Formalismi di calcolo e introduzione al λ-calcolo

Claudio Sacerdoti Coen

<sacerdot@cs.unibo.it>

Universitá di Bologna

2023-2024

Formalismi di calcolo

Un formalismo di calcolo è un formalismo che risponde alla domanda “cosa vuol dire calcolare”

Un formalismo è una descrizione matematicamente rigorosa di un fenomeno, in genere ottenuta tramite manipolazione di espressioni simboliche.

Tantissmi formalismi di calcolo: λ-calcolo, macchine di Turing, sistemi di Post, funzioni primitive ricorsive con operatore di minimizzazione, Random Access Machines, linguaggi di programmazione rigorosamente specificati, ...

La tesi di Church-Turing

Tesi (nel senso di congettura!) di Church-Turing: Ogni funzione calcolabile da un formalismo di calcolo sufficientemente espressivo è calcolabile da una macchina di Turing e viceversa.

Tutti i formalismi sono equivalenti dal punto di vista di cosa calcolano.

Formalismi diversi hanno punti di forza/debolezza diversi (come i linguaggi di programmazione).

- calcolare = modificare un supporto fisico di celle discrete, ognuna contentente una quantità finita di informazione
- calcolo ottenuto tramite operazioni locali
  (una testina r/w si muove sul supporto fisico)
- oltre al supporto fisico, la macchina è in uno stato scelto da un insieme finito

λ-calcolo (Church)

- calcolare = semplificare espressioni
- espressioni: tutto è una funzione
  in particolare: le funzioni prendono in input funzioni e danno in output funzioni

Macchine di Turing vs λ-calcolo

Macchina di Turing	**λ-calcolo**
Ogni passo $O(1)$ (tempo)	Implementazone naif di un passo: $O(n^2)$
Ogni cella $O(1)$ (spazio)	Implementazione efficiente: $O(???)$
Ottimo per studio complessità	Pessimo per studio complessità

Macchine di Turing vs λ-calcolo

Macchina di Turing	**λ-calcolo**
Imperativo, ma $\neq$ linguaggi imperativi	Cuore di tutti i linguaggi funzionali
Non composizionale	Composizionale
Di basso livello	Di alto livello
Difficile implementare costrutti/dati/meccanismi	Facile implementare costrutti/dati/meccanismi
Definizione non ricorsiva $\Rightarrow$ prove complesse	Definizioni ricorsive $\Rightarrow$ prove per induzione
Pessimo per studio linguaggi di programmazione	Ottimo per studio linguaggi di programmazione

Macchina di Turing	**λ-calcolo**
Ad-hoc	Controparte computazionale della logica

Una macchina di Turing è definita da una tupla (A, Q, q_0, q_f, δ) dove:

- L'alfabeto A è un insieme non vuoto, finito di simboli
- L'insieme di stati Q è un insieme non vuoto, finito di stati
- $q_0 \in Q$ è lo stato iniziale
- $q_f \in Q$ è lo stato finale
- La funzione di transizione δ ha dominio $A \times Q$ e codominio $A \times Q \times \{L, R\}$

Lo stato di una macchina di Turing (A, Q, q_0, q_f, δ) è una tripla (α, i, q) dove:

- Il nastro infinito α è una funzione da $\mathbb{Z}$ a A
  Intuizione: $\alpha(k) = a$ sse la k-esima cella del nastro contiene il simbolo a
- $i \in \mathbb{Z}$ è la posizione della testina sul nastro
  Intuizione: la testina è posizionata sulla i-esima cella di contenuto $\alpha(i)$
- $q \in Q$ è lo stato corrente

Uno stato (α, i, q) è iniziale di input α sse $i = 0$ e $q = q_0$ e finale di output α sse $q = q_f$.

Esecuzione di una macchina di Turing

Una macchina di Turing (A, Q, q_0, q_f, δ) in uno stato (α, i, q) non finale transisce in un nuovo stato (α', i', q') se:

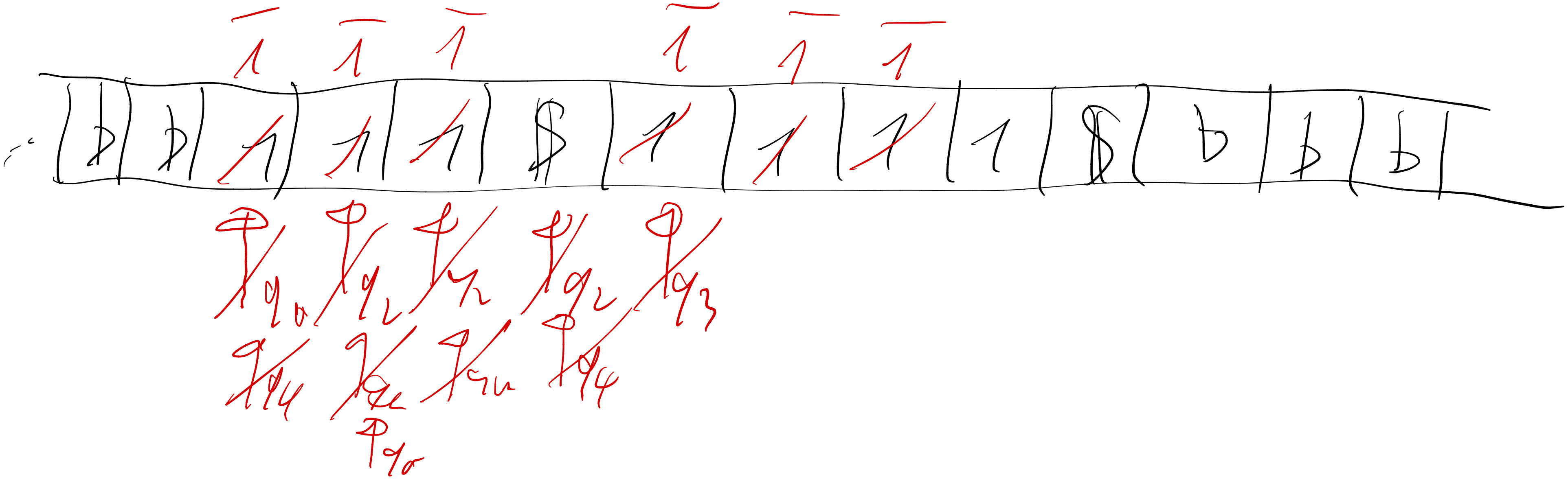
- $\delta(\alpha(i), q) = a, q', x$
  Intuizione: la testina legge il contenuto $\alpha(i)$ della cella corrente i, lo stato corrente si aggiorna da q a q' e ...
- $\alpha'(i) = a$ e $\alpha'(n) = \alpha(n)$ per $n \neq i$
  ...la testina sovrascrive il valore della cella con a e ...
- $i' = i + 1$ se $x = R$; $i' = i - 1$ se $x = L$
  ...si muove a destra o a sinistra a seconda del valore di x

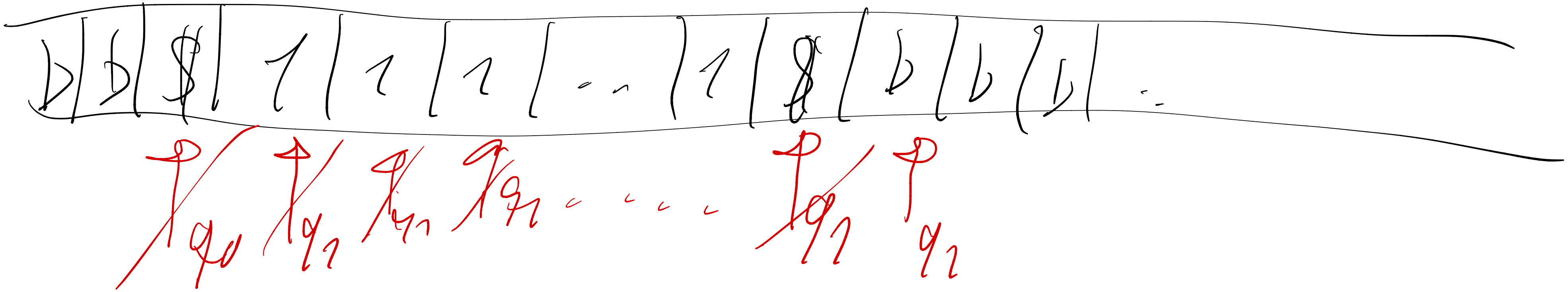
Programmazione di una macchina di Turing

Esercizio: confronto di due numeri espressi in base 1
(ovvero il numero *n* è rappresentato da *n* uni di seguito)

Definire una macchina di Turing che prenda in input α e dia in output β dove:

- $A = \{b, 1, 0, \$\}$
- $\alpha = \ldots bb11\ldots1\$11\ldots1\$bb\ldots$
  t.c. 0 (la posizione iniziale della testina) corrisponda all'1 più a sinistra (o al dollaro più a sinistra se non ci sono uni a sinistra del dollaro)
- $\beta = \ldots bb11\ldots1\$11\ldots1\$Xbb\ldots$
  dove *X* sia 1 se il numero di uni nella prima sequenza è minore o uguale del numero di uni nella seconda e 0 altrimenti





δ $A \setminus Q$	q_0	q_1	q_2	q_3	q_4
\$	\$, q_1, R	\$, q_1, R	\$, q_3, R		\$, q_4, L
1	$\bar{1}$, q_2, R	1, q_1, R	1, q_2, R	$\bar{1}$, q_4, L	1, q_4, L
b		1, q_4, R			
$\bar{1}$		1, q_1, R	$\bar{1}$, q_2, R		$\bar{1}$, q_0, R

Il λ-calcolo: sintassi

Tutto è una funzione unaria (= con un solo input) anonima:

$$t ::= x \mid tt \mid \lambda x.t$$

dove:

- t viene chiamato termine
- useremo $t, s, u, v, M, N, \ldots$ per indicare un termine
- x (ma anche $y, z, w, \ldots$) indica l'occorrenza di una variabile
- $t_1 t_2$ (chiamata applicazione) è la chiamata di funzione: passo t_2 in input alla funzione t_1 notazione matematica standard: $t_1(t_2)$
- $\lambda x.t$ (chiamata astrazione) è una funzione anonima il cui parametro formale è x e il cui corpo è t notazione matematica standard: $x \mapsto t$ o anche $f(x) = t$ se la funzione avesse un nome f

Il λ-calcolo: esempi

- $\lambda x.x$ è la funzione identità: prende in input x e lo restituisce in output
- $(\lambda x.x)(\lambda y.y)$ applica la funzione identità a un'altra copia della funzione identità. Il risultato atteso è la funzione identità $\lambda y.y$
- $\lambda x.y$ è la funzione costante che ignora l'input x e restituisce sempre y
- $(\lambda x.y)(\lambda z.z)$ applica la funzione costante di prima alla funzione identità. Il risultato atteso è y
- $\lambda x.xx$ prende in input una funzione x e la applica a se stessa
- $\lambda x.\lambda y.xy$ prende in input una funzione x e restituisce una funzione che prende in input una y e applica x a y
- $(\lambda x.\lambda y.xy)(\lambda z.z)$ ridurrà a $\lambda y.(\lambda z.z)y$ che ridurrà alla funzione identità $\lambda y.y$

Il λ-calcolo: sintassi

$$t ::= x \mid tt \mid \lambda x.t$$

Regole di precedenza e associatività:

- l'applicazione ha la precedenza sull'astrazione:
  $\lambda x.xx$ si legge come $\lambda x.(xx)$ e non come $(\lambda x.x)x$
- l'applicazione è associativa a sinistra:
  xyz si legge come $(xy)z$ e non come $x(yz)$

Il λ-calcolo: funzioni n-arie e applicazione parziale

$$t ::= x \mid tt \mid \lambda x.t$$

Una funzione binaria $f(x, y) = g(x, y)$ può essere vista come una funzione unaria che restituisce una funzione unaria: $\lambda x.\lambda y.gxy$

Notate che il passaggio simultaneo di una coppia di input $g(x, y)$ viene codificato con il passaggio sequenziale di un input alla volta: gxy, ovvero $(gx)y$.

Vantaggio: le funzioni possono essere applicate parzialmente passando solamente il primo parametro ($(\lambda x.\lambda y.x + y)2$ riduce alla funzione $\lambda y.2 + y$ che incrementa un numero di 2.

Il λ-calcolo: riduzione (intuizione)

Un λ-termine t può ridurre a un altro λ-termine t' rimpiazzando una chiamata di funzione $(\lambda x.M)N$ con il corpo M dove sostituisco x con N.

Esempio: $(\lambda x.yx)(zz)$ riduce a $y(zz)$.

Tuttavia devo fare attenzione a definire correttamente la nozione di sostituzione.

Variabili libere e legate

$$t ::= x \mid tt \mid \lambda x.t$$

I nomi dati ai parametri formali non sono importanti: $\lambda x.x$ e $\lambda y.y$ sono la stessa funzione.

Tuttavia i nomi delle variabili globali lo sono eccome: $\lambda x.y$ e $\lambda x.z$ sono due programmi diversi.
Intuizione: non posso rimpiazzare uno con l'altro in un contesto dove $y = 0$ e $z = 1$ senza ottenere risultati diversi.

Introduciamo una terminologia.

Variabili libere e legate

$$t ::= x \mid tt \mid \lambda x.t$$

Il λ nell'astrazione $\lambda x.t$ è un binder: esso lega la variabile x nel corpo t.

Una variabile che non è legata si dice *libera*.

Nel λ-calcolo: le variabili legate sono tutte parametri formali (c'è un solo binder) e quelle libere sono tutte variabili globali.

$$t ::= x \mid tt \mid \lambda x.t$$

L'insieme $FV(t)$ delle variabili libere di t si calcola come segue:

- $FV(x) = \{x\}$
- $FV(MN) = FV(M) \cup FV(N)$
- $FV(\lambda x.M) = FV(M) \setminus \{x\}$

Esempio: $FV(\lambda x.xy(\lambda y.yz)) = \{y, z\}$.

Nota: nell'esempio la prima occorrenza di y è libera, mentre la seconda è legata.

$FV(\lambda x.xy(\lambda y.yz)) = FV(xy(\lambda y.yz)) \setminus \{x\} =$
$= (FV(xy) \cup FV(\lambda y.yz)) \setminus \{x\} = (FV(x) \cup FV(y) \cup (FV(yz) \setminus \{y\})) \setminus \{x\}$

$$= \left(\{x\} \cup \{y\} \cup \left(\{y\} \cup \{z\} \right) \setminus \{y\} \right) \setminus \{x\}$$

$$= \left(\{x, y\} \cup \{z\} \right) \setminus \{x\} = \{x, y, z\} \setminus \{x\} = \{y, z\}$$

α-conversione $\equiv_\alpha$

Due λ-termini t_1 e t_2 sono α-convertibili (i.e. $t_1 \equiv_\alpha t_2$) se posso ottenere l'uno dall'altro ridenominando le sole variabili legate in modo tale che le occorrenze legate di una variabile in posizione corrispondente nei due termini siano legate dai binder in posizione corrispondente e che le occorrenze di variabili libere abbiano in posizione corrispondente una variabile libera con lo stesso nome.

$\lambda x.\lambda y.xyz \equiv_\alpha \lambda y.\lambda w.ywz$

$\not\equiv_\alpha$ $\not\equiv_\alpha$

$\lambda x.\lambda z.xzz$ $\lambda x.\lambda y.yxz$

Esempi:

$\lambda x.\lambda y.xyz \equiv_\alpha \lambda y.\lambda w.ywz$

$\lambda x.\lambda y.xyz \not\equiv_\alpha \lambda x.\lambda z.xzz$

$\lambda x.\lambda y.xyz \not\equiv_\alpha \lambda x.\lambda y.yxz$

$\lambda x.\lambda y.xyz \not\equiv_\alpha \lambda x.\lambda y.xyw$

Da questo momento in avanti considereremo (quasi) sempre i termini α-equivalenti come uguali.

Più formalmente: l'α-equivalenza è una relazione di equivalenza (simmetrica, riflessiva e transitiva) e noi lavoreremo con le classi di equivalenza di λ-termini modulo l'α-equivalenza.

Sostituzione

Quando si sostituisce in M un termine N al posto di una variabile x (scrittura: $M\{N/x\}$) bisogna stare molto attenti a evitare catture accidentali delle variabili libere.

$(\lambda x.xy)\{xx/y\}$

Esempio:

$(\lambda x.xy)\{zz/y\} = \lambda x.x(zz)$ ma

$(\lambda x.xy)\{xx/y\} \neq \lambda x.x(xx)$

$\lambda x.x(xx)$

(a sx le x in xx sono globali, a dx sono parametri formali)

Soluzione via α-conversione:

$(\lambda x.xy)\{xx/y\} \equiv_\alpha (\lambda z.zy)\{xx/y\} = \lambda z.z(xx)$

Sostituzione

Formalmente: $M\{N/x\}$ (M dove sostituisco alle occorrenze libere di x il termine N) è definito come

- $x\{N/x\} = N$
- $y\{N/x\} = y$
- $(t_1 t_2)\{N/x\} = t_1\{N/x\}t_2\{N/x\}$
- $(\lambda x.M)\{N/x\} = \lambda x.M$
- $(\lambda y.M)\{N/x\} = \lambda z.M\{z/y\}\{N/x\}$ per $z \notin FV(M) \cup FV(N)$

Terminologia: z è sufficientemente fresca se $z \notin FV(M) \cup FV(N)$ e fresca se non è mai stata utilizzata prima. Ogni variabile fresca è automaticamente sufficientemente fresca.

β-riduzione

t_1 β-riduce a t_2 in un passo (indicato $t_1 \rightarrow_\beta t_2$) sse ottengo t_2 da t_1 rimpiazzando da qualche parte in t_1 il redex $(\lambda x.M)N$ con il suo ridotto $M\{N/x\}$.

Esempio: $\lambda x.(\lambda y.xy)x \rightarrow_\beta \lambda x.xx$ dove è stato ridotto il redex $(\lambda y.xy)x$.

β-riduzione

Formalmente definiamo la relazione binaria $\rightarrow_\beta$ tramite un sistema di inferenza (cfr. deduzione naturale).

$$\frac{}{(\lambda x.M)N \rightarrow_\beta M\{N/x\}}$$

$$\frac{M \rightarrow_\beta M'}{MN \rightarrow_\beta M'N}$$

$$\frac{M \rightarrow_\beta M'}{NM \rightarrow_\beta NM'}$$

$$\frac{M \rightarrow_\beta M'}{\lambda x.M \rightarrow_\beta \lambda x.M'}$$

Osservazione: l'ultima regola dice che posso ridurre il corpo di una funzione prima che questa sia invocata! (come quando si ottimizza il codice)

β-riduzione: esempio

$$
\cfrac{\cfrac{\overline{(\lambda x.yx)y \to_\beta yy}}{y((\lambda x.yx)y) \to_\beta y(yy)}}{\lambda y.y((\lambda x.yx)y) \to_\beta \lambda y.y(yy)}
$$

β-riduzione

$t_1 \to_\beta^n t_{n+1}$ (t_1 riduce in n passi a t_{n+1}) sse
$t_1 \to_\beta t_2 \to_\beta \ldots \to_\beta t_{n+1}$

Formalmente:

- $t \to_\beta^0 t$
- $t \to_\beta^{n+1} t''$ sse $t \to_\beta t'$ e $t' \to_\beta^n t''$.

$t \to_\beta^* t'$ (t riduce in 0 o più passi a t') sse $\exists n.t \to_\beta^n t'$

Esempio: $(\lambda x.\lambda y.xy)(\lambda z.z)(\lambda z.z) \to_\beta^3 \lambda z.z$.

$\to_\beta (\lambda y.\overbrace{(\lambda z.z)y})(\lambda z.z) \to_\beta (\lambda y.y)(\lambda z.z) \to_\beta$

Forme normali

t è una forma normale (anche scritto $t \not\to_\beta$) sse $\nexists t'.t \to_\beta t'$

t ha forma normale t' sse $t \to_\beta^* t' \wedge t' \not\to_\beta$

t ha una forma normale o può convergere sse esiste un t' t.c. t ha forma normale t'

Non determinismo

La relazione $\rightarrow_\beta$ è non deterministica, ovvero ci sono dei t t.c. esistono t_1, t_2 distinti t.c. $t_1 \, {}_\beta\!\leftarrow t \rightarrow_\beta t_2$

Esempio: $y \, {}_\beta\!\leftarrow (\lambda x.y)((\lambda z.z)w) \rightarrow_\beta (\lambda x.y)w$

Intuizione: non viene specificato in quale ordine vanno ridotti i redessi e un'implementazione può scegliere liberamente. Questo potenzialmente può portare a forme normali distinte, ma vedremo che non sarà così. (Continando l'esempio: $(\lambda x.y)w \rightarrow_\beta y$).

Quanto è espressivo il λ-calcolo?

Un linguaggio Turing completo sembra necessitare di:

1. tipi di dato
   almeno i numeri naturali con cui codificare il resto
2. scelta (if-then-else)
   a input diversi deve restituire output diversi
3. ripetizione (while, ricorsione)
   un codice di dimensione prestabilita deve poter analizzare input di dimensione arbitrariamente grande

Il λ-calcolo sembra non avere nessuno di questi meccanismi!

Somma dei numeri pari $\leq n$

Il numero 3 viene rappresentato come S (S (S 0))
(S = successore).

let rec f n =
  match n **with**
  | 0 $\Rightarrow$ 0
  | S m $\Rightarrow$ **if** even(S m) **then** S m + f m **else** f m

Somma dei numeri pari $\leq n$

Esempio:

f (S (S 0))
$\rightarrow$ **match** S (S 0)) **with** | 0 $\Rightarrow$ 0 | S m $\Rightarrow$ **if** even(S m) **then** S m + f m **else** f m
$\rightarrow$ **if** even(S (S 0)) **then** S (S 0) + f (S 0) **else** f (S 0)
$\rightarrow$ S (S 0) + f (S 0)
$\rightarrow$ S (S 0) + **match** S 0 **with** | 0 $\Rightarrow$ 0 | S m $\Rightarrow$ **if** even(S m) **then** S m + f m **else** f m
$\rightarrow$ S (S 0) + **if** even(S 0) **then** S 0 + f 0 **else** f 0
$\rightarrow$ S (S 0) + f 0
$\rightarrow$ S (S 0) + 0

$\rightarrow$ S (S 0)

PARADOSSO DI RUSSELL:

ASSIOMA DI COMPRENSIONE*: DATA UNA PROPRIETÀ P, ESISTE $\{x \mid P(x)\}$ E SI HA

$$\forall y.\, (y \in \{x \mid P(x)\} \Leftrightarrow P(y))$$

RUSSELL: $X \stackrel{\text{def}}{=} \{y \mid y \notin y\}$

$$X \in X \Leftrightarrow \neg(X \in X) \Leftrightarrow \neg(\neg(X \in X))$$

$$\Leftrightarrow \neg(\neg(\neg(X \in X)))$$

$$\ldots \Leftrightarrow \neg \ldots \neg(X \in X) \Leftrightarrow \ldots$$

* INCONSISTENTE!

TUTTO È UN INSIEME

$$X \in X$$

$$\neg$$

$$X \notin X$$

ASSIOMA DI COMPRENSIONE

- DA $P(y)$ RICAVA UN'INSIEME $\{y \mid P(y)\}$

$$\{y \mid y \notin y\} \in \{y \mid y \notin y\}$$

$$\iff \neg (\ldots)$$

TUTTO È UNA FUNZIONE

$$x\,x$$

$$f$$

$$f\,(x\,x)$$

λ-ASTRAZIONE:

- DA M (IN CUI OCCORRA y) RICAVA $\lambda y.M$

$$(\lambda y. f\,(y\,y))\,(\lambda y. f(y\,y))$$

$$\to_\beta f\,((\lambda y. f(y\,y))\,(\lambda y. f(y\,y)))$$

PER ESSERE TURING-COMPLETI ~~SERVE~~ È SUFFICIENTE RIPETERE LO STESSO CODICE PIÙ VOLTE

- CICLI
- RICORSIONE

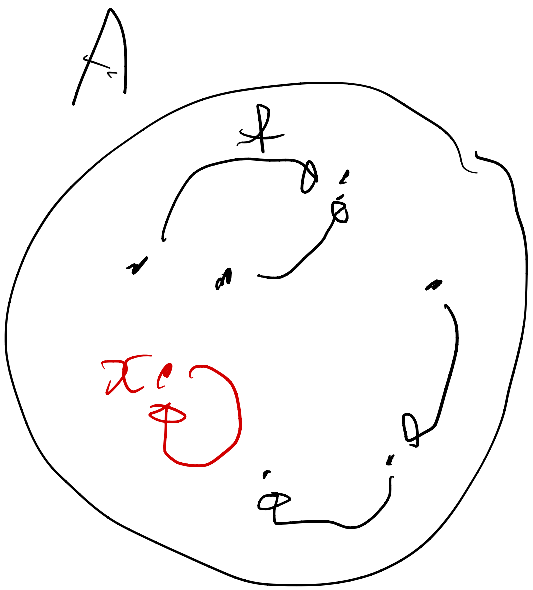
OPPURE ESEGUO OGNI VOLTA UNA NUOVA COPIA DEL CODICE

- $\lambda x . f (x\ x)$

Sia A un insieme e f una funzione di dominio e codominio A

x è un PUNTO FISSO di f sse

$$x = f(x)$$



es. $|\cdot|$ ha infiniti punti fissi su $\mathbb{Z}$

es. $x \mapsto x+1$ non ha punti fissi

TEOREMA: in λ-calcolo ogni termine M ha almeno un punto fisso

DIM. $(\lambda x. M(xx))(\lambda x. M(xx))$

è un punto fisso di M

INFATTI $(\lambda x. M(xx))(\lambda x. M(xx))$

$\to M((\lambda x. M(xx))(\lambda x. M(xx)))$

QED

Definizione: Y è un OPERATORE DI PUNTO FISSO sse $\forall M$, YM è un punto fisso di M

TEOREMA: $Y \stackrel{def}{=} \lambda f.\ (\lambda x.\ f(xx))(\lambda x.\ f(xx))$ è un operatore di punto fisso

DIM: OVVIA,

$$YM = (\lambda f.\ (\lambda x.\ f(xx))(\lambda x.\ f(xx)))\ M$$

$$\to_\beta (\lambda x.\ M(xx))(\lambda x.\ M(xx))$$ CHE È UN PUNTO FISSO DI M

QED.

Esempio di programma funzionale in OCaml

Somma dei numeri pari $\leq n$

let rec f n =
  match n **with**
    | 0 $\Rightarrow$ 0
    | S m $\Rightarrow$ **if** even(S m) **then** S m + f m **else** f m

Ingredienti

1. Il tipo di dato dei numeri naturali e la possibilità di farci pattern-matching sopra
2. Scelta: **if-then-else**, ma anche il pattern-matching stesso
3. Ricorsione: la funzione *f* invoca sé stessa

Programma

Mostreremo come esprimere in λ-calcolo ricorsione, scelta e tipi di dato assumendo di volta in volta di avere già a disposizione gli altri meccanismi.

Le funzioni anonime non possono richiamare sé stesse esplicitamente e non ci sono altri meccanismi per ciclare.

Chiediamo aiuto alla logica!

Ci sono situazioni/meccanismi in logica nei quali si arriva a ragionamenti circolari?

Paradosso di Russell

Paradosso di Russell

$$X = \{Y \mid \neg(Y \in Y)\}$$

$X \in X \iff \neg(X \in X) \iff \neg(\neg(X \in X)) \iff \neg(\neg(\neg(X \in X))) \iff \ldots$

Ingredienti

1. un predicato binario $\in$ per il quale gli insiemi siano sia oggetto (il contenuto) che soggetto (il contenitore)
2. l'auto-applicazione $Y \in Y$ e poi nuovamente $X \in X$
3. l'introduzione di una negazione intorno all'autoapplicazione
4. la trasformazione del predicato $\neg(Y \in Y)$ in insieme autoapplicabile (via assioma inconsistente di comprensione)

Paradosso di Russell nel λ-calcolo

Ingredienti

1. un predicato binario $\in$ per il quale gli insiemi siano sia oggetto (il contenuto) che soggetto (il contenitore)
2. l'auto-applicazione $Y \in Y$ e poi nuovamente $X \in X$
3. l'introduzione di una negazione intorno all'autoapplicazione
4. la trasformazione del predicato $\neg(Y \in Y)$ in insieme autoapplicabile (via assioma inconsistente di comprensione)

Ingredienti

1. applicazione di funzione per la quale i λ-termini sono sia oggetto (l'argomento) che soggetto (la funzione invocata)
2. l'auto-applicazione YY e poi nuovamente XX
3. l'introduzione di una funzione g intorno all'autoapplicazione
4. la trasformazione del λ-termine $g(YY)$ in una funzione autoapplicabile (via λ-astrazione)

Paradosso di Russell nel λ-calcolo

Paradosso di Russell

$$X = \{Y \mid \neg(Y \in Y)\}$$

$X \in X \iff \neg(X \in X) \iff \neg(\neg(X \in X)) \iff \neg(\neg(\neg(X \in X))) \iff \ldots$

Nel λ-calcolo

$$\lambda Y.g(YY)$$

$(\lambda Y.g(YY))(\lambda Y.g(YY)) \rightarrow_\beta g((\lambda Y.g(YY))(\lambda Y.g(YY))) \rightarrow_\beta g(g((\lambda Y.g(YY))(\lambda Y.g(YY)))) \rightarrow_\beta \ldots$

Il più piccolo λ-termine divergente

Caso particolare per g funzione identità:

$$(\lambda x.xx)(\lambda x.xx) \rightarrow_\beta (\lambda x.xx)(\lambda x.xx) \rightarrow_\beta \ldots$$

Divergenza

Un termine t_0 può divergere sse $\forall i. \exists t_{i+1}. t_i \to_\beta t_{i+1}$.

Esempio: $(\lambda x.xx)(\lambda x.xx) \to_\beta (\lambda x.xx)(\lambda x.xx) \to_\beta \ldots$

Un termine può divergere e può convergere allo stesso tempo.
Esempio:

$$y \; {}_\beta\!\leftarrow (\lambda x.y)((\lambda x.xx)(\lambda x.xx)) \to_\beta (\lambda x.y)((\lambda x.xx)(\lambda x.xx)) \to_\beta \ldots$$

Punti fissi

Punto fisso in matematica

Sia A un insieme e $f : A \to A$. Un $x \in A$ è punto fisso di f sse $f(x) = x$.

In matematica esistono funzioni che non ammettono punti fissi (es. la funzione $x \mapsto x + 1$)

Punto fisso in λ-calcolo

Un λ-termine t è punto fisso di f sse $f(t) =_\beta t$, dove $=_\beta$ (β-conversione) è la chiusura riflessiva, simmetrica e transitiva di $\to_\beta$.

I punti fissi esistono sempre: $(\lambda x.f(xx))(\lambda x.f(xx))$ è punto fisso di f

Operatore di punto fisso

Un λ-termine Y è un operatore di punto fisso sse per ogni λ-termine f, Yf è punto fisso di f.

$\lambda f.((\lambda x.f(xx))(\lambda x.f(xx)))$ è un operatore di punto fisso.
Ne esitono infiniti altri.

Codifica delle funzioni ricorsive

Somma dei numeri pari $\leq n$ via ricorsione

let rec f : (nat $\rightarrow$ nat) =
 λn.
  match n **with**
   | 0 $\Rightarrow$ 0
   | S m $\Rightarrow$ **if** even(S m) **then** S m + f m **else** f m

Funtore (non ricorsivo) associato

let F : (nat $\rightarrow$ nat) $\rightarrow$ (nat $\rightarrow$ nat) =
λf.
 λn.
  match n **with**
   | 0 $\Rightarrow$ 0
   | S m $\Rightarrow$ **if** even(S m) **then** S m + f m **else** f m

Codifica delle funzioni ricorsive

Funtore (non ricorsivo) associato

let F : (nat $\rightarrow$ nat) $\rightarrow$ (nat $\rightarrow$ nat) =
 λf.
  λn.
   match n **with**
    | 0 $\Rightarrow$ 0
    | S m $\Rightarrow$ **if** even(S m) **then** S m + f m **else** f m

Codifica di f in λ-calcolo

$f = YF$ dove Y è un operatore di punto fisso.

Infatti $f = YF \rightarrow_\beta F(YF) = Ff$.

Ancora più esplicitamente

Somma dei numeri pari $\leq n$ via ricorsione

let rec f =
 λn.
  match n **with**
   | 0 $\Rightarrow$ 0
   | S m $\Rightarrow$ **if** even(S m) **then** S m + f m **else** f m

Codifica in λ-calcolo

Y
 (λf.
  λn.
   match n **with**
    | 0 $\Rightarrow$ 0
    | S m $\Rightarrow$ **if** even(S m) **then** S m + f m **else** f m)

Ancora più esplicitamente

Fattoriale

let rec fact =
  λn.
   match n **with**
    | 0 $\Rightarrow$ 1
    | S m $\Rightarrow$ S m * fact m

Codifica in λ-calcolo

Y
 (λfact.
  λn.
   match n **with**
    | 0 $\Rightarrow$ 1
    | S m $\Rightarrow$ S m * fact m)

SIA fact $= Y\,(\lambda \mathit{fact}\ \ldots)$

SI HA

$\mathit{fact}\ (S\ 0) =$

$Y\,(\lambda \mathit{fact}\ \ldots)\ (S\ 0) \to_\beta$

$(\lambda \mathit{fact}\ \ldots)\ (Y\,(\lambda \mathit{fact}\ \ldots))\ (S\ 0) \to_\beta$

$(S\ 0)$

$(\lambda n.\ \mathrm{MATCH}\ n\ \mathrm{WITH}\ 0 \Rightarrow 1 \mid S\ m \Rightarrow S\ m * \boxed{Y\,(\lambda \mathit{fact}\ \ldots)}\ m)$

$\to_\beta^* \quad S\ 0 * Y\,(\lambda \mathit{fact}\ \ldots)\ 0$

$\to_\beta^* \quad S\ 0 * 1$

$\to_\beta^* \quad S\ 0$

$$\to_\beta \Big[(\lambda n.\ \text{MATCH } n \text{ WITH } 0 \Rightarrow 1 \mid S\ m \Rightarrow S\ m\ *$$

$$(\lambda n.\ \text{MATCH } n \text{ WITH } 0 \Rightarrow 1 \mid S\ m \Rightarrow S\ m\ *\ \underbrace{Y\,(\lambda \mathrm{Fact}.\ldots)}\ m)$$

$$(S\ 0)\ \Big]$$

$$\to_\beta \ \ldots$$

$$\to_\beta \ \ldots$$

$$\to_\beta \ \ldots$$

Tipi di dato algebrici: esempi

(NOME DEL) TIPO

COSTRUTTORI

type $\mathbb{B}$ = true : $\cancel{\mathbb{N}}$ $\mathbb{B}$ | false : $\mathbb{B}$
Es: true : $\mathbb{B}$

TYPE SEME = CUORI : SEME
| QUADRI : SEME
| PICCHE : SEME
| FIORI : SEME

type $\mathbb{N}$ = 0 : $\mathbb{N}$ | S : $\mathbb{N} \to \mathbb{N}$
Es: S (S 0) : $\mathbb{N}$

SI LEGGE "CONS"

type *List T* = [] : *List T* | (::) : $T \to$ *List T* $\to$ *List T*
Es: $\underbrace{(S\ 0)}_{\text{TESTA}}$ :: $\underbrace{\underbrace{0}_{\text{TESTA}} :: \underbrace{[]}_{\text{CODA}}}_{\text{CODA}}$: *List* $\mathbb{N}$

TIPO PARAMETRICO

PARAMETRO, VARIABILE DI TIPO

TYPE N^2 = PAIR : $N \to N \to N^2$

es. PAIR 3 5 : N^2

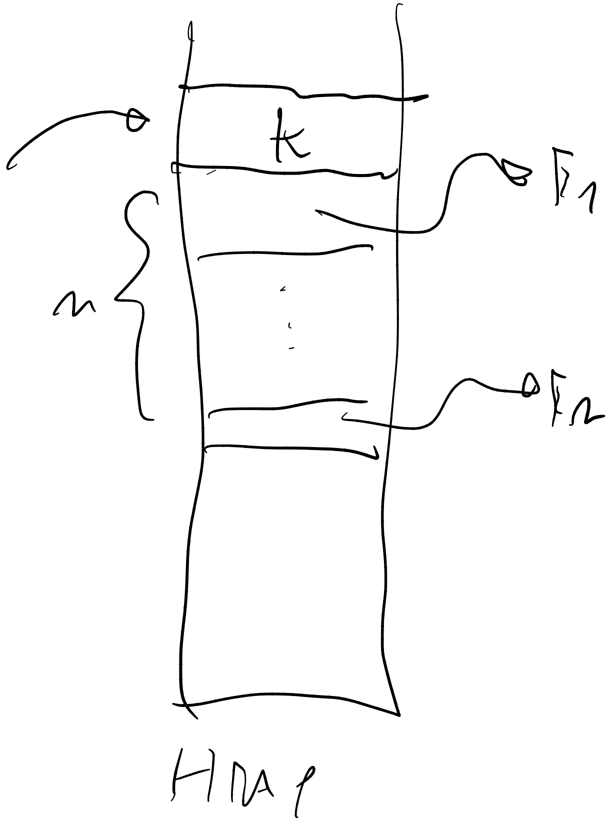
IMPLEMENTAZIONE A BASSO LIVELLO:

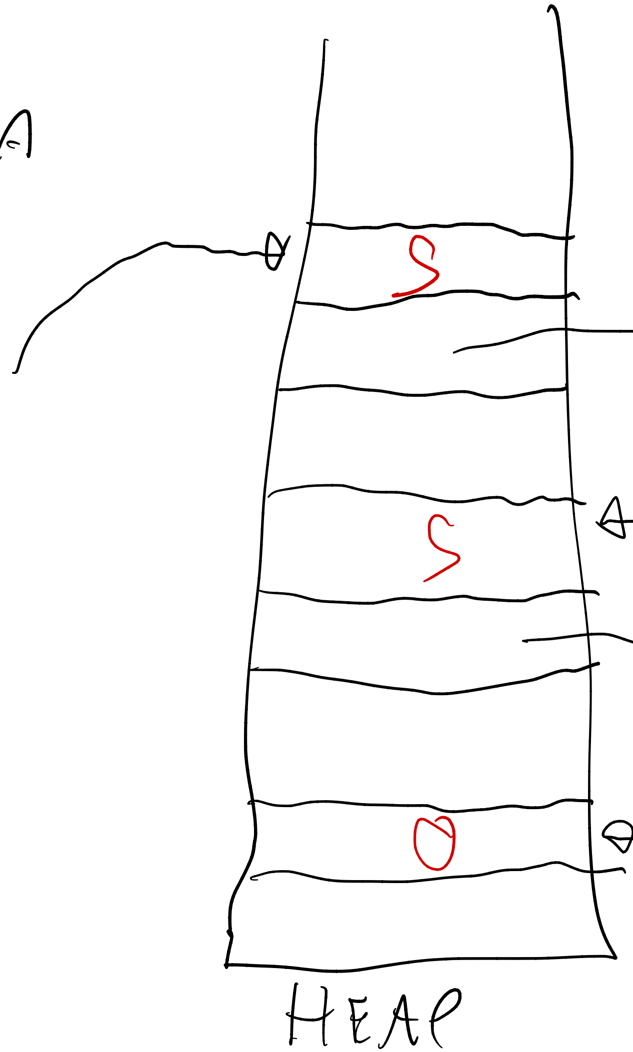
UN COSTRUTTORE K APPLICATO A n INPUT $E_1 \dots E_n$

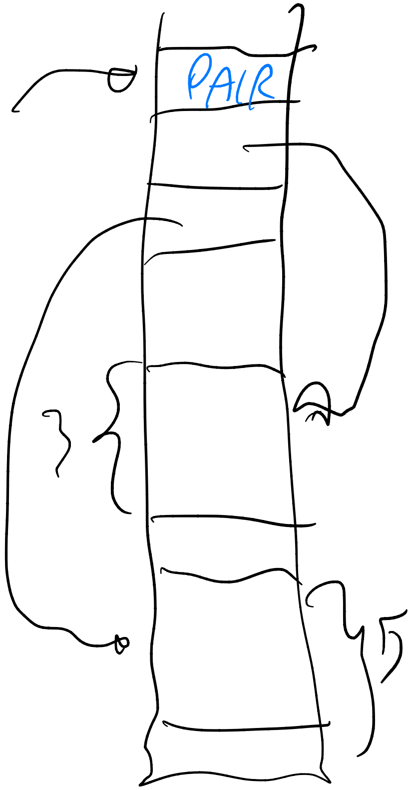
es. S (S 0)

PAIR 3 5

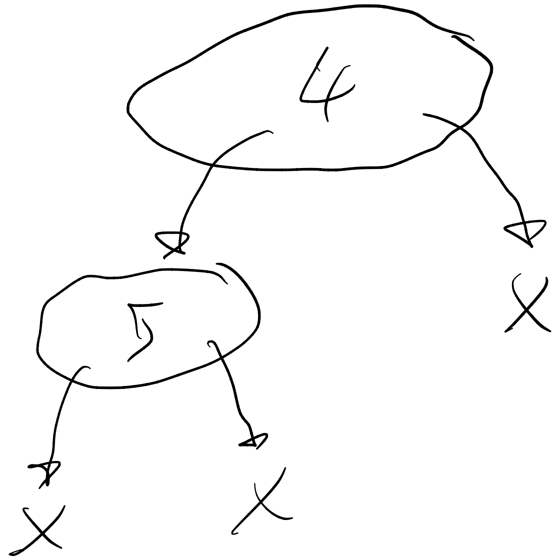
IN MEMORIA DIVENTA





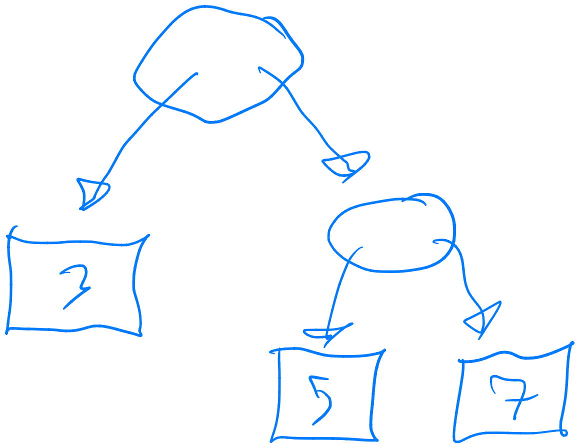


ALBERI DI TIPO 1:



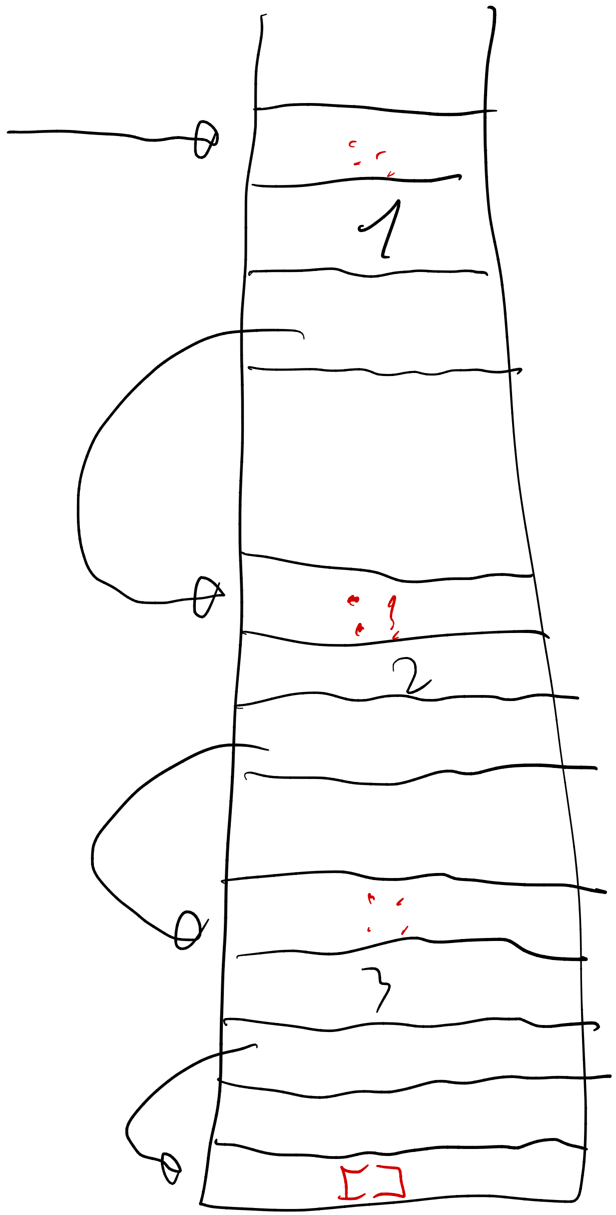
TYPE TREE1 T ::= X : TREE1 T | ◯ : TREE1 T → T → TREE1 T → TREE1 T

ALBERO DI TIPO 2:



TYPE TREE2 T ::= □ : T → TREE2 T | ◯ : TREE2 T → TREE2 T → TREE2 T

$1 :: (2 :: (3 :: []))$



HEAP

$\underline{\text{MATCH}}$ x $\underline{\text{WITH}}$

$\vdots$

$| \; K_i \; \underbrace{x_1 \dots x_n} \Rightarrow \; M_i$

NOME DELL'i-ESIMO COSTRUTTORE

VARIABILI, UNA PER OGNI ARGOMENTO DEL COSTRUTTORE

LEGATE IN M_i

CODICE DA ESEGUIRE

PUÒ USARE LE VARIABILI $x_1 \dots x_n$

ESEMPIO $\mathbb{B}$: SIA $b : \mathbb{B}$

$$\begin{aligned}&\underline{\text{MATCH}}\ b\ \underline{\text{WITH}}\\ &\quad |\ \text{true} \Rightarrow M_1\\ &\quad |\ \text{false} \Rightarrow M_2\end{aligned} \quad \equiv \quad \underline{\text{IF}}\ b\ \underline{\text{THEN}}\ M_1\ \underline{\text{ELSE}}\ M_2$$

ESEMPIO SU LIST $\mathbb{N}$: CALCOLARE LA SOMMA

$$\begin{aligned}&\underline{\text{LET}}\ \underline{\text{REC}}\ \text{SUM}\ l = \quad \text{LIST } \mathbb{N}\\ &\quad \underline{\text{MATCH}}\ l\ \underline{\text{WITH}}\\ &\qquad |\ [\,] \Rightarrow 0\\ &\qquad |\ \underbrace{x}_{\text{TESTA } \mathbb{N}} :: \underbrace{l}_{\text{CODA LIST } \mathbb{N}} \Rightarrow x + \text{sum}\ l\end{aligned}$$

MATCH $5 :: (2 :: [])$ WITH

$\mid [\,] \Rightarrow 0$

$\mid x :: l \Rightarrow x + \text{sum } l$

$x :: l = 5 :: (2 :: [\,])$

$$\longrightarrow (x + \text{sum } l)\{5/x\}\{2 :: [\,]/l\}$$

$$= 5 + \text{sum } (2 :: [\,])$$

MATCH l WITH

| [] => 0

| x::l => x + sum l

IF ((*l).tag == "[]")

{

0

}

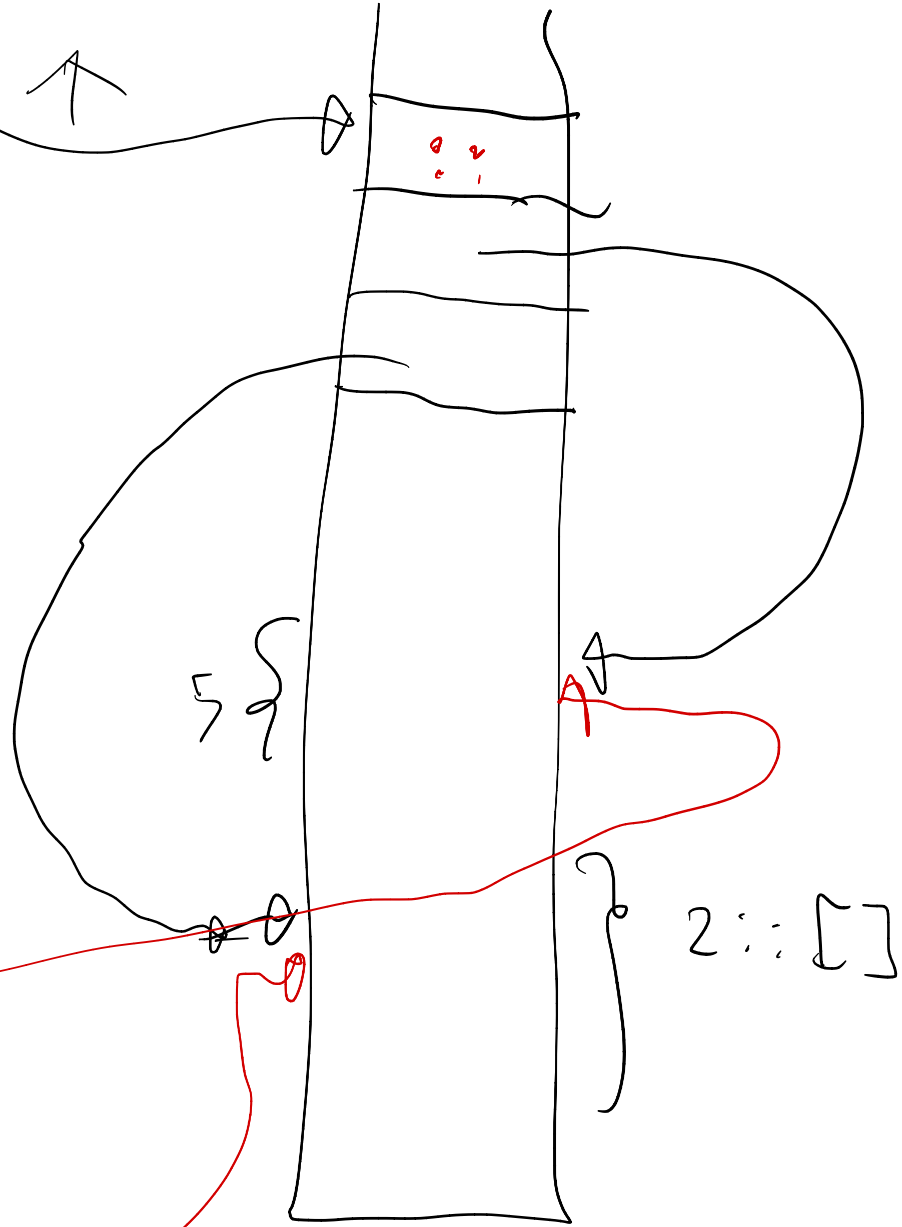
{

INT *x = (*l).f1;

INTLIST *l = (*l).f2;

x + sum l

}



TYPE LIST T ::= $[\,] : \text{LIST } T \mid (::) : T \to \text{LIST } T \to \text{LIST } T$

let rec sum =
$\lambda l.$
match l with
| $[\,] \Rightarrow 0$
| $x :: l \Rightarrow x + \text{sum } l$

let rec sum =
$\lambda l.$
$\text{match}_{\text{LIST}}$
l
0
$(\lambda x.\, \lambda l.\, x + \text{sum } l)$

$\text{empty} : \text{LIST } T$

$\text{cons} : T \to \text{LIST } T \to \text{LIST } T$

$\text{match}_{\text{LIST}} : \forall X.\, \text{LIST } T \to X \to (T \to \text{LIST } T \to X) \to X$

$\text{match}_{\text{LIST}}\ \text{empty}\ M_e\ M_c \to^*_\beta M_e$

$\text{match}_{\text{LIST}}\ (\text{cons}\ M_x\ M_l)\ M_e\ M_c \to^*_\beta M_c\ M_x\ M_l$

$\mathrm{empty} : \mathrm{LIST}\ T$

$\mathrm{cons} : T \to \mathrm{LIST}\ T \to \mathrm{LIST}\ T$

$\mathrm{match}_{\mathrm{LIST}} : \cancel{\forall X. \mathrm{LIST}\ T \to X \to (T \to \mathrm{LIST}\ T \to X) \to X}$

$\mathrm{LIST}\ T \to \forall X.\ X \to (T \to \mathrm{LIST}\ T \to X) \to X$

$$\mathrm{LIST}\ T \stackrel{\mathrm{def}}{=} \forall X. X \to (T \to \mathrm{LIST}\ T \to X) \to X$$

$$\mathrm{match}_{\mathrm{LIST}} \stackrel{\mathrm{def}}{=} \lambda \underbrace{x}_{\mathrm{LIST}\ T}.\underbrace{x}_{\mathrm{LIST}\ T}$$

$$\mathrm{empty} : \mathrm{LIST}_T = \forall X.\ X \to (T \to \mathrm{LIST}\ T \to X) \to X$$

$$\stackrel{\mathrm{def}}{=} \lambda \underbrace{e}_{X}.\lambda \underbrace{c}_{T \to \mathrm{LIST}\ T \to X}.\ e$$

$\mathrm{match}_{\mathrm{LIST}}\ \mathrm{empty}\ M_e\ M_c \to_\beta^* M_e$

$\mathrm{match}_{\mathrm{LIST}}\ (\mathrm{cons}\ M_x\ M_L)\ M_e\ M_c \to_\beta^* M_c\ M_x\ M_L$

$\mathrm{match}_{\mathrm{LIST}}\ \mathrm{empty}\ M_e\ M_c$

$= (\lambda x.x)(\lambda e.\lambda c.\ )\ M_e\ M_c$

$\to_\beta (\lambda e.\lambda c.\ e)\ M_e\ M_c$

$\to_\beta^2 M_e$

$$\mathrm{cons} : T \to \mathrm{List}\ T \to \mathrm{List}\ T$$

$$= T \to \mathrm{List}\ T \to \forall X.\ X \to (T \to \mathrm{List}\ T \to X) \to X$$

$$\overset{\mathrm{def}}{=} \lambda \underset{\cdot\cdot}{x}^{T}.\ \lambda \underset{\cdot\cdot}{l}^{\mathrm{List}\ T}.\ \lambda \underset{\cdot\cdot}{e}^{X}.\ \lambda \underset{\cdot\cdot}{c}^{T \to \mathrm{List}\ T \to X}.\ c\ x\ l$$

$$\mathrm{match}_{\mathrm{List}}\ (\mathrm{cons}\ M_x\ M_l)\ M_e\ M_c$$

$$= (\lambda x.\ x)\ (\mathrm{cons}\ M_x\ M_l)\ M_e\ M_c$$

$$\to_\beta \mathrm{cons}\ M_x\ M_l\ M_e\ M_c$$

$$= (\lambda x.\lambda l.\lambda e.\lambda c.\ c\ x\ l)\ M_x\ M_l\ M_e\ M_c$$

$$\to_\beta^{4} M_c\ M_x\ M_l$$

$$\text{TYPE List } L = \underbrace{\overset{0 \text{ ARGOMENTI}}{[\,]} : \text{List } L \;\Big|\; (::) : \overset{2 \text{ ARGOMENTI}}{T \to \text{List } T \to \text{List } T}}_{2 \text{ COSTRUTTORI (EMPTY/CONS)}}$$

$$\text{match}_{List} \overset{def}{=} \lambda x.x$$

$$\text{empty} \overset{def}{=} \lambda e.\lambda c.\, e$$

$$\text{cons} \overset{def}{=} \lambda x.\lambda l.\lambda e.\lambda c.\; c\; x\; l$$

$$\text{TYPE } \mathbb{B} = \text{true} : \mathbb{B} \;|\; \text{false} : \mathbb{B}$$

$$\text{match}_{\mathbb{B}} \overset{def}{=} \lambda x.x$$

$$\text{true} \overset{def}{=} \lambda t.\lambda e.\, t$$

$$\text{false} \overset{def}{=} \lambda t.\lambda e.\, e$$

$$\text{match}_{\mathbb{B}}\ \text{true}\ M_T\ M_E$$
$$= (\lambda x.x)\ \text{true}\ M_T\ M_E$$
$$\to_\beta \text{true}\ M_T\ M_E = (\lambda t.\lambda e.t)\, M_T\, M_E$$
$$\to_\beta^2 M_T$$

$$\text{TYPE } \mathbb{N} = 0 : \mathbb{N} \;|\; S : \mathbb{N} \to \mathbb{N}$$

$$\text{match}_{\mathbb{N}} = \lambda x.x$$

$$0 = \lambda t.\lambda s.\, t$$

$$S = \lambda n.\lambda t.\lambda s.\, s\, n$$

$$\text{match}_{\mathbb{N}}\ (S\ N)\ M_t\ M_s$$
$$\to_\beta S\ N\ M_t\ M_s$$
$$= (\lambda n.\lambda t.\lambda s.\, s\, n)\ N\ M_t\ M_s$$
$$\to_\beta^3 M_s\ N$$

$$\text{TYPE SEME} = \underbrace{\text{cuori} \overset{\Pi^{0}}{:} \text{SEME} \mid \text{quadri} \overset{\Pi^{0}}{:} \text{SEME} \mid \text{picche} \overset{\Pi^{0}}{:} \text{SEME} \mid \text{fiori} \overset{\Pi^{0}}{:} \text{SEME}}_{4 \text{ COSTRUTTORI}}$$

$$\text{match}_{\text{SEME}} \overset{\text{def}}{=} \lambda x.\, x$$

$$\text{cuori} \overset{\text{def}}{=} \lambda c.\, \lambda q.\, \lambda p.\, \lambda f.\, c$$

$$\text{quadri} \overset{\text{def}}{=} \lambda c.\, \lambda q.\, \lambda p.\, \lambda f.\, q$$

$$\text{picche} \overset{\text{def}}{=} \lambda c.\, \lambda q.\, \lambda p.\, \lambda f.\, p$$

$$\text{fiori} \overset{\text{def}}{=} \lambda c.\, \lambda q.\, \lambda p.\, \lambda f.\, f$$

```
VAR a = 2;
f(x, y) {
    VAR z = 2;
    x = z · y;
    z = y + a;
    a = 3;
    x = x + g();
    RETURN x + z;
    g() {
        z = z + 1;
        RETURN 3;
    }
}
```

```
f(x, y, a) {
    VAR z = 2;
    VAR x' = z · y;
    VAR z' = y + a;
    VAR a' = 3;
    VAR <z'', RES> = g(z');
    VAR x'' = x' + RES;
    RETURN <a', x'', x'' + z''>
    g(z) {
        VAR z' = z + 1;
        RETURN <z', 3>
    }
}
```

VAR $x = 4;$

$\ldots g(x);$

$(\ldots g(x) \ldots)\{4/x\}$

$= \ldots g(4) \ldots$

$(\lambda x \ldots g(x))\ 4$

LET $x = 4$ IN $\ldots g(x) \ldots$

CICLI:

```
VAR x = 10;
VAR RES = 0;
WHILE x > 0 DO
    RES = RES + x;
    x = x - 1;
DONE
```

```
let rec while x res =
  if x > 0 then
    while (x-1) (res+x)
  else
    <x, res>
```

```
struct persona {
    nome = "claudio";
    età = 47;
    città = "Bologna"
}

if persona.età > 20
then persona.nome
```

```
type persona =
    mk : string → int → string
        → persona

età p =
  match p with
    mk nome età città
      ⇒ età

if età persona > 20
then nome persona
```

OGGETTI :

```
object persona {
    età = 41;
    nome = "claudio";

    invecchia ( n ) {
        if self.età + n > 100 then
            self.muori()
        else
            self.età = self.età + n }
    muori() { self.nome = "RIP"; }
}
```

```
struct persona {
    età = 41;
    nome = "claudio";
    invecchia =
    λself. λn.
      if self.età + n
         > 100 then
         self.muori self
      else
      { età = self.età + n;
        nome = self.nome;
        invecchia = self.invecchia
      } muori = self.muori
```

ECCEZIONI:

TYPE $e = E_1 : \mathbb{N} \to e$
$\mid E_2 : \text{STRING} \to e$
$\mid E_3 : \mathbb{N} \to \mathbb{N} \to e$

- throw e

- try M with
  $\mid E_1\ x \Rightarrow M_1$
  $\mid E_2\ x \Rightarrow M_2$
  $\mid E_3\ x\ y \Rightarrow M_3$

try
(if PARI (4) then
true
else
throw (E_1 7)) OR false
with
$\mid E_1\ x \Rightarrow$ PARI(x)
$\mid E_2\ x \Rightarrow$ false
$\mid E_3\ x\ y \Rightarrow$ true

try true
with ... | ... | ...
$\to$ true

try

(**IF** PARI(5)

THEN true

ELSE

three (E1 7)

) OR FALSE

with

| – —

| – —

| – ⌒

try three (E1 7) OR FALSE

with

| E1 x ⇒ PARI(x)

| E2 _ ⇒ false

| E3 x y ⇒ true

↓

PARI (7)

$M : \mathbb{B} \vee \mathbb{N} \vee \text{STRING} \vee (\mathbb{N} \times \mathbb{N})$

$M : I \longrightarrow O_1 \vee O_2 \vee \dots \vee O_n$

IN λ-CALCOLO: HO SOLO

$M : I_1 \to I_2 \to \dots \to I_k \longrightarrow O$

$\simeq I_1 \wedge I_2 \wedge \dots \wedge I_k \longrightarrow O$

$$\boxed{I \multimap O_1 \vee O_2}$$

$$\equiv I \multimap \neg\neg(O_1 \vee O_2)$$

$$\equiv I \multimap \neg(\neg O_1 \wedge \neg O_2)$$

$$\equiv I \multimap ((O_1 \multimap \bot) \wedge (O_2 \multimap \bot) \multimap \bot)$$

$$\equiv \boxed{I \multimap (O_1 \multimap \bot) \multimap (O_2 \multimap \bot) \multimap \bot}$$

$$\neg F \equiv F \multimap \bot$$

$$\neg(P \vee Q) \equiv \neg P \wedge \neg Q$$

$$\neg(P \wedge Q) \equiv \neg P \vee \neg Q$$

$$\neg\neg F \equiv F$$

$$F_1 \wedge F_2 \multimap G$$

$$\equiv F_1 \multimap F_2 \multimap G$$

$f : \mathbb{Z} \to \mathbb{Z} \vee \mathbb{Z} \vee \text{STRING}$

TERMINA CON SUCCESSO THROW NEGATIVE THROW STRING

$=$

$\lambda x.$

(IF $x < 0$ THEN
THROW (NEGATIVE x)
ELSE IF $x > 10$ THEN
THROW (TOOBIG "RIDUCI")
ELSE
x) $\ast$ 10

$$f : \mathbb{Z} \to (\mathbb{Z} \to \bot) \to (\mathbb{Z} \to \bot) \to (\text{STRING} \to \bot) \to \bot$$

$$= \lambda x \; \underbrace{\lambda k_{return}.\; \lambda k_{E1}.\; \lambda k_{E2}.}_{\text{CONTINUATIONI}}$$

$\underline{\text{IF}}$ $x < 0$ $\underline{\text{THEN}}$

$k_{E1}\; x$

$\underline{\text{ELSE}}$ IF $x > 10$ $\underline{\text{THEN}}$

k_{E2} "RIDUCI"

$\underline{\text{ELSE}}$

$k_{return}\; \boxed{\boxed{x} * 10}$

try

$f\ 2$

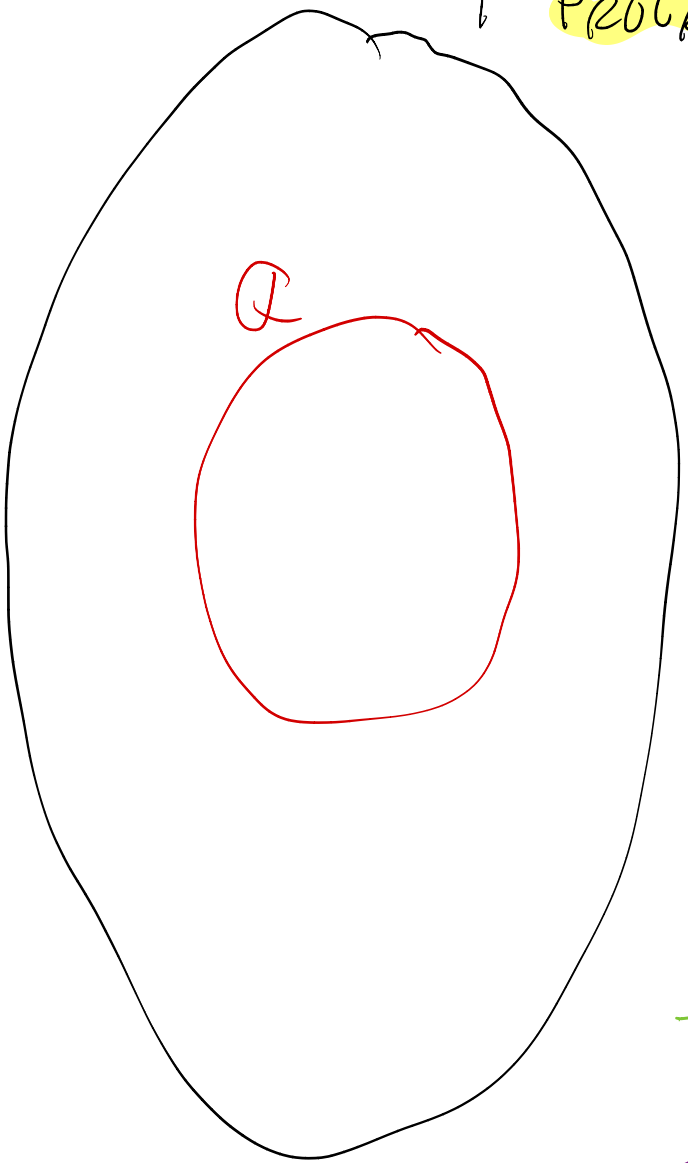
with

$\mid E_1\ x \Rightarrow x+2$

$\mid E_2\ \_ \Rightarrow 0$

$\rightsquigarrow f\ (\lambda x.\, x)\ (\lambda x.\, x+2)\ (\lambda \_.\, 0)$

P = L'INSIEME DI TUTTI I PROGRAMMI ← COME È SCRITTO



- Una proprietà è un sottoinsieme di P
- x ha la proprietà Q se $x \in Q$
- una proprietà Q è banale se $Q = \emptyset \vee Q = P$
- esempio: $Q = \{ \pi \in P \,/\, \pi$ usa 2 variabili$\}$
- esempio $Q = \{ \pi \in P \,/\, \pi(\alpha) = 1 \}$

CALCOLANO LA STESSA FUNZIONE

— una proprietà Q è estensionale se

$\forall p, q \in P. \; (\forall i.\ p(i) \text{ restituisce } \sigma \Leftrightarrow q(i) \text{ restituisce } \sigma)$

$\Rightarrow p \in Q \Leftrightarrow q \in Q$

— una proprietà è intensionale se non è estensionale

- una proprietà Q è decidibile se $\exists p \in P.$

$$\forall q \in P. \; (q \in Q \iff p(q) = true \;\wedge\; q \notin Q \iff p(q) = false)$$

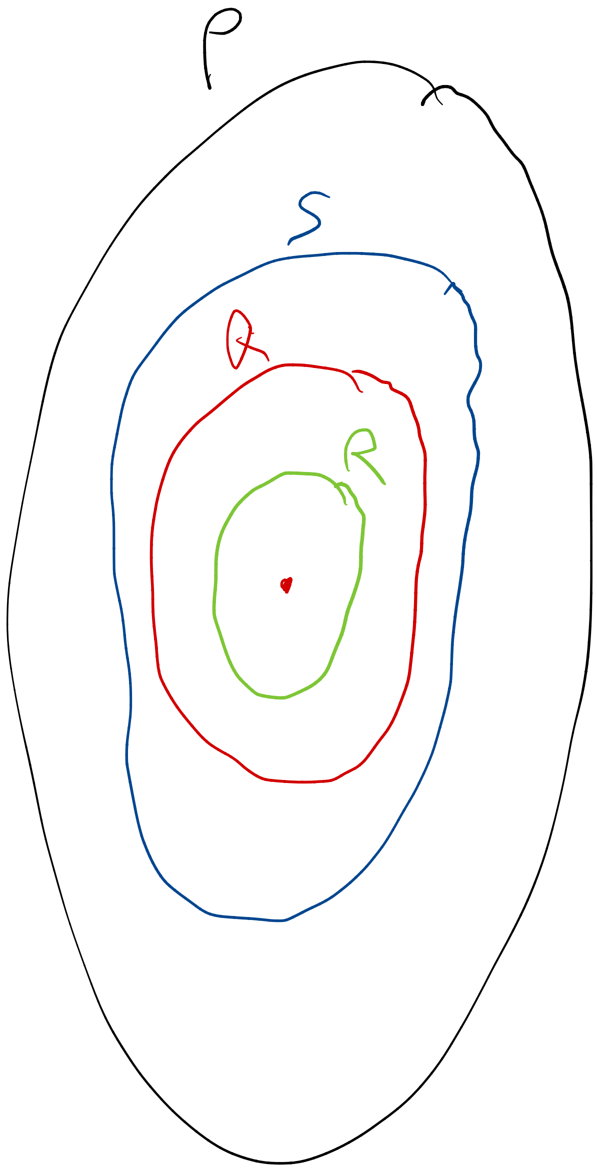
- esempio: $Q = \{ p \in P \,/\, |p| < 100 \text{ caratteri} \}$

  Q è decidibile

TEOREMA DI RICE:

$\forall Q$. se Q non banale e Q estensionale

allora Q <u>NON</u> è decidibile



- R è un' approssimazione da dentro di Q se $R \subseteq Q$

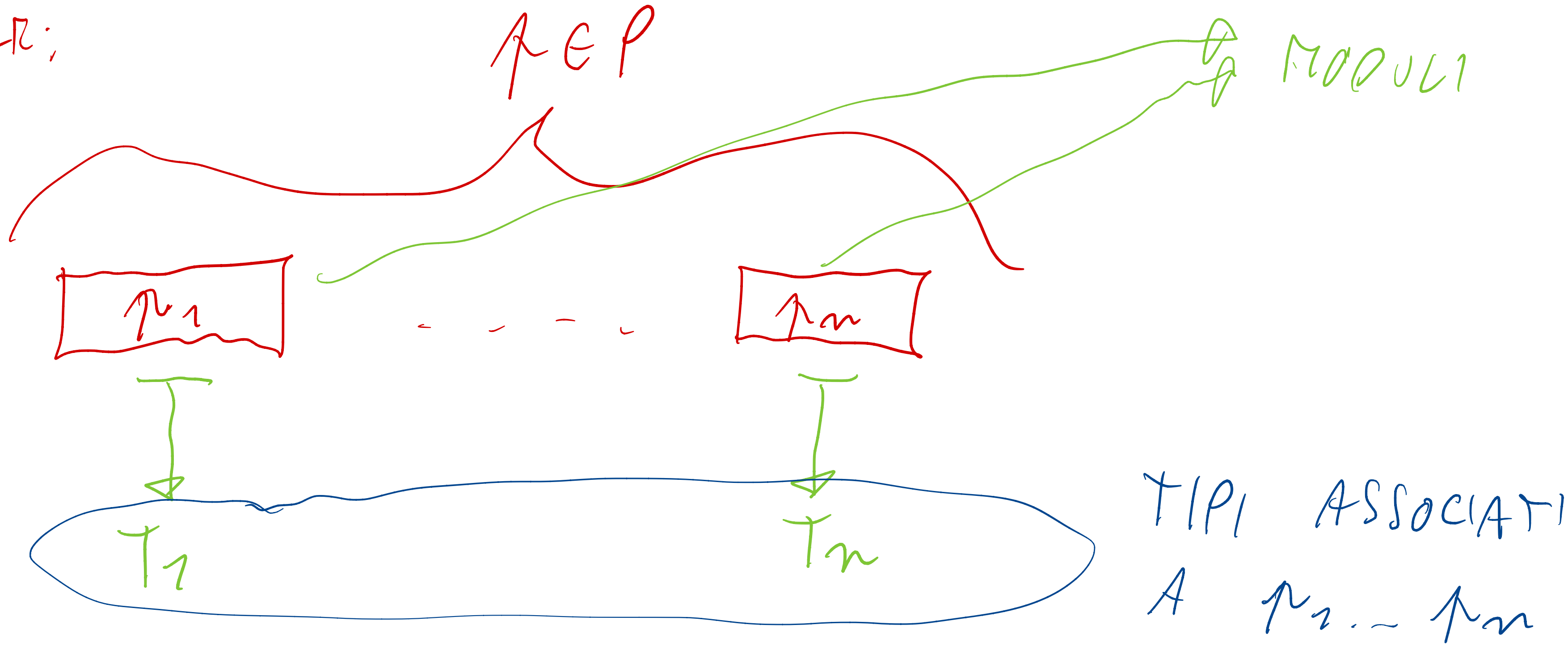
- S è un' approssimazione da fuori di Q se $Q \subseteq S$

- supponiamo che R/S siano decidibili e decise da un programma r/s

TEOREMA: $\forall p. \, (r(p) = true \Rightarrow p \in Q) \wedge (s(p) = false \Rightarrow p \notin Q)$

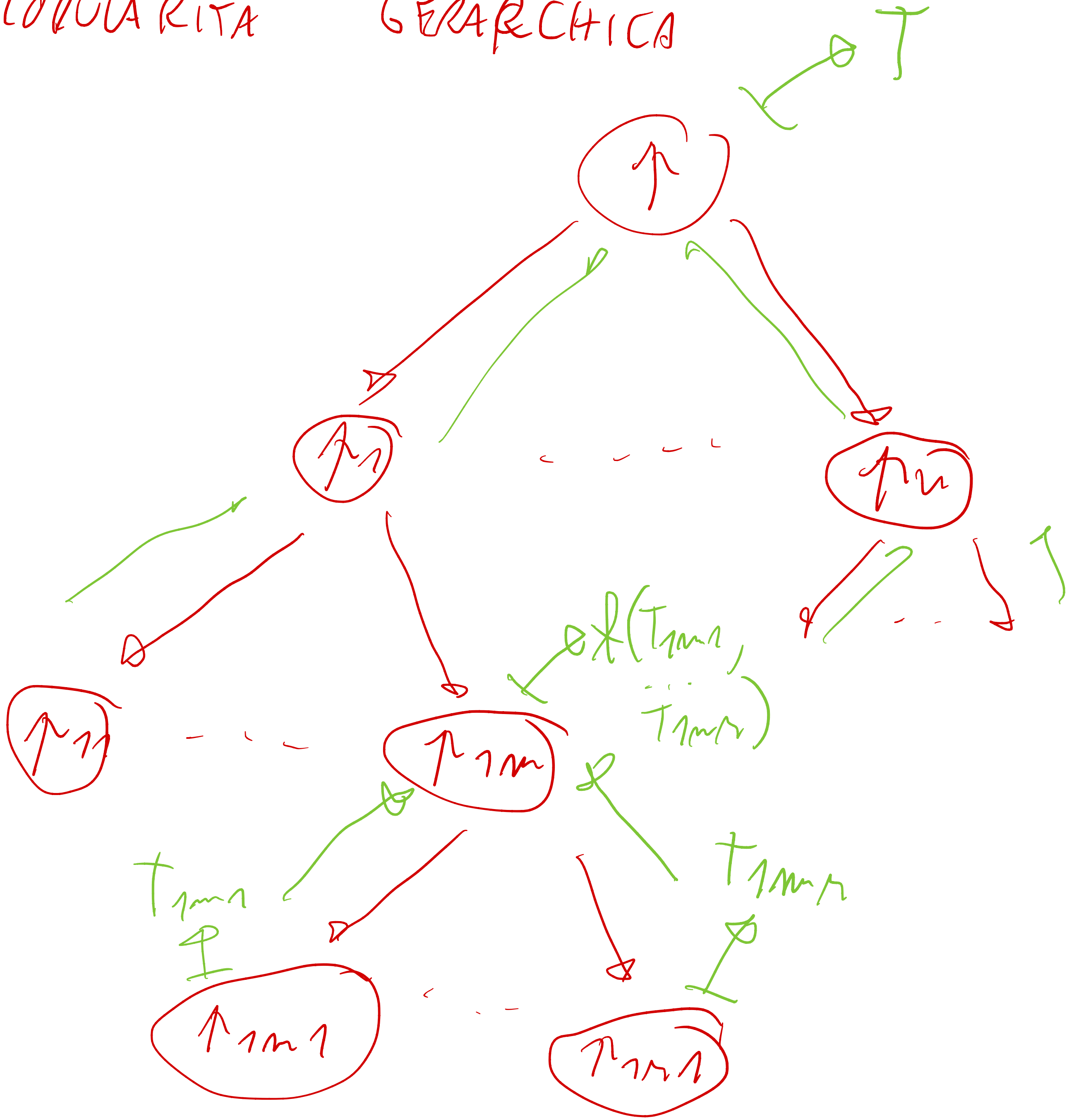
— un sistema di tipi è l'implementazione di un programma che decide un'approssimazione da dentro o da fuori IN MANIERA MODULARE

— MODULARE:



$$\mu(r) = true \iff r \in R \iff \bar{\mu}(T_1, \dots, T_n)$$

— MODULARITÀ GERARCHICA



λ-CALCOLO TIPATO SEMPLICE

TIPI $T ::= A \mid T \to T$

TIPO DELLE FUNZIONI CON UN CERTO INPUT/OUTPUT

VARIABILE DI TIPO $A \mid B \mid C \mid \dots$

intuizione: $\mathbb{B} / \mathbb{Z} / \mathbb{N} /$ STRING $/ \dots$

es. $A \to B$ è il tipo delle funzioni che dato un input di tipo A restituiscono output di tipo B

- $\multimap$ È ASSOCIATIVO A DESTRA

- es. $A \multimap B \multimap C = A \multimap (B \multimap C)$

- CONTESTO $\Gamma ::= \;|\; \Gamma, x : T$

VARIABILI, NON TERMINI

- es. $x : A,\ y : B,\ z : A \multimap B$

- IPOTESI: IN Γ NESSUNA VARIABILE È RIPETUTA

- SCRIVERÒ $(x:T) \in \Gamma$ PER DIRE $\Gamma = \ldots, x:T, \ldots$

- JUDGEMENT DI TIPAGGIO $\Gamma \vdash t : T$

è una relazione ternaria

si legge "in Γ, t ha tipo T"

- DEFINISCO $\Gamma \vdash t : T$ ATTRAVERSO UN

sistema di inferenza

$$\frac{(x:T)\in\Gamma}{\Gamma\vdash x:T}\qquad\frac{\Gamma\vdash M:T_1\to T_2\qquad\Gamma\vdash N:T_1}{\Gamma\vdash MN:T_2}\qquad\frac{\Gamma, x:T_1\vdash M:T_2}{\Gamma\vdash \lambda x.M:T_1\to T_2}$$

$$\dfrac{\dfrac{(f:(A\to A)\to B)\in\Gamma}{\underbrace{f:(A\to A)\to B,\ x:A\to A}_{\Gamma}\vdash f:(A\to A)\to B}\qquad\dfrac{(x:A\to A)\in\Gamma}{\underbrace{f:(A\to A)\to B,\ x:A\to A}_{\Gamma}\vdash x:A\to A}}{\dfrac{f:(A\to A)\to B,\ x:A\to A\vdash f\,x:B}{f:(A\to A)\to B\vdash\lambda x.\,f\,x:(A\to A)\to B}}$$

$\lambda x.xx$ NON È BEN TIPATO

$\lambda x.xx$ NON HA LA PROPRIETÀ MISTERIOSA

VERO SOLO SE

$$T_3? = T_3? \to T_4?$$

IMPOSSIBILE

$$\dfrac{\dfrac{(x : T_3? \to T_4?) \in \Gamma}{x : \cancel{T_1?}\, T_3? \to T_4? \vdash x : T_3? \to T_4?} \qquad \dfrac{(x : T_3?) \in x : T_3? \to T_4?}{x : T_3? \to T_4? \vdash x : T_3?}}{x : \cancel{T_1?}^{\,T_3? \to T_4?} \vdash x\,x : T_2?}$$

$$\vdash \lambda x.xx :$$

LOGICA PROPOSIZIONALE MINALE

FORMULE $F ::= A \mid F \to F$

A → VARIABILE PROPOSIZIONALE

$F \to F$ → IMPLICAZIONE MATERIALE
SE ... ALLORA

$\to$ È ASSOCIATIVO A DESTRA

es. $A \to B \to A = A \to (B \to A)$

CONTESTO DI IPOTESI $\Gamma ::= \; \mid \Gamma, F$

→ SUPPONGO CHE
F VALGA

JUDGEMENT DI DERIVAZIONE LOGICA $\quad \Gamma \vdash F$

a partire dalle ipotesi Γ dimostra F

DEDUZIONE NATURALE:

$$\frac{F \in \Gamma}{\Gamma \vdash F} \qquad \frac{\Gamma \vdash F_1 \to F_2 \qquad \Gamma \vdash F_1}{\Gamma \vdash F_2}\ \text{MODUS PONENS} / \to_e \qquad \frac{\Gamma, F_1 \vdash F_2}{\Gamma \vdash F_1 \to F_2}\ \to_i$$

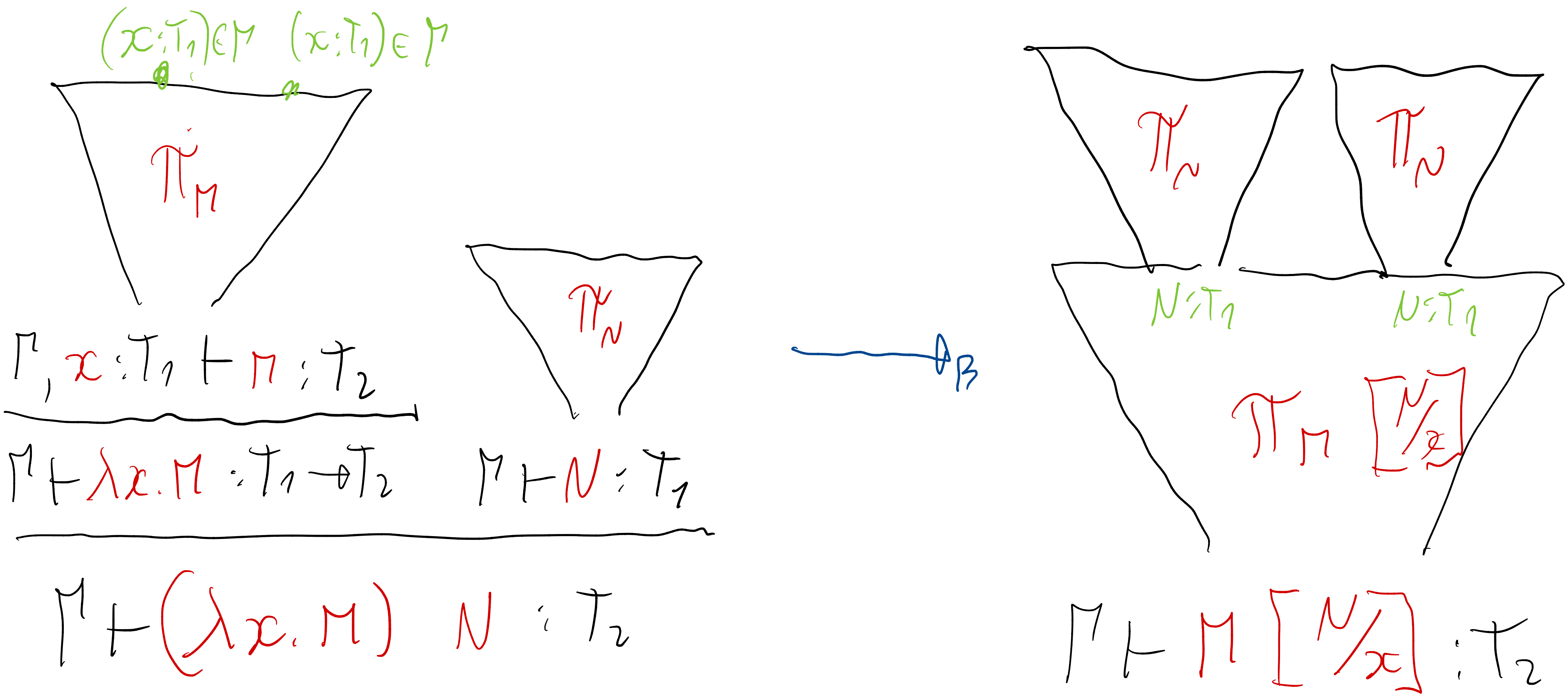
$$\dfrac{\dfrac{\dfrac{\dfrac{A \to B \to C \in \Gamma}{A \to B \to C, B, A \vdash A \to B \to C} \qquad \dfrac{A \in \Gamma}{A \to B \to C, B, A \vdash A}}{A \to B \to C, B, A \vdash B \to C} \qquad \dfrac{B \in \Gamma}{A \to B \to C, B, A \vdash B}}{\dfrac{A \to B \to C, B, A \vdash C}{A \to B \to C, B \vdash A \to C}}}{A \to B \to C \vdash B \to A \to C}$$

$$\frac{F \in \Gamma}{\Gamma \vdash F} \qquad \frac{\Gamma \vdash F_1 \to F_2 \qquad \Gamma \vdash F_1}{\Gamma \vdash F_2}\ \text{modus ponens} / \to_e \qquad \frac{\Gamma, F_1 \vdash F_2}{\Gamma \vdash F_1 \to F_2}\ \to_i$$

$$\frac{(x : T) \in \Gamma}{\Gamma \vdash x : T} \qquad \frac{\Gamma \vdash M : T_1 \to T_2 \qquad \Gamma \vdash N : T_1}{\Gamma \vdash M N : T_2} \qquad \frac{\Gamma, x : T_1 \vdash M : T_2}{\Gamma \vdash \lambda x . M : T_1 \to T_2}$$

ISOMORFISMO DI CURRY-HOWARD-KOLMOGOROV

TIPO	FORMULA
TERMINI	PROVE
COSTRUTTORE DI TIPO	CONNETTIVO
COSTRUTTORI DI TERMINI	PASSI DI PROVA
VARIABILI: LIBERE, LEGATE	IPOTESI: GLOBALI, LOCALI (SCARICATE)
TYPE CHECKING	PROOF CHECKING
TYPE INHABITATION (dato Γ, T cerca t t.c. $\Gamma \vdash t : T$)	RICERCA DI PROVE
RIDUZIONE	NORMALIZZAZIONE


$(x:T_1) \in \Gamma$
$(x:T_1) \in \Gamma$
Π_M
$\Gamma, x : T_1 \vdash M : T_2$
Π_N
$\Gamma \vdash \lambda x.M : T_1 \to T_2$
$\Gamma \vdash N : T_1$
$\Gamma \vdash (\lambda x.M)\ N : T_2$
$\to_\beta$
Π_N
Π_N
$N : T_1$
$N : T_1$
$\Pi_M [N/x]$
$\Gamma \vdash M[N/x] : T_2$

$$
\dfrac{
\dfrac{
\dfrac{(A \multimap A \multimap B) \in \Gamma}{\Gamma \vdash A \multimap A \multimap B}
\qquad
\dfrac{(A) \in \Gamma}{\Gamma \vdash A}
}{A \multimap A \multimap B,\ A \vdash A \multimap B}
\qquad
\dfrac{(A) \in \Gamma}{A \multimap A \multimap B,\ A \vdash A}
}{
\dfrac{A \multimap A \multimap B,\ A \vdash B}{A \multimap A \multimap B \vdash A \multimap B}\ \Rightarrow_i
}
$$

$$
\dfrac{
  \dfrac{
    \dfrac{(f : A \multimap A \multimap B) \in \Gamma}{\Gamma \vdash f : A \multimap A \multimap B}
    \qquad
    \dfrac{(x : A) \in \Gamma}{\Gamma \vdash x : A}
  }{
    f : A \multimap A \multimap B,\ x : A \vdash f\,x : A \multimap B
  }
  \qquad
  \dfrac{(x : A) \in \Gamma}{f : A \multimap A \multimap B,\ x : A \vdash x : A}
}{
  \dfrac{f : A \multimap A \multimap B,\ x : A \vdash f\,x\,x : B}{f : A \multimap A \multimap B \vdash \lambda x. f\,x\,x : A \multimap B}\ \Rightarrow_i
}
$$

$$\boxed{\lambda x.\, f\, x\, x}$$

$$T ::= \dots \mid T \times T$$

$$t ::= \dots \mid \langle t, t \rangle \mid t.1 \mid t.2 \mid \underline{\text{match}}\ t\ \underline{\text{with}}\ \langle x_1, x_2 \rangle \Rightarrow t$$

$$\frac{\Gamma \vdash M_1 : T_1 \qquad \Gamma \vdash M_2 : T_2}{\Gamma \vdash \langle M_1, M_2 \rangle : T_1 \times T_2}$$

$$\frac{\Gamma \vdash C : T_1 \times T_2}{\Gamma \vdash C.1 : T_1} \qquad \frac{\Gamma \vdash C : T_1 \times T_2}{\Gamma \vdash C.2 : T_2}$$

$$\frac{\Gamma \vdash C : T_1 \times T_2 \qquad \Gamma, x_1 : T_1, x_2 : T_2 \vdash M : T}{\Gamma \vdash \underline{\text{match}}\ C\ \underline{\text{with}}\ \langle x_1, x_2 \rangle \Rightarrow M : T}$$

$\underline{\text{let}}\ \langle x_1, x_2 \rangle = C\ \underline{\text{in}}\ M$

$$F ::= \dots \mid F \wedge F$$

$$\frac{\Gamma \vdash F_1 \qquad \Gamma \vdash F_2}{\Gamma \vdash F_1 \wedge F_2}\ \wedge_i$$

$$\frac{\Gamma \vdash F_1 \wedge F_2}{\Gamma \vdash F_1}\ \wedge_{e_1} \qquad \frac{\Gamma \vdash F_1 \wedge F_2}{\Gamma \vdash F_2}\ \wedge_{e_2}$$

$$\frac{\Gamma \vdash F_1 \wedge F_2 \qquad \Gamma, F_1, F_2 \vdash F}{\Gamma \vdash F}\ \wedge_e$$

$$\langle M_1, M_2 \rangle . 1 \longrightarrow M_1$$

$$\langle M_1, M_2 \rangle . 2 \longrightarrow M_2$$

$$\underline{\text{match}}\ \langle M_1, M_2 \rangle\ \underline{\text{with}}\ \langle x_1, x_2 \rangle \Rightarrow M \longrightarrow M\left[{}^{M_1}/_{x_1}\right]\left[{}^{M_2}/_{x_2}\right]$$

$$\dfrac{\dfrac{\dfrac{\Pi_{M_1}}{\Gamma \vdash M_1 : T_1} \qquad \dfrac{\Pi_{M_2}}{\Gamma \vdash M_2 : T_2}}{\Gamma \vdash \langle M_1, M_2 \rangle : T_1 \times T_2}}{\Gamma \vdash \langle M_1, M_2 \rangle . 1 : T_1} \longrightarrow \dfrac{\Pi_{M_1}}{\Gamma \vdash M_1 : T_1}$$

$$\dfrac{\dfrac{\dfrac{\Pi_{M_1}}{\Gamma \vdash M_1 : T_1} \qquad \dfrac{\Pi_{M_2}}{\Gamma \vdash M_2 : T_2}}{\Gamma \vdash \langle M_1, M_2 \rangle : T_1 \times T_2} \qquad \dfrac{\Pi_M}{\Gamma, x_1 : T_1, x_2 : T_2 \vdash M : T}}{\Gamma \vdash \underline{\text{match}}\ \langle M_1, M_2 \rangle\ \text{with}\ \langle x_1, x_2 \rangle \Rightarrow M : T} \quad \rightsquigarrow \quad \dfrac{\Pi_{M_1} \quad \Pi_{M_2} \quad \Pi_M[M_1/x_1][M_2/x_1]}{\Gamma \vdash M[M_1/x_1][M_2/x_1] : T}$$

Unit

$$T ::= \dots \mid \mathbb{1} \qquad F ::= \dots \mid \top$$

()

$$t ::= \dots \mid () \mid \texttt{let}\ () = t\ \texttt{in}\ t$$

$$\frac{}{\Gamma \vdash () : \mathbb{1}} \qquad \frac{}{\Gamma \vdash \top}\ \top_i$$

$$\frac{\Gamma \vdash M : \mathbb{1} \qquad \Gamma \vdash N : T}{\Gamma \vdash \texttt{match}\ M\ \texttt{with}\ () \Rightarrow N : T} \qquad \frac{\Gamma \vdash \top \qquad \Gamma \vdash F}{\Gamma \vdash F}\ \top_e$$

$\texttt{let}\ () = M\ \texttt{in}\ N$

$$\dfrac{\overline{\Gamma \vdash () : \mathbb{1}} \qquad \begin{array}{c}\Pi_N \\ \vdots \\ \Gamma \vdash N : \tau\end{array}}{\Gamma \vdash \underline{\mathrm{let}}\ () = ()\ \underline{\mathrm{in}}\ N : \tau} \quad \rightsquigarrow \quad \begin{array}{c}\Pi_N \\ \vdots \\ \Gamma \vdash N : \tau\end{array}$$

EMPTY

$T ::= \ldots \mid 0$

$t ::= \ldots \mid \mathrm{abort}(t)$

$$\frac{\Gamma \vdash M : \bot}{\Gamma \vdash \mathrm{abort}(M) : T}$$

match M with

$F ::= \ldots \mid \bot$

$$\frac{\Gamma \vdash \bot}{\Gamma \vdash F}$$

$$T ::= \ldots \mid T_1 + T_2$$

$$t ::= \ldots \mid \underline{L}(t) \mid \underline{R}(t) \mid \underline{\text{match}} \ldots$$

$$\frac{\Gamma \vdash M_1 : T_1}{\Gamma \vdash \underline{L}(M_1) : T_1 + T_2} \qquad \frac{\Gamma \vdash M_2 : T_2}{\Gamma \vdash \underline{R}(M_2) : T_2}$$

$$\frac{\Gamma \vdash M : T_1 + T_2 \quad \Gamma, x_1 : T_1 \vdash N_1 : T \quad \Gamma, x_2 : T_2 \vdash N_2 : T}{\Gamma \vdash \underline{\text{match}}\ M\ \underline{\text{with}}\ \underline{L}(x_1) \Rightarrow N_1 \mid \underline{R}(x_2) \Rightarrow N_2 : T}$$

$$F ::= \ldots \mid F_1 \vee F_2$$

$$\frac{\Gamma \vdash F_1}{\Gamma \vdash F_1 \vee F_2}\,\vee_{i1} \qquad \frac{\Gamma \vdash F_2}{\Gamma \vdash F_1 \vee F_2}\,\vee_{i2}$$

$$\frac{\Gamma \vdash F_1 \vee F_2 \quad \Gamma, F_1 \vdash F \quad \Gamma, F_2 \vdash F}{\Gamma \vdash F}\,\vee_e$$

TEOREMA (CONSISTENZA DELLA LOGICA PROPOSIZIONALE):

$$\nvdash \bot$$

DIMOSTRAZIONE:

ASSUMO $\vdash \bot$

PER CURRY-HOWARD-KOLMOGOROV $\exists M. \vdash M : \bot$

SIA M T.C. $\vdash M : \bot$

SIA N LA FORMA NORMALE DI M CHE ESISTE PER IL TEOREMA DI NORMALIZZAZIONE FORTE

SI HA $\vdash N : \bot$

CHE FORMA HA N?

— N <u>NON</u> È UNA REGOLA DI INTRODUZIONE

(il + non ne ha)

— N NON È UNA VARIABILE

— N SIA UNA REGOLA DI ELIMINAZIONE:

es. N = <u>match</u> M <u>with</u>

$\equiv$

COME È FATTO M ?

— M NON È UNA VARIABILE

— M NON È UNA REGOLA DI INTRODUZIONE

PERCHÈ N È NORMALE

— M SIA UNA REGOLA DI ELIMINAZIONE

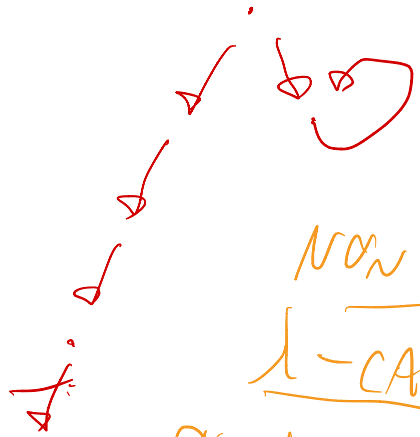
LOOP!

ASSURDO MANCA IL CASO BASE

QUINDI $\nvdash \bot$

QED.

DEF: un termine t si dice (debolmente) normalizzante quando ha una forma normale



NON NEL λ-CALCOLO QUESTO DISEGNO

DEF: un termine t si dice fortemente normalizzante se $\nexists (t_i)_{i \in \mathbb{N}}$.
$t = t_0 \wedge \forall i,\ t_i \to_\beta t_{i+1}$

DEF.: un λ-calcolo si dice avere una proprietà Q se ogni termine ce l'ha

TEOREMA DI NORMALIZZAZIONE FORTE:

$$\forall \Gamma, M, T. \quad \underbrace{\Gamma \vdash M : T}_{\text{APPROSSIMAZIONE DA DENTRO}} \Longrightarrow \underbrace{M \text{ è fortemente normaliz.}}_{\text{Q INDECIDIBILE}}$$

OSSERVAZIONE: $\lambda x.xx$ NON È TIPABILE,

MA $\lambda x.xx$ È FORTEMENTE NORMALIZZANTE

PROVA ERRATA PER INDUZIONE:

procediamo per induzione sull'albero di prova $\Gamma \vdash M : T$

caso $\dfrac{(x:T) \in \Gamma}{\Gamma \vdash x : T}$:

devo dimostrare x è fortemente normalizzante

OVVIO

caso $\dfrac{\Gamma, x:T_1 \vdash M : T_2}{\Gamma \vdash \lambda x.M : T_1 \to T_2}$

per ipotesi induttiva: M è forte. normaliz. (I) devo dimostrare $\lambda x.M$ è fort. norm.

per assurdo, supponiamo $\lambda x.M$ non lo sia, ovvero

$$M \to_\beta M_1 \to_\beta M_2 \to_\beta M_3 \to \ldots$$

$$\lambda x.M \to_\beta \lambda x.M_1 \to_\beta \lambda x.M_2 \to_\beta \lambda x.M_3 \to \ldots$$

assurdo poiché M è fort. norm.

caso $\dfrac{\boxed{\Gamma \vdash M : T_1 \to T_2} \qquad \boxed{\Gamma \vdash N : T_1}}{\Gamma \vdash MN : T_2}$

per ipotesi induttiva M è fort. norm. (I_1) e
N è fort. norm. (I_2)

devo dimostrare che MN è fort. norm.

OSSERVAZIONE: M PUÒ ESSERE DELLA FORMA $\lambda x.t$ (con t fort. norm.), MA $MN = (\lambda x.t)N \to_\beta t[N/x]$

CHI MI GARANTISCE CHE $N[t/x]$ SIA FORT. NORM.?

XXX NESSUNO: $(\lambda x.xx)(\lambda x.xx)$ DIVERGE MA $\lambda x.xx$ è FORT. NORM. XXX

Conclusioni:

— Il tipaggio è modulare

— La norm. forte non lo è

— Deve esserci una proprietà intermedia modulare che approssimi meglio la normalità forte

$$SN \overset{def}{=} \{ t \,/\, t \text{ è fort. norm.} \}$$

$$WT \overset{def}{=} \{ t \,/\, \exists \Gamma, T.\ \Gamma \vdash t : T \}$$

RED_T = "insieme dei termini RIDUCIBILI di tipo T"

$\bigcirc = \bigcup_{T \text{ tipo}} RED_T$

PIANO DEI LAVORI:

1. TROVARE UNA BUONA DEFINIZIONE DI RED_T

2. DIMOSTRARE $RED_T \subseteq SN$

3. DIMOSTRARE $WT \subseteq RED_T$

DEF. DI RED_T: per ricorsione su T

$$RED_A \stackrel{def}{=} \{ M \;/\; \exists \Gamma.\ \Gamma \vdash M : A \ \wedge\ M \in SN \}$$

$$RED_{T_1 \to T_2} \stackrel{def}{=} \{ M \;/\; \exists \Gamma.\ \Gamma \vdash M : T_1 \to T_2 \ \wedge$$

$$\cancel{M \in SN\ \wedge} \quad \leftarrow \text{RIDONDANTE}$$

$$\forall N \in RED_{T_1}.\ MN \in RED_{T_2} \}$$

Teorema (tentativo): $\forall T \subseteq RED$

ovvero $\forall \Gamma, M, T, \quad \Gamma \vdash M : T \implies M \in RED_T$

per induzione sull'albero $\Gamma \vdash M : T$

caso
$$\frac{\boxed{\Gamma \vdash M : T_1 \to T_2} \qquad \boxed{\Gamma \vdash N : T_1}}{\Gamma \vdash MN : T_2}$$

per ipotesi induttiva $M \in RED_{T_1 \to T_2}$ (I_1)

e $N \in RED_{T_1}$ (I_2)

devo dimostrare $MN \in RED_{T_2}$

OVVIO PER I_1, I_2 E LA DEFINIZIONE DI $RED_{T_1 \to T_2}$

caso $\dfrac{(x:T) \in \Gamma}{\Gamma \vdash x:T}$

devo dimostrare che $x \in RED_T$

OVVIO PER IL LEMMA CR4

caso $\dfrac{\boxed{\Gamma, x:T_1 \vdash M:T_2}}{\Gamma \vdash \lambda x.M : T_1 \to T_2}$

per ipotesi induttiva $M \in RED_{T_2}$

devo dimostrare $\lambda x.M \in RED_{T_1 \to T_2}$

ovvero $\exists \Gamma. \Gamma \vdash \lambda x.M : T_1 \to T_2$ ovvio $\wedge$

$\forall N \in RED_{T_1}. (\lambda x.M)N \in RED_{T_2}$?

Mi serve:

✓ • C3 (caso particolare): $(\lambda x.M)N \to t \in RED_{T_2} \Rightarrow$

$(\lambda x.M)N \in RED_{T_2}$

XXX • Mi servirebbe $M \in RED_{T_2} \Rightarrow M[N/x] \in RED_{T_2}$ XXX

Nuovo piano dei lavori: Proof & Types: 4.4, 6.1 – 6.3

1. Definiamo RED_T
2. Identifichiamo CR1 – CR3 le proprietà dei candidati di riducibilità
3. Dimostriamo CR1 – CR3 per RED_T
4. Generalizziamo l'enunciato $WT \subseteq RED_T$ e lo dimostriamo usando CR1 – CR3
5. Dimostriamo $RED_T \subseteq SN$ usando CR1

- $CR1(T)$: $RED_T \subseteq SN$
- $CR2(T)$: $\forall M, N.\quad M \in RED_T \land M \to_\beta^* N \Rightarrow N \in RED_T$
- $CR3(T)$: $\forall M\ (M$ neutrale $\land (\forall N.\ M \to_\beta N \Rightarrow N \in RED_T)$
  $\Rightarrow M \in RED_T)$
- CR4 COROLLARIO DI CR3: $\forall T.\ x \in RED_T$ OVVIO PER CR3

Def: un termine M è neutrale quando
i redex di $N[M/x]$ sono redex di M
o di N (non ne ha creati di nuovi)

TEOREMA: M è neutrale se M non è una λ-astrazione

TEOREMA: $\forall T.\ CR1(T) \wedge CR2(T) \wedge CR3(T)$

DIMOSTRAZIONE: per induzione (mutua) su T

Caso A:

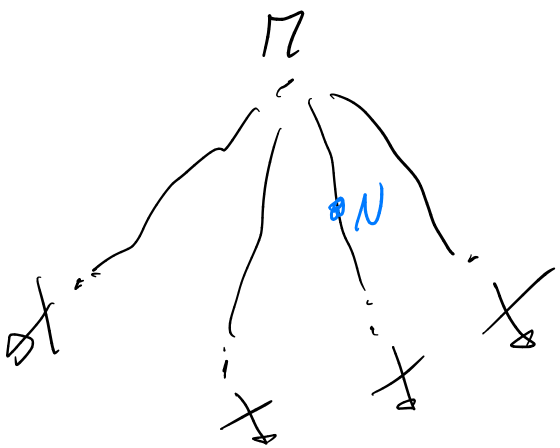
devo dimostrare

- $CR1(A)$: $RED_A \subseteq SN$ ovvero

  $\{M / \exists \Gamma.\ \Gamma \vdash M : A \wedge M \in SN\} \subseteq SN$ <u>OVVIO</u>

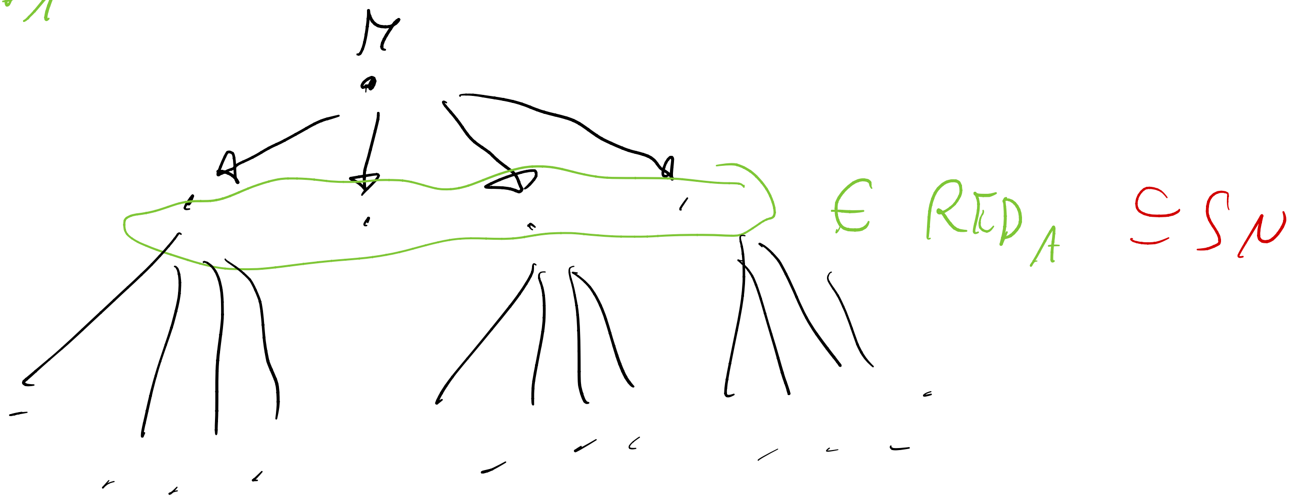
- $CR2(A)$: $\forall M, N.\ M \in RED_A \wedge M \to_\beta^* N \Rightarrow N \in RED_A$

  <u>OVVIO</u>



se ci fosse un cammino infinito da N ci sarebbe anche da M

- $CR3(A)$: $\forall M.\ M$ neutrale $\land (\forall N.\ M \to_\beta N \Rightarrow N \in RED_A) \to M \in RED_A$



OVVIO

se M avesse un cammino infinito, uno degli N t.c. $M \to_\beta N$ avrebbe un cammino infinito caso $T_1 \to T_2$:

per ipotesi induttiva, $CR1(T_1) \wedge CR2(T_1) \wedge CR3(T_1)$

e $CR1(T_2) \wedge CR2(T_2) \wedge CR3(T_2)$

dobbiamo dimostrare

- $CR1(T_1 \to T_2)$: $RED_{T_1 \to T_2} \subseteq SN$ ovvero

$$\{M \mid \exists \Gamma,\ \Gamma \vdash M : T_1 \to T_2 \wedge \forall N \in RED_{T_1}.\ MN \in RED_{T_2}\} \subseteq SN$$

poiché per ipotesi induttiva vale $CR3(T_1)$ allora vale $CR4(T_1)$ ovvero $x \in RED_{T_1}$

fisso M t.c. $\exists \Gamma$, $\Gamma \vdash M : T_1 \to T_2$ e

$\underbrace{\forall N \in RED_{T_1} . \; MN \in RED_{T_2}}_{H}$ e dimostro $M \in SN$

da H e da $x \in RED_{T_1}$ ho $Mx \in RED_{T_2}$

per ipotesi induttiva $CR_1(T_2)$, $Mx \in SN$

quindi $M \in SN$ (ovvio)

se $M \to M_1 \to M_2 \to \ldots$

allora $Mx \to M_1 x \to M_2 x \to \ldots$

impossibile

- $CR2(T_1 \to T_2)$: $\forall M, N.\ M \in RED_{T_1 \to T_2} \land M \to_\beta^* N \Rightarrow N \in RED_{T_1 \to T_2}$

fisso M, N t.c. $M \in RED_{T_1 \to T_2}$ (H_1) e $M \to_\beta^* N$ (H_2)

dimostro $N \in RED_{T_1 \to T_2} = \{ U \,/\, \exists \Gamma.\ \Gamma \vdash U : T_1 \to T_2 \land$
$\forall W \in RED_{T_1}.\ NW \in RED_{T_2} \}$

ovvero

1. dimostro $\exists \Gamma.\ \Gamma \vdash N : T_1 \to T_2$. Da H_1 ho $\exists \Gamma.\ \Gamma \vdash M : T_1 \to T_2$
QUINDI DA H_2 E SUBJECT REDUCTION
OVVIO

2. dimostro $\forall W \in RED_{T_1}.\ NW \in RED_{T_2}$

fisso $W \in RED_{T_1}$ e dimostro $NW \in RED_{T_2}$

ESEMPIO: $\lambda x.M$ NON È NEUTRALE POICHÉ

$$(z\,y)\left[{}^{\lambda x.M}/_{z}\right] = \underbrace{(\lambda x.M)\,y}_{\text{REDEX}}$$

MA $(\lambda x.M)\,y \notin \lambda x.M$ E $(\lambda x.M)\,y \notin (z\,y)$

ESEMPIO: $(M\ N)$ È NEUTRALE POICHÉ

SE $R\left[{}^{MN}/_{x}\right]$ CONTIENE UN REDEX DELLA FORMA $(\lambda z.U)\,W$ ALLORA NON PUÒ ESSERE STATO CREATO

$\cancel{(\lambda z.U)\,x}$ $\qquad$ $\cancel{x/R}$

per ipotesi induttiva, vale CR2(T_2) ovvero

$\forall V, V' \in RED_{T_2}.\ V \to V' \Rightarrow V' \in RED_{T_2}$

e per H_1, $MV \in RED_{T_2}$

poiché $M \to_\beta^* N$ per H_2 si ha $MV \to_\beta^* NV$

QUINDI $NV \in RED_{T_2}$

- CR3($T_1 \to T_2$): $\forall M.\ M$ neutrale $\land$ $(\forall N.\ M \to_\beta N \Rightarrow N \in RED_{T_1 \to T_2})$

$\Rightarrow M \in RED_{T_1 \to T_2}$

fisso M t.c. M neutrale (H_1) e $\forall N.\ M \to_\beta N \Rightarrow N \in RED_{T_1 \to T_2}$ (H_2)

dimostro $M \in RED_{T_1 \to T_2}$ ovvero

1. dimostro $\exists M.\ \Gamma \vdash M : T_1 \multimap T_2$

TODO: ???

[es. $(\lambda x.y)(\lambda z.zz)$ È NEUTRALE E NON È TIPATO

MA $(\lambda x.y)(\lambda z.zz) \to_\beta y$ CHE È BEN TIPATO]

2. dimostro $\forall U.\ U \in RED_{T_1} \Rightarrow MU \in RED_{T_2}$

fisso U t.c. $U \in RED_{T_1}$ (H3) e dimostro $MU \in RED_{T_2}$

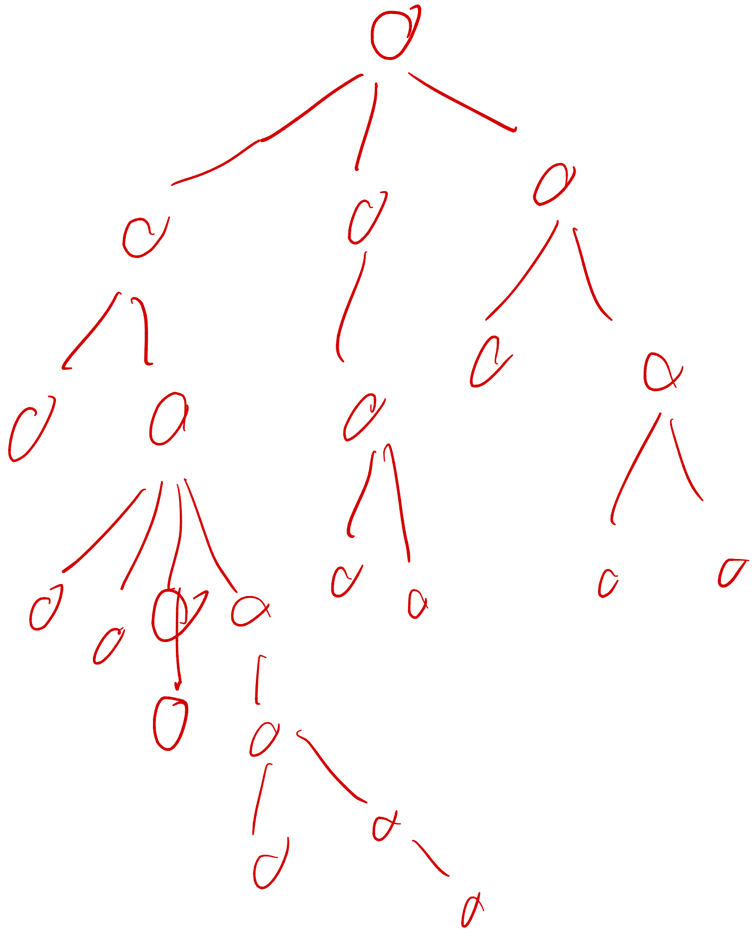
procedo per induzione su $\nu(U)$ per dimostrare:

$MU \in RED_{T_2}$

CONSIDERIAMO UN ALBERO FINITELY BRANCHING T

(= UN NODO HA UN NUMERO FINITO DI FIGLI)

E SENZA RAMI INFINITI



USANDO L'ASSIOMA DELLA SCELTA

VALE

$\exists\, \nu(T) \in \mathbb{N}$.

NESSUN RAMO

È PIÙ LUNGO DI $\nu(T)$

caso base: $\nu(\cancel{U})$ è 0 ovvero ~~U~~ ~~fo~~

PER $CR_3(T_2)$ MI RIDUCO A DIMOSTRARE $MU \to V \Rightarrow V \in RED_{T_2}$

SIA V T.C. $MU \to_\beta V$. CI SONO DUE POSSIBILITÀ:

1. $$\frac{M \to_\beta M'}{MU \to_\beta M'U = V}$$ PER H_2, $M' \in RED_{T_1 \to T_2}$

   QUINDI $M'U \in RED_{T_2}$ QUINDI $V \in RED_{T_2}$

2. $MU \to_\beta V$ IN QUANTO M È UNA λ-ASTRAZIONE

   IMPOSSIBILE PER H_1

caso induttivo: $\nu(U) = n+1$ E SIA $U \to_\beta U'$ T.C. $\nu(U') = n$

PER IPOTESI INDUTTIVA, $MU' \in RED_{T_2}$ (II)

devo dimostrare $MU \in RED_{T_2}$

PER $CR_3(T_2)$ MI RIDUCO A DIMOSTRARE $MU \to V \Rightarrow V \in RED_{T_2}$

CI SONO 3 CASI: primi due identici a prima

TERZO CASO:
$$\frac{U \to_\beta U'}{MU \to_\beta MU'}$$
PER I.I., $MU' \in RED_{T_2}$

Q.E.D.

TEOREMA: $\forall \Gamma, M, T.$ DATO $\{ N_i \mid (x_i : T_i) \in \Gamma,\ N_i \in RED_{T_i} \}$,

$$\Gamma \vdash M : T \Rightarrow M\left[\vec{N_i} / \vec{x_i} \right] \in RED_T$$

COROLLARIO: $\forall \Gamma, M, T.\quad \Gamma \vdash M : T \Rightarrow M \in RED_T$

DIMOSTRAZIONE DEL COROLLARIO:

OVVIO PER CR4 SCEGLIENDO $\forall i.\quad N_i = x_i \in RED_{T_i}$

DIMOSTRAZIONE DEL TEOREMA PER INDUZIONE SU $\Gamma \vdash M : T$.

CASO $\dfrac{(x_j : T_j) \in \Gamma}{\Gamma \vdash x_j : T_j}$

DIMOSTRO $x_j\left[\vec{N_i} / \vec{x_i}\right] \in RED_{T_j}$

$$\|$$

$$N_j$$

OVVIO PER LA PREMESSA CHE ASSICURA

CHE OGNI $N_i \in RED_{T_i}$

CASO $\dfrac{\boxed{\Gamma \vdash M : T_1 \to T_2} \quad \boxed{\Gamma \vdash N : T_1}}{\Gamma \vdash MN : T_2}$

PER IPOTESI INDUTTIVE: $M[\vec{N}/\vec{x}] \in RED_{T_1 \to T_2}$ (II_1)

E $N[\vec{N}/\vec{x}] \in RED_{T_1}$ (II_2)

DOBBIAMO DIMOSTRARE $(MN)[\vec{N}/\vec{x}] \in RED_{T_2}$

$$\| $$

$$M[\vec{N}/\vec{x}] \; N[\vec{N}/\vec{x}]$$

OVVIO PER II_1, II_2 E DEFINIZIONE DI $RED_{T_1 \to T_2}$

CASO $\dfrac{\boxed{\Gamma, x_{n+1} : T_{n+1} \vdash M : T}}{\Gamma \vdash \lambda x_{n+1}. M : T_{n+1} \to T}$ DOVE $n = |\Gamma|$

PER IPOTESI INDUTTIVA $M\left[\vec{N_i}/\vec{x_i}\right] \in RED_T$ (II)

$\forall N_{n+1} \in RED_{T_{n+1}}$

DEVO DIMOSTRARE $(\lambda x_{n+1}. M)\left[\vec{N_i}/\vec{x_i}\right] \in RED_{T_{n+1} \to T}$

OVVERO

1) $\exists \Gamma. \Gamma \vdash (\lambda x_{n+1}. M)\left[\vec{N_i}/\vec{x_i}\right] : T_{n+1} \to T$ OVVIO PER L'IPOTESI

$\Gamma \vdash \lambda x_{n+1}. M : T_{n+1} \to T$, PER IL FATTO CHE $N_i : T_i$ E PER UN LEMMA

2. $\forall N_{n+1} \in RED_{T_{n+1}} \quad (\lambda x_{n+1}.\, M)[\tilde{N}/\tilde{x}]\, N_{n+1} \in RED_{T_{n+1}}$

FISSO $N_{n+1} \in RED_{T_{n+1}}$ (H)

SI SFRUTTA CR3 POICHÉ $(\lambda x_{n+1}.\, M)[\tilde{N}/\tilde{x}]\, N_{n+1}$ NEUTRALE

POICHÉ $N_{n+1} \in RED_{T_{n+1}}$ PER (H) E

POICHÉ $M[\tilde{N}/\tilde{x}] \equiv M[\tilde{N}/\tilde{x};\, x_{n+1}/x_{n+1}] \in RED_{T_{n+1}}$

PER (II)

ALLORA $\exists\ \nu(N_{n+1})$ e $\nu(M[\tilde{N}/\tilde{x}])$ DEFINITI

CON LA VOLTA SCORSA.

PROCEDO PER INDUZIONE SU $\nu(N_{n+1}) + \nu(M[\vec{N_i}/\vec{x_i}])$

PER DIMOSTRARE $(\lambda x_{n+1}.M)[\vec{N_i}/\vec{x_i}]\, N_{n+1} \to^* U \in RED_T$ PER OGNI $M, \vec{N_i}, N_{n+1}$

(E CONCLUDERE QUINDI PER CR3)

CASO

$$\frac{M[\vec{N_i}/\vec{x_i}] \to U'}{(\lambda x_{n+1}.M)[\vec{N_i}/\vec{x_i}]\, N_{n+1} \to (\lambda x_{n+1}.U')\, N_{n+1}}\ \beta$$

POICHÈ $M[\vec{N_i}/\vec{x_i}] \to_\beta U'$ SI HA

$$\nu(M[\vec{N_i}/\vec{x_i}]) = \nu(U')+1$$

E CONCLUDO PER IPOTESI INDUTTIVA

- CASO
$$\frac{N_{n+1} \to_\beta W}{(\lambda x_{n+1}.\, M[\vec{N_i}/\vec{x_i}])\, N_{n+1} \to_\beta (\lambda x_{n+1}.\, M)[\vec{N_i}/\vec{x_i}]\, W}$$

ANALOGO AL PRECEDENTE, $\nu(N_{n+1}) = 1 + \nu(W)$

E QUINDI CONCLUDO PER IPOTESI INDUTTIVA

CASO
$$\frac{}{(\lambda x_{n+1}.\, M)[\vec{N_i}/\vec{x_i}]\, N_{n+1} \to_\beta M[\vec{N_i}/\vec{x_i}, N_{n+1}/x_{n+1}]}$$

VERA PER II

QED.

NEI LINGUAGGI DI PROGRAMMAZIONE VI È COSTRUTTO ESPLICITO DI RICORSIONE:

$$f : T := M$$

FUNZIONE DICHIARATA AL TOPLEVEL AVENTE TIPO T E CORPO M

È ZUCCHERO SINTATTICO PER IL COSTRUTTO

$$f := (\overbrace{\nu f : T . M}^{\text{FUNZIONE RICORSIVA ANONIMA}})$$

NOME TOP-LEVEL DI UN TERMINE

BINDER DI PUNTO FISSO

TERMINE CHE DICHIARA UNA FUNZIONE RICORSIVA DI TIPO T CHE NEL CORPO PUÒ RICHIAMARSI USANDO IL NOME f

$$\frac{\Gamma, f:T \vdash M:T}{\Gamma \vdash (\nu f:T.\, M):T}$$

QUESTA REGOLA È LA CAUSA DELLA POSSIBILITÀ DI DIVERGERE!

Es.

$$\dfrac{\dfrac{\dfrac{f:N\to N, x:N \vdash f:N\to N \qquad f:N\to N, x:N \vdash x:N}{f:N\to N, x:N \vdash f\,x:N}}{f:N\to N \vdash \lambda x:N.\, f\,x : N\to N}}{\vdash (\nu f:N\to N.\, \lambda x:N.\, f\,x):N\to N}$$

TALE REGOLA IMPLICA LA <u>NON</u> CONSISTENZA DEL SISTEMA LOGICO:

$$\frac{\vdash Y f : T \to \bot .\ \lambda x : T.\ f\,x : T \to \bot \qquad \vdash I : T}{\vdash \underbrace{(Y f : T \to \bot .\ \lambda x : T.\ f\,x)}_{T \to \bot}\ \underbrace{I}_{T} : \bot}$$

COROLLARIO DELLE OSSERVAZIONI PRECEDENTI:

$$Y = \lambda f.\ (\lambda x.\ f\,(x\,x))\,(\lambda x.\ f\,(x\,x))$$

NON È TIPABILE

LOGICA PROPOSIZIONALE DEL SECONDO ORDINE:

$$F ::= \ldots \mid \forall A. F$$

VARIABILE PROPOSIZIONALE

es. $\forall A, B, C. \; (A \land B \to C) \to \neg C \to \neg(A \land B)$

[INVECE NELLA LOGICA DEL PRIM'ORDINE

$$F ::= \ldots \mid \forall x. F \mid P^n(x_1, \ldots, x_n)$$

PREDICATO

es. ESSERE PARI, ESSERE NATURALI

VARIABILE DI TERMINE, ELEMENTO DEL DOMINIO DEL DISCORSO

es. x UN NUMERO, UN INSIEME, UNA PERSONA, ...]

es. $\forall x. \; x \leq x$

VARIABILE FRESCA, NON USATA IN Γ

$$\frac{\Gamma \vdash F[B/A]}{\Gamma \vdash \forall A.F}\ \forall_i \qquad\qquad \frac{\Gamma \vdash \forall A.F}{\Gamma \vdash F[G/A]}\ \forall_e$$

es.

$$\dfrac{\dfrac{\dfrac{(\forall A.(A\to B)) \in \forall A.(A\to B)}{\forall A.(A\to B) \vdash \forall A.(A\to B)}}{\forall A.(A\to B) \vdash (D\to D)\to B}\ \forall_e}{\forall A.(A\to B) \vdash \forall C.(C\to C)\to B}\ \forall_i$$

Polimorfismo Uniforme o Generico o Template:

$$T ::= \ldots \mid \forall A.T$$

$\langle A \rangle T$ ← Sintassi di molti linguaggi di programmazione

$$\frac{\Gamma \vdash M : T[B/A]}{\Gamma \vdash M : \forall A.T}\ \forall_i \qquad \frac{\Gamma \vdash M : \forall A.T}{\Gamma \vdash M : T[T'/A]}\ \forall_e$$

$M\langle T' \rangle$

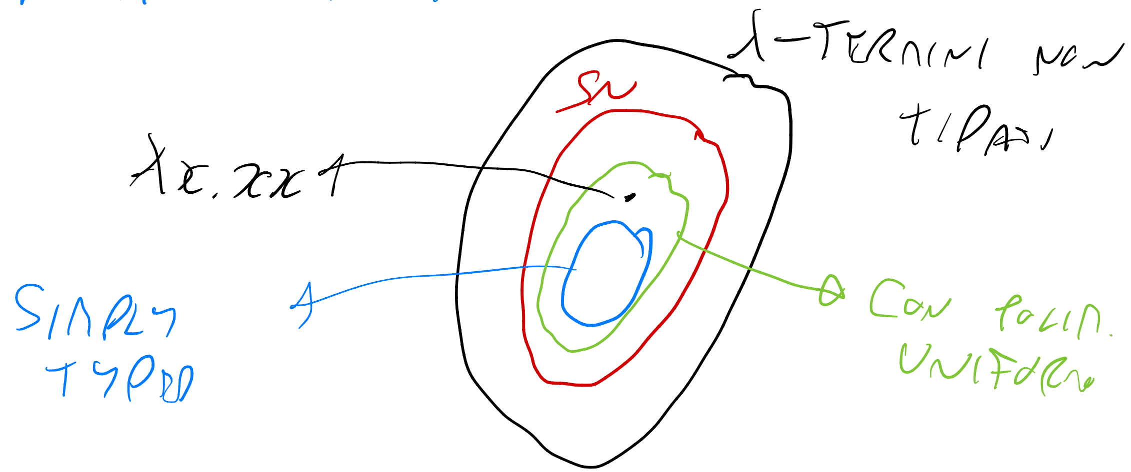
$\lambda x.xx$ È NON TIPATO NEL λ-CALCOLO TIPATO SEMPLICE

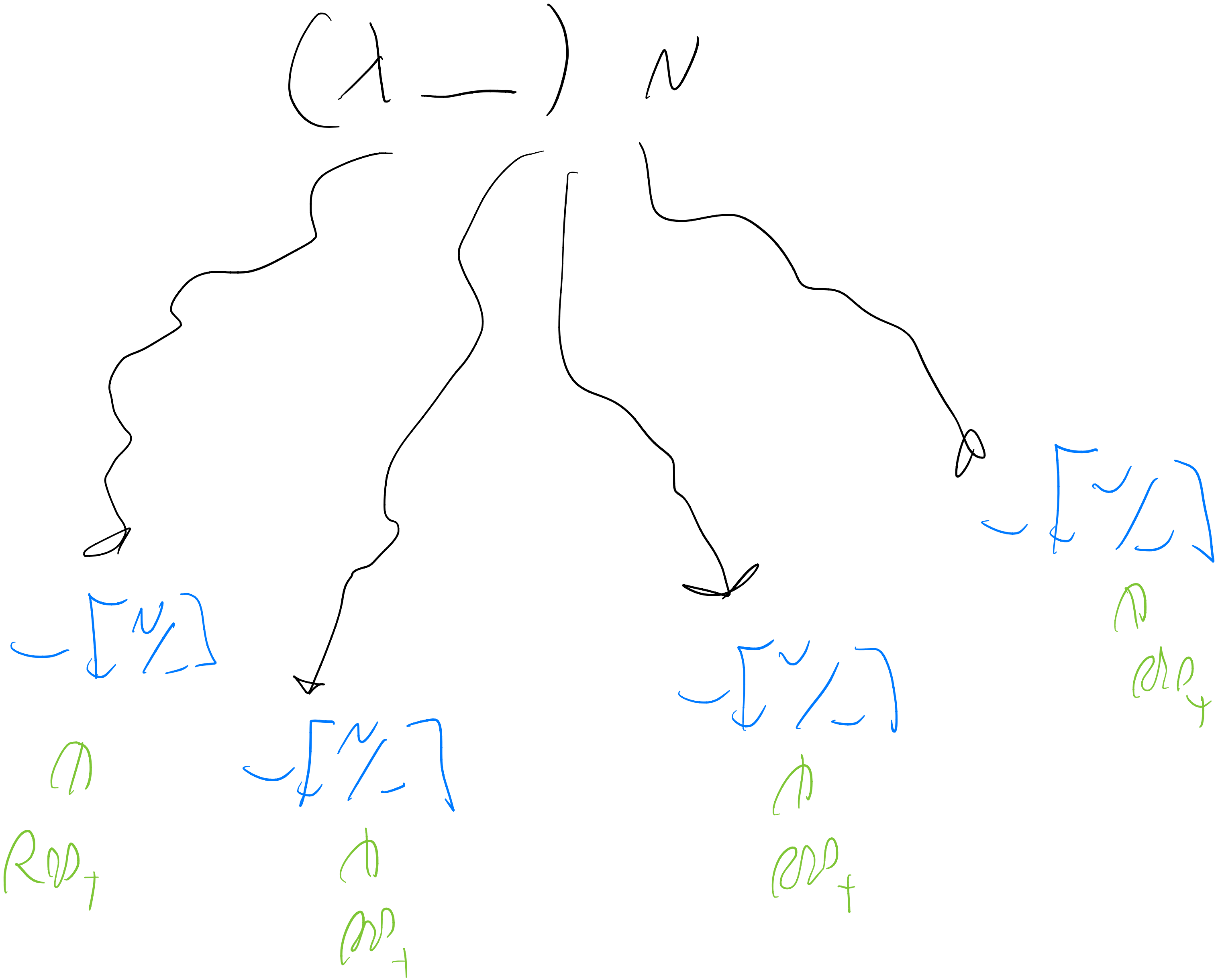
$\lambda x.xx$ È TIPATO NEL λ-CALCOLO CON POLIMORFISMO UNIFORME

$$
\dfrac{
\dfrac{
\dfrac{(x : \forall A.A \to A) \in \Gamma}{x : \forall A.A \to A \vdash x : \forall A.A \to A}
}{x : \forall A.A \to A \vdash x : (\forall A.A \to A) \to (\forall A.A \to A)}\,\forall_e
\qquad
\dfrac{(x : \forall A.A \to A) \in \Gamma}{x : \forall A.A \to A \vdash x : \forall A.A \to A}
}{
\dfrac{x : \forall A.A \to A \vdash xx : \forall A.A \to A}{\vdash \lambda x.xx : (\forall A.A \to A) \to (\forall A.A \to A)}
}
$$

1. il λ-calcolo con polimorfismo uniforme è un'approssimazione migliore della proprietà di normalizzazione forte

2. $\lambda x.xx$ mostra che non è sempre possibile MONOMORFIZZARE programmi che usano il polimorfismo $\Rightarrow$ ABBIAMO INCREMENTATO LA POTENZA ESPRESSIVA





$F ::= \dots \mid \exists A.F$ QUANTIFICATORE ESISTENZIALE AL SECONDO ORDINE

A: VARIABILE PROPOSIZIONALE

$$\dfrac{\Gamma \vdash F[G/A]}{\Gamma \vdash \exists A.F}\ \exists_i \qquad \dfrac{\Gamma \vdash \exists A.F \qquad \Gamma, F[B/A] \vdash G}{\Gamma \vdash G}\ \exists_e$$

B: FRESCA, NON USATA IN Γ E G

$$\dfrac{\dfrac{\dfrac{\dots, E \to B, E \wedge D \vdash E \to B}{} \qquad \dfrac{\dfrac{}{\dots, E\to B, E \wedge D \vdash E \wedge D}}{\dots, E \to B, E \wedge D \vdash E}\ \wedge_{e_1}}{\dots, E \to B, E \wedge D \vdash B}}{\dots, E \to B \vdash E \wedge D \to B}$$

$$\dfrac{\dfrac{}{\exists A.(A \to B) \vdash \exists A.(A \to B)} \qquad \dfrac{\dots, E \to B \vdash E \wedge D \to B}{\exists A.(A \to B), \underline{E \to B} \vdash \exists C.(C \wedge D \to B)}\ \exists_i}{\exists A.(A \to B) \vdash \exists C.(C \wedge D \to B)}\ \exists_e$$

$T ::= \ldots \mid \exists A.T$ INTERFACCIA / TIPO DI DATO ASTRATTO / CLASSE / MIXIN / MODULO / TRAIT / ...

$t ::= \ldots \mid \text{open } t \text{ as } x \text{ in } t$

$$\frac{\Gamma \vdash M : F[G/A]}{\Gamma \vdash M : \exists A.F}$$

MODULE INSTANCE
TYPE A = G
M : F[G/A]
END

$$\frac{\overbrace{\Gamma \vdash M : \exists A.F}^{\text{IMPLEMENTO M SENZA CONOSCERE N}} \qquad \overbrace{\Gamma, f : F[B/A] \vdash N : G}^{\text{IMPLEMENTO N SENZA CONOSCERE M}}}{\underbrace{\Gamma \vdash \text{open } M \text{ as } f \text{ in } N : G}_{\text{FA IL LAVORO DEL LINKER}}}$$

REGOLA DI RIDUZIONE:

IMPLEMENTAZIONE DEL MODULO

USO IL MODULO

$$\dfrac{\dfrac{\Gamma \vdash M : F[T/A]}{\Gamma \vdash M : \exists A.F}\,\exists_i \qquad \Gamma, x : F[B/A] \vdash N : G}{\Gamma \vdash \text{open } M \text{ as } x \text{ in } N : G}\,\exists_e \longrightarrow \Gamma \vdash N[M/x] : G$$

LINKER

CODICE EQUIVALENTE SENZA USARE MODULI

TIPO DI DATO ASTRATTO

È un tipo di dato del quale non viene data l'implementazione, ma solo l'INTERFACCIA come INSIEME DI SEGNATURE DI FUNZIONI

ES: STACK DI INTERI COME TIPO DI DATO ASTRATTO

MODULE STACK

TYPE STACK // ESISTE IL TIPO DEGLI STACK

FUN EMPTY : STACK

FUN PUSH : $STACK \times \mathbb{Z} \to STACK$

FUN POP : $STACK \to \mathbb{1} + \mathbb{Z} \times STACK$

END

OPEN STACK

$(\lambda x. \lambda s.\ PUSH \langle x, s \rangle)\ 2\ EMPTY$

MODULE INSTANCE STACK

TYPE STACK $= \text{ARRAY}\langle \mathbb{Z} \rangle \times \mathbb{N}$

FUN EMPTY $= \langle [\,], 0 \rangle$

FUN PUSH $\langle x, s \rangle = \langle s.1[s.2 \leftarrow x], s.2+1 \rangle$

...

END

$\exists\, \text{STACK}.\ \text{STACK} \times (\text{STACK} \times \mathbb{Z} \to \text{STACK}) \times (\text{STACK} \to 1 + \mathbb{Z} \times \text{STACK})$

OPEN M AS f IN .... $\underset{\text{EMPTY}}{\underset{\uparrow}{f.1}}$ ... $\underset{\text{PUSH}}{\underset{\uparrow}{f.2.1}}$ ... $\underset{\text{POP}}{\underset{\uparrow}{f.2.2}}$ ...

Un ABSTRACT REWRITING SYSTEM (ARS) è

una coppia $(A, \to)$ t.c.

1. $A \neq \emptyset$ INSIEME DEGLI STATI

2. $\to \subseteq A \times A$ RELAZIONE DI TRANSIZIONE

- λ-CALCOLO COME ARS: $(\Pi, \to_\beta)$

INSIEME DEI λ-TERMINI

- UNA MACCHINA DI TURING (A, Q, q_0, Q_f, δ) COME ARS:

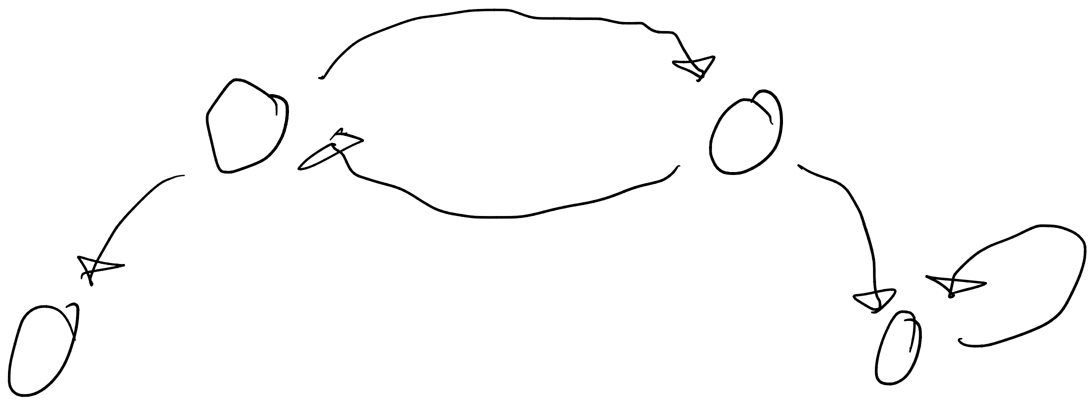
$(A^{\mathbb{Z}} \times \mathbb{Z} \times Q, \to)$

-

- UN LINGUAGGIO DI PROGRAMMAZIONE:

$$(\underbrace{\mathbb{R}}_{\text{REGISTRI}} \times \underbrace{\mathbb{Z}^{\mathbb{N}}}_{\text{MEMORIA}}, \underbrace{\to}_{\text{FETCH-DECODE-EXECUTE}})$$

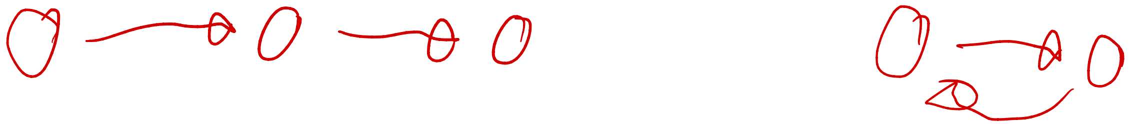
- ESEMPIO ARTIFICIALE:



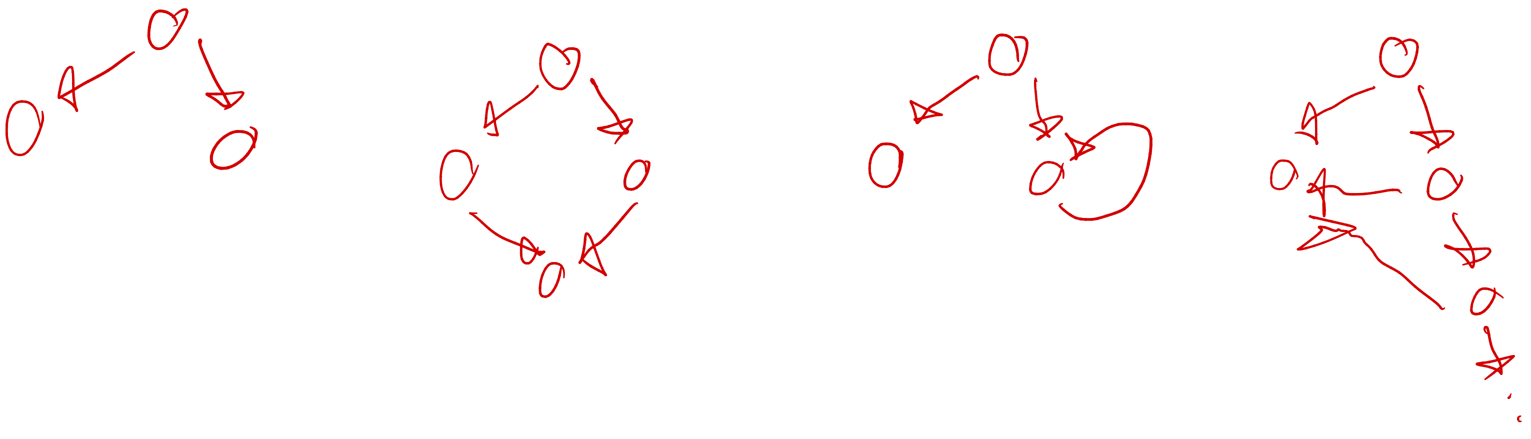
Un ARS $(A, \to)$ è DETERMINISTICO quando

$$\forall q_1, q_2, q_2' \in A. \quad q_1 \to q_2 \ \wedge \ q_1 \to q_2' \implies q_2 = q_2'$$

ESEMPI:

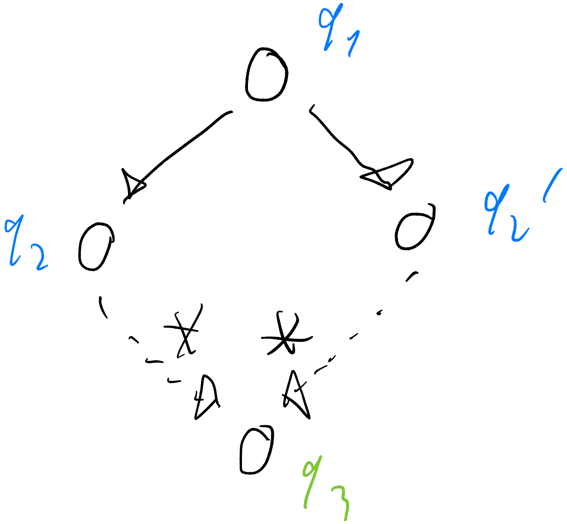


CONTROESEMPI:



TIPI DI CONFLUENZA:

CONFLUENZA LOCALE



$\forall q_1, q_2, q_2'$:

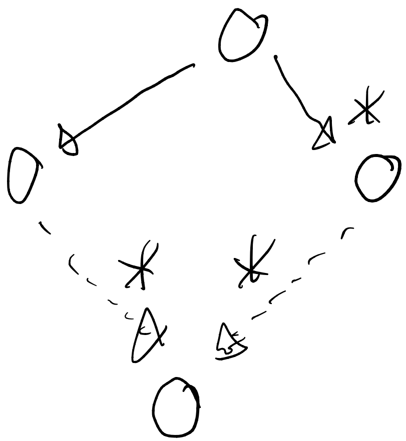
$q_1 \to q_2 \wedge q_1 \to q_2' \Rightarrow$

$\exists q_3 : q_2 \xrightarrow{*} q_3 \wedge q_2' \xrightarrow{*} q_3$

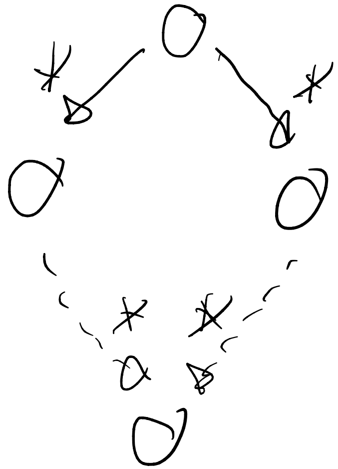
$\forall q_1, q_2, q_2'$: $q_2 \leftarrow q_1 \to q_2'$

$\Rightarrow \exists q_3 .\ q_2 \xrightarrow{*} q_3 \xleftarrow{*} q_2'$

SEMICONFLUENZA

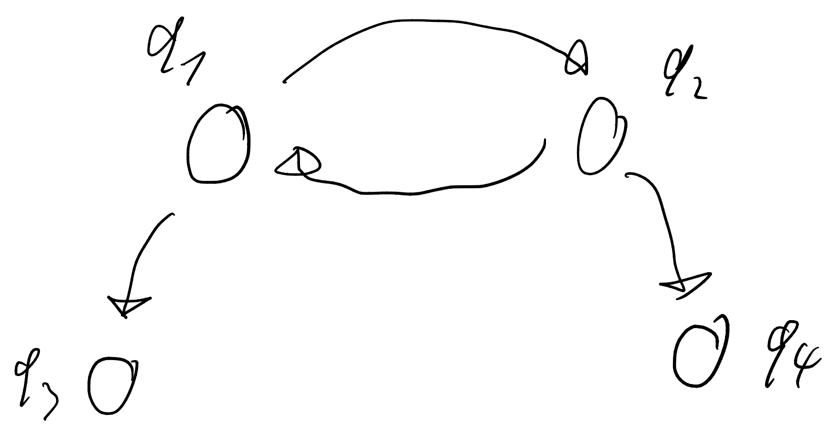


CONFLUENZA

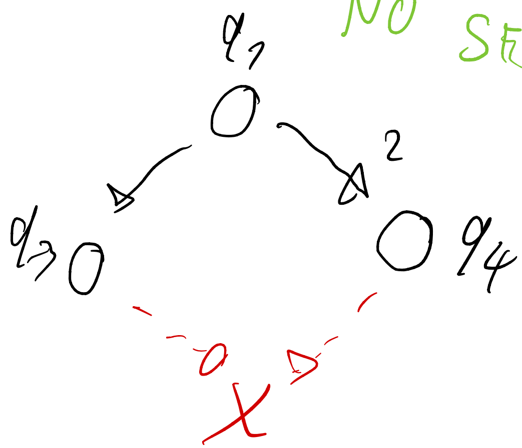


CONFLUENZA LOCALE $\not\Rightarrow$ SEMICONFLUENZA

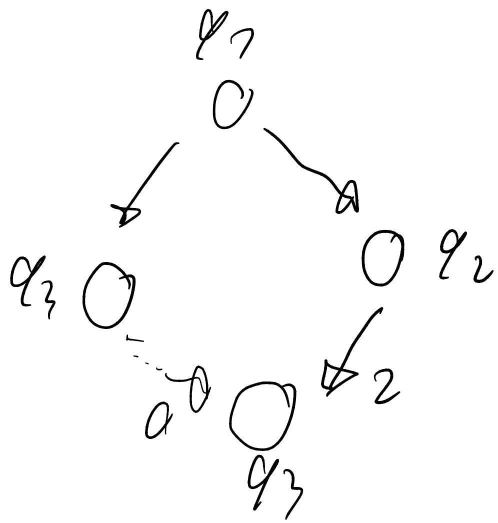
CONTROESEMPIO:



NO SEMIC



CONF. LOCALE



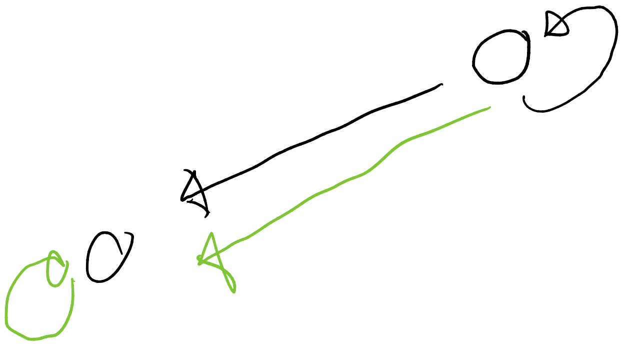
OSSERVAZIONE:

IL CONTROESEMPIO NON È FORTEMENTE NORMALIZZANTE

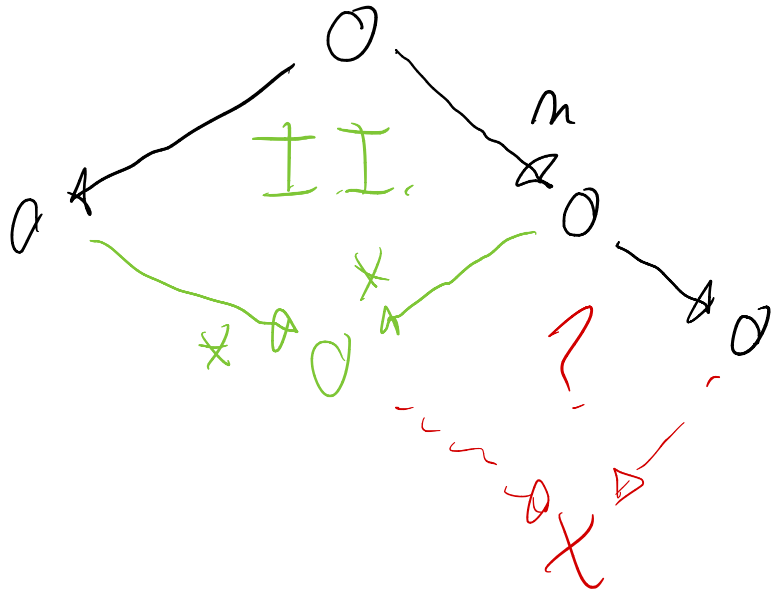
TEOREMA: FORT. NORM. $\land$ CONFLUE. LOCALE $\Rightarrow$ SEMICONFLUENZA

DIMOSTR. ERRATA DI: CONFL. LOCALE $\Rightarrow$ SEMICONFLUENZA

CASO 0 PASSI:

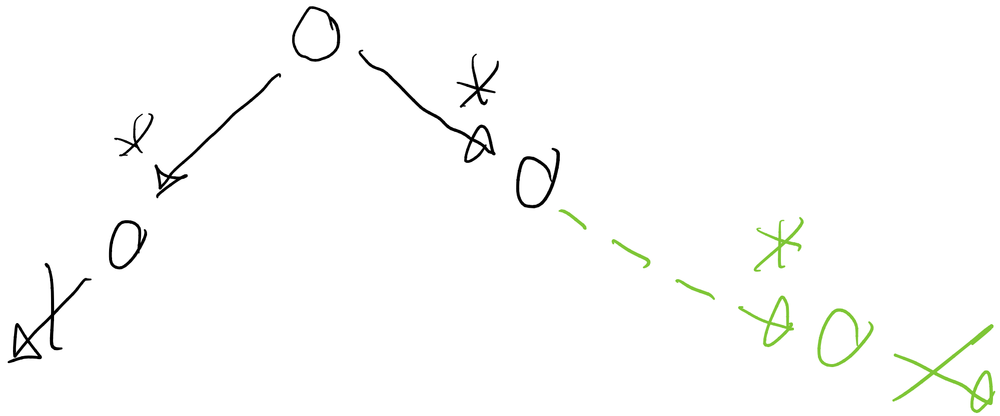


CASO n+1 PASSI:

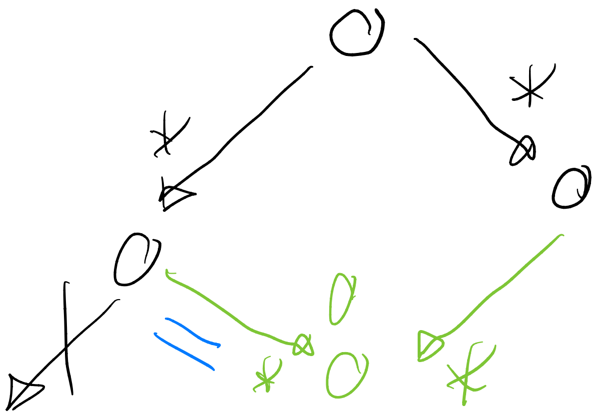


TEOREMA: CONFLUENZA $\Longrightarrow$ SAFETY

DEF DI SAFETY:



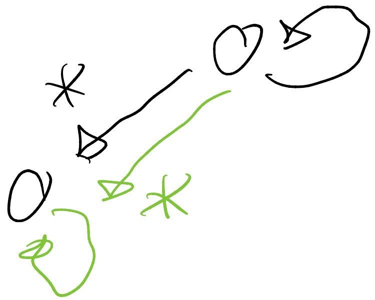
DIMOSTRAZIONE:



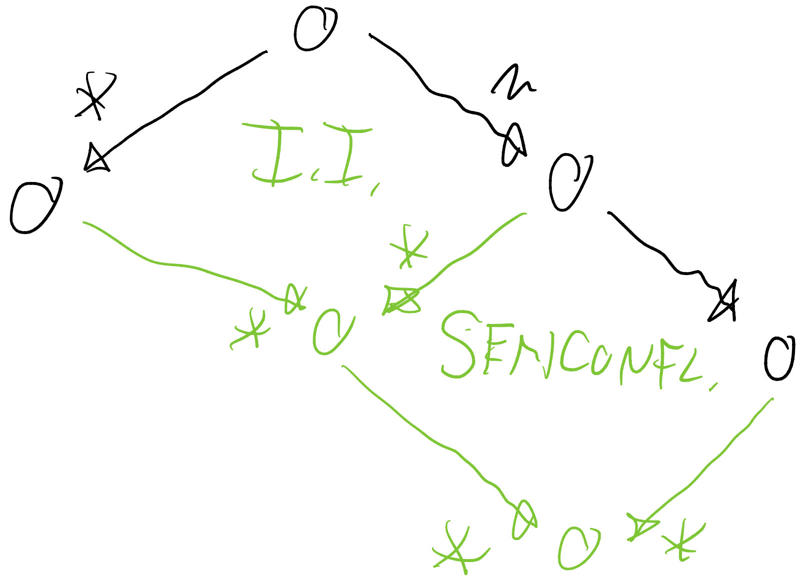
QED

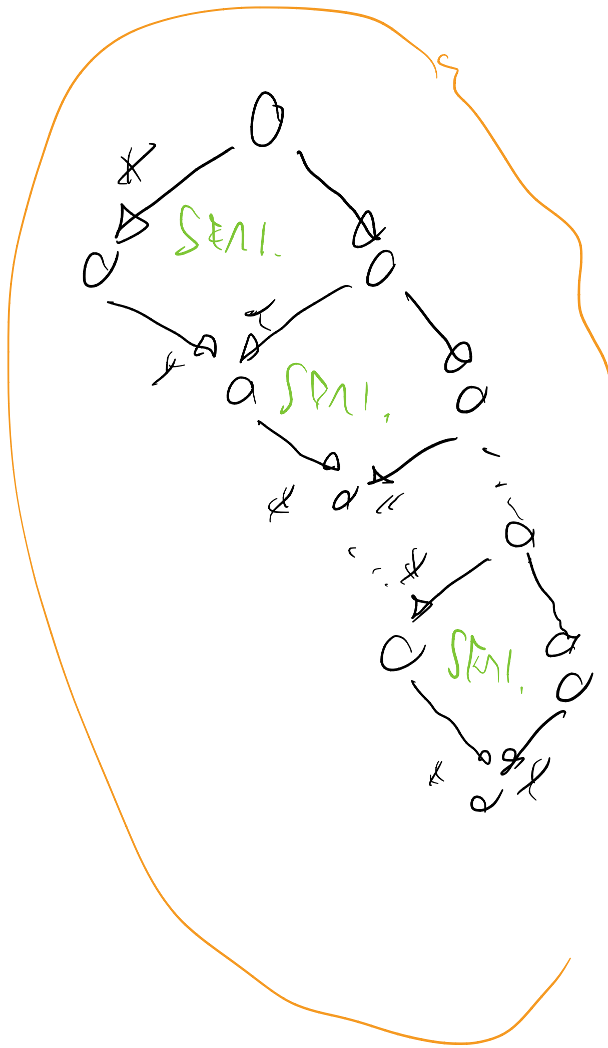
TEOREMA: SEMICONFLUENZA $\Rightarrow$ CONFLUENZA

CASO 0:



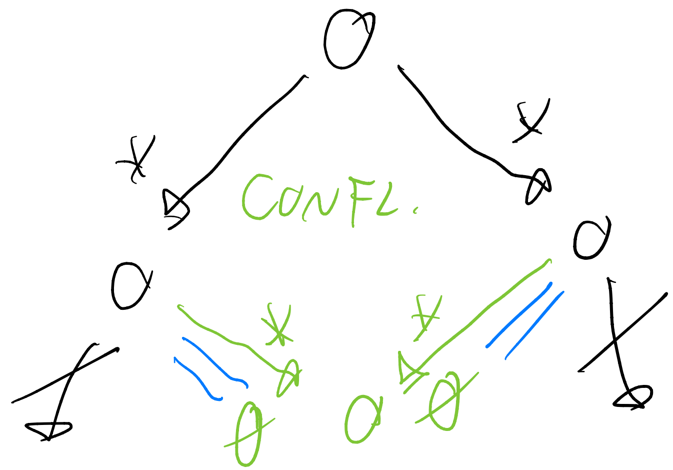
CASO $n+1$:





QED.

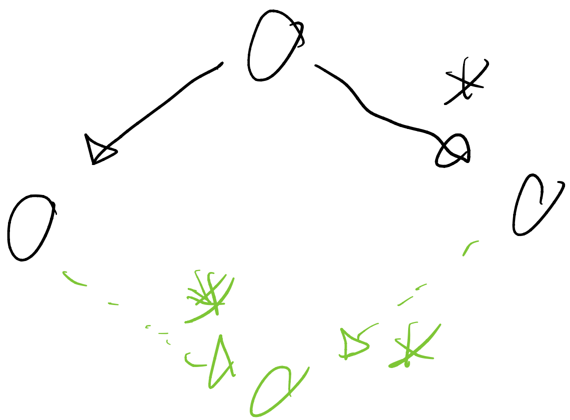
TEOREMA: CONFLUENZA $\Longrightarrow$ UNICITÀ DELLE FORME NORMALI



LE "DUE" FORME NORMALI SONO UGUALI

Q.E.D.

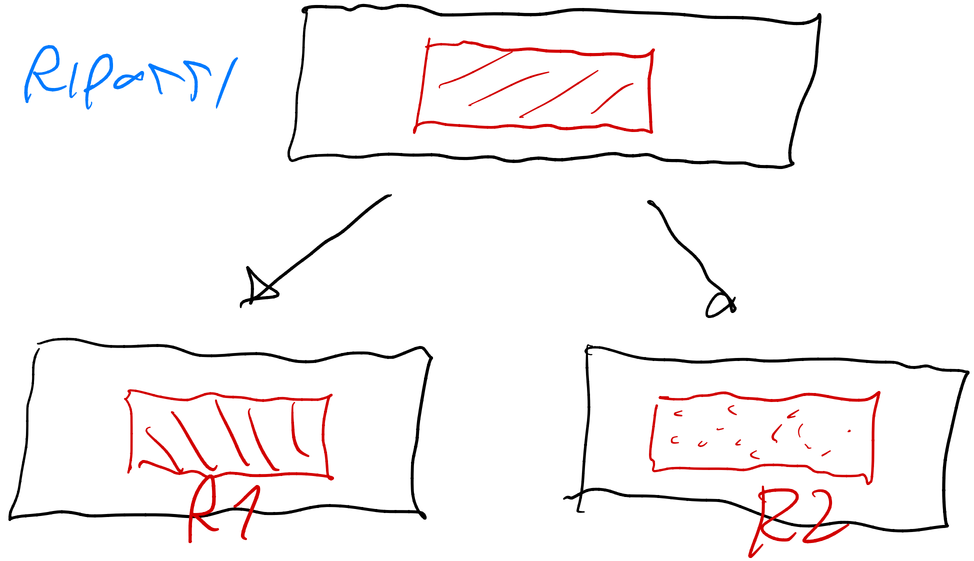
TEOREMA: IL λ-CALCOLO È SEMI-CONFLUENTE



PROVA: OMESSA

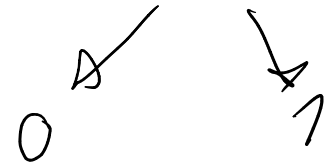
Fonti del non-determinismo:

① Un redesso ha due ridotti

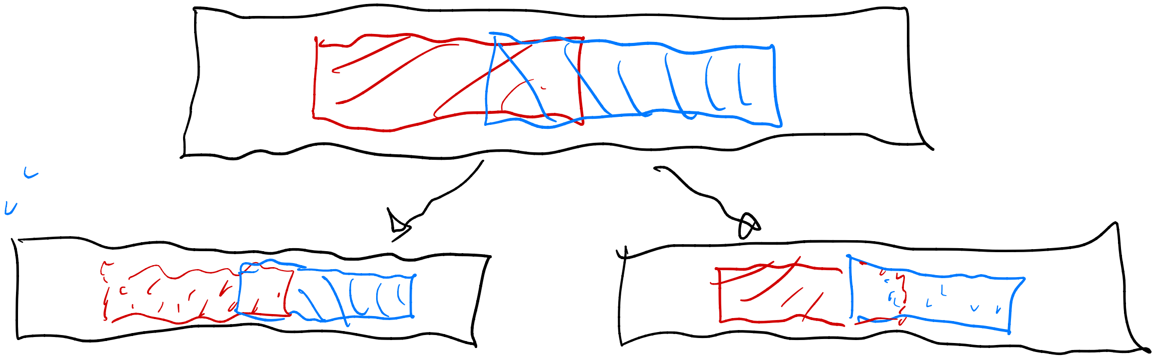


Non nel λ-calcolo

es.: FLIP()



② Due redessi possono essere overlapping, ma non uno strettamente incluso nell'altro



La parte blu non è più un redesso di prima!

La parte rossa non è più il redesso di prima

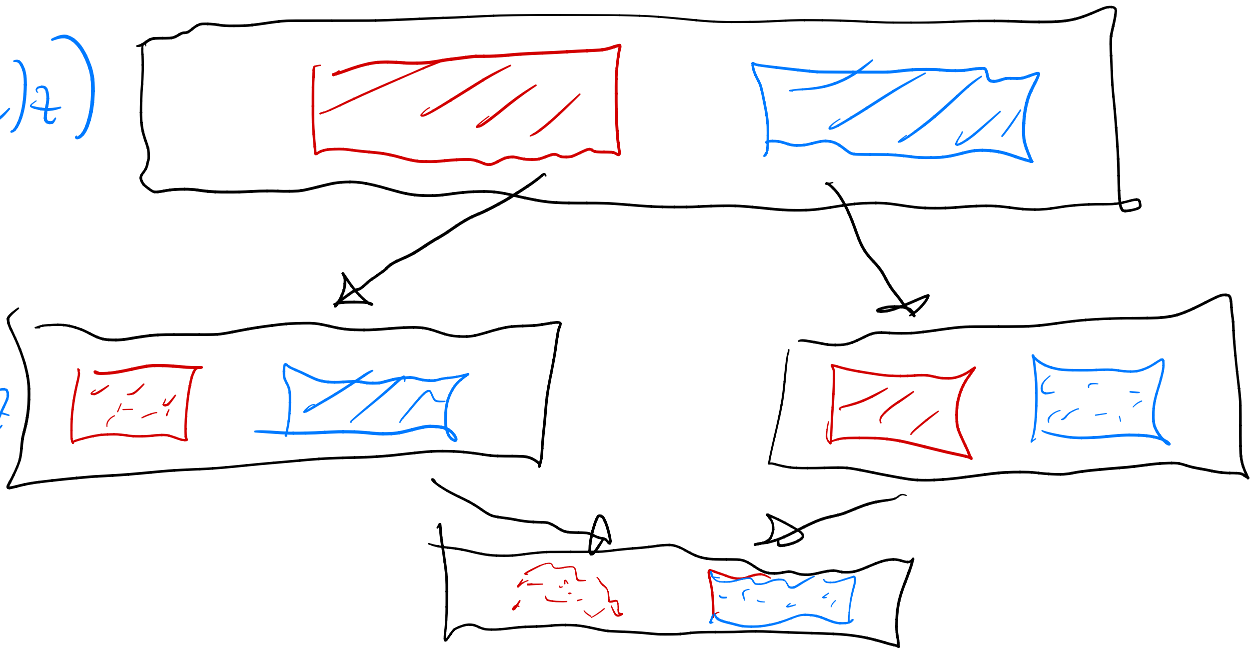
NON NEL λ-CALCOLO

MUTEX m = 1;

$$\left(m.in(); \ \ldots \right) \ \| \ \left(m.in(); \ \ldots \right)$$

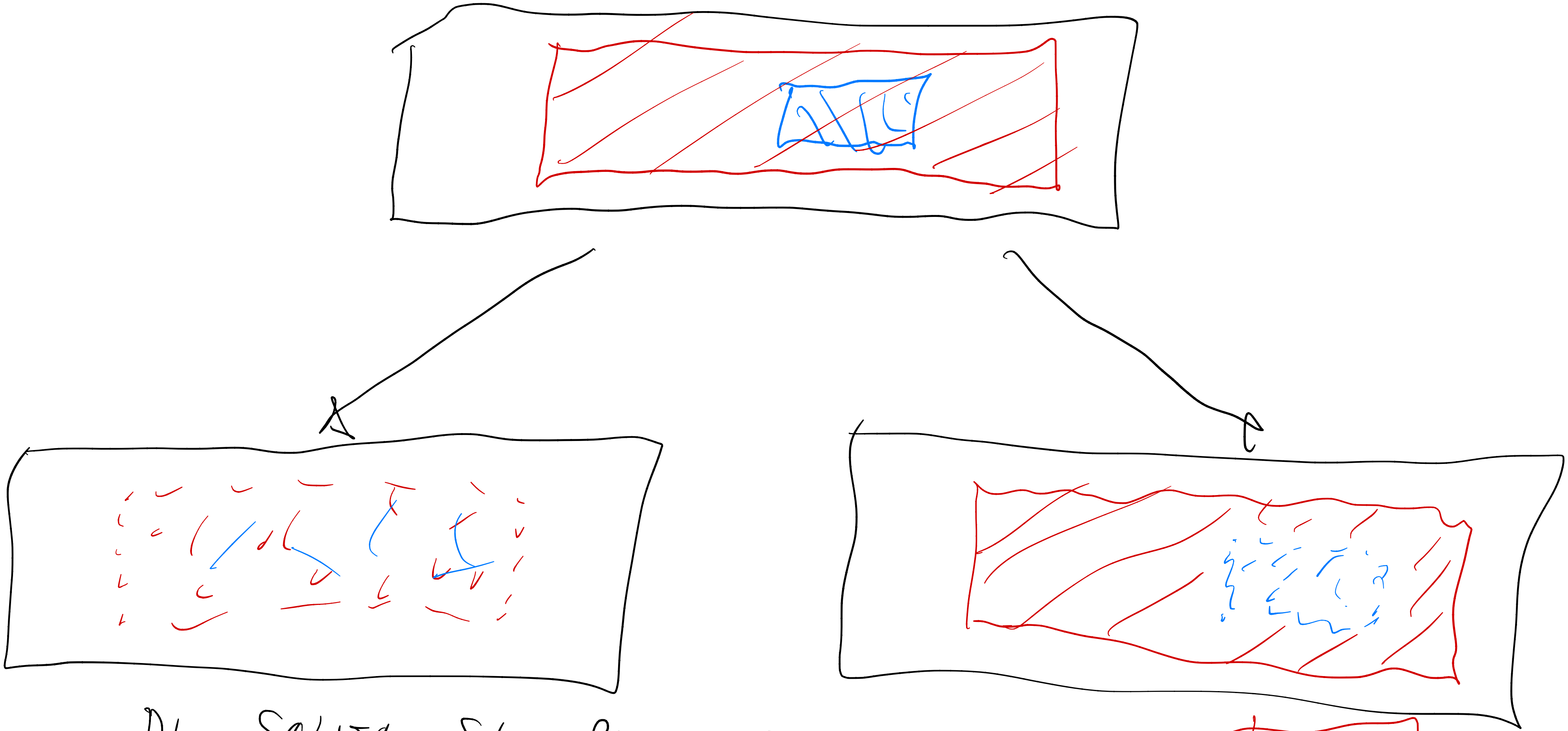
3) REDEXSI NON OVERLAPPING O PARALLELI C'È NEL λ-CALCOLO

$$\begin{array}{ccc} & x\,((\lambda z.z)\,y)\,((\lambda w.w)\,z) & \\ \swarrow & & \searrow \\ x\,y\,((\lambda w.w)\,z) & & x\,((\lambda z.z)\,y)\,z \\ & \searrow \quad \swarrow & \\ & x\,y\,z & \end{array}$$



4 UN REDEX INTERAMENTE CONTENUTO NELL'ALTRO

C'È NEL λ-CALCOLO



DI SOLITO SI PERDE CONFLUENZA

$0 * (1/0) \to 0$

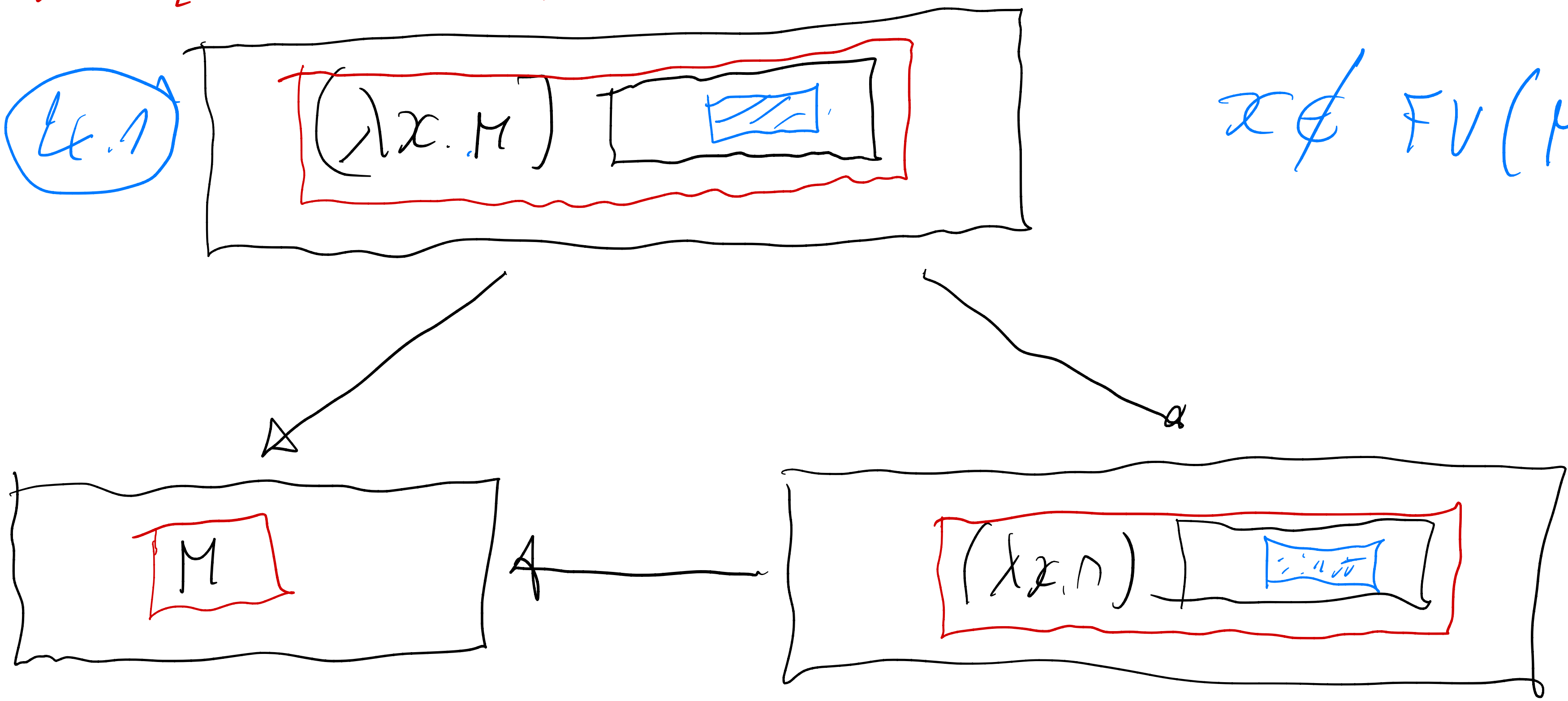
$0 * (1/0) \to 0 * \varepsilon \to \varepsilon$

$0 * X \to 0$

$X / 0 \to \varepsilon$

$X * \varepsilon \to \varepsilon$

CASO 4 NEL λ-CALCOLO:



$x \notin FV(M)$

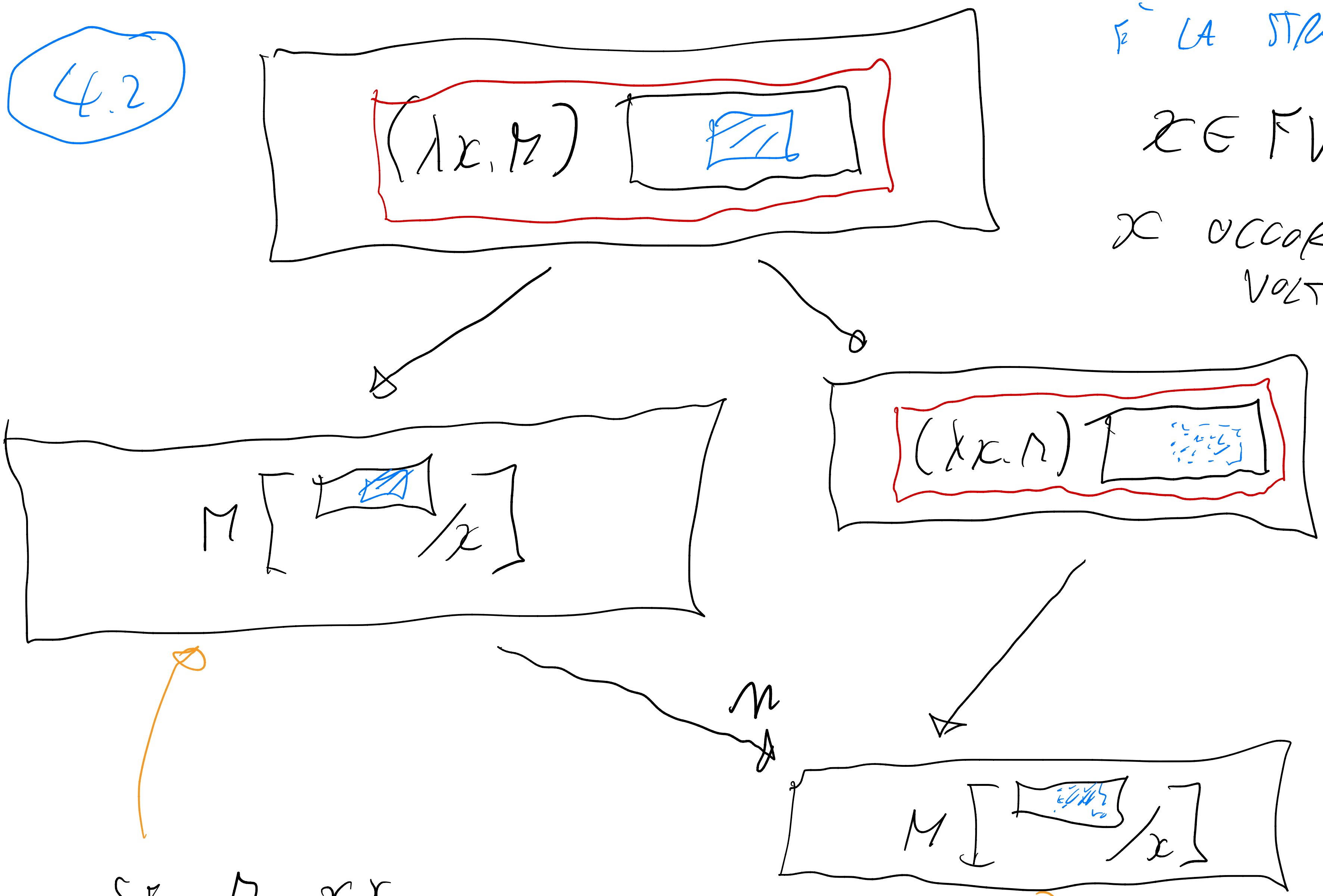
OSSERVAZIONE: CON LA CALL-BY-VALUE (C/JAVA/...) SI RISCHIA DI DIVERGERE QUANDO NON NECESSARIO

4.2

LA CALL-BY-VALUE È LA STRADA PIÙ CORTA

$x \in FV(M)$

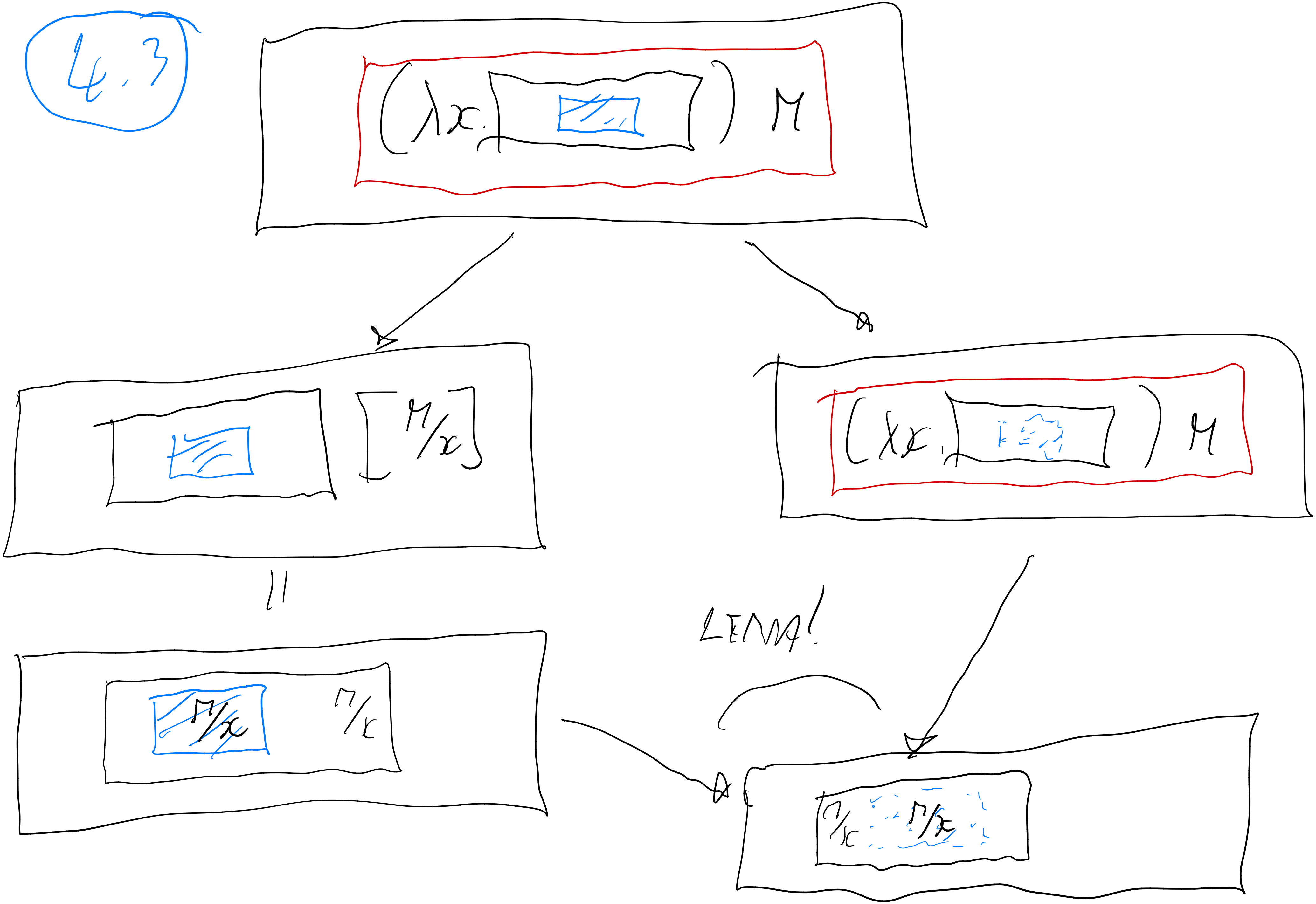
x OCCORRE n VOLTE IN M



Es. SE $M = xx$

HO

4.3



LEMMA: SE $M \to_\beta N$ ALLORA $M[R/x] \to_\beta N[R/x]$

DIM: OMESSA