

Linguaggi di programmazione

Numeri naturali

In base a chi lo si chiede si avranno risposte diverse sul dare una definizione di numeri naturali

es

$3 = \{\{3\}, \{\{3\}\}, \{\{\{3\}\}\}, \{\{\{\{3\}\}\}\}\}$ Von Neumann → definizione insiemistica

$3 = \lambda xy.x(x(y))$ Alonzo Church → definizione ricorsiva

Immaginiamo di parlare con un marziano e cerchiamogli di dare una definizione di successore nei numeri naturali (lui tenterà di smontarla, quindi bisogna fare attenzione) secondo gli assiomi di Peano

1. $0 \in \mathbb{N}$

2. $n \in \mathbb{N} \rightarrow \text{succ}(n) \in \mathbb{N}$

3. $\nexists n \text{ t.c. } 0 = \text{succ}(n)$

4. $\forall n, m \text{ succ}(n) = \text{succ}(m) \Rightarrow n = m$

5. $\forall S \subseteq \mathbb{N} \exists 0 \in S \quad \Rightarrow S = \mathbb{N}$

$\{n \in S \Rightarrow \text{succ}(n) \in S\}$

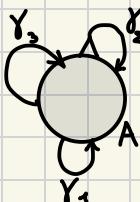
una proprietà e il sottoinsieme corrispondente sono in biezione (dal sottoinsieme posso ricavare la proprietà originale)

induzione

$$\begin{array}{c} \forall P \quad P(0) \quad P(n) \rightarrow P(n+1) \\ \hline \forall n \quad P(n) \end{array}$$

Definire un'algebra

(A, Γ) dove $\Gamma = \{\gamma_1, \gamma_2, \gamma_3, \dots\}$



Ma posso anche definire operazioni che prendono elementi dall'esterno

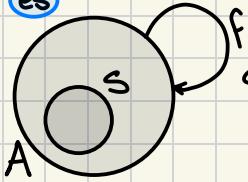
Chiusura di un'algebra

Sia $f: A^n \times K \rightarrow A$ una operazione su A con parametri esterni in $K = K_1 \times \dots \times K_m$.

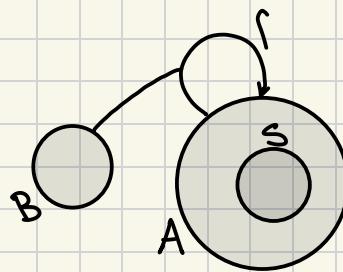
Un insieme $S \subseteq A$ si dice **chiuso** rispetto a f quando $a_1, \dots, a_n \in S \rightarrow f(a_1, \dots, a_n, K_1, \dots, K_m) \in S$

Ovvero è chiuso se per ogni input possibile che pesco da A e che proviene da S , questo rimane in f (quelli che provengono da $A \setminus S$ o dall'esterno non ci interessano)

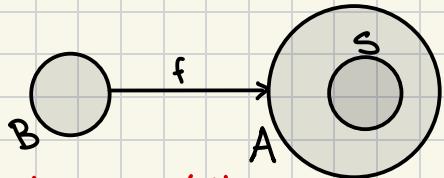
es



S è chiuso rispetto a f se
 $\forall x \in S \rightarrow f(x) \in S$



S è chiuso rispetto a S se
 $\forall x \in B, y \in A \rightarrow f(x, y) \in S$



S è chiuso rispetto a f se
 $\forall x \in B \rightarrow f(x) \in S$

Applicando la def vedremo che, visto che possiamo usare qualsiasi elemento di S , la seconda parte dell'implicazione (visto che prende input solo da B) è vera se per qualsiasi input di f finisce in S

Algebra induttiva

Un'algebra (A, Γ) si dice **induttiva** quando:

- tutte le γ_i sono iniettive

un elemento non può essere immagine

- tutte le γ_i sono disgiunte

di due funzioni differenti

- $\forall S \subseteq A$ se S è chiuso rispetto a tutte le γ_i allora $S = A$ → principio di induzione

es IN

in \mathbb{N} però abbiamo parlato dell'elemento 0 (ora non contemplato), definiamo quindi la seguente algebrā

$(\mathbb{N}, \{\emptyset, \text{succ}\})$ dove $\text{I} \xrightarrow{\emptyset} \mathbb{N}$, $\mathbb{N} \xrightarrow{\text{succ}} \mathbb{N}$

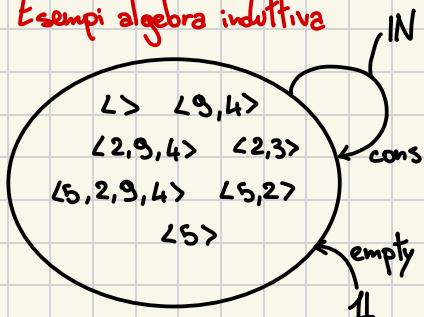
funzione che associa un insieme di un solo elemento all'elemento 0 in \mathbb{N}

Notiamo quindi che:

- tutte le γ_i sono iniettive
- tutte le immagini di γ_i sono disgiunte (\emptyset punta a 0, mentre succ a tutti gli altri)
- \mathbb{N} è chiuso rispetto a succ
- \mathbb{N} è chiuso rispetto a \emptyset

Il costruire un'algebra induttiva mi garantisce che non esistono sottosalgabre in essa contenute. Inoltre tutte le algebre induttive sono **isomorfismi**

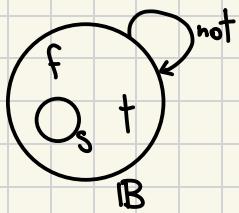
Esempio algebra induttiva



$\text{cons}(2, \langle 3, 4 \rangle) = \langle 2, 3, 4 \rangle \leftarrow$ con append non sarebbe stata induttiva, non disgiunta

L'insieme che contiene le liste finite è induttivo, mentre quello che contiene quelle finite e infinite poiché contiene l'algebra delle liste finite

esistono algebre induttive senza operazioni nullarie? exp.



L'insieme $S = \emptyset$ è chiuso poiché non ci sono elementi in S su cui poter applicare la funzione, la prima parte dell'implicazione della def. di chiusura è falsa, rimane solo da vedere se l'output è

X X X

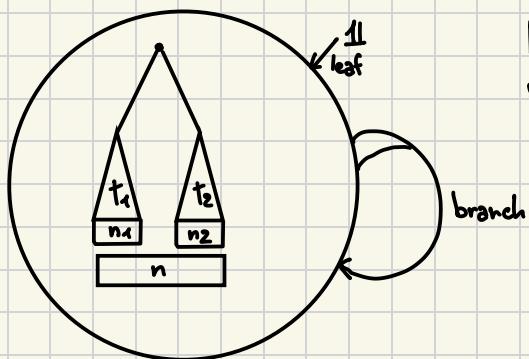
chiuso rispetto a not

con un costruttore che pesca solo da fuori non lo era

TEOREMA

Un albero binario con n foglie ha $2n-1$ nodi
dim

Per dimostrarlo dovranno usare l'induzione strutturale



Nell'induzione strutturale assumiamo che la proprietà sia valida per t_1 e t_2

$$\begin{aligned} n &= n_1 + n_2 \\ (2n_1 - 1) + (2n_2 - 1) + 1 &= 2(n_1 + n_2) - 1 \end{aligned}$$

radice

\downarrow

n

Espresione link

Definiamo un linguaggio L come insieme di stringhe. Per descrivere la sintassi di linguaggi formali, usiamo la seguente sintassi:

$\langle \text{simbolo} \rangle ::= \langle \text{espressione} \rangle$

es

$M, N ::= s \mid ? \mid M+N \mid M*N$

$\underline{\quad} \quad \underline{\quad} \quad \underline{\quad}$ assegnazione "secca"

dove se M e N sono espressioni è possibile costruire un'altra espressione unendoli tramite "+" o "*"

Definiamo quindi una funzione $\text{eval}: \text{Exp} \rightarrow \text{IN}$

es elemento di Exp

$\text{eval}(s) = s$ elemento di IN

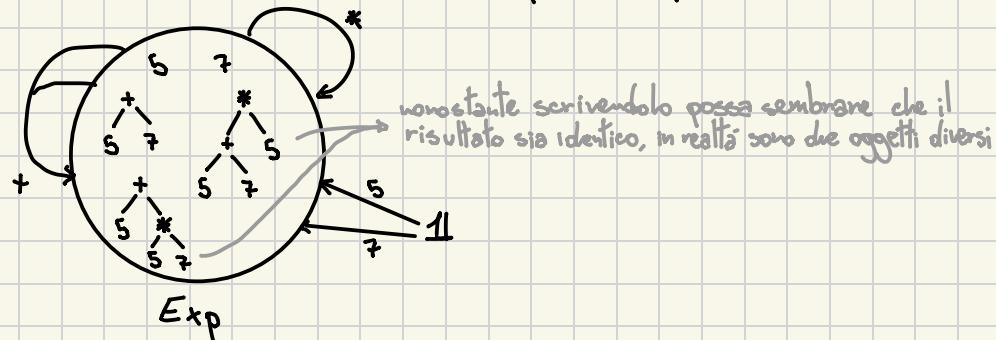
$\text{eval}(?) = ?$

$\text{eval}(M+N) = \text{eval}(M) + \text{eval}(N)$

$\text{eval}(M*N) = \text{eval}(M) * \text{eval}(N)$

$\text{eval}(\boxed{s+?*s}) = ?$

infatti in base a come considero l'espressione posso avere risultati diversi, dobbiamo quindi **disambiguare**



nonostante scrivendolo possa sembrare che il risultato sia identico, in realtà sono due oggetti diversi

sicuramente non si tratta di una grammatica induttiva, ma ho usato l'induzione per definirla

Equazione ricorsive di domini

es

$$\begin{aligned} \mathbb{N} &= 1\mathbb{N} + \mathbb{N} \\ L &= \mathbb{N} \cup (\mathbb{N} \times L) \end{aligned}$$

isomorfismo 2
insieme di liste

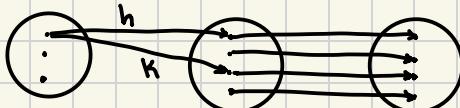
Brief sulla teoria delle categorie

Nella teoria delle categorie non si può usare la parola "elemento"

Come faccio quindi a descrivere un morfismo (funzione) f iniettivo (monomorfismo) non potendo parlare di elementi?

RIS $h \circ f = k \circ f \Rightarrow h = k$

$$A \xrightarrow{h} B \xrightarrow{k} C$$



essendo f iniettiva ad ogni elemento viene associato un elemento diverso del codominio, dunque nella composizione $h \circ f$ e $k \circ f$ risulta che gli elementi di destinazione sono esattamente gli stessi allora $h = k$

Segnatura di un'algebra

Due algebre (A, Γ_A) , (B, Γ_B) hanno la stessa segnatura se $\forall \gamma_A \in \Gamma_A \exists \gamma_B \in \Gamma_B$ con stessa aritmetà e se, sostituendo tutte le B in γ_B con A ottengo γ_A

es

algebra di D

$$\begin{aligned} f_D: A \times D &\rightarrow D \\ g_D: 1\mathbb{N} &\rightarrow D \\ h_D: A \times B \times D &\rightarrow D \end{aligned}$$

algebra di C

$$\begin{aligned} f_C: A \times C &\rightarrow C \\ g_C: 1\mathbb{N} &\rightarrow C \\ h_C: A \times B \times C &\rightarrow C \end{aligned}$$

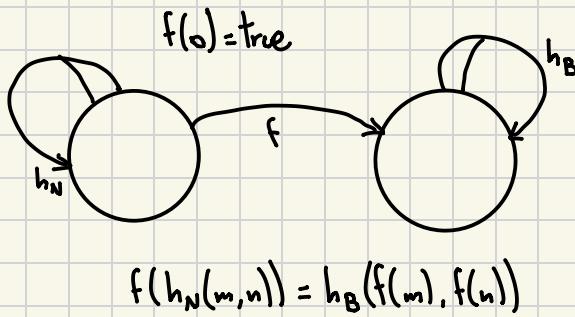
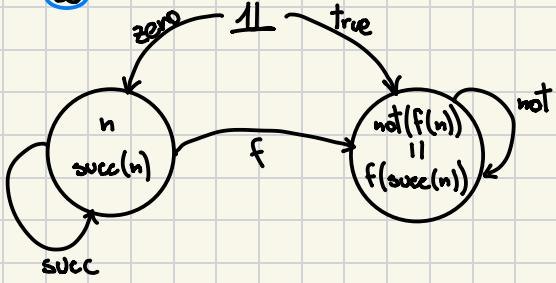
D e C hanno la stessa segnatura

Omomorfismo

Due algebre con la stessa segnatura (A, Γ) e $(B, \Delta = \{\delta_1, \dots, \delta_K\})$, un omomorfismo è una funzione $A \rightarrow B$ tale che:

$$\forall i: f(\gamma_i(a_1, \dots, a_n)) = \delta_i(f(a_1), \dots, f(a_n)) \quad \text{rispetta le operazioni}$$

es



Se applico l'operazione da una parte o dall'altra il risultato non cambia, e lo devo fare per tutte le operazioni dell'algebra

TEOREMA

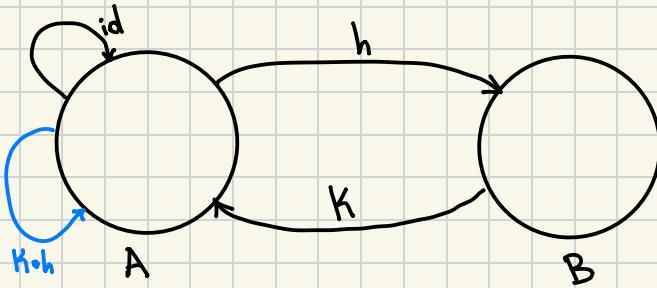
Prese due algebre induttive che hanno la stessa segnatura allora esiste un unico omomorfismo

DEF Isomorfismo

Un isomorfismo è un omomorfismo in cui la funzione è una biezione

LEMMA di Lambek

Avendo due algebre induttive esistono due omomorfismi (nelle due direzioni) e componendoli otteniamo un ulteriore omomorfismo



componendo h e K otteniamo dunque, per quanto detto, un ulteriore omomorfismo, ma in A già esiste un omomorfismo ovvero l'identità. Per il teorema precedente però può esistere un solo omomorfismo dunque $id = K \circ h$

da ciò si deduce che K e h sono inversi per cui la loro composizione è chiaramente biettiva, il che ci fa concludere che $K \circ h$ è un isomorfismo

Linguaggio Exp

link

es

$$M, N, L ::= K \mid x \mid M + N \mid \text{let } x = M \text{ in } N$$

ML Standard

dove:

- $K \rightarrow$ costante

- $x \rightarrow$ variabile $\in Var$

- $\text{let } x = M \text{ in } N \rightarrow$ assegna a x il valore M all'interno di N

Vediamo alcuni esempi di uso:

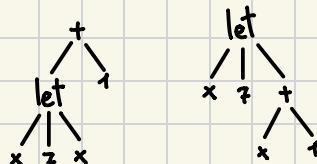
$$\text{let } x = 3 \text{ in } x + x + 2 \Rightarrow 8$$

$$\text{let } x = x + 2 \text{ in } 4 \Rightarrow 4$$

$$\text{let } x = \text{let } y = 3 \text{ in } y + 1 \text{ in } x + y \Rightarrow 7$$

$$\text{let } x = 7 \text{ in } x + 1 \Rightarrow ??$$

quest'ultima espressione può creare ambiguità
dato che può essere doppiamente interpretata



DEF Variabili libere e legate

Una variabile x si dice **libera** se data un'exp $\text{let } x = M \text{ in } N$, x non compare in N

Si dice invece **legata** se invece compare in N

es

$$\begin{array}{c} \text{legata} \\ \text{let } x = 5 \text{ in } x + x \\ \text{libera} \end{array}$$

DEF free

Definiamo la funzione free: $\text{Exp} \rightarrow \wp(\text{Var})$ che restituisce l'insieme delle variabili libere di un'espressione

es

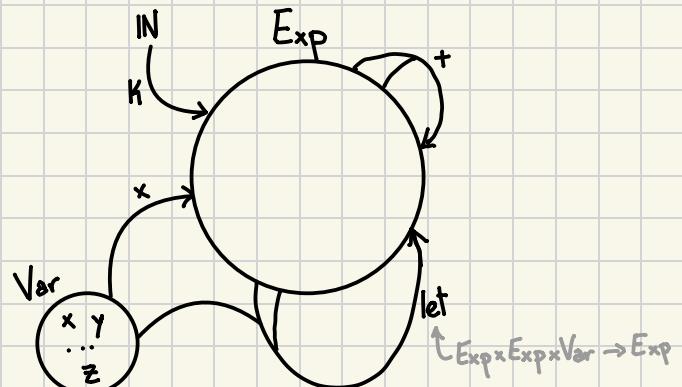
$$\text{free}(\text{let } x = 7 \text{ in } x + y) = \{y\}$$

$$\text{free}(K) = \emptyset$$

$$\text{free}(x) = \{x\}$$

$$\text{free}(M + N) = \text{free}(N) \cup \text{free}(M)$$

$$\text{free}(\text{let } x = M \text{ in } N) = \text{free}(M) \cup (\text{free}(N) - x) \leftarrow \text{free}(\text{let } x = x \text{ in } x) = x$$



Semantica operazionale

La semantica operazionale permette di rispondere alla domanda quanto fa, permettendoci di eliminare le ambiguità viste finora

Definisco quindi un'operazione per calcolare il valore, ma dipende anche dall'**ambiente** che si sta considerando

$$M \in \text{Exp} \times \text{Env} \rightsquigarrow \text{Val}$$

dove $\rightsquigarrow \subseteq \text{Exp} \times \text{Exp} \times \text{Val}$

↑
associazione tra
variabile e valore

↑
ambiente in cui
valutare Exp

Poiché ci sono casi in cui non viene restituito alcun risultato, si tratta quindi di una **relazione** e non di una funzione

Mi sto quindi chiedendo se $(M, E, v) \in \rightsquigarrow$

questa espressione ha questo valore

es

$$(\text{let } x=3 \text{ in } x+1, [x \mapsto 7], 10) \in \rightsquigarrow$$

$$(s, [E], 7) \notin \rightsquigarrow$$

Passiamo ora a una notazione più snella

deduzione

$$E \vdash M \rightsquigarrow v$$

↓ sequenze

che si legge "M deduce E con valore v"

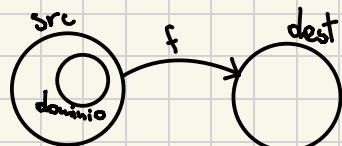
Ambiente

funzione non definita per
alcuni elementi del dominio

L'ambiente è una funzione parziale a dominio finito da variabile a valore

\rightarrow dominio finito

$$E: \text{Var} \xrightarrow{\text{fin}} \text{Val}$$



$$f: \text{src} \longrightarrow \text{dest}$$

$$(E_1, E_2)(x) = \begin{cases} E_2(x) & \text{se } x \in \text{dom}(E_2) \\ E_1(x) & \text{altrimenti} \end{cases}$$

es

$$E_1 = \{(z, 3), (y, 9)\}$$

$$E_2 = \{(z, 4)\}$$

$$(E_1, E_2)(y) = 9$$

$$(E_1, E_2)(z) = 4$$

Regola di inferenza eager

- ① $E \vdash K \rightsquigarrow K$ const
 ② $E \vdash x \rightsquigarrow E(x)$ x non esiste non esiste nessuna triple var
 ↗ quando fa x
 nell'ambiente E ?

- $$\textcircled{3} \quad \begin{array}{c} E \vdash M \rightsquigarrow v_1 & E \vdash N \rightsquigarrow v_2 \\ \hline E \vdash M + N \rightsquigarrow v \end{array} \quad (\text{dove } v = v_1 + v_2) \quad \begin{array}{l} \text{condizioni laterali} \\ \text{plus} \end{array}$$

conclusione

- $$\begin{array}{c} \textcircled{4} \\ \text{valuta M con E} \\ \text{e mettilo in } v \\ \text{E} \vdash M \rightsquigarrow v \\ \hline \text{E} \vdash \text{let } x = M \text{ in } N \rightsquigarrow v' \end{array} \quad \begin{array}{c} \text{valuta N con E insieme a} \\ (x, v) \text{ e mettilo in } v' \\ \text{E} \{ (x, v) \} \vdash N \rightsquigarrow v' \\ \text{let} \end{array}$$

- es** $\emptyset \vdash \text{let } x=3 \text{ in } ((\text{let } x=(\text{let } y=2 \text{ in } x+y) \text{ in } x+7) + x)$

adesso appliciamo le regole di inferenza dall'esterno all'interno

per prima uso quella del let, spostando la parentesi se vedete stata quella del +

$$\begin{array}{c}
 \frac{(x,3)(y,2) \vdash x \rightsquigarrow 3 \quad (x,3)(y,2) \vdash y \rightsquigarrow 2}{(x,3) \vdash 2 \rightsquigarrow 2} \text{ plus} \\
 \frac{(x,3) \vdash 2 \rightsquigarrow 2 \quad (x,3)(y,2) \vdash x+y \rightsquigarrow 5}{(x,3) \vdash \text{let } y=2 \text{ in } x+y \rightsquigarrow 5} \text{ let} \\
 \frac{(x,3) \vdash \text{let } y=2 \text{ in } x+y \rightsquigarrow 5 \quad (x,3)(x,5) \vdash x+7 \rightsquigarrow 12}{(x,3) \vdash \text{let } x=(\text{let } y=2 \text{ in } x+y) \text{ in } x+7 \rightsquigarrow 12} \text{ let} \\
 \frac{(x,3) \vdash \text{let } x=(\text{let } y=2 \text{ in } x+y) \text{ in } x+7 \rightsquigarrow 12 \quad (x,3) \vdash x \rightsquigarrow 3}{(x,3) \vdash \text{let } x=(\text{let } y=2 \text{ in } x+y) \text{ in } x+7+x \rightsquigarrow 15} \text{ plus} \\
 \hline
 \emptyset \vdash 3 \rightsquigarrow 3 \quad \boxed{\{(x,3)\} \vdash (\text{let } x=(\text{let } y=2 \text{ in } x+y) \text{ in } x+7) + x \rightsquigarrow 15} \text{ let} \\
 \hline
 \emptyset \vdash \text{let } x=3 \text{ in } ((\text{let } x=(\text{let } y=2 \text{ in } x+y) \text{ in } x+7) + x) \rightsquigarrow 15
 \end{array}$$

L'approccio eager però ha uno svantaggio, infatti in un caso del tipo `let x = f()` ~~f~~ viene valutato prima il "mostro" nonostante in questo caso non sia assolutamente necessario.

Semantica lazy-dinamica

Rispetto alle regole di inferenza eager cambiano quella del let e quella della variabile

- $$\textcircled{1} \quad \frac{E \vdash M \rightsquigarrow J}{E \vdash x \rightsquigarrow J} (\text{se } Ex = M) \quad \text{var}$$

- oss $\emptyset \vdash \text{let } x = y \text{ in let } y = 7 \text{ in } x$ in lazy dinamico
f2 7 ma dovrebbe essere non definito

- $$\textcircled{2} \quad \frac{E(x,M) \vdash N \rightsquigarrow v}{E \vdash \text{let } x = M \text{ in } N \rightsquigarrow v} \text{ let}$$

- (es) let $x=2$ in let $y=x+1$ in let $x=7$ in y

eager

$$\frac{(x,2) \vdash x \rightsquigarrow 2 \quad (x,2) \vdash 1 \rightsquigarrow 1}{(x,2) \vdash x+1 \rightsquigarrow 3} \quad \frac{(x,2)(y,3) \vdash ? \rightsquigarrow ? \quad (x,2)(y,3)(x,?) \vdash y \rightsquigarrow 3}{(x,2)(y,3) \vdash \text{let } x=? \text{ in } y \rightsquigarrow 3}$$

- $\emptyset \vdash 2 \rightsquigarrow 2$ $(x, 2) \vdash \text{let } y = x + 1 \text{ in let } x = 7 \text{ in } y \rightsquigarrow 3$

- $\emptyset \vdash \text{let } x = 2 \text{ in let } y = x + 1 \text{ in let } x = 7 \text{ in } y$ and 3

lazy-dinamico

$$\begin{array}{c}
 \text{_____} \\
 \underline{(x,2)(y,x+1)(x,7) \vdash x \rightsquigarrow 7} \quad (x,2)(y,x+1)(x,7) \vdash 1 \rightsquigarrow 1 \\
 \underline{(x,2)(y,x+1)(x,7) \vdash x+1 \rightsquigarrow 8} \\
 \underline{(x,2)(y,x+1)(x,7) \vdash y \rightsquigarrow 8} \\
 \underline{(x,2)(y,x+1) \vdash \text{let } x=7 \text{ in } y \rightsquigarrow 8} \\
 \emptyset (x,2) \vdash \text{let } y=x+1 \text{ in let } x=7 \text{ in } y \rightsquigarrow 8
 \end{array}$$

diverso

Introducendo il lazy ci siamo esposti al problema dello scoping (in questo caso è dinamico). La semantica lazy-dinamica è considerata "sbagliata" poiché ci dà un risultato diverso dalle altre 3 combinazioni di metodologie.

Semantica lazy-statica

Rispetto alle regole di inferenza eager cambiano quella del let e quella della variabile.

①

$$\frac{E \vdash M \rightsquigarrow v}{E' \vdash x \rightsquigarrow v} \quad (\text{se } E'(x) = (M, E)) \quad \text{var}$$

②

$$\frac{\begin{array}{l} E(x, M, E) \vdash N \rightsquigarrow v \\ \text{let } E \vdash \text{let } x = M \text{ in } N \rightsquigarrow v \end{array}}{E \vdash \text{let } x = M \text{ in } N \rightsquigarrow v} \quad \text{let} \quad E : \text{Var} \xrightarrow{\text{fin}} \text{Exp} \times \text{Env}$$

OSS in questo caso let $x = \dots$ in $x+x+x+x+x$ la semantica lazy è molto lenta (devo ricalcolare ogni volta x)

es

$$\begin{array}{c} \frac{\emptyset \vdash 2 \rightsquigarrow 2}{(x, 2, \emptyset) \vdash x \rightsquigarrow 2} \quad \frac{}{(x, 2, \emptyset) \vdash 1 \rightsquigarrow 1} \\ \frac{(x, 2, \emptyset) \vdash x+1 \rightsquigarrow 3}{E(x, 7, E) \vdash y \rightsquigarrow 3} \\ \hline E \vdash (x, 2, \emptyset)(y, x+1, (x, 2, \emptyset)) \text{ let } x=7 \text{ in } y \rightsquigarrow 3 \\ \hline (x, 2, \emptyset) \vdash \text{let } y = x+1 \text{ in let } x = 7 \text{ in } y \rightsquigarrow 3 \\ \hline \emptyset \vdash \text{let } x = 2 \text{ in let } y = x+1 \text{ in let } x = 7 \text{ in } y \rightsquigarrow 3 \end{array}$$

TEOREMA

Semantica eager e lazy-statica sono equivalenti.

	statico	dinamico
lazy	○	✗
eager	○	○

il linguaggio Exp è troppo semplice per poter cogliere la differenza tra eager statico e dinamico

Linguaggio Fun

es

$$M, N ::= \lambda x M, N | \text{let } x = M \text{ in } N | F_n x \Rightarrow M | MN$$

\hookrightarrow funzione M applicata a argomento N

$$(F_n x \Rightarrow x+1) S$$

$M \quad N$