

# Un modello fully abstract del PCF

Grilletti Gianluca    Barbarino Giovanni

Università di Pisa

May 16, 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,  $S \rightarrow T$  è un tipo

## Example

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

## Grammatica per generare i termini di PCF

$$\langle \text{nat\_exp} \rangle ::= 0 | 1 | 2 | \dots | \langle \text{nat\_exp} \rangle + \langle \text{nat\_exp} \rangle$$

$$\langle \text{bool\_exp} \rangle ::= \text{true} | \text{false} | \text{Eq? } \langle \text{nat\_exp} \rangle \langle \text{nat\_exp} \rangle$$

$$\langle \sigma \rightarrow \tau\_exp \rangle ::= \lambda(x : \sigma). \langle \tau\_exp \rangle$$

$$\langle \sigma \times \tau\_exp \rangle ::= \langle \langle \sigma\_exp \rangle, \langle \tau\_exp \rangle \rangle$$

$$\langle \sigma\_exp \rangle ::= \langle \sigma\_var \rangle |$$

$$\text{if } \langle \text{bool\_exp} \rangle \text{ then } \langle \sigma\_exp \rangle \text{ else } \langle \sigma\_exp \rangle |$$

$$\langle \sigma\_application \rangle | \langle \sigma\_projection \rangle | \langle \sigma\_fixed\_point \rangle$$

$$\langle \sigma\_application \rangle ::= \langle \tau \rightarrow \sigma\_exp \rangle \langle \tau\_exp \rangle$$

$$\langle \sigma\_projection \rangle ::= \text{Proj}_1 \langle \sigma \times \tau\_exp \rangle | \text{Proj}_2 \langle \tau \times \sigma\_exp \rangle$$

$$\langle \sigma\_fixed\_point \rangle ::= Y_\sigma \langle \sigma \rightarrow \sigma\_exp \rangle$$

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

### Example

- $(\underline{n} + \underline{m}) + \underline{n} : \text{Nat}$
- $\text{Eq?}(\underline{n})(\underline{m}) : \text{Bool}$
- $\langle \text{true}, \underline{n} \rangle : \text{Bool} \times \text{Nat}$
- $\text{Proj}_1 \langle \text{true}, \underline{n} \rangle : \text{Bool}$
- $\lambda(x : \text{Nat}).x + 1 : \text{Nat} \rightarrow \text{Nat}$  (indichiamolo con  $\text{Succ}$ )
- $\text{Succ}(\underline{n}) : \text{Nat}$
- $\text{if}[\text{Eq?}(\underline{n})(\underline{m})] \text{ then}[\underline{n}] \text{ else}[\text{Succ}]$  non è ben formato
- $\text{if}[\text{Eq?}(\underline{n})(\underline{m})] \text{ then}[\underline{n}] \text{ else}[\text{Succ}(\underline{n})] : \text{Nat}$
- $\lambda(x : \text{Nat}).\text{if}[\text{Eq?}(\underline{0})(x)] \text{ then}[\text{true}] \text{ else}[\text{false}] : \text{Nat} \rightarrow \text{Bool}$   
(Indichiamolo con  $\text{IsZero}$ )
- $Y[\text{Succ}] : \text{Nat}$
- $Y[\text{IsZero}]$  non è ben formato

# Programmi

Un programma di PCF è un termine:

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

## Example

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

# Semantica operativa

Diamo le seguenti regole di riduzione:

$$\text{add} \quad \underline{n} + \underline{m} \rightarrow \underline{n + m}$$

$$\text{Eq?} \quad \text{Eq?}(\underline{n})(\underline{n}) \rightarrow \text{true}$$

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

$$\text{cond} \quad \text{if}[\text{true}] \text{ then}[M] \text{ else}[N] \rightarrow M$$

$$\text{if}[\text{false}] \text{ then}[M] \text{ else}[N] \rightarrow N$$

$$\text{proj} \quad \text{Proj}_1 < M, N > \rightarrow M$$

$$\text{Proj}_2 < M, N > \rightarrow N$$

$$\alpha \quad \lambda(x : \sigma).M \rightarrow \lambda(y : \sigma).[y/x]M \text{ (con } y \text{ non libera in } M)$$

$$\beta \quad [\lambda(x : \sigma).M](N) \rightarrow [N/x]M$$

$$Y \quad Y_\sigma \rightarrow \lambda(f : \sigma \rightarrow \sigma).f(Y_\sigma f)$$

- 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 operativa 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

$Eval(M) = \text{undef}$

# Equivalenza osservazionale

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

## Example

$$C[] \equiv \lambda(x : \text{Nat}).x + []$$

Porre il termine  $\underline{n}$  nel contesto  $C[]$  significa considerare il termine

$$C[\underline{n}] \equiv \lambda(x : \text{Nat}).x + \underline{n}$$

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



# Espressività di PCF

Diciamo che una funzione parziale  $f : \mathbb{N} \rightarrow \mathbb{N}$  è *calcolabile* se esiste un programma per computer<sup>1</sup>  $P$  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

---

<sup>1</sup>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  $Eval(t(\underline{n})) = \text{undef}$

## 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  $N$ :

$$M \stackrel{\text{obs}}{=} N \Leftrightarrow \llbracket M \rrbracket = \llbracket N \rrbracket$$

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
- $\lambda_A$  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  $\sigma$  è un insieme di partite di lunghezza pari (l'ultima mossa è di  $P$ ) tali che:

- $\sigma$  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

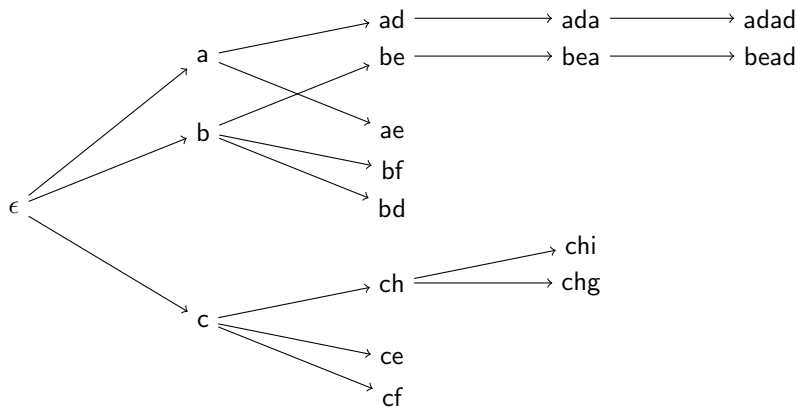
Prendiamo un gioco, il cui set di mosse  $M$  è suddiviso dalla funzione di labelling  $\lambda$  in

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

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*

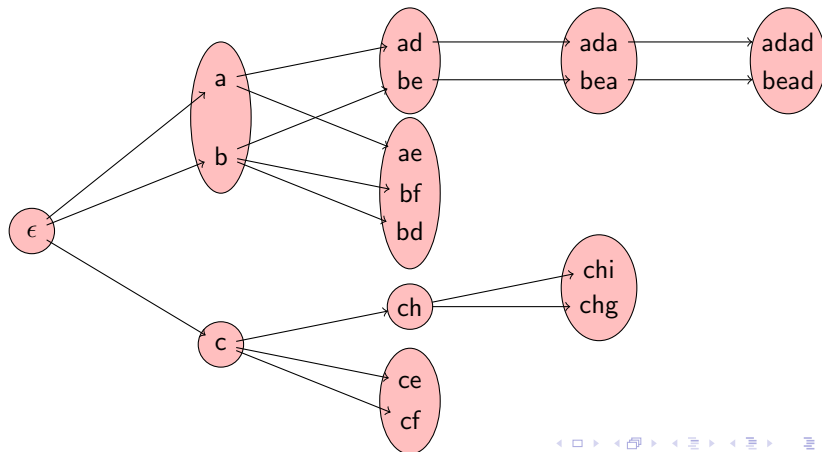
$$M_{QO} = \{a, b, c\}, \quad M_{AO} = \{g, i\}$$

$$M_{QP} = \{h\}, \quad M_{AP} = \{d, e, f\}$$



$s \approx t$  se:

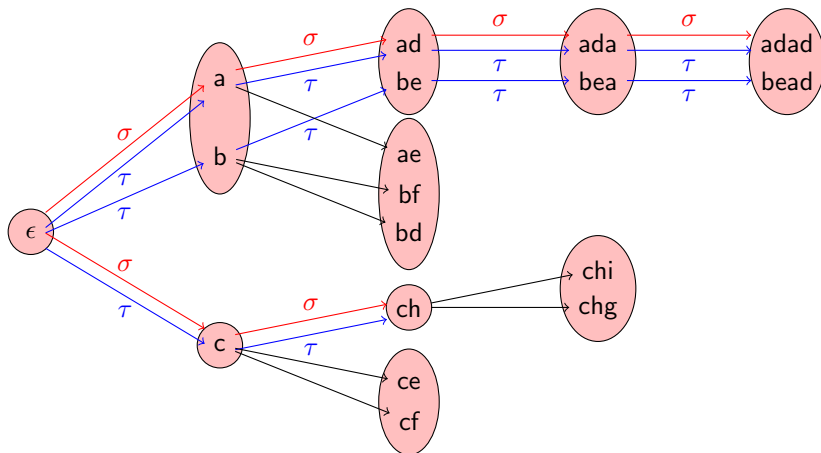
- $s$  e  $t$  hanno la stessa etichettatura
- per ogni  $s'$  e  $t'$  sottostringhe iniziali di  $s$  e  $t$  tali che  $|s'| = |t'|$ , si ha  $s' \approx t'$
- se  $sa$  è una partita valida, allora esiste  $b$  tale che  $tb \approx sa$





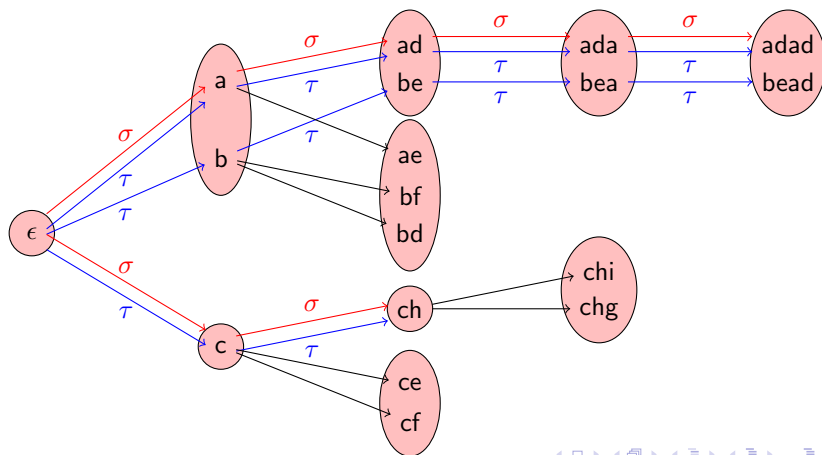
Una strategia  $\sigma$  è un insieme di partite di lunghezza pari tali che:

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



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

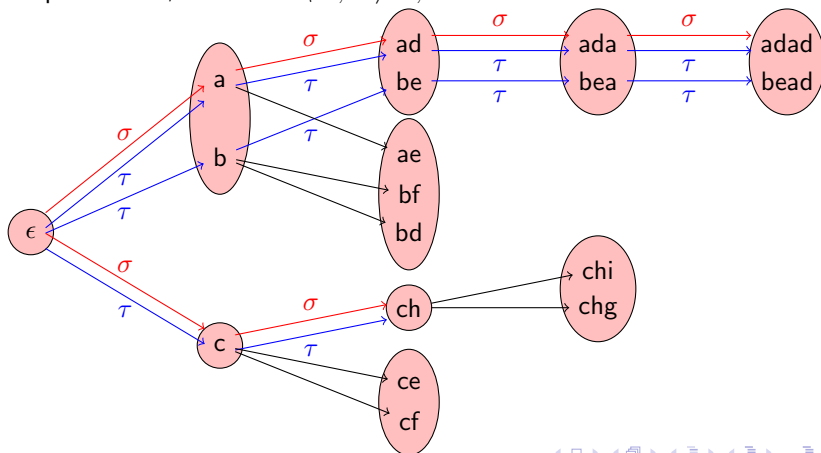
$$f_{\tau}(x) = \begin{cases} d & \text{se } x = a \\ e & \text{se } x = b \\ h & \text{se } x = c \\ d & \text{se } x = g \end{cases}$$



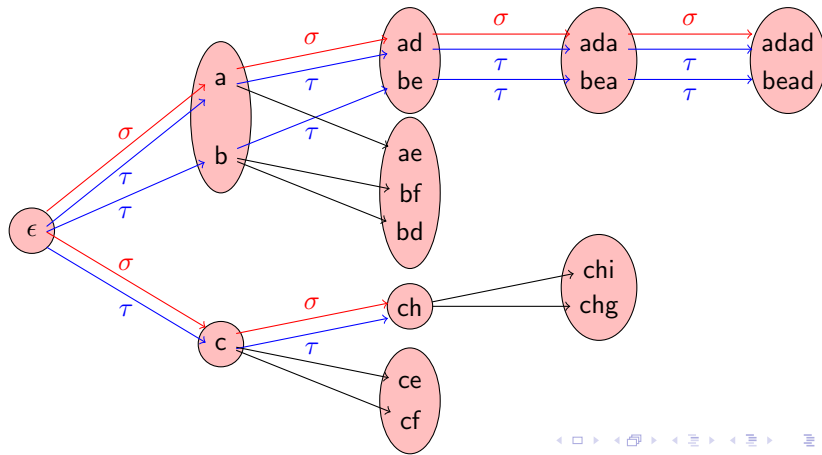
Estendiamo la relazione  $\approx$  alle strategie; poniamo:

- $\sigma \preceq \tau$  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  $sab \approx s'a'b'$
- $\sigma \approx \tau$ ; iif  $\sigma \preceq \tau \wedge \tau \preceq \sigma$

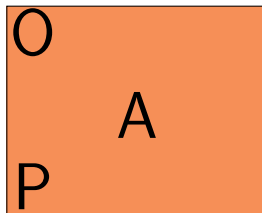
In questo caso, avremo  $\sigma \preceq \tau, \sigma \not\approx \tau, \tau \approx \tau$



- $\preceq$  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)

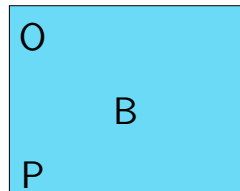
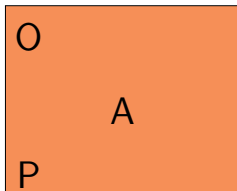


# Il gioco $A \otimes B$

- $M_{A \otimes B} = M_A \amalg M_B$
- $\lambda_{A \otimes B} = \lambda_A \amalg \lambda_B$
- $P_{A \otimes B}$  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  $B$
- $s \approx_{A \otimes B} t \Leftrightarrow s|_A \approx_A t|_A \wedge s|_B \approx_B t|_B \wedge fst(s) = fst(t)$

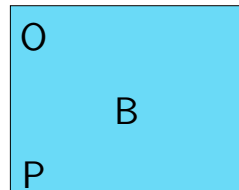
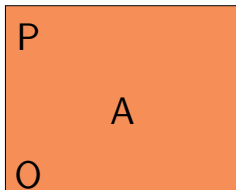
## Proprietà

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







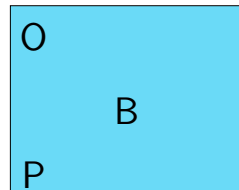
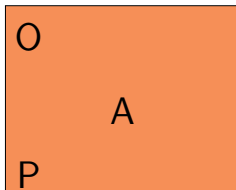


# Il gioco $A \& B$

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

## 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  $B$



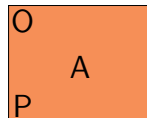
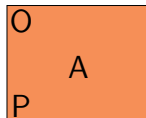
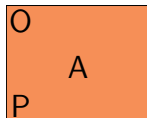
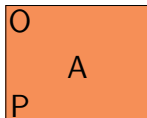
# Il gioco !A

- $M_{!A} = \omega \times M_A$
- $\lambda_{!A}(i, a) = \lambda_A(a)$
- $s$  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  $\pi : \omega \rightarrow \omega$  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  $A \otimes A \otimes A \otimes A \otimes \dots$

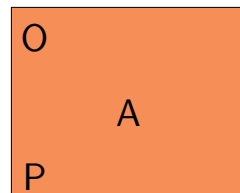
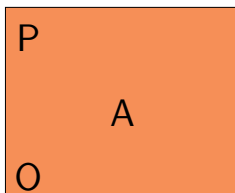


# Alcune strategie

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

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  $\sigma$  e  $\tau$ ,  $\tau \circ \sigma = \sigma \circ \tau = \sigma^\dagger; \tau$
- 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  $!A \multimap B$ ; cioè  $Mor(A \& B, C) \cong Mor(A, B \Rightarrow C)$ )

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

# order enrichment

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

Definiamo una categoria cartesiana chiusa  $\mathcal{C}$  *pointed poset enriched* se:

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

Definiamo una categoria cartesiana chiusa  $\mathcal{C}$  *razionale* se:

- è ppo-enriched
- per ogni  $f : A \times B \rightarrow B$  si ha:
  - La catena  $(f^{(k)} | k \in \omega)$  definita da  $f^{(0)} = \perp_{A,B}$  e  $f^{k+1} = f \circ \langle \text{id}_A, f^{(k)} \rangle$  ammette  $\text{lub } f^\nabla$
  - Dati  $g : C \rightarrow A$  e  $h : B \rightarrow D$ ,  $g \circ f^\nabla \circ h = \bigsqcup_{k \in \omega} g \circ f^{(k)} \circ h$

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

$$[\sigma] \sqsubseteq_A [\tau] \Leftrightarrow \sigma \preceq_A \tau$$

### Teorema

$K_1(\mathcal{G})$  con l'ordine  $\sqsubseteq$  è razionale

### Teorema

Sia  $\mathcal{C}$  una categoria cartesiana chiusa razionale. Si ha che:

- Fissata la denotazione dei tipi base di PCF in  $\mathcal{C}$  (ogni tipo viene denotato con un oggetto)
- Fissata la denotazione delle costanti di PCF in  $\mathcal{C}$  (ogni termine di tipo  $\tau$  viene denotato con un morfismo di  $1 \rightarrow \llbracket \tau \rrbracket$ )

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  $\tau$  di PCF, posto  $T = \llbracket \tau \rrbracket$ , si ha che  $1 \rightarrow T$  è un dl-domain; in particolare  $\mathcal{M}(K_I(\mathcal{G}))$  è un *cpo-based model* algebrico

## Teorema

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

# Full abstraction

Definiamo il gioco di *Sierpinsky*  $\Sigma$  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  $A$ :

$$x \lesssim_A y \Leftrightarrow \forall \alpha \rightarrow \Sigma.x; \alpha \preceq_\Sigma y; \alpha$$

Definiamo  $\mathcal{E} = K_1(\mathcal{G}) / \lesssim$ , cioè la categoria tale che:

- $\mathcal{E}_0 = K_1(\mathcal{G})_0$
- $Mor_{\mathcal{E}}(A, B) = Mor_{K_1(\mathcal{G})}(A, B) / \lesssim_{A \Rightarrow B}$

## Teorema

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

DA SCRIVERE MEGLIO

# Universalità

Definiamo un gioco  $A$  *effettivamente dato* se:

- Esiste una mappa  $e_A : \omega \rightarrow M_A$  suriettiva; chiamiamo questa funzione codifica
- Rispetto alla codifica le seguenti funzioni sono calcolabili:
  - $\lambda_A$  (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  $K_I(\mathcal{G}_{rec})$  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?