Un modello fully abstract del PCF

Grilletti Gianluca Barbarino Giovanni

Università di Pisa

May 17, 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

Example

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

Example

- $(\underline{n} + \underline{m}) + \underline{n} : Nat$
- $Eq?(\underline{n})(\underline{m}) : Bool$
- $< true, \underline{n} >: Bool \times Nat$
- $Proj_1 < true, \underline{n} >: Bool$
- $\lambda(x : Nat).x + 1 : Nat \to Nat$ (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*)

Example

- $Eq?(\underline{n})(\underline{m}) : Bool$ 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:

$$
\begin{array}{rl}
\text{add} & \underline{n} + \underline{m} \to \underline{n+m} \\
\text{Eq?} & Eq?(\underline{n})(\underline{n}) \to true \\
 & Eq?(\underline{n})(\underline{m}) \to false \text{ (per } n \text{ ed } m \text{ distinti)} \\
\text{cond} & if[true] \quad then[M] \quad else[N] \to M \\
 & if[false] \quad then[M] \quad else[N] \to N \\
\text{proj} & Proj_1 < M, N > \to M \\
 & Proj_2 < M, N > \to N \\
\alpha & \lambda(x:\sigma).M \to \lambda(y:\sigma).[y/x]M \text{ (con } y \text{ non libera in } M\text{)} \\
\beta & [\lambda(x:\sigma).M](N) \to [N/x]M \\
Y & Y_\sigma \to \lambda(f:\sigma\to\sigma).f(Y_\sigma f)
\end{array}
$$

- 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 $Eval(M) = N$

Theorem (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 $Eval(M) = undef$

Equivalenza osservazionale

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

Example

$C[\,] \equiv \lambda(x : Nat).x + [\,]$
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 $M \overset{obs}{=} N$

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:

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

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

IL CONTENUTO DI QUESTA SLIDE DIPENDE DA QUANTO VOGLIAMO DIRE ALLA FINE

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:

- 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 di mosse tale che:
  1. La prima mossa è di O
  2. P e O si alternano
  3. In ogni momento della partita, 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

Strategie

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

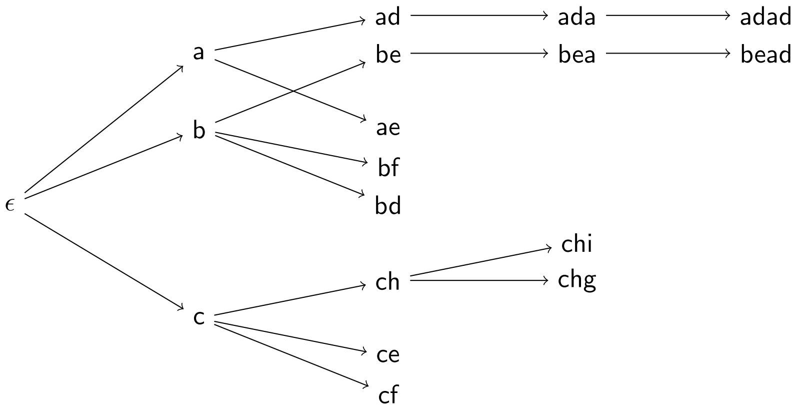
- σ è prefix-closed
- le strategie sono history free, cioè
  - $sab, tac \in \sigma \Rightarrow b = c$
  - $sab \in \sigma, ta \in P_A \Rightarrow tab \in \sigma$

Albero di Gioco

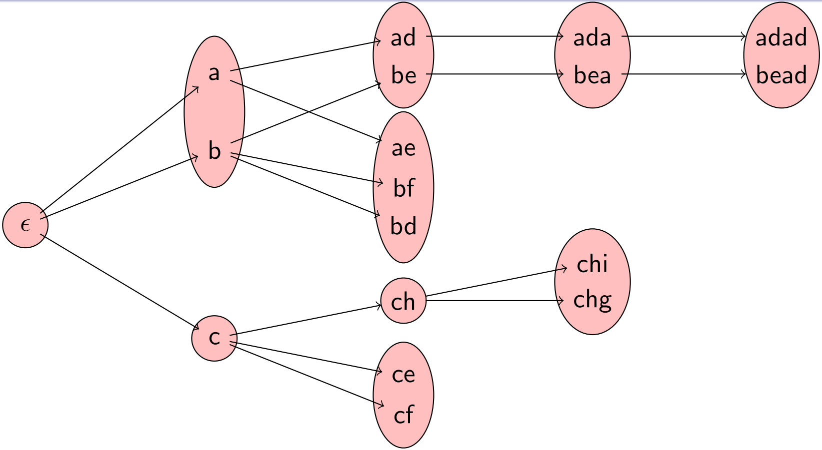
Prendiamo un gioco, il cui set di mosse M è suddiviso dalla funzione di labelling λ in

$$M_{QO} = \{a, b, c\}, \quad M_{AO} = \{g, i\}$$
$$M_{QP} = \{h\}, \quad M_{AP} = \{d, e, f\}$$

Il set di partite valide P, la relazione di equivalenza $\approx$, e le strategie del gioco si possono rappresentare in maniera semplice tramite il *Game Tree*

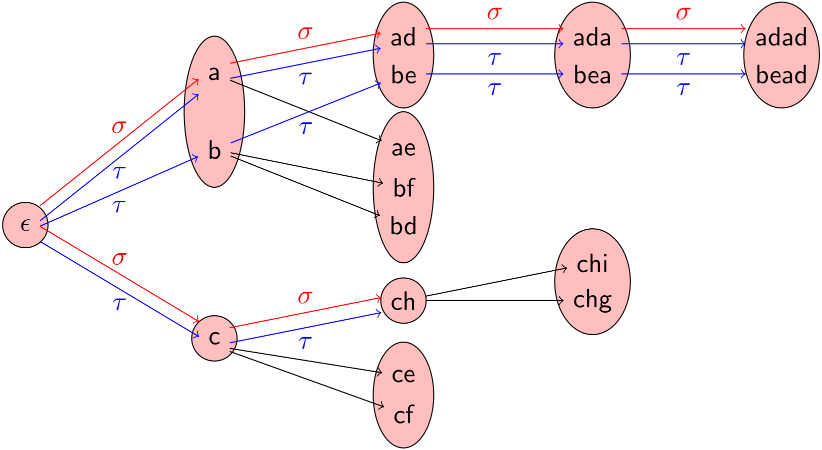


$$M_{QO} = \{a, b, c\}, \quad M_{AO} = \{g, i\}$$
$$M_{QP} = \{h\}, \quad M_{AP} = \{d, e, f\}$$



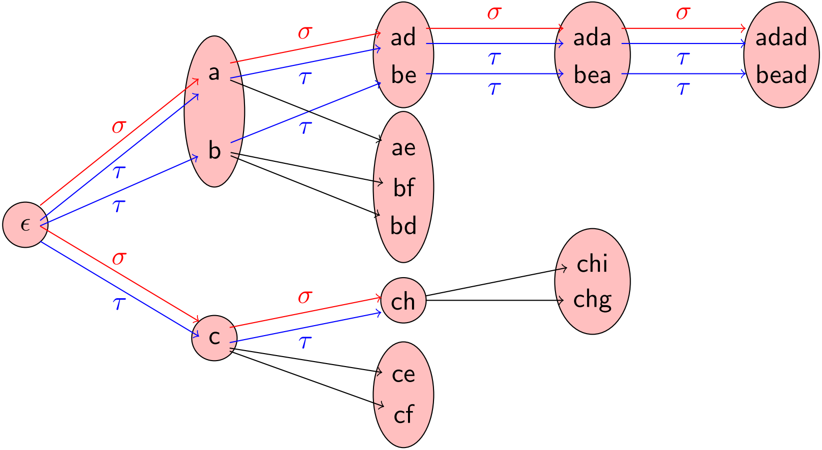
$s \approx t$ se:

- s e t hanno la stessa etichettatura
- se s' e t' sottostringhe iniziali di s e t tali che $|s'| = |t'|$, $s' \approx t'$
- se sa è una partita valida, allora esiste b tale che $tb \approx sa$

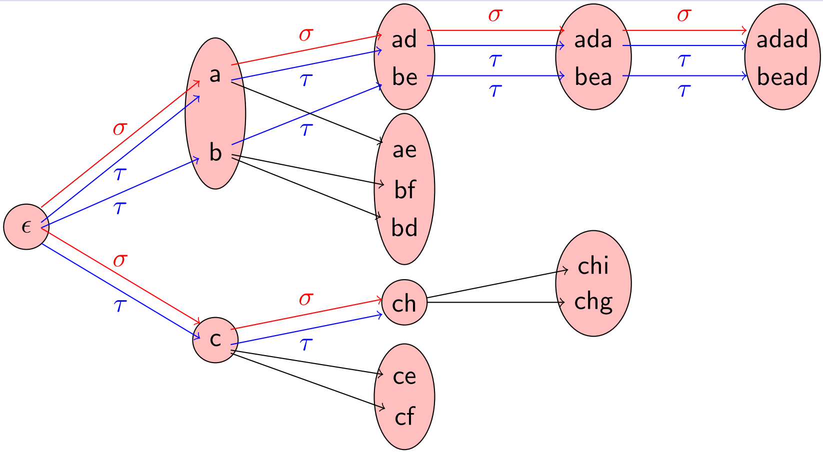


Una strategia σ è un insieme di partite di lunghezza pari tali che:

- σ è prefix-closed
- $sab, tac \in \sigma \Rightarrow b = c$
- $sab \in \sigma, ta \in P_A \Rightarrow tab \in \sigma$



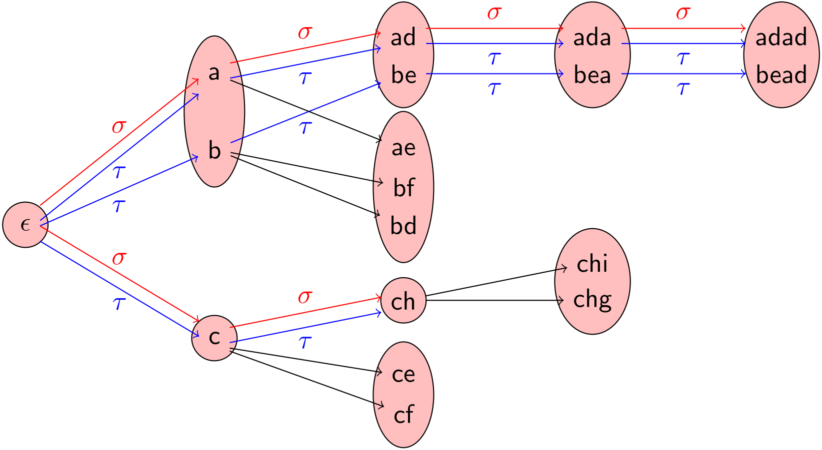
$$f_\sigma(x) = \begin{cases} d \text{ se } x = a \\ / \text{ se } x = b \\ h \text{ se } x = c \\ d \text{ se } x = g \end{cases} \qquad f_\tau(x) = \begin{cases} d \text{ se } x = a \\ e \text{ se } x = b \\ h \text{ se } x = c \\ d \text{ se } x = g \end{cases}$$



Estendiamo la relazione $\approx$ alle strategie; poniamo:

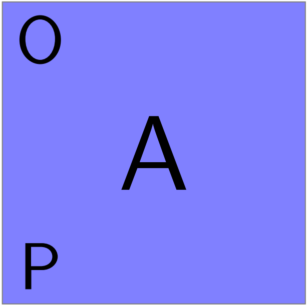
- $\underline{\sigma \preccurlyeq \tau}$ se per ogni $sab \in \sigma$ e $s' \in \tau$, se $sa \approx s'a'$ allora esiste b' tale che $s'a'b' \in \tau$ e $sab \approx s'a'b'$
- $\underline{\sigma \approx \tau}$; iif $\sigma \preccurlyeq \tau \wedge \tau \preccurlyeq \sigma$

In questo caso, avremo $\sigma \preccurlyeq \tau, \sigma \not\approx \sigma, \tau \approx \tau$



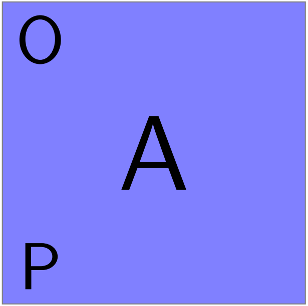
- $\preccurlyeq$ è un preordine sulle strategie; di conseguenza $\approx$ è una relazione di equivalenza parziale
- Nel caso l'equivalenza $\approx$ del gioco sia l'identità, il game tree diventa un albero semplice, e l'ordine tra strategie si può vedere come inclusione di insiemi o tra le funzioni parziali

Come rappresentiamo i giochi (il tavolo insomma)

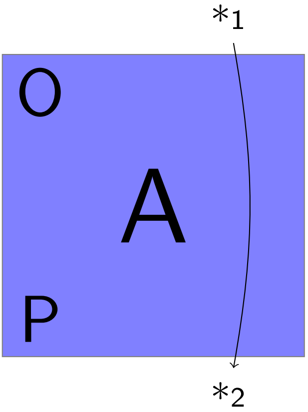


Come rappresentiamo i giochi (il tavolo insomma)

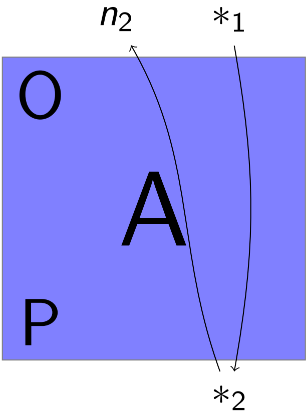
$*_1$



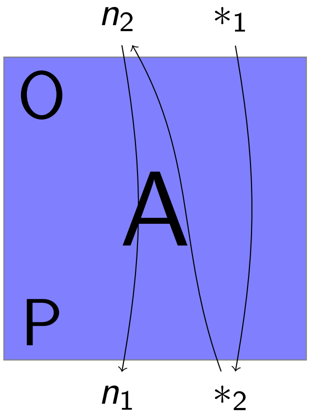
Come rappresentiamo i giochi (il tavolo insomma)



Come rappresentiamo i giochi (il tavolo insomma)



Come rappresentiamo i giochi (il tavolo insomma)

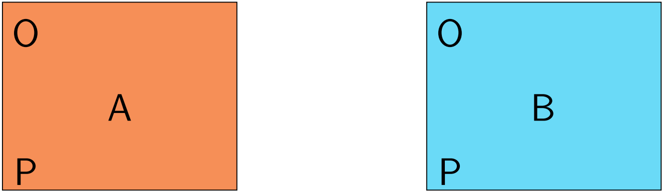


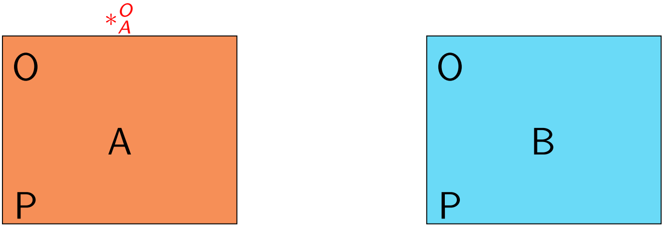
Il gioco $A \otimes B$

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

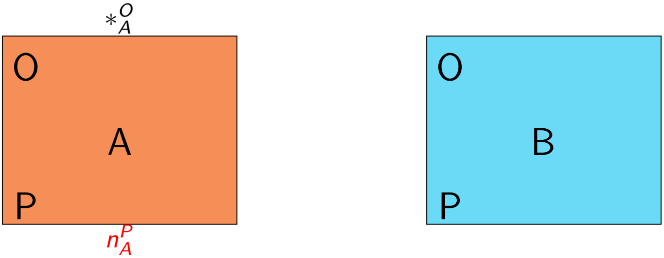
Proprietà

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

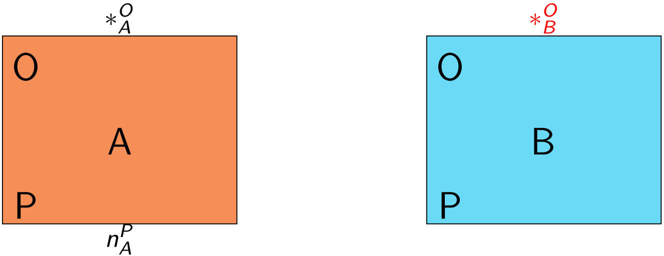

O
A
P
O
B
P



$$*_A^O$$

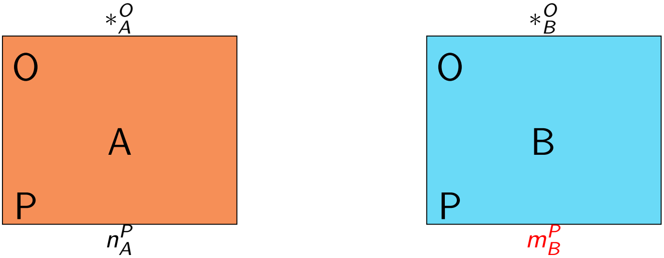


$$*^O_A \quad n^P_A$$



$$*^O_A \quad n^P_A \quad *^O_B$$

giochi e strategie



$$*_A^O \quad n_A^P \quad *_B^O \quad m_B^P$$

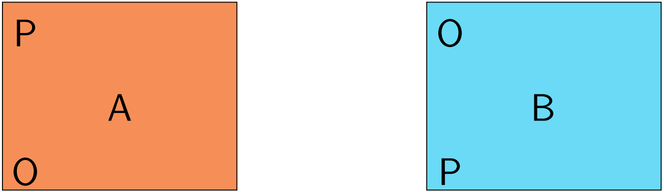
Il gioco $A \multimap B$

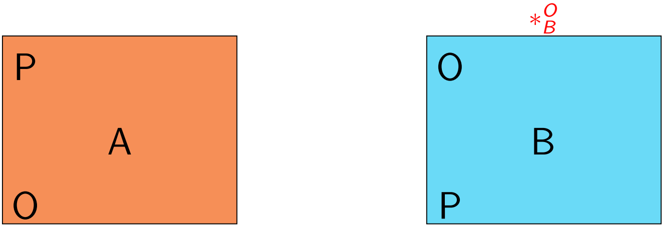
- $M_{A \multimap B} = M_A \coprod M_B$
- $\lambda^{QA}_{A \multimap B} = \lambda^{QA}_A \coprod \lambda^{QA}_B$

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

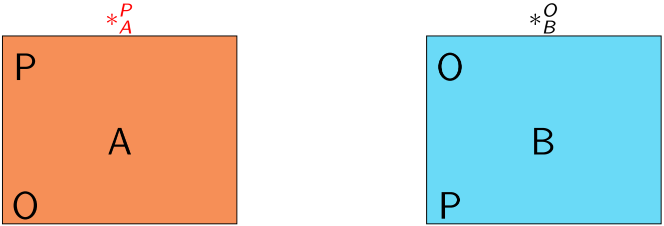
- Solamente il giocatore P può cambiare componente di gioco


P
A
O
O
B
P

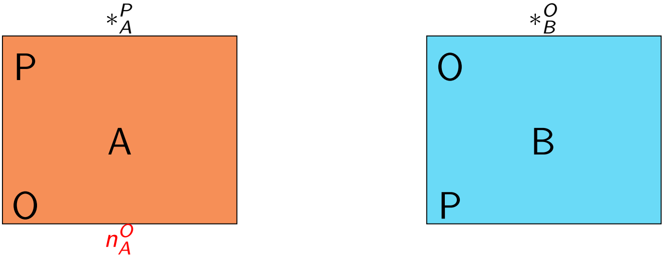


$$*^{O}_{B}$$

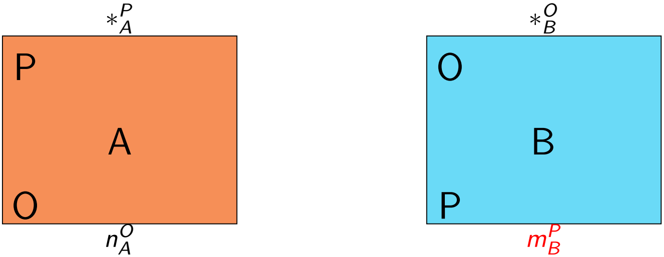
giochi e strategie



$$*_B^O \quad *_A^P$$



$$*^O_B \quad *^P_A \quad n^O_A$$



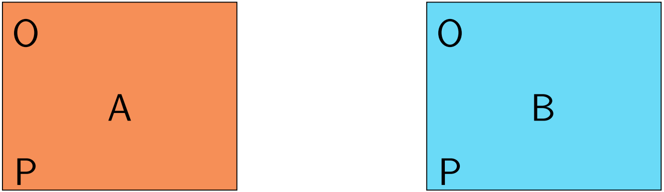
$$*_B^O \quad *_A^P \quad n_A^O \quad m_B^P$$

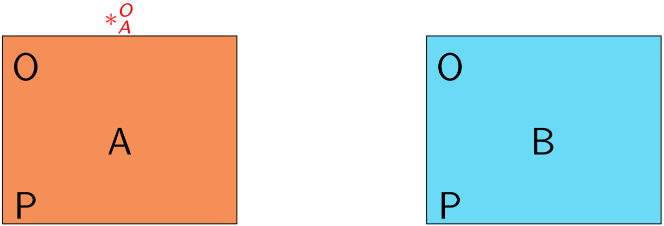
Il gioco $A\&B$

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

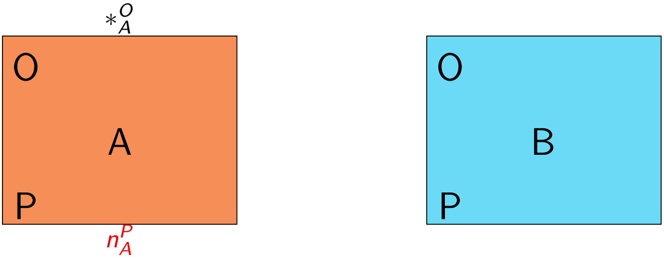
Proprietà

- 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


O
A
P
O
B
P



$$*^O_A$$



$$*_A^O \quad n_A^P$$

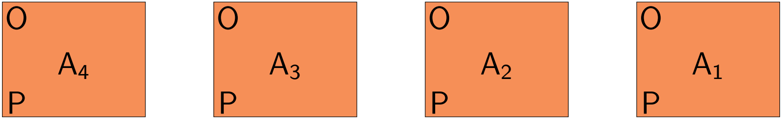
Il gioco $!A$

- $M_{!A} = \omega \times M_A$
- $\lambda_{!A}(i, a) = \lambda_A(a)$
- 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 $A \otimes A \otimes A \otimes A \otimes \ldots$

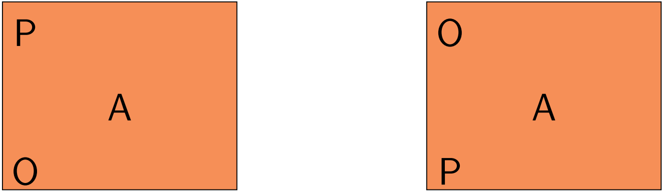

O
A_4
P
O
A_3
P
O
A_2
P
O
A_1
P

Alcune strategie

DOBBIAMO DECIDERE COME SPIEGARE LE STRATEGIE; QUESTE ANDREBBERO DETTE:

- $\sigma ; \tau$ [NO]
- $id_{A \multimap A}$ (la copy-cat) [SI]
- $App : (A \Rightarrow B) \& A \Rightarrow B$ [SI]

La strategia copycat



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 $\{\sigma \text{ strategia di } A \multimap B | \sigma \approx \sigma\}/ \approx$
- Date $[\sigma] : A \to B$ e $[\tau] : B \to C$, $[\tau] \circ [\sigma] = [\sigma; \tau]$

In particolare $\mathcal{G}$ è dotata di:

- un oggetto finale (1)
- è una categoria monoidale (è definito $\otimes$ bifuntore associativo e con elemento neutro)
- è una categoria autonoma (per ogni gioco A esiste il suo gioco duale $1 \multimap A$)
- NON è una categoria cartesiana chiusa (mancano i prodotti)

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'oggetto esponente $(A \Rightarrow B$ definito come $!A \multimap B$; cioé $Mor(A \& B, C) \cong Mor(A, B \Rightarrow C))$

SI PUÒ TAGLIARE UN PO' QUESTA? (FORSE!)

order enrichement

Definiamo un *pointed poset* 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 *lub* f^{∇}
  - Dati $g : C \to A$ e $h : B \to D$, $g \circ f^{\nabla} \circ h = \bigsqcup_{k \in \omega} g \circ f^{(k)} \circ h$

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

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

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

DOBBIAMO METTERE UN PAIO DI INTERPRETAZIONI
(Exodd) un paio? Sta tutto qui il difficile!

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 *cpo-based model* 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

DA SCRIVERE MEGLIO

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

(Exodd) Dov'è finita la proprietà di universalità?