa*:
  $f(*_O) = \underline{pull}_1, \quad f(click_n) = \underline{pull}_{n+1} \forall n, \quad f(bang_n) = n_P$



Il gioco $A \Rightarrow B$

$$A \Rightarrow B \quad \equiv \quad !A \multimap B$$

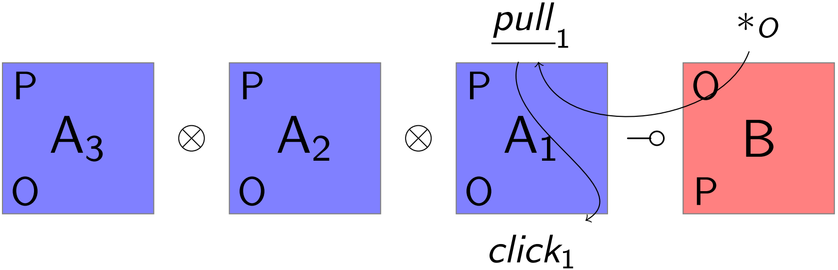
- $Gun = A$ con $P_A = \{\epsilon,\ \underline{pull},\ \underline{pull}\ click,\ \underline{pull}\ bang\}$
- $Life = B$ con $P_B = \{\epsilon,\ *_O,\ *_O\checkmark_P,\ *_O1_P,\ *_O2_P,\ *_O3_P,\ \ldots\}$
- Strategia *Roulette Russa*:
  $f(*_O) = \underline{pull}_1, \quad f(click_n) = \underline{pull}_{n+1} \forall n, \quad f(bang_n) = n_P$



Il gioco $A \Rightarrow B$

$$A \Rightarrow B \quad \equiv \quad !A \multimap B$$

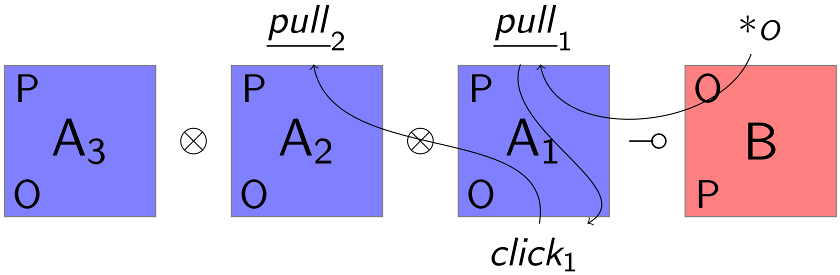
- $Gun = A$ con $P_A = \{\epsilon,\ \underline{pull},\ \underline{pull}\ click,\ \underline{pull}\ bang\}$
- $Life = B$ con $P_B = \{\epsilon,\ *_O,\ *_O\checkmark_P,\ *_O 1_P,\ *_O\, 2_P,\ *_O 3_P,\ \ldots\}$
- Strategia *Roulette Russa*:
  $f(*_O) = \underline{pull}_1, \quad f(click_n) = \underline{pull}_{n+1}\forall n, \quad f(bang_n) = n_P$



Il gioco $A \Rightarrow B$

$$A \Rightarrow B \quad \equiv \quad !A \multimap B$$

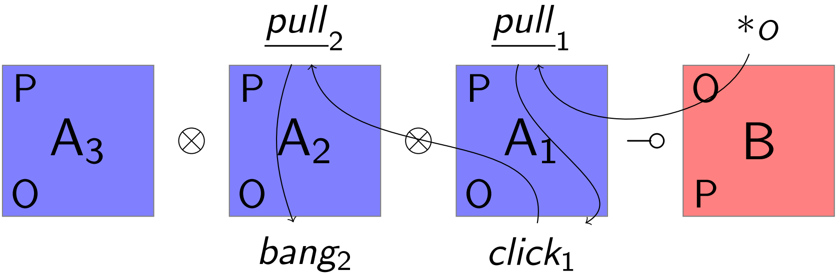
- $Gun = A$ con $P_A = \{\epsilon,\ \underline{pull},\ \underline{pull}\ click,\ \underline{pull}\ bang\}$
- $Life = B$ con $P_B = \{\epsilon,\ *_O,\ *_O\checkmark_P,\ *_O1_P,\ *_O2_P,\ *_O3_P,\ \dots\}$
- Strategia *Roulette Russa*:
  $f(*_O) = \underline{pull}_1$, $\quad f(click_n) = \underline{pull}_{n+1}\ \forall n,\quad f(bang_n) = n_P$



Il gioco $A \Rightarrow B$

$$A \Rightarrow B \quad \equiv \quad !A \multimap B$$

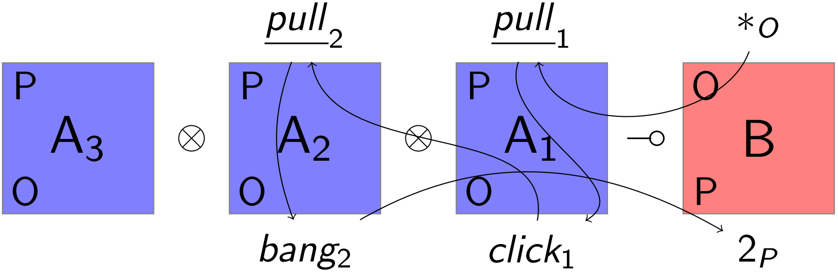
- *Gun* = *A* con $P_A = \{\epsilon,\ \underline{pull},\ \underline{pull}\ click,\ \underline{pull}\ bang\}$
- *Life* = *B* con $P_B = \{\epsilon,\ *_O,\ *_O\checkmark_P,\ *_O1_P,\ *_O2_P,\ *_O3_P,\ \dots\}$
- Strategia *Roulette Russa*:
  $f(*_O) = \underline{pull}_1, \quad f(click_n) = \underline{pull}_{n+1} \forall n, \quad f(bang_n) = n_P$



Il gioco $A \Rightarrow B$

$$A \Rightarrow B \quad \equiv \quad !A \multimap B$$

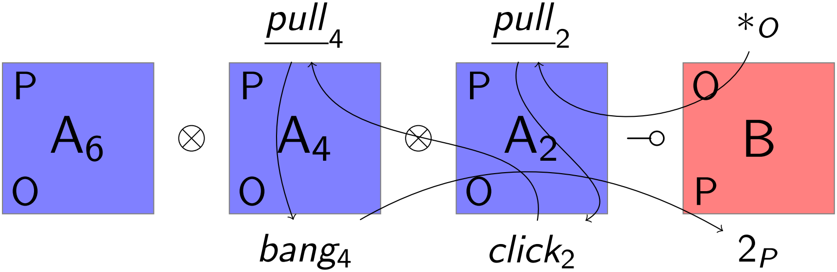
- $Gun = A$ con $P_A = \{\epsilon,\ \underline{pull},\ \underline{pull}\ click,\ \underline{pull}\ bang\}$
- $Life = B$ con $P_B = \{\epsilon,\ *_O,\ *_O \checkmark_P,\ *_O 1_P,\ *_O 2_P,\ *_O 3_P,\ \dots\}$
- Strategia *Roulette Russa*:
  $f(*_O) = \underline{pull}_1,\quad f(click_n) = \underline{pull}_{n+1} \forall n,\quad f(bang_n) = n_P$



Il gioco $A \Rightarrow B$

$$A \Rightarrow B \quad \equiv \quad !A \multimap B$$

- $Gun = A$ con $P_A = \{\epsilon,\ \underline{pull},\ \underline{pull}\ click,\ \underline{pull}\ bang\}$
- $Life = B$ con $P_B = \{\epsilon,\ *_O,\ *_O\checkmark_P,\ *_O1_P,\ *_O2_P,\ *_O3_P,\ \dots\}$
- Strategia *Roulette Russa*:
  $f(*_O) = \underline{pull}_2, \quad f(click_{2n}) = \underline{pull}_{2n+2} \forall n, \quad f(bang_{2n}) = n_P$



La strategia $\sigma;\tau$

Date σ strategia di $A \multimap B$ e τ di $B \multimap C$, allora

La strategia $\sigma;\tau$

Date σ strategia di $A \multimap B$ e τ di $B \multimap C$, allora

- $\sigma;\tau = \{s|_{A,C} : s \in (M_A \coprod M_B \coprod M_C)^*,\ s|_{A,B} \in \sigma,\ s|_{B,C} \in \tau\}$

La strategia $\sigma;\tau$

Date σ strategia di $A \multimap B$ e τ di $B \multimap C$, allora

- $\sigma;\tau = \{s|_{A,C} : s \in (M_A \coprod M_B \coprod M_C)^*,\ s|_{A,B} \in \sigma,\ s|_{B,C} \in \tau\}$
- $\sigma;\tau$ è una strategia di $A \multimap C$

La strategia $\sigma;\tau$

Date σ strategia di $A \multimap B$ e τ di $B \multimap C$, allora

- $\sigma;\tau = \{s|_{A,C} : s \in (M_A \coprod M_B \coprod M_C)^*,\ s|_{A,B} \in \sigma,\ s|_{B,C} \in \tau\}$
- $\sigma;\tau$ è una strategia di $A \multimap C$

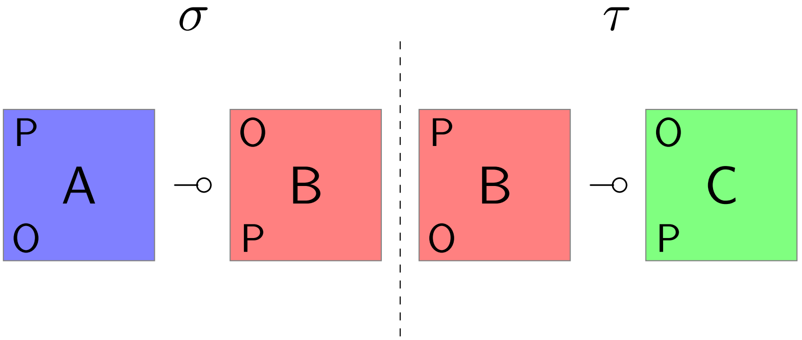
Si può dare una costruzione di $\sigma;\tau$ algoritmica:

La strategia $\sigma;\tau$

Date σ strategia di $A \multimap B$ e τ di $B \multimap C$, allora

- $\sigma;\tau = \{s|_{A,C} : s \in (M_A \coprod M_B \coprod M_C)^*,\ s|_{A,B} \in \sigma,\ s|_{B,C} \in \tau\}$
- $\sigma;\tau$ è una strategia di $A \multimap C$

Si può dare una costruzione di $\sigma;\tau$ algoritmica:

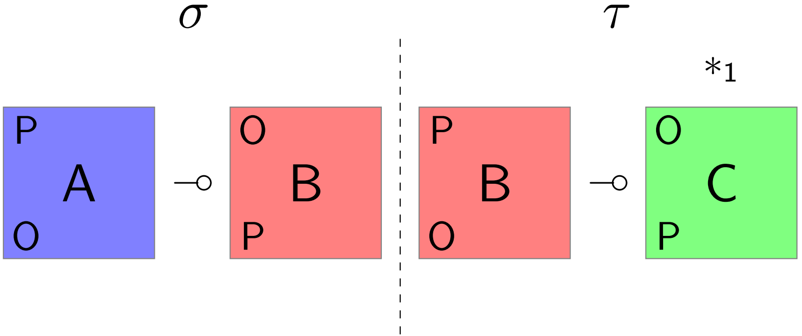


La strategia $\sigma;\tau$

Date σ strategia di $A \multimap B$ e τ di $B \multimap C$, allora

- $\sigma;\tau = \{s|_{A,C} : s \in (M_A \coprod M_B \coprod M_C)^*,\ s|_{A,B} \in \sigma,\ s|_{B,C} \in \tau\}$
- $\sigma;\tau$ è una strategia di $A \multimap C$

Si può dare una costruzione di $\sigma;\tau$ algoritmica:

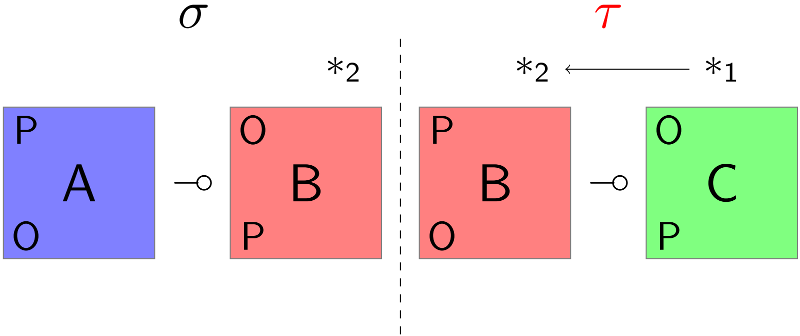


La strategia $\sigma;\tau$

Date σ strategia di $A \multimap B$ e τ di $B \multimap C$, allora

- $\sigma;\tau = \{s|_{A,C} : s \in (M_A \coprod M_B \coprod M_C)^*,\ s|_{A,B} \in \sigma,\ s|_{B,C} \in \tau\}$
- $\sigma;\tau$ è una strategia di $A \multimap C$

Si può dare una costruzione di $\sigma;\tau$ algoritmica:

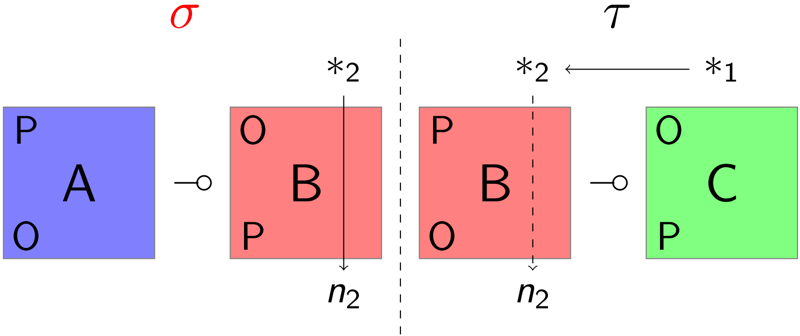


La strategia $\sigma;\tau$

Date σ strategia di $A \multimap B$ e τ di $B \multimap C$, allora

- $\sigma;\tau = \{s|_{A,C} : s \in (M_A \coprod M_B \coprod M_C)^*,\ s|_{A,B} \in \sigma,\ s|_{B,C} \in \tau\}$
- $\sigma;\tau$ è una strategia di $A \multimap C$

Si può dare una costruzione di $\sigma;\tau$ algoritmica:

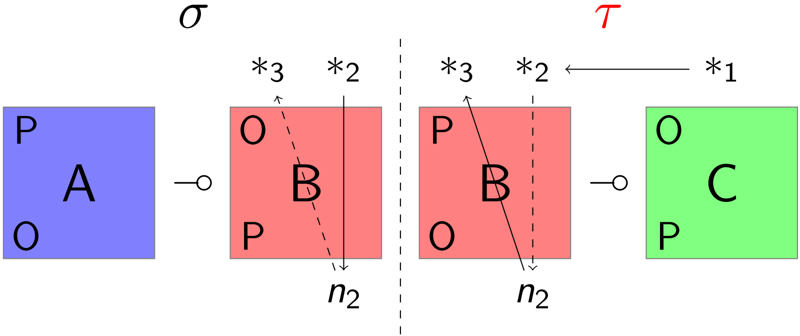


La strategia $\sigma;\tau$

Date σ strategia di $A \multimap B$ e τ di $B \multimap C$, allora

- $\sigma;\tau = \{s|_{A,C} : s \in (M_A \coprod M_B \coprod M_C)^*,\ s|_{A,B} \in \sigma,\ s|_{B,C} \in \tau\}$
- $\sigma;\tau$ è una strategia di $A \multimap C$

Si può dare una costruzione di $\sigma;\tau$ algoritmica:

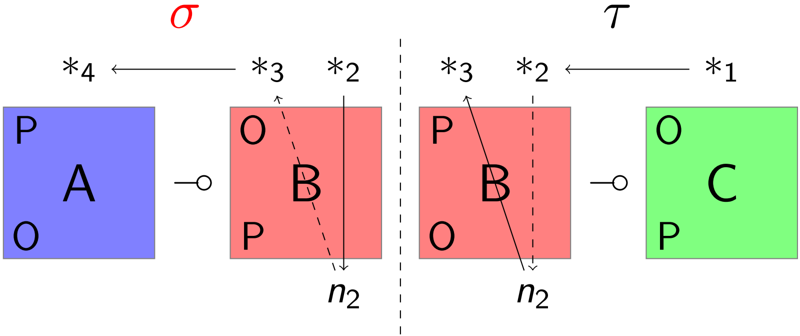


La strategia $\sigma;\tau$

Date σ strategia di $A \multimap B$ e τ di $B \multimap C$, allora

- $\sigma;\tau = \{s|_{A,C} : s \in (M_A \coprod M_B \coprod M_C)^*,\ s|_{A,B} \in \sigma,\ s|_{B,C} \in \tau\}$
- $\sigma;\tau$ è una strategia di $A \multimap C$

Si può dare una costruzione di $\sigma;\tau$ algoritmica:

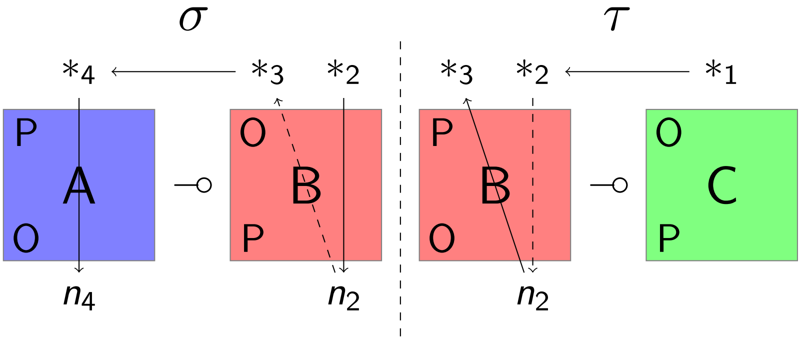


La strategia $\sigma;\tau$

Date σ strategia di $A \multimap B$ e τ di $B \multimap C$, allora

- $\sigma;\tau = \{s|_{A,C} : s \in (M_A \coprod M_B \coprod M_C)^*,\ s|_{A,B} \in \sigma,\ s|_{B,C} \in \tau\}$
- $\sigma;\tau$ è una strategia di $A \multimap C$

Si può dare una costruzione di $\sigma;\tau$ algoritmica:

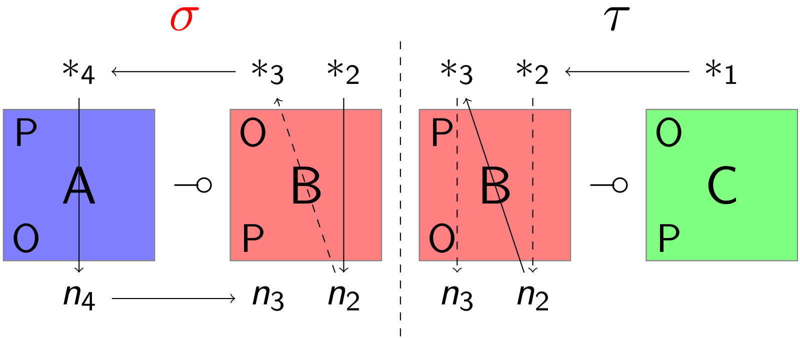


La strategia $\sigma;\tau$

Date σ strategia di $A \multimap B$ e τ di $B \multimap C$, allora

- $\sigma;\tau = \{s|_{A,C} : s \in (M_A \coprod M_B \coprod M_C)^*,\ s|_{A,B} \in \sigma,\ s|_{B,C} \in \tau\}$
- $\sigma;\tau$ è una strategia di $A \multimap C$

Si può dare una costruzione di $\sigma;\tau$ algoritmica:

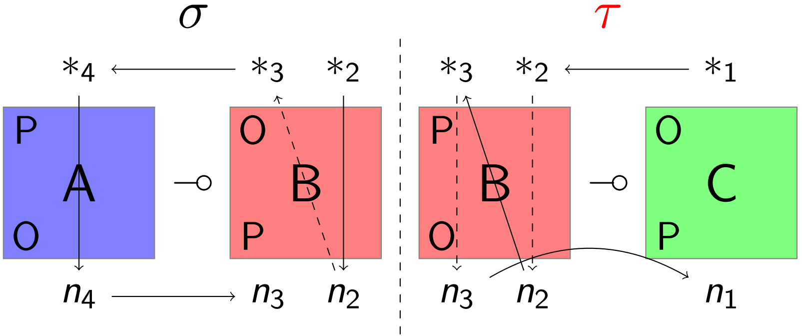


La strategia $\sigma;\tau$

Date σ strategia di $A \multimap B$ e τ di $B \multimap C$, allora

- $\sigma;\tau = \{s|_{A,C} : s \in (M_A \coprod M_B \coprod M_C)^*,\ s|_{A,B} \in \sigma,\ s|_{B,C} \in \tau\}$
- $\sigma;\tau$ è una strategia di $A \multimap C$

Si può dare una costruzione di $\sigma;\tau$ algoritmica:

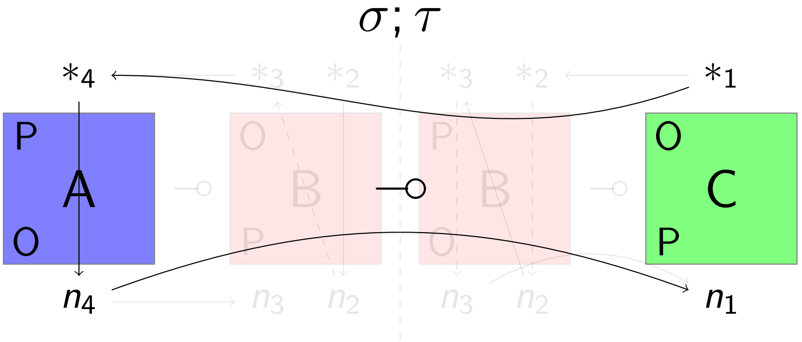


La strategia $\sigma;\tau$

Date σ strategia di $A \multimap B$ e τ di $B \multimap C$, allora

- $\sigma;\tau = \{s|_{A,C} : s \in (M_A \coprod M_B \coprod M_C)^*,\ s|_{A,B} \in \sigma,\ s|_{B,C} \in \tau\}$
- $\sigma;\tau$ è una strategia di $A \multimap C$

Si può dare una costruzione di $\sigma;\tau$ algoritmica:



La strategia *copycat* su $A \multimap A$

Dato A, è sempre possibile costruire la strategia *copycat* su $A \multimap A$:

La strategia *copycat* su $A \multimap A$

Dato A, è sempre possibile costruire la strategia *copycat* su $A \multimap A$:

- $id_A = \{s \in P_{A \multimap A} : s|_1 = s|_2\}$

La strategia *copycat* su $A \multimap A$

Dato A, è sempre possibile costruire la strategia *copycat* su $A \multimap A$:

- $id_A = \{s \in P_{A \multimap A} : s|_1 = s|_2\}$
- la strategia consiste nel copiare le mosse di O sull'altra componente, quindi la funzione parziale associata a questa strategia è

$$f(x_1) = x_2, \quad f(x_2) = x_1 \quad \forall x \in M_A$$

La strategia *copycat* su $A \multimap A$

Dato A, è sempre possibile costruire la strategia *copycat* su $A \multimap A$:

- $id_A = \{s \in P_{A \multimap A} : s|_1 = s|_2\}$
- la strategia consiste nel copiare le mosse di O sull'altra componente, quindi la funzione parziale associata a questa strategia è

$$f(x_1) = x_2, \quad f(x_2) = x_1 \quad \forall x \in M_A$$

- Data σ strategia di $A \multimap B$, avremo che $id_A; \sigma = \sigma$

La categoria dei giochi $\mathcal{G}$

Definiamo $\mathcal{G}$ la categoria tale che:

La categoria dei giochi $\mathcal{G}$

Definiamo $\mathcal{G}$ la categoria tale che:

- $\mathcal{G}_0$ sono i giochi

La categoria dei giochi $\mathcal{G}$

Definiamo $\mathcal{G}$ la categoria tale che:

- $\mathcal{G}_0$ sono i giochi
- dati due giochi A e B, i morfismi $A \to B$ sono $\{\sigma \text{ strategia di } A \multimap B \mid \sigma \approx_s \sigma\} / \approx_s$

La categoria dei giochi $\mathcal{G}$

Definiamo $\mathcal{G}$ la categoria tale che:

- $\mathcal{G}_0$ sono i giochi
- dati due giochi A e B, i morfismi $A \to B$ sono $\{\sigma \text{ strategia di } A \multimap B \mid \sigma \approx_s \sigma\}/\approx_s$
- Date $[\sigma] : A \to B$ e $[\tau] : B \to C$, $[\tau] \circ [\sigma] = [\sigma; \tau]$

La categoria dei giochi $\mathcal{G}$

Definiamo $\mathcal{G}$ la categoria tale che:

- $\mathcal{G}_0$ sono i giochi
- dati due giochi A e B, i morfismi $A \to B$ sono $\{\sigma \text{ strategia di } A \multimap B \mid \sigma \approx_s \sigma\}/\approx_s$
- Date $[\sigma] : A \to B$ e $[\tau] : B \to C$, $[\tau] \circ [\sigma] = [\sigma; \tau]$
- Il morfismo identico è dato dalla strategia id_A

La categoria dei giochi $\mathcal{G}$

Definiamo $\mathcal{G}$ la categoria tale che:

- $\mathcal{G}_0$ sono i giochi
- dati due giochi A e B, i morfismi $A \to B$ sono $\{\sigma \text{ strategia di } A \multimap B \mid \sigma \approx_s \sigma\}/\approx_s$
- Date $[\sigma] : A \to B$ e $[\tau] : B \to C$, $[\tau] \circ [\sigma] = [\sigma; \tau]$
- Il morfismo identico è dato dalla strategia id_A

In particolare abbiamo che $\mathcal{G}$:

La categoria dei giochi $\mathcal{G}$

Definiamo $\mathcal{G}$ la categoria tale che:

- $\mathcal{G}_0$ sono i giochi
- dati due giochi A e B, i morfismi $A \rightarrow B$ sono $\{\sigma \text{ strategia di } A \multimap B \mid \sigma \approx_s \sigma\}/\approx_s$
- Date $[\sigma] : A \rightarrow B$ e $[\tau] : B \rightarrow C$, $[\tau] \circ [\sigma] = [\sigma; \tau]$
- Il morfismo identico è dato dalla strategia id_A

In particolare abbiamo che $\mathcal{G}$:

- è dotata di un oggetto finale (1)

La categoria dei giochi $\mathcal{G}$

Definiamo $\mathcal{G}$ la categoria tale che:

- $\mathcal{G}_0$ sono i giochi
- dati due giochi A e B, i morfismi $A \to B$ sono $\{\sigma \text{ strategia di } A \multimap B \mid \sigma \approx_s \sigma\}/\approx_s$
- Date $[\sigma] : A \to B$ e $[\tau] : B \to C$, $[\tau] \circ [\sigma] = [\sigma; \tau]$
- Il morfismo identico è dato dalla strategia id_A

In particolare abbiamo che $\mathcal{G}$:

- è dotata di un oggetto finale (1)
- è una categoria monoidale

  ($\otimes$ è un bifuntore associativo con elemento neutro)

La categoria dei giochi $\mathcal{G}$

Definiamo $\mathcal{G}$ la categoria tale che:

- $\mathcal{G}_0$ sono i giochi
- dati due giochi A e B, i morfismi $A \to B$ sono $\{\sigma \text{ strategia di } A \multimap B \mid \sigma \approx_s \sigma\}/\approx_s$
- Date $[\sigma] : A \to B$ e $[\tau] : B \to C$, $[\tau] \circ [\sigma] = [\sigma; \tau]$
- Il morfismo identico è dato dalla strategia id_A

In particolare abbiamo che $\mathcal{G}$:

- è dotata di un oggetto finale (1)
- è una categoria monoidale

  ($\otimes$ è un bifuntore associativo con elemento neutro)
- NON è una categoria cartesiana chiusa (mancano i prodotti)

Il gioco $A \& B$

Dati due giochi A e B definiamo il gioco $A \& B$ come:

Il gioco $A\&B$

Dati due giochi A e B definiamo il gioco $A\&B$ come:

- $M_{A\&B} = M_A \coprod M_B$
- $\lambda_{A\&B} = \lambda_A \coprod \lambda_B$
- $P_{A\&B} = P_A \coprod P_B$
- $\approx_{A\&B} = \approx_A \coprod \approx_B$

Il gioco $A\&B$

Dati due giochi A e B definiamo il gioco $A\&B$ come:

- $M_{A\&B} = M_A \coprod M_B$
- $\lambda_{A\&B} = \lambda_A \coprod \lambda_B$
- $P_{A\&B} = P_A \coprod P_B$
- $\approx_{A\&B} = \approx_A \coprod \approx_B$

Proprietà

Il gioco $A\&B$

Dati due giochi A e B definiamo il gioco $A\&B$ come:

- $M_{A\&B} = M_A \coprod M_B$
- $\lambda_{A\&B} = \lambda_A \coprod \lambda_B$
- $P_{A\&B} = P_A \coprod P_B$
- $\approx_{A\&B} = \approx_A \coprod \approx_B$

Proprietà

- Una partita di $A\&B$ è giocata su una sola delle due componenti

Il gioco $A\&B$

Dati due giochi A e B definiamo il gioco $A\&B$ come:

- $M_{A\&B} = M_A \coprod M_B$
- $\lambda_{A\&B} = \lambda_A \coprod \lambda_B$
- $P_{A\&B} = P_A \coprod P_B$
- $\approx_{A\&B} = \approx_A \coprod \approx_B$

Proprietà

- Una partita di $A\&B$ è giocata su una sola delle due componenti
- Ogni strategia di $A\&B$ è unione di una strategia di A e di una strategia di B (anche vuota)

strategie *der* e $\dagger$

Dato un gioco A, definiamo la strategia *der* su $!A \multimap A$:

strategie *der* e †

Dato un gioco A, definiamo la strategia *der* su $!A \multimap A$:

- Se A_i è l'i-esima componente di $!A$, allora der_A^i si comporta come la strategia copycat su $A_i \multimap A \cong A \multimap A$, e non agisce sulle altre componenti

strategie *der* e $\dagger$

Dato un gioco A, definiamo la strategia *der* su $!A \multimap A$:

- Se A_i è l'i-esima componente di $!A$, allora der_A^i si comporta come la strategia copycat su $A_i \multimap A \cong A \multimap A$, e non agisce sulle altre componenti
- Dato che le componenti di $!A$ sono equivalenti, avremo che $der_A^i \approx_s der_A^j \; \forall i, j$, quindi lo indicheremo semplicemente con der_A

strategie *der* e †

Dato un gioco A, definiamo la strategia *der* su $!A \multimap A$:

- Se A_i è l'i-esima componente di $!A$, allora der_A^i si comporta come la strategia copycat su $A_i \multimap A \cong A \multimap A$, e non agisce sulle altre componenti
- Dato che le componenti di $!A$ sono equivalenti, avremo che $der_A^i \approx_s der_A^j \; \forall i, j$, quindi lo indicheremo semplicemente con der_A

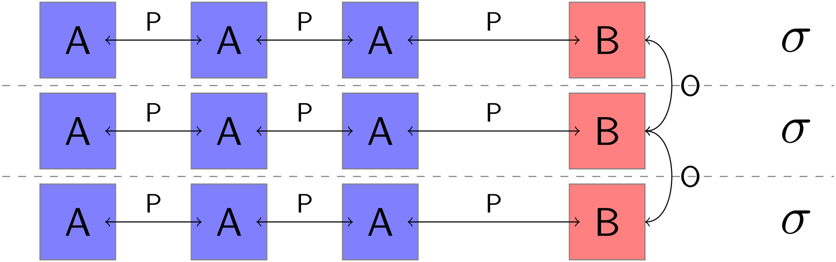
Data una strategia σ su $!A \multimap B$, definiamo la strategia $\sigma^\dagger$ su $!A \multimap !B$:

strategie *der* e †

Dato un gioco A, definiamo la strategia *der* su $!A \multimap A$:

- Se A_i è l'i-esima componente di $!A$, allora der_A^i si comporta come la strategia copycat su $A_i \multimap A \cong A \multimap A$, e non agisce sulle altre componenti
- Dato che le componenti di $!A$ sono equivalenti, avremo che $der_A^i \approx_s der_A^j \; \forall i, j$, quindi lo indicheremo semplicemente con der_A

Data una strategia σ su $!A \multimap B$, definiamo la strategia $\sigma^\dagger$ su $!A \multimap !B$:



categoria di co-Kleisli

categoria di co-Kleisli

Definiamo $K_!(\mathcal{G})$ la categoria di *co-Kleisli* di $\mathcal{G}$ rispetto a !
In particolare:

categoria di co-Kleisli

Definiamo $K_!(\mathcal{G})$ la categoria di *co-Kleisli* di $\mathcal{G}$ rispetto a !
In particolare:

- $K_!(\mathcal{G})_0 = \mathcal{G}_0$

categoria di co-Kleisli

Definiamo $K_!(\mathcal{G})$ la categoria di *co-Kleisli* di $\mathcal{G}$ rispetto a !
In particolare:

- $K_!(\mathcal{G})_0 = \mathcal{G}_0$
- Dati due giochi A, B, $Mor_{K_!(\mathcal{G})}(A, B) = Mor_{\mathcal{G}}(!A, B)$

categoria di co-Kleisli

Definiamo $K_!(\mathcal{G})$ la categoria di *co-Kleisli* di $\mathcal{G}$ rispetto a !
In particolare:

- $K_!(\mathcal{G})_0 = \mathcal{G}_0$
- Dati due giochi A, B, $Mor_{K_!(\mathcal{G})}(A, B) = Mor_{\mathcal{G}}(!A, B)$
- Date due strategie σ e τ, $\tau \circ \sigma = \sigma \,⨾\, \tau := \sigma^{\dagger} ; \tau$

categoria di co-Kleisli

Definiamo $K_!(\mathcal{G})$ la categoria di *co-Kleisli* di $\mathcal{G}$ rispetto a !
In particolare:

- $K_!(\mathcal{G})_0 = \mathcal{G}_0$
- Dati due giochi A, B, $Mor_{K_!(\mathcal{G})}(A, B) = Mor_{\mathcal{G}}(!A, B)$
- Date due strategie σ e τ, $\tau \circ \sigma = \sigma \fatsemi \tau := \sigma^{\dagger}; \tau$
- Dato un gioco A, il morfismo identico è der_A

categoria di co-Kleisli

Definiamo $K_!(\mathcal{G})$ la categoria di *co-Kleisli* di $\mathcal{G}$ rispetto a !
In particolare:

- $K_!(\mathcal{G})_0 = \mathcal{G}_0$
- Dati due giochi A, B, $Mor_{K_!(\mathcal{G})}(A, B) = Mor_{\mathcal{G}}(!A, B)$
- Date due strategie σ e τ, $\tau \circ \sigma = \sigma \fatsemi \tau := \sigma^\dagger ; \tau$
- Dato un gioco A, il morfismo identico è der_A

In particolare $K_!(\mathcal{G})$ è una *categoria cartesiana chiusa*, cioè:

categoria di co-Kleisli

Definiamo $K_!(\mathcal{G})$ la categoria di *co-Kleisli* di $\mathcal{G}$ rispetto a !
In particolare:

- $K_!(\mathcal{G})_0 = \mathcal{G}_0$
- Dati due giochi A, B, $Mor_{K_!(\mathcal{G})}(A, B) = Mor_{\mathcal{G}}(!A, B)$
- Date due strategie σ e τ, $\tau \circ \sigma = \sigma \fatsemi \tau := \sigma^\dagger ; \tau$
- Dato un gioco A, il morfismo identico è der_A

In particolare $K_!(\mathcal{G})$ è una *categoria cartesiana chiusa*, cioè:

- Dati due oggetti esiste il *prodotto* $(A \& B)$

categoria di co-Kleisli

Definiamo $K_!(\mathcal{G})$ la categoria di *co-Kleisli* di $\mathcal{G}$ rispetto a !
In particolare:

- $K_!(\mathcal{G})_0 = \mathcal{G}_0$
- Dati due giochi A, B, $Mor_{K_!(\mathcal{G})}(A, B) = Mor_{\mathcal{G}}(!A, B)$
- Date due strategie σ e τ, $\tau \circ \sigma = \sigma \mathbin{⨾} \tau := \sigma^{\dagger} ; \tau$
- Dato un gioco A, il morfismo identico è der_A

In particolare $K_!(\mathcal{G})$ è una *categoria cartesiana chiusa*, cioè:

- Dati due oggetti esiste il *prodotto* $(A \& B)$
- Esiste un oggetto *finale* (1)

categoria di co-Kleisli

Definiamo $K_!(\mathcal{G})$ la categoria di *co-Kleisli* di $\mathcal{G}$ rispetto a !
In particolare:

- $K_!(\mathcal{G})_0 = \mathcal{G}_0$
- Dati due giochi A, B, $Mor_{K_!(\mathcal{G})}(A, B) = Mor_{\mathcal{G}}(!A, B)$
- Date due strategie σ e τ, $\tau \circ \sigma = \sigma \fatsemi \tau := \sigma^\dagger ; \tau$
- Dato un gioco A, il morfismo identico è der_A

In particolare $K_!(\mathcal{G})$ è una *categoria cartesiana chiusa*, cioè:

- Dati due oggetti esiste il *prodotto* $(A \& B)$
- Esiste un oggetto *finale* (1)
- Dati due oggetti, esiste l'esponente
  ("$\Rightarrow$" è tale che $Mor_{K_!(\mathcal{G})}(A \& B, C) \cong Mor_{K_!(\mathcal{G})}(A, B \Rightarrow C)$)

order enrichement e razionalità

order enrichement e razionalità

Definiamo un *pointed poset (ppo)* come un poset con un minimo (generalmente indicato con $\perp$)

order enrichement e razionalità

Definiamo un *pointed poset (ppo)* come un poset con un minimo (generalmente indicato con $\perp$)   Definiamo una categoria cartesiana chiusa C *pointed poset enriched* se:

order enrichement e razionalità

Definiamo un *pointed poset (ppo)* come un poset con un minimo (generalmente indicato con $\perp$) Definiamo una categoria cartesiana chiusa C *pointed poset enriched* se:

- Dati due oggetti A, B, $(Mor(A, B), \sqsubseteq_{A,B}, \perp_{A,B})$ è un pointed poset

order enrichement e razionalità

Definiamo un *pointed poset (ppo)* come un poset con un minimo (generalmente indicato con $\perp$)   Definiamo una categoria cartesiana chiusa C *pointed poset enriched* se:

- Dati due oggetti A, B, $(Mor(A, B), \sqsubseteq_{A,B}, \perp_{A,B})$ è un pointed poset
- Composizione, prodotto e currying sono monotoni

order enrichement e razionalità

Definiamo un *pointed poset (ppo)* come un poset con un minimo (generalmente indicato con $\perp$) Definiamo una categoria cartesiana chiusa C *pointed poset enriched* se:

- Dati due oggetti A, B, $(Mor(A,B), \sqsubseteq_{A,B}, \perp_{A,B})$ è un pointed poset
- Composizione, prodotto e currying sono monotoni
- Per ogni $f : A \to B$, per ogni gioco C, $\perp_{B,C} \circ f = \perp_{A,B}$

order enrichement e razionalità

Definiamo un *pointed poset (ppo)* come un poset con un minimo (generalmente indicato con $\perp$) Definiamo una categoria cartesiana chiusa C *pointed poset enriched* se:

- Dati due oggetti A, B, $(Mor(A, B), \sqsubseteq_{A,B}, \perp_{A,B})$ è un pointed poset
- Composizione, prodotto e currying sono monotoni
- Per ogni $f : A \to B$, per ogni gioco C, $\perp_{B,C} \circ f = \perp_{A,B}$

Definiamo una categoria cartesiana chiusa C *razionale* se:

order enrichement e razionalità

Definiamo un *pointed poset (ppo)* come un poset con un minimo (generalmente indicato con $\perp$)   Definiamo una categoria cartesiana chiusa C *pointed poset enriched* se:

- Dati due oggetti A, B, $(Mor(A,B), \sqsubseteq_{A,B}, \perp_{A,B})$ è un pointed poset
- Composizione, prodotto e currying sono monotoni
- Per ogni $f : A \to B$, per ogni gioco C, $\perp_{B,C} \circ f = \perp_{A,B}$

Definiamo una categoria cartesiana chiusa C *razionale* se:

- è ppo-enriched

order enrichement e razionalità

Definiamo un *pointed poset (ppo)* come un poset con un minimo (generalmente indicato con $\perp$)   Definiamo una categoria cartesiana chiusa C *pointed poset enriched* se:

- Dati due oggetti A, B, $(Mor(A, B), \sqsubseteq_{A,B}, \perp_{A,B})$ è un pointed poset
- Composizione, prodotto e currying sono monotoni
- Per ogni $f : A \to B$, per ogni gioco C, $\perp_{B,C} \circ f = \perp_{A,B}$

Definiamo una categoria cartesiana chiusa C *razionale* se:

- è ppo-enriched
- per ogni $f : A \times B \to B$ si ha:
  - La catena $(f^{(k)} | k \in \omega)$ definita da $f^{(0)} = \perp_{A,B}$ e $f^{k+1} = f \circ < id_A, f^{(k)} >$ ammette *least upper bound* f^{∇}
  - Dati $g : C \to A$ e $h : B \to D$, $g \circ f^{\nabla} \circ h = \bigsqcup_{k \in \omega} g \circ f^{(k)} \circ h$

Denotazione di PCF

Denotazione di PCF

Dato A gioco, date $[\sigma], [\tau]$ classi di strategie di A, definiamo

$$[\sigma] \sqsubseteq [\tau] \Leftrightarrow \sigma \preccurlyeq_s \tau$$

Denotazione di PCF

Dato A gioco, date $[\sigma], [\tau]$ classi di strategie di A, definiamo

$$[\sigma] \sqsubseteq [\tau] \Leftrightarrow \sigma \precsim_s \tau$$

Teorema

$K_!(\mathcal{G})$ con l'ordine $\sqsubseteq$ è razionale

Denotazione di PCF

Dato A gioco, date $[\sigma], [\tau]$ classi di strategie di A, definiamo

$$[\sigma] \sqsubseteq [\tau] \Leftrightarrow \sigma \precsim_s \tau$$

Teorema

$K_!(\mathcal{G})$ con l'ordine $\sqsubseteq$ è razionale

Teorema

Sia C una categoria cartesiana chiusa razionale. Si ha che:

- Fissata la denotazione dei tipi base di PCF in C (ogni tipo viene denotato con un oggetto)
- Fissata la denotazione delle costanti di PCF in C (ogni termine di tipo τ viene denotato con un morfismo di $1 \to [\![\tau]\!]$)

allora la denotazione può essere estesa a tutti i termini di PCF

example

Bool

- $M_{Bool} = \{*, t, f\}$
- $\lambda_{Bool} = \{(*, OQ); (t, PA); (f, PA)\}$
- $P_{Bool} = \{\epsilon, *, *t, *f\}$
- $\approx_{Bool} = id_{Bool}$

Nat

- $M_{Nat} = \{*, \underline{0}, \underline{1}, \dots\}$
- $\lambda_{Nat} = \{(*, OQ), (\underline{0}, PA), (\underline{1}, PA), \dots\}$
- $P_{Nat} = \{\epsilon, *, *\underline{0}, *\underline{1}, \dots\}$
- $\approx_{Nat} = id_{Nat}$

Interpretazione dei termini

Per poter usare il teorema prima dobbiamo fissare la denotazione dei giochi e delle costanti Indichiamo con $[\![\cdot]\!]$ la denotazione

Tipi

La denotazione di un tipo è un gioco

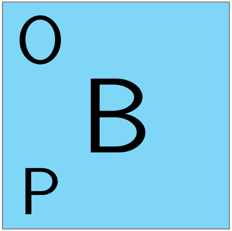
- $[\![Bool]\!] = Bool$
- $[\![Nat]\!] = Nat$
- $[\![S \times T]\!] = [\![S]\!] \& [\![T]\!]$
- $[\![S \to T]\!] = [\![S]\!] \Rightarrow [\![T]\!]$

Termini

La denotazione di un termine di tipo T è una strategia di $(1\&A_1\&\ldots\&A_n) \to T$

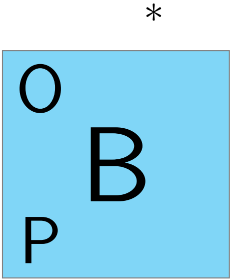
- $[\![true : Bool]\!] = \sigma_{tt} : 1 \to Bool$
- $[\![false : Bool]\!] = \sigma_{ff} : 1 \to Bool$
- $[\![n : Nat]\!] = \sigma_n : 1 \to Nat$
- $[\![M + N]\!] =< \sigma_{add}, < [\![M]\!], [\![N]\!] >> ; App$
- $[\![Eq?MN]\!] =< \sigma_{eq}, < [\![M]\!], [\![N]\!] >> ; App$
- $[\![< s, t >]\!] =< [\![s]\!], [\![t]\!] >$
- $[\![Proj_1 < s, t >]\!] = [\![s]\!]$
- $[\![Proj_2 < s, t >]\!] = [\![t]\!]$

La strategia σ_{tt} è la seguente



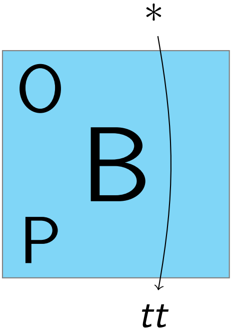
La strategia σ_{tt} è la seguente

- Alla domanda di O...



La strategia σ_{tt} è la seguente

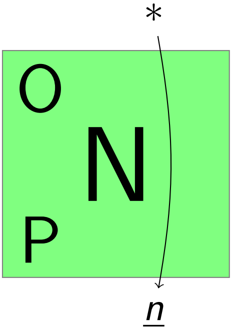
- Alla domanda di O...
- ...P risponde tt



La strategia σ_{tt} è la seguente

- Alla domanda di O...
- ... P risponde tt

La strategia σ_n funziona allo stesso modo



$[\![< s, t >]\!]$

Dato il termine $< s, t >$ ad esso associamo la strategia prodotto $< [\![s]\!], [\![t]\!] >$.
Ad esempio nel caso $[\![< n, m >]\!]$:

O

$N_1 \& N_2$

P

$[\![< s, t >]\!]$

Dato il termine $< s, t >$ ad esso associamo la strategia prodotto $< [\![s]\!], [\![t]\!] >$.
Ad esempio nel caso $[\![< n, m >]\!]$:

- Se O fa una domanda sulla prima componente...

$*_1$

O

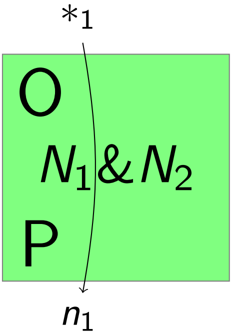
$N_1 \& N_2$

P

$[\![< s, t >]\!]$

Dato il termine $< s, t >$ ad esso associamo la strategia prodotto $< [\![s]\!], [\![t]\!] >$.
Ad esempio nel caso $[\![< n, m >]\!]$:

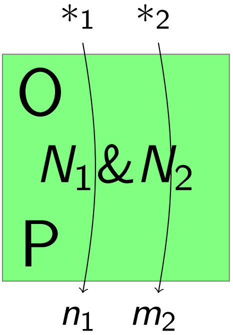
- Se O fa una domanda sulla prima componente...
- ... P risponde usando la strategia $[\![n]\!]$



$[\![< s, t >]\!]$

Dato il termine $< s, t >$ ad esso associamo la strategia prodotto $< [\![s]\!], [\![t]\!] >$.
Ad esempio nel caso $[\![< n, m >]\!]$:

- Se O fa una domanda sulla prima componente...
- ... P risponde usando la strategia $[\![n]\!]$
- Altrimenti risponde usando la strategia $[\![m]\!]$



P   O

$$\mathcal{N}_1 \;\&\; \mathcal{N}_2 \Rightarrow \mathcal{N}_3$$

P   O   O   P

La strategia σ_{add} rappresenta la somma tra numeri naturali

$*_3$

$$ {}^{P}_{O}N_1 \;\&\; {}^{P}_{O}N_2 \Rightarrow {}^{O}_{P}N_3 $$

La strategia σ_{add} rappresenta la somma tra numeri naturali

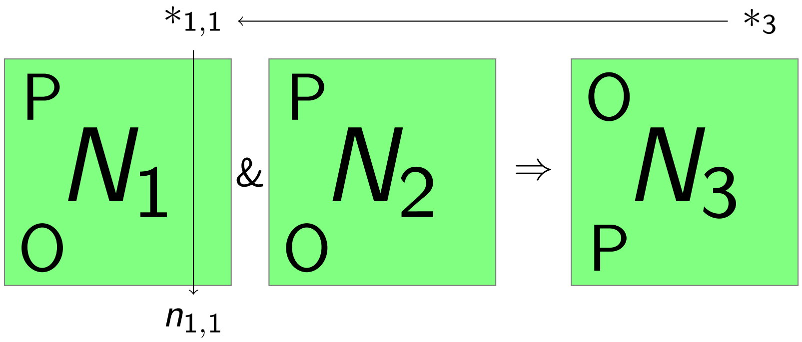
- La domanda di O $(*_3)$...

$$*_{1,1} \longleftarrow *_3$$

$$\begin{array}{c} P \\ N_1 \\ O \end{array} \;\&\; \begin{array}{c} P \\ N_2 \\ O \end{array} \;\Rightarrow\; \begin{array}{c} O \\ N_3 \\ P \end{array}$$

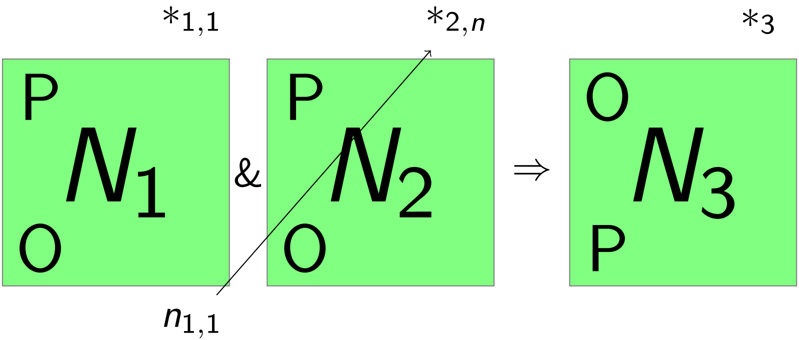
La strategia σ_{add} rappresenta la somma tra numeri naturali

- La domanda di O $(*_3)$...
- ...viene girata a N_1 da P.
  O risponde con il primo addendo



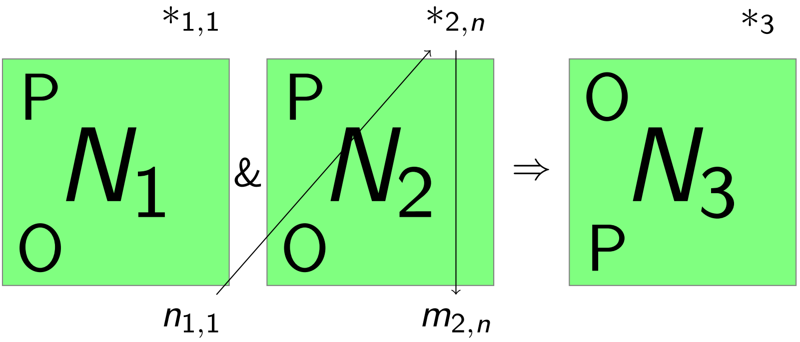
La strategia σ_{add} rappresenta la somma tra numeri naturali

- La domanda di O $(*_3)$...
- ...viene girata a N_1 da P.
  O risponde con il primo addendo



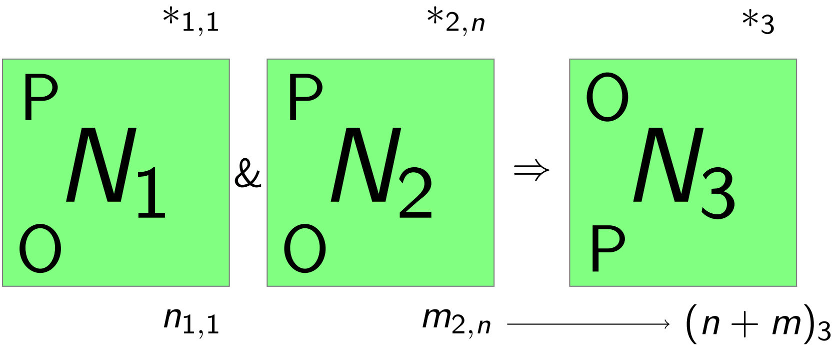
La strategia σ_{add} rappresenta la somma tra numeri naturali

- La domanda di O ($*_3$)...
- ...viene girata a N_1 da P. O risponde con il primo addendo
- P chiede il secondo addendo ($*_{2,n}$). O risponde



La strategia σ_{add} rappresenta la somma tra numeri naturali

- La domanda di O $(*_3)$...
- ...viene girata a N_1 da P. O risponde con il primo addendo
- P chiede il secondo addendo $(*_{2,n})$. O risponde



La strategia σ_{add} rappresenta la somma tra numeri naturali

- La domanda di O $(*_3)$...
- ...viene girata a N_1 da P. O risponde con il primo addendo
- P chiede il secondo addendo $(*_{2,n})$. O risponde
- P risponde alla domanda iniziale con la somma degli addendi

$[\![N + M]\!] =< \sigma_{add}, < [\![N]\!], [\![M]\!] >> ; App$

P

$N_1 \& N_2 \Rightarrow N_3 \&$

O

P

$N_4 \& N_5$

O

$\Rightarrow$

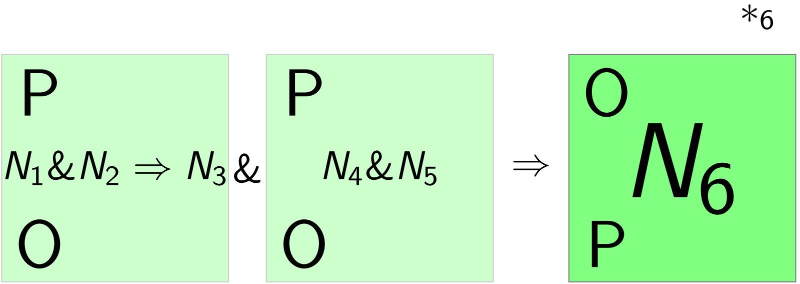
O

N_6

P

App rappresenta l'applicazione tra termini. Nell'esempio:

$[\![N+M]\!] = <\sigma_{add}, <[\![N]\!], [\![M]\!]>> \,;\, App$



App rappresenta l'applicazione tra termini. Nell'esempio:

- Ogni mossa di O viene copiata da P sulla componente corrispondente

$[\![N + M]\!] =< \sigma_{add}, < [\![N]\!], [\![M]\!] >> ; App$

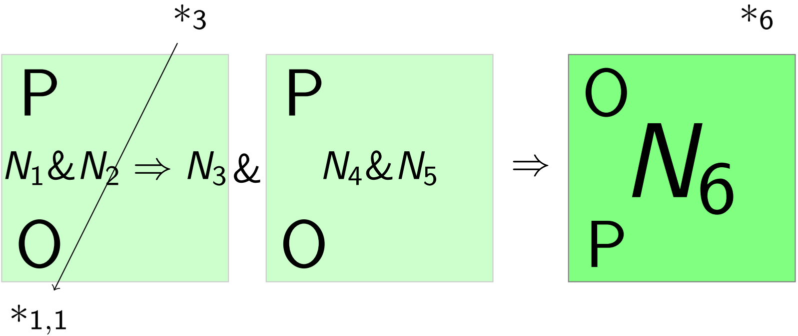
$$*_3 \longleftarrow *_6$$

P	P	O
$N_1 \& N_2 \Rightarrow N_3$ &	$N_4 \& N_5$ ⇒	N_6
O	O	P

App rappresenta l'applicazione tra termini. Nell'esempio:

- Ogni mossa di O viene copiata da P sulla componente corrispondente

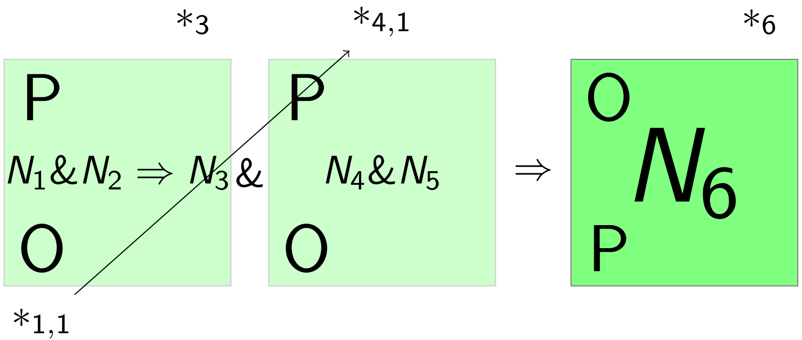
$[\![N + M]\!] =< \sigma_{add}, < [\![N]\!], [\![M]\!] >> ; App$



App rappresenta l'applicazione tra termini. Nell'esempio:

- Ogni mossa di O viene copiata da P sulla componente corrispondente
- O è costretto a muovere rispettando le regole di σ_{add} e $< [\![N]\!], [\![M]\!] >$

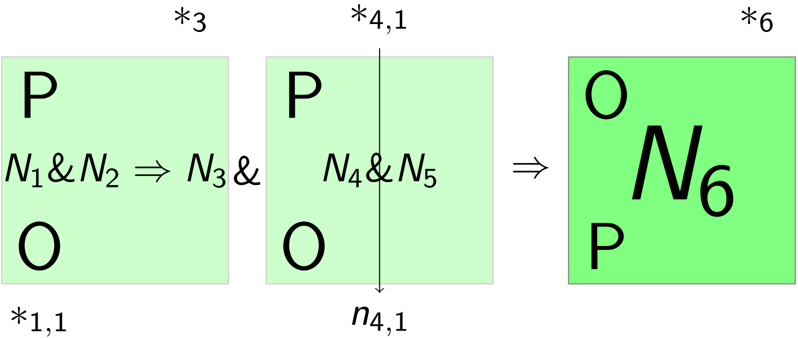
$[\![N+M]\!] = <\sigma_{add}, <[\![N]\!],[\![M]\!]>> \,;\, App$



App rappresenta l'applicazione tra termini. Nell'esempio:

- Ogni mossa di O viene copiata da P sulla componente corrispondente
- O è costretto a muovere rispettando le regole di σ_{add} e $<[\![N]\!],[\![M]\!]>$

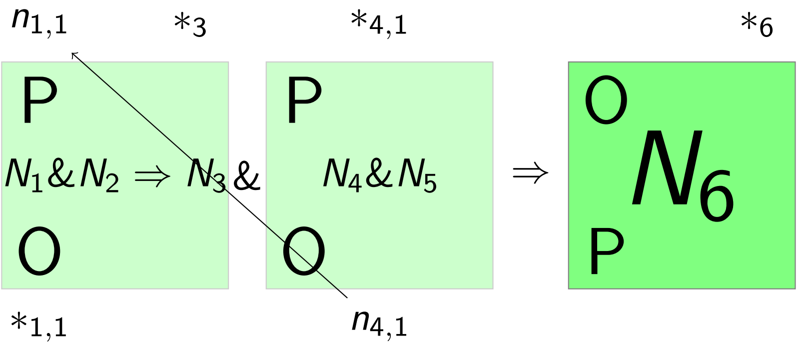
$[\![N+M]\!] =< \sigma_{add}, < [\![N]\!], [\![M]\!] >> ⨾ App$



App rappresenta l'applicazione tra termini. Nell'esempio:

- Ogni mossa di O viene copiata da P sulla componente corrispondente
- O è costretto a muovere rispettando le regole di σ_{add} e $< [\![N]\!], [\![M]\!] >$

$[\![N+M]\!] = <\sigma_{add}, <[\![N]\!], [\![M]\!]>> \fatsemi App$



App rappresenta l'applicazione tra termini. Nell'esempio:

- Ogni mossa di O viene copiata da P sulla componente corrispondente
- O è costretto a muovere rispettando le regole di σ_{add} e $<[\![N]\!], [\![M]\!]>$

$[\![N+M]\!] =< \sigma_{add}, < [\![N]\!], [\![M]\!] >> \,;\, App$

$n_{1,1}$ $\quad *_3$ $\quad *_{4,1}$ $\quad *_6$

P   P   O

$$N_1 \& N_2 \Rightarrow N_3 \; \& \; N_4 \& N_5 \Rightarrow N_6$$

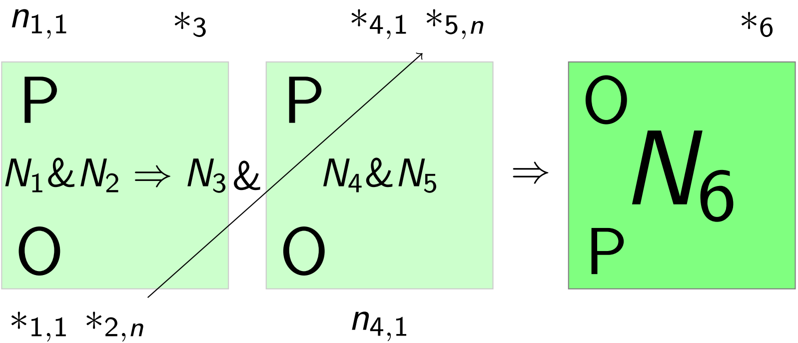
O   O   P

$*_{1,1}$ $\quad *_{2,n}$ $\quad n_{4,1}$

App rappresenta l'applicazione tra termini. Nell'esempio:

- Ogni mossa di O viene copiata da P sulla componente corrispondente
- O è costretto a muovere rispettando le regole di σ_{add} e $< [\![N]\!], [\![M]\!] >$

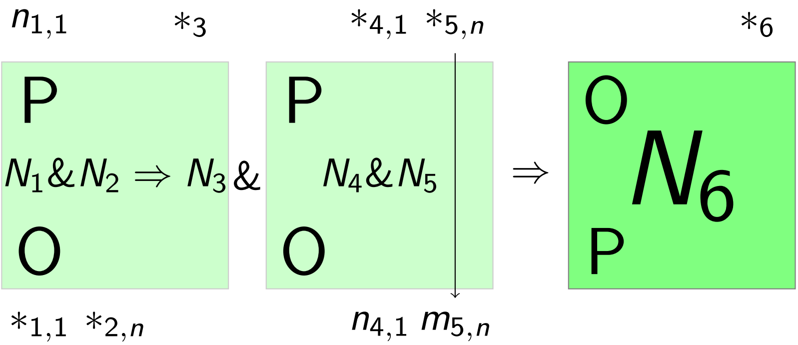
$[\![N+M]\!] =< \sigma_{add}, < [\![N]\!], [\![M]\!] >> ⨾ App$



App rappresenta l'applicazione tra termini. Nell'esempio:

- Ogni mossa di O viene copiata da P sulla componente corrispondente
- O è costretto a muovere rispettando le regole di σ_{add} e $< [\![N]\!], [\![M]\!] >$

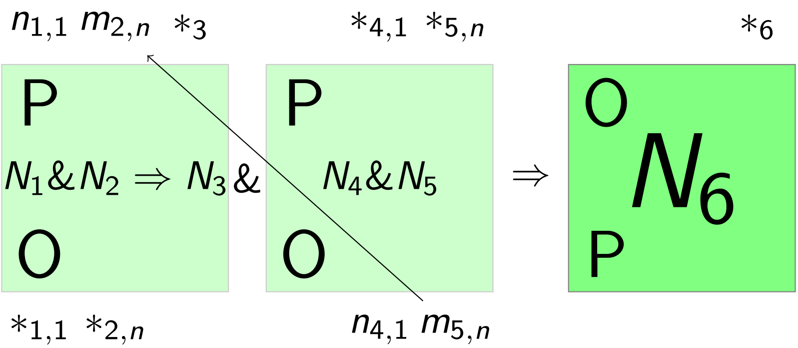
$[\![N + M]\!] =< \sigma_{add}, < [\![N]\!], [\![M]\!] >> ; App$



App rappresenta l'applicazione tra termini. Nell'esempio:

- Ogni mossa di O viene copiata da P sulla componente corrispondente
- O è costretto a muovere rispettando le regole di σ_{add} e $< [\![N]\!], [\![M]\!] >$

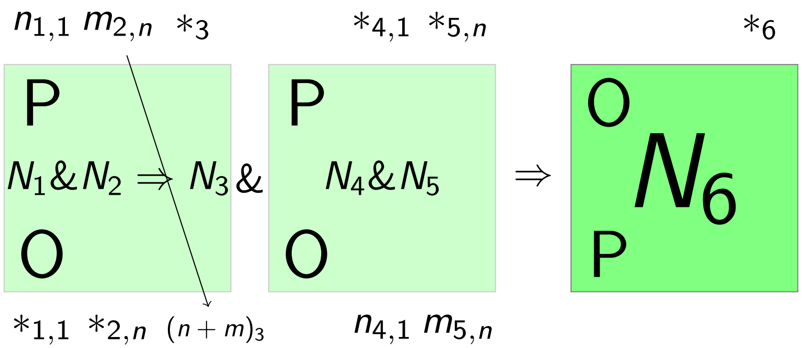
$[\![N + M]\!] =< \sigma_{add}, < [\![N]\!], [\![M]\!] >> \fatsemi App$



App rappresenta l'applicazione tra termini. Nell'esempio:

- Ogni mossa di O viene copiata da P sulla componente corrispondente
- O è costretto a muovere rispettando le regole di σ_{add} e $< [\![N]\!], [\![M]\!] >$

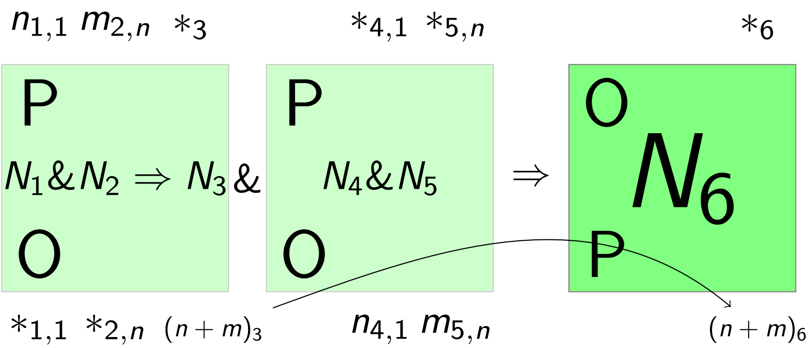
$[\![N+M]\!] = < \sigma_{add}, < [\![N]\!], [\![M]\!] >> ; App$



App rappresenta l'applicazione tra termini. Nell'esempio:

- Ogni mossa di O viene copiata da P sulla componente corrispondente
- O è costretto a muovere rispettando le regole di σ_{add} e $< [\![N]\!], [\![M]\!] >$

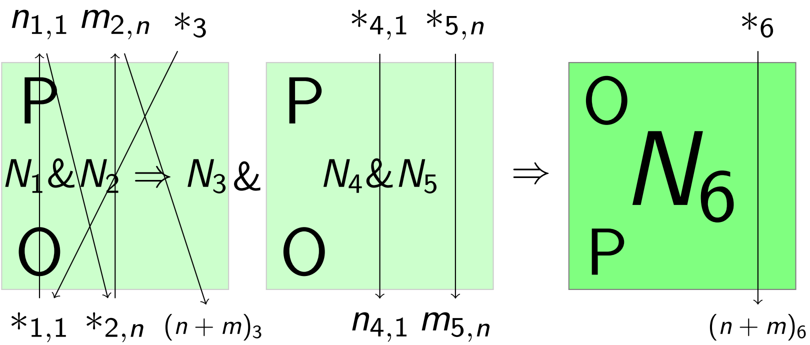
$[\![N+M]\!] =< \sigma_{add}, < [\![N]\!], [\![M]\!] >> \fatsemi App$



App rappresenta l'applicazione tra termini. Nell'esempio:

- Ogni mossa di O viene copiata da P sulla componente corrispondente
- O è costretto a muovere rispettando le regole di σ_{add} e $< [\![N]\!], [\![M]\!] >$

$[\![N+M]\!] =< \sigma_{add}, < [\![N]\!], [\![M]\!] >> \fatsemi App$



Notiamo che le sequenze di mosse su ogni singolo tavolo sono partite valide del gioco corrispondente, come richiesto da $\fatsemi$

$[\![Eq?mn]\!]$

P

$N_1 \& N_2 \Rightarrow B_3 \&$

O

P

$N_4 \& N_5$

O

$\Rightarrow$

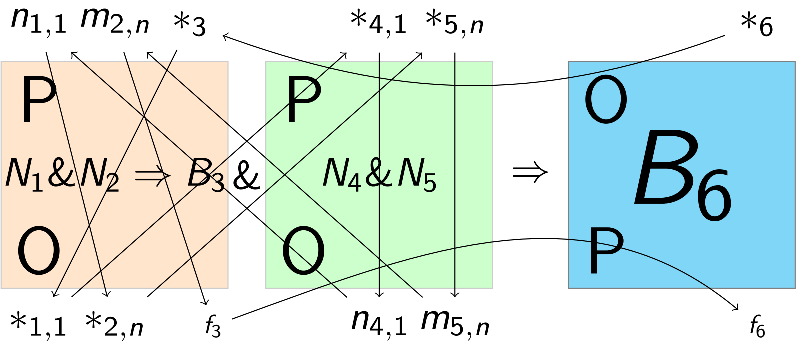
O

B_6

P

In modo analogo al precedente possiamo dare la strategia
$[\![Eq?MN]\!] = < \sigma_{eq}, < [\![M]\!], [\![N]\!] >> ; App$

$[\![Eq?mn]\!]$



In modo analogo al precedente possiamo dare la strategia
$[\![Eq?MN]\!] = < \sigma_{eq}, < [\![M]\!], [\![N]\!] >> ; App$

Per parlare di variabili occorre introdurre il concetto di *ambiente*

Ambiente

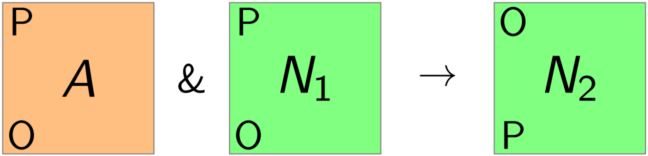
Dato il termine $t : T$, siano $x_1 : S_1, \ldots, x_n : S_n$ le variabili libere che vi compaiono
Chiamiamo $\Pi = \{x_1 : S_1, \ldots, x_n : S_n\}$ l'ambiente di t

Con $\Delta \vdash t : T$ indichiamo che $\Pi \subseteq \Delta$
Lo scopo dell'ambiente quello di fare da *input*, cioé assegnare un valore alle variabili libere

Termini

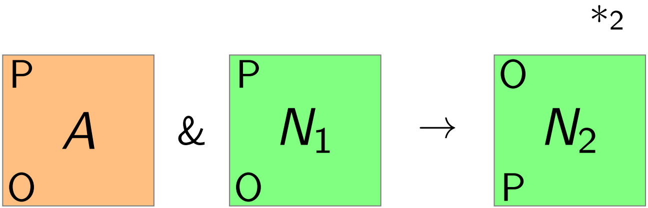
- $[\![x : S]\!] = Cc : 1\&[\![S]\!] \to [\![S]\!]$
- $[\![\Pi \vdash \lambda(x : S).t : T]\!] = Cur([\![t]\!]) : 1\&[\![\Pi]\!] \to [\![S]\!] \Rightarrow [\![T]\!]$ dove $\Pi, x : S \vdash t : T$
- $[\![if\ \ B\ then\ M\ else\ N]\!] =< Cond, < [\![B]\!], [\![M]\!], [\![N]\!] >> \fatsemi App$
- $[\![M(N)]\!] =< [\![M]\!], [\![N]\!] > \fatsemi App$

$[\![x]\!]$



La strategia $[\![x]\!]$ consiste nel copiare le mosse di O sul primo tavolo (strategia *copycat*)

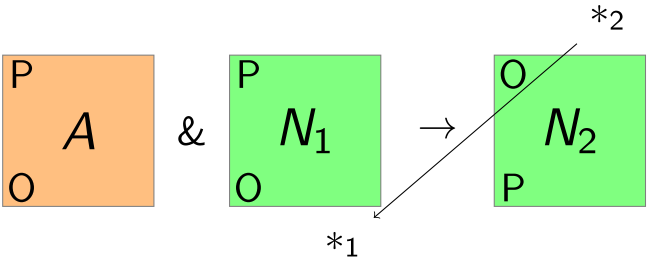
$[\![x]\!]$



La strategia $[\![x]\!]$ consiste nel copiare le mosse di O sul primo tavolo (strategia *copycat*)

- P copia le mosse di O tra i due tavoli

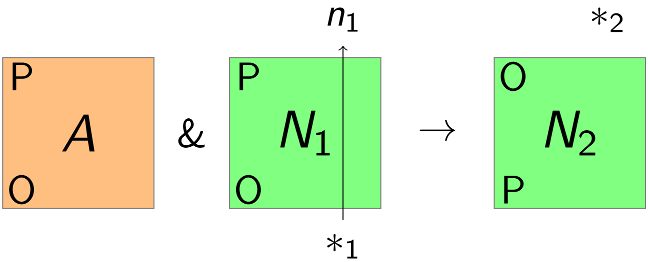
$[\![x]\!]$



La strategia $[\![x]\!]$ consiste nel copiare le mosse di O sul primo tavolo (strategia *copycat*)

- P copia le mosse di O tra i due tavoli

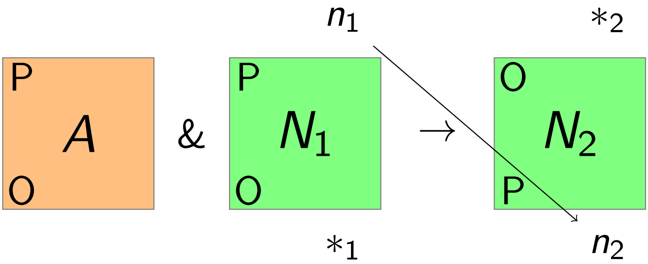
$[\![x]\!]$



La strategia $[\![x]\!]$ consiste nel copiare le mosse di O sul primo tavolo (strategia *copycat*)

- P copia le mosse di O tra i due tavoli
- O muove in maniera arbitraria

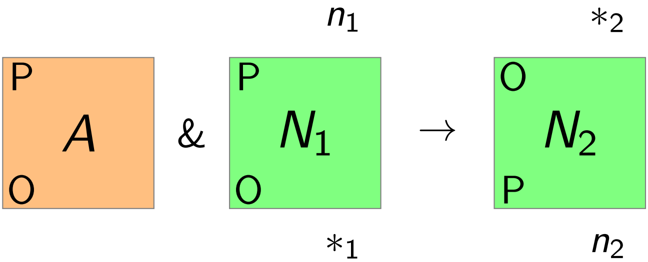
$[\![x]\!]$



La strategia $[\![x]\!]$ consiste nel copiare le mosse di O sul primo tavolo (strategia *copycat*)

- P copia le mosse di O tra i due tavoli
- O muove in maniera arbitraria

$[\![x]\!]$

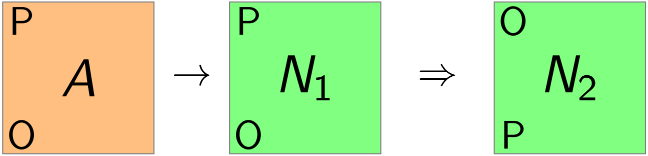


La strategia $[\![x]\!]$ consiste nel copiare le mosse di O sul primo tavolo (strategia *copycat*)

- P copia le mosse di O tra i due tavoli
- O muove in maniera arbitraria

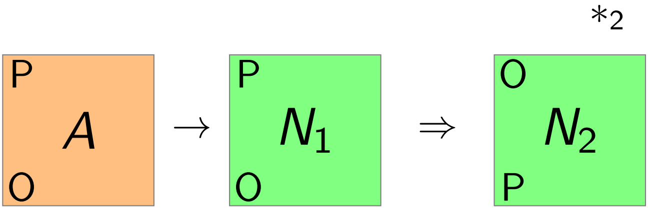
Le mosse di O nell'ambiente influenzano il comportamento di P durante la partita

$[\![\Pi \vdash \lambda(x : S).t : T]\!] = Cur([\![t]\!]) : 1\&[\![\Pi]\!] \to [\![S]\!] \Rightarrow [\![T]\!]$



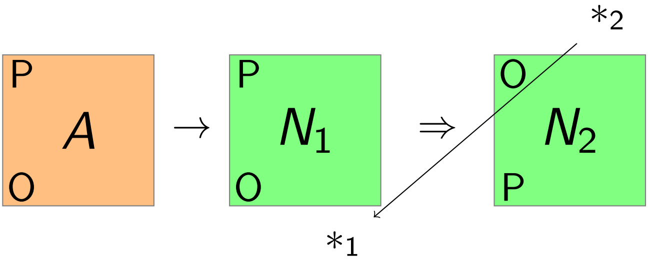
Notiamo che una strategia $\sigma : A\&B \to C$ può essere canonicamente interpretata come una strategia di $A \to B \Rightarrow C$

$[\![\Pi \vdash \lambda(x : S).t : T]\!] = Cur([\![t]\!]) : 1\&[\![\Pi]\!] \to [\![S]\!] \Rightarrow [\![T]\!]$



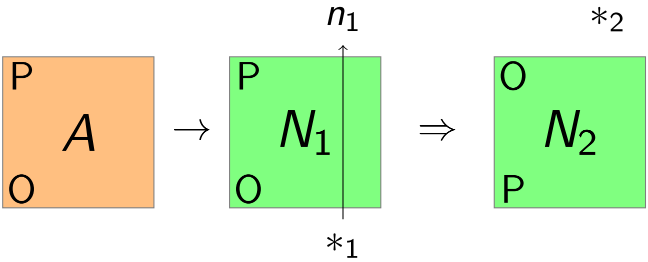
Notiamo che una strategia $\sigma : A\&B \to C$ può essere canonicamente interpretata come una strategia di $A \to B \Rightarrow C$

$[\![\Pi \vdash \lambda(x : S).t : T]\!] = Cur([\![t]\!]) : 1\&[\![\Pi]\!] \to [\![S]\!] \Rightarrow [\![T]\!]$



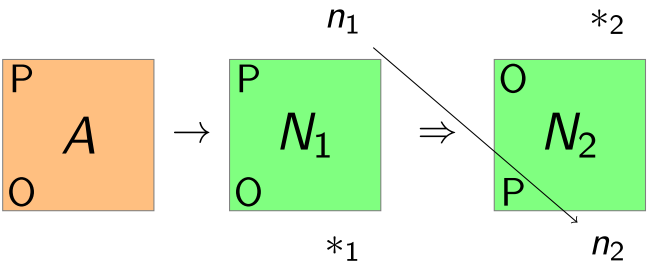
Notiamo che una strategia $\sigma : A\&B \to C$ può essere canonicamente interpretata come una strategia di $A \to B \Rightarrow C$

$[\![\Pi \vdash \lambda(x : S).t : T]\!] = Cur([\![t]\!]) : 1\&[\![\Pi]\!] \to [\![S]\!] \Rightarrow [\![T]\!]$



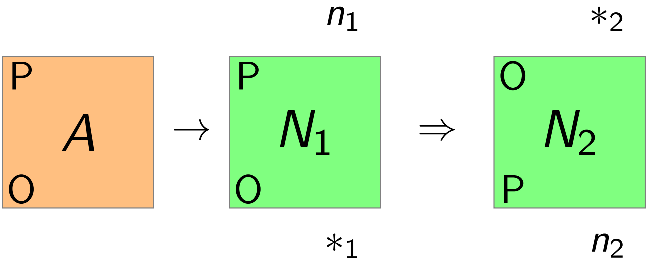
Notiamo che una strategia $\sigma : A\&B \to C$ può essere canonicamente interpretata come una strategia di $A \to B \Rightarrow C$

$[\![\Pi \vdash \lambda(x : S).t : T]\!] = Cur([\![t]\!]) : 1\&[\![\Pi]\!] \to [\![S]\!] \Rightarrow [\![T]\!]$



Notiamo che una strategia $\sigma : A\&B \to C$ può essere canonicamente interpretata come una strategia di $A \to B \Rightarrow C$

$[\![\Pi \vdash \lambda(x : S).t : T]\!] = Cur([\![t]\!]) : 1\&[\![\Pi]\!] \to [\![S]\!] \Rightarrow [\![T]\!]$



Notiamo che una strategia $\sigma : A\&B \to C$ può essere canonicamente interpretata come una strategia di $A \to B \Rightarrow C$
Chiamiamo la strategia ottenuta $Cur(\sigma)$

P

$B_1 \& N_2 \& N_3$

O

$\Rightarrow$

O

N_4

P

La strategia *cond* rappresenta il costrutto *if* . . . *then* . . . *else* . . .

$$*_4$$

P $B_1 \& N_2 \& N_3$ O	$\Rightarrow$	O N_4 P

La strategia *cond* rappresenta il costrutto *if* . . . *then* . . . *else* . . .

- Alla domanda di O, P controlla la guardia (B_1)

$$*_1 \longleftarrow *_4$$

P

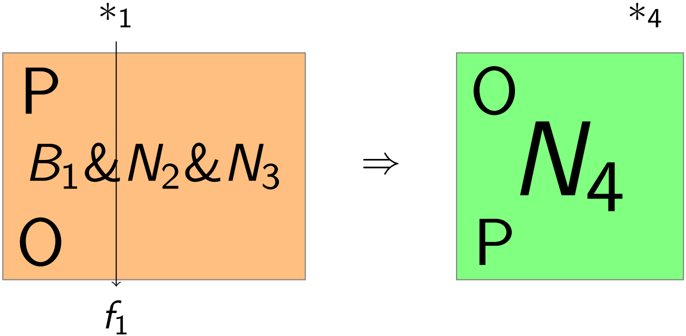
$$B_1 \& N_2 \& N_3 \Rightarrow N_4$$

O O

P

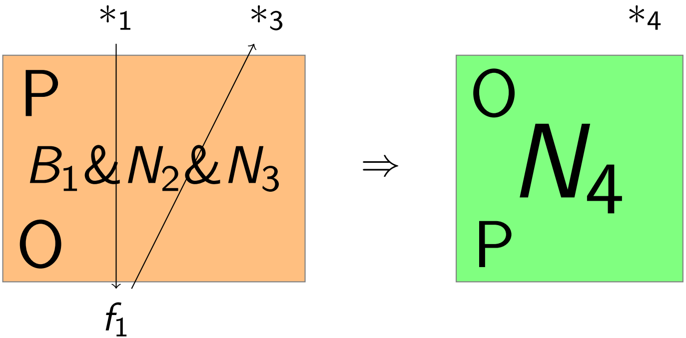
La strategia *cond* rappresenta il costrutto *if . . . then . . . else . . .*

- Alla domanda di O, P controlla la guardia (B_1)



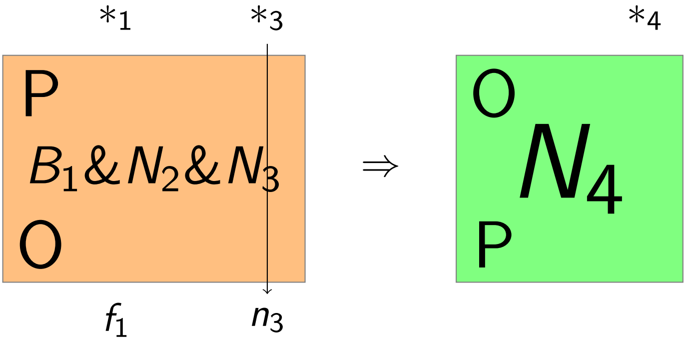
La strategia *cond* rappresenta il costrutto *if* ... *then* ... *else* ...

- Alla domanda di O, P controlla la guardia (B_1)
- Se la guardia risulta vera P consulta N_2, altrimenti N_3



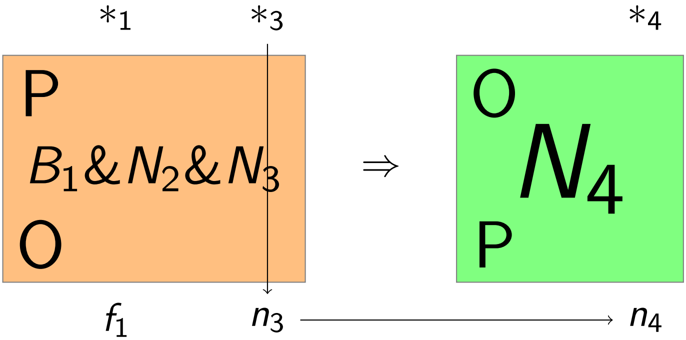
La strategia *cond* rappresenta il costrutto *if* . . . *then* . . . *else* . . .

- Alla domanda di O, P controlla la guardia (B_1)
- Se la guardia risulta vera P consulta N_2, altrimenti N_3



La strategia *cond* rappresenta il costrutto *if* . . . *then* . . . *else* . . .

- Alla domanda di O, P controlla la guardia (B_1)
- Se la guardia risulta vera P consulta N_2, altrimenti N_3
- P copia la risposta data su N_4



La strategia *cond* rappresenta il costrutto *if* ... *then* ... *else* ...

- Alla domanda di O, P controlla la guardia (B_1)
- Se la guardia risulta vera P consulta N_2, altrimenti N_3
- P copia la risposta data su N_4

Teorema

La semantica dei giochi data finora è *sound*, cioè:

- se $[\![s]\!] = [\![t]\!]$
- se s e t sono costruiti senza usare il ricursore Y

allora $s \stackrel{\text{obs}}{=} t$

Ricursore

- Definiamo Θ come segue

$$[\![F : (A \to A) \Rightarrow A \vdash \lambda(f : A \to A).f(F(f))]\!] =$$
$$\Theta : 1\&((A \Rightarrow A) \Rightarrow A) \to (A \Rightarrow A) \Rightarrow A$$

- $[\![Y]\!] = \Theta^{\nabla} : 1 \to (A \Rightarrow A) \Rightarrow A$

Notiamo che per le definizioni date precedentemente si ha $[\![Y]\!] = \bigsqcup_{n \in \omega} [\![Y^{(n)}]\!]$ dove:

$$Y^{(0)} \equiv \lambda(f : A \to A).\perp_A$$
$$Y^{(k+1)} \equiv \lambda(f : A \to A).f(Y^{(k)}(f))$$

Esempio

Secondo la nostra semantica si ha $[\![Y(\lambda x.5)]\!] = \sigma_5$?

$$Y^{(0)}(\lambda x.5) \twoheadrightarrow \perp_{Nat}$$
$$[\![Y^{(0)}(\lambda x.5)]\!] = \{\epsilon\}$$

$$\begin{aligned} Y^{(1)}(\lambda x.5) &\twoheadrightarrow [\lambda x.5](Y^{(0)}(\lambda x.5)) \\ &\twoheadrightarrow 5 \\ [\![Y^{(1)}(\lambda x.5)]\!] &= \sigma_5 \\ [\![Y(\lambda x.5)]\!] &= \sigma_5 \end{aligned}$$

Esempio

Posto $F \equiv \lambda f.\lambda x.f(x+1)$ secondo la nostra semantica si ha $[\![Y(F)]\!] = \{\epsilon\}$?

$$Y^{(0)}(F) \twoheadrightarrow \perp_{N\to N}$$
$$[\![Y^{(0)}(F)]\!] = \{\epsilon\}$$

Supponiamo $Y^{(k)}(F) \twoheadrightarrow \perp_{N\to N}$

$$\begin{aligned} Y^{(k+1)}(F) &\twoheadrightarrow F(Y^{(k)}(F)) \\ &\twoheadrightarrow F(\perp_{N\to N}) \\ &\twoheadrightarrow \lambda x.\ \perp_{N\to N} (x+1) \\ &\twoheadrightarrow \lambda x.\ \perp_N \\ &\equiv \perp_{N\to N} \end{aligned}$$

Quindi possiamo affermare $[\![Y(F)]\!] = \bigsqcup_{n\in\omega} [\![Y^{(n)}(F)]\!] = \bigsqcup_{n\in\omega} \{\epsilon\} = \{\epsilon\}$

Intensional full abstraction

Teorema

Per ogni tipo τ di PCF, posto $T = [\![\tau]\!]$, si ha che $1 \to T$ è un dI-domain; in particolare $\mathcal{M}(K_!(\mathcal{G}))$ è un modello algebrico

Teorema

$\mathcal{M}(K_!(\mathcal{G}))$ è un modello intensionally fully abstract di PCF

Full abstraction

Definiamo il gioco di *Sierpinsky* Σ tale che:

- $M_\Sigma = \{q, a\}$ dove $\lambda_\Sigma(q) = OQ$ e $\lambda_\Sigma(a) = PA$
- $P_\Sigma = \{\epsilon, q, qa\}$ e $\approx_\Sigma = id_{P_\Sigma}$

Definiamo il preordine $\lesssim_A$ sulle strategie del gioco A:
$x \lesssim_A y \;\Leftrightarrow\; \forall \alpha \to \Sigma . x; \alpha \preccurlyeq_\Sigma y; \alpha$
Definiamo $\mathcal{E} = K_!(\mathcal{G}) / \lesssim$, cioè la categoria tale che:

- $\mathcal{E}_0 = K_!(\mathcal{G})_0$
- $Mor_{\mathcal{E}}(A, B) = Mor_{K_!(\mathcal{G})}(A, B) / \lesssim_{A \Rightarrow B}$

Teorema

$\mathcal{E}$ è un modello fully abstract per PCF

Universalità

Definiamo un gioco A *effettivamente dato* se:

- Esiste una mappa $e_A : \omega \to M_A$ suriettiva; chiamiamo questa funzione codifica
- Rispetto alla codifica le seguenti funzioni sono calcolabili:
  - λ_A (rispetto a qualche codifica di $\{P, O, Q, A\}$)
  - la funzione caratteristica di P_A
  - la funzione caratteristica di $\approx_A$

Definiamo una strategia *ricorsiva* se la sua funzione parziale associata è calcolabile

Definiamo $\mathcal{G}_{rec}$ la categoria dei giochi effettivamente dati con morfismi le strategie ricorsive

Fatti

- Possono essere definite le categorie $K_!(\mathcal{G}_{rec})$ ed $\mathcal{E}_{rec}$ con ragionamenti analoghi ai precedenti
- $\mathcal{E}_{rec}$ è un modello fully abstract per PCF

Universalità

- Ogni strategia di $\mathcal{M}(\mathcal{E}_{rec})$ è definibile in PCF, cioè è interpretazione di un termine di PCF
- $\mathcal{M}(\mathcal{E}_{rec})$ è un *oggetto iniziale* della categoria FAMOD(PCF)