

3/03/14

## DEFINIZIONE FORMALE DI SEMANTICA

Semantica

$$(L, M, E_{i \in I}, \equiv)$$

equivalenza  
semantica  
 $\equiv \subseteq M \times M$

$$l, l', l = l' \Leftrightarrow E[l] = E[l']$$

Linguaggio oggetto  
 $f \in L$

metà linguaggio  
definisce gli  
oggetti per  
assegnare significato  
ad  $f, m \in M$

funzioni di interpretazione  
semantico.

$$E_{i \in I}: L \rightarrow M$$

## OPERAZIONE

$L$  = Linguaggio Oggetto che definiamo

$$M = \langle S, \rightarrow \rangle \text{ tc. } S = \{ \langle l, g: \text{Var}(l) \rightarrow \text{Val} \rangle \mid l \in L \}$$

relazione di transizione  
che mette in relazione  
configurazioni successive

$$\langle l_0, g_0 \rangle \rightarrow \langle l_1, g_1 \rangle$$

eseguendo istruzione  $l$   
aggiornando  $\cancel{g_0}$  con i  
valori manipolati dall'istruzione  
corrente.

$\mathcal{E}$  = definizione della " $\rightarrow$ " di  $M$

$\equiv_m$  = identità sui grafici

### DENOTAZIONALE

$L$  = linguaggio da definire.

$M = \{g : \text{Var}(l) \rightarrow \text{Val} \text{ tc le } l\}$

$\mathcal{E} = \#^L \rightarrow M$  ovvero:  $l \rightarrow \text{Val}$

$\equiv$  = identità

### ASSIOMATICA

$L$  = linguaggio da definire.

$M = \{\text{albero di derivazione di le } l\}$

$\mathcal{E}$  = data dalla definizione di assiomi e delle regole di inferenze

$\equiv$  = identità sugli alberi di derivazione.

## PRINCIPIO SOSTITUTIVITÀ

$\ell, \ell' \in \mathcal{L}$  che possono comparire in un contesto  
 $C[\cdot]$

$$\ell \underset{\mathcal{E}_\ell}{=} \ell' \Rightarrow \mathcal{H}C[\cdot], C[\ell] \underset{\mathcal{E}_\ell}{=} C[\ell']$$

Usato per ottimizzare il codice, in quanto viene sostituito un pezzo di codice con uno con lo stesso significato equivalente.

## Principio di FULL ABSTRACTION

$\ell, \ell' \in \mathcal{L}, C[\cdot] \not\models C[\cdot]. C[\ell] \underset{\mathcal{E}_\ell}{=} C[\ell'] \Rightarrow \ell \underset{\mathcal{E}_\ell}{=} \ell'$

# APPUNTI PER ESAME

Sol ①: Il principio di sostituzione dice che dati due programmi equivalenti, se li sostituisci in ogni contesto di programma ottengo ancora programmi equivalenti.

~~Non scrivere così~~

Sol ②: Dati  $l, l' \in \mathcal{L}$ .  $l \underset{\mathcal{E}}{\equiv} l' \Rightarrow \mathcal{A}(l) \underset{\mathcal{C}}{\equiv} \mathcal{A}(l')$ .

$\uparrow \downarrow$  soltuione ottimale

Sol ③: Data una semantica  $(\mathcal{L}, M, \mathcal{E}, \underset{m}{\equiv})$  che genera  $\underset{\mathcal{E}}{\equiv}$  come

$$l \underset{\mathcal{E}}{\equiv} l' \Leftrightarrow \mathcal{E}(l) \underset{m}{\equiv} \mathcal{E}(l')$$

+ Sol ②

# SEMANTICA STATICÀ - TYPE CHECKING

Tipo: Insieme di valori con definita una relazione  
[ $\Sigma$ ] di uguaglianza decidibile

es

$(\mathbb{Z}, =)$  è un tipo perché è definita un'operazione di uguaglianza sugli interi decidibile.

Teoria dei tipi: si occupa di determinare la validità di alcune proprietà di elementi di un insieme utilizzando delle afferzioni o judgments:

DATO:  $\Delta = \{ \begin{array}{l} x_1 : \tau_1, \dots, x_n : \tau_n \mid i \neq j \Rightarrow x_i \neq x_j \end{array} \}$   
none: tipo

AMBIENTI STATICI: Associano oggetti a tipi

-  $\Delta \vdash \tau$ : da  $\Delta$  posso dimostrare/inferire che è di tipo  $\tau$

~~=  $\Delta \vdash \tau = \tau'$~~

-  $\Delta \not\vdash X : \tau$ : da  $\Delta$  posso inferire che  $X$  è di tipo  $\tau$

$\{x: \text{int}, z: \text{bool} \dots\} \models \text{int}$

Assumendo che int sia un tipo permesso e superi la fase di analisi sintattica.

$x \in \tau$   
 $z \in N \Rightarrow z : \text{int}$

esecuzione di  $x$  genera  $x' \in \tau$

$2, 3, 4 \in N \Rightarrow 2+3 \cdot 4 : \text{int}$

es

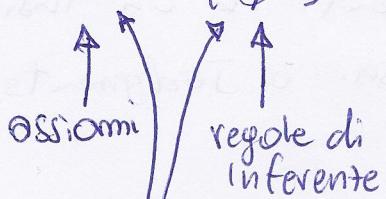
$$\{x:\text{int}, y:\text{int}, z:\text{int}\} \vdash x+y+z : \text{int}$$

$$\{x:\text{int}, y:\text{int}, z:\text{bool}\} \not\vdash x+y+z$$

non posso inferire il tipo.

$\mathcal{U}$ : universo premesse

$$\mathbb{I}_{\mathcal{U}}: \left\{ \frac{\emptyset}{y} \right\} \cup \left\{ \frac{x}{y} \right\}, y \in \mathcal{U}, x \subseteq \mathcal{U}$$



Conclusione.

$\mathbb{I}$ -derivation:  $\frac{\emptyset}{y}, y \in \mathcal{U}$  oppure  $\frac{d_1 \dots d_n}{y}$ ,  $d_1 \dots d_n$  sono  $\mathbb{I}_{\mathcal{U}}$ -derivation.

INDUZIONE

$\mathbb{I}_{\mathcal{U}}$ : regole di inferenza,  $\frac{\mathbb{I}_{\mathcal{U}}}{\mathbb{I}_{\mathcal{U}'}}$  derivationi su  $\mathcal{U}' \supset \mathcal{U}$  a partire da

$\forall d \in \mathbb{I}_{\mathcal{U}}. P(d) \Leftrightarrow \forall d \in \mathbb{I}_{\mathcal{U}'}. ([\forall d' \in d. P(d')] \Rightarrow P(d))$

$\Delta \vdash x : \text{int}$

$\Delta \vdash y : \text{int}$

$\Delta \vdash z : \text{int}$

$$\frac{\Delta \vdash y * z : \text{int}}{\Delta \vdash y + z : \text{int}}$$

$* : \text{int} \cdot \text{int} \rightarrow \text{int}$   
 $+ : \text{int} + \text{int} \rightarrow \text{int}$

$$\{x:\text{int}, y:\text{int}, z:\text{int}\} \vdash x+y+z : \text{int}$$

# IMP: PROTOTIPO DI LINGUAGGIO INTERATIVO

TIPI = {bool, int}  $\ni \Sigma$

bool = {{tt, ff}, id<sub>bool</sub>}      int = {z, =}

IDENTIFICATORI = {var, cosa, ...}      ~~che~~ basta che iniziino con una lettera

Uop = operatori unari = {not, -, ...}  $\ni uop$

Bop = operatori binari = {+, -, \*, =, and, ...}  $\ni bop$

Con = costanti = {n/t  $\leftarrow$  n oppure t}

DATA lo sddetto notazione, definiamo la grammatica in BNF

C ::= nil |  $\leftarrow$  comando vuoto.  
id ::=  $\Sigma$  |  $\leftarrow$  assegnamento.

C; C |  $\leftarrow$  composizione di comandi.

if E then C else C |  $\leftarrow$  E = espressione con valore bool  
while E do C |  
p( $\sigma e$ ) |

|  $\leftarrow$  chiamata di procedura con fargappi  
di parentesi  $\sigma e$   
|  $\leftarrow$  dichiarazione di una procedura.

E ::= k | id | E bop E | uop E

Unione dei risultati delle dichiarazioni

D ::= nil | const x: z = E | var x: z = E | D; D | D and D |  
D in D | procedure p(form) C

form ::= • | const x: z, form | var x: z, form

$\sigma e ::=$  • | E,  $\sigma e$

SEMANTICA  $\vdash \langle L, M, E_{i \in I}, = \rangle$

||  
MAP

$(\_, \$) - \text{fun} = (\text{bad}, \{B, +\}) - \text{bad}$

new assign to street start  $\{ - \text{zero}, \text{one}\} = \text{it's a different id}$   
existing one

$\text{you} \in \{\text{you}, \text{+ bad}\} = \text{it's a different id} = \text{good}$

$\text{good} \in \{\text{you}, \text{+}, \text{-}, \text{+}\} = \text{it's a different id} = \text{good}$

$+ \text{one} \in \{+ \text{bad}, \text{+}\} = \text{it's a different id} = \text{good}$

an extension of extended, insertion of block d after

factory obvious  $\rightarrow | 10 = 2$   
otherwise  $\rightarrow | 3 = 1$

Answers to new questions  $\rightarrow | 3(2)$

food order was 3000000000  $\rightarrow | 3(2) \rightarrow \text{salt } 3(2)$

$| 3(2) \rightarrow \text{shallow}$

$(30)q$

apples were ordered as changing  $\rightarrow | 3(2)$   
300000000  $\rightarrow | 3(2)$   
answering one by one  $\rightarrow | 3(4)$

newish many  $\rightarrow | 3(2) | 3(2) | 3(2) | 3(2) \rightarrow | 3(2)$   
numbers still  $\rightarrow | 3(2) | 3(2) | 3(2) | 3(2) \rightarrow | 3(2)$

$| 3(2) | 3(2) | 3(2) | 3(2) \rightarrow | 3(2) | 3(2) | 3(2) | 3(2) \rightarrow | 3(2)$   
 $| 3(2) | 3(2) | 3(2) | 3(2) \rightarrow | 3(2) | 3(2) | 3(2) | 3(2) \rightarrow | 3(2)$

and, same  $\rightarrow | \text{not}, 3(2) | 3(2) | 3(2) \rightarrow | 3(2)$

$3(2) | 3(2) \rightarrow | 3(2)$

17/09/13

A insieme di valori.

TIPO =  $(A, = \subseteq A \times A)$

↑  
ugualanza decidibile

Axiom



Axiom



$$\frac{\Delta \vdash G_1, \dots, \Delta \vdash G_n}{\Delta \vdash G}$$

$\Delta = \{x_1 : \tau_1, \dots, x_n : \tau_n \mid i \neq j \Rightarrow x_i \neq x_j\}$   
ambiente statico.

Albero di derivazione

Ottivio semantico statico: determinare il maggior numero di errori prima di generare il codice.

Ci sono due tipi di errori:

- 1) TRAPPED: se quando si verifica blocca l'esecuzione del programma  
 2) UNTRAPPED: se non blocca il programma

↑  
semantico statico cerca questo tipo di errori.

TYPE = SYSTEM  $\rightarrow$  SAFE, se elimina tutti gli errori UNTRAPPED  
 (Tipi + Semantico statico)

$\boxed{x:y \text{ se } \Delta \vdash y:\tau \Rightarrow y \text{ err}}$

non ha errori UNTRAPPED.

ogni configurazione del mio programma

ha un tipo  $\tau$  nel senso che si riesce a costruire un Albero di derivazione con foglie ha Assiomi

ma il teorema precedente è inutile senza il

## SUBJECT REDUCTION

$$\boxed{\Delta \vdash f : \tau \wedge f \rightarrow f' \Rightarrow \Delta \vdash f' : \tau'}$$

↓      ↓      ↓

Partendo da una configurazione e applicando la semantica statica  $\Rightarrow$  otengo ancora una configurazione valida. tutte le configurazioni che ~~deriveranno~~ verranno derivate saranno corrette.

$$\boxed{f : \tau \vdash f : \tau' \Leftrightarrow f : \tau' \vdash f}$$

↓      ↓

semplicemente se ho una configurazione valida allora la sua derivazione è anche valida.

# SEMANTICA DINAMICA

Interpreta nodi di grafi del tipo:

$$(\langle P, G \rangle, \rightarrow_e \subseteq \langle P, G \rangle \times \langle P, G \rangle)$$

to una configurazione iniziale:

$$I = \langle P, G \rangle$$

e una finale

$$F = \langle P', G' \rangle$$

$$\Sigma \equiv \rightarrow_e^*, = \subseteq \{ \Sigma[\epsilon] \times \Sigma[\epsilon] \}$$

dobbiamo definire questa relazione per induzione, sulla struttura della sintassi.

$$K \triangleleft / \quad \left. \begin{array}{l} \\ \end{array} \right\} \text{Assiommi} \quad id \triangleleft / \quad \left. \begin{array}{l} \\ \end{array} \right\} \text{Regole} \quad id) \quad \rho \vdash \langle id, G \rangle \xrightarrow{e} \langle K, G \rangle, \\ G(\rho(id)) = K$$

$$\boxed{E_0} \triangleleft E_0 \text{ bop } E_1 \quad \left. \begin{array}{l} \\ \end{array} \right\} \text{Regole} \quad E_1 \triangleleft E_0 \text{ bop } E_1 \quad \left. \begin{array}{l} \\ \end{array} \right\} \text{proprie} \\ E \triangleleft vop E \quad \left. \begin{array}{l} \\ \end{array} \right\} E_0) \quad \rho \vdash \langle E_0 \text{ bop } E_2, G \rangle \xrightarrow{e} \\ \langle E_1 \text{ bop } E_2, G \rangle \xrightarrow{e} \\ \langle E_0 \text{ bop } E_1, G \rangle \xrightarrow{e} \\ \langle K_0 \text{ bop } K_1, G \rangle \xrightarrow{e} K$$

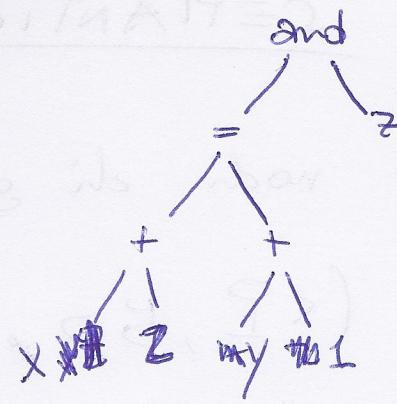
ipotesi  
induttiva

$E = [(x+z) = (y+1)] \text{ and } z$

$D = \{x: \text{int}, y: \text{int}, z: \text{int}\}$

$\mathcal{G} = \{x: l_x, y: l_y, z: l_z\}$

$L = \{l_x=1, l_y=2, l_z=\#t\}$



PROBLEMS

$\mathcal{G} \vdash < E, h_0 >$

①:

$\mathcal{G} \vdash < x, h_0 > \rightarrow_e < z, h_0 >$

$\mathcal{G} \vdash < x+z, h_0 > \rightarrow_e < z+z, h_0 >$

$\mathcal{G} \vdash < (x+z) = (y+1), h_0 > \rightarrow_e < (z+z) = (y+1), h_0 >$

$\mathcal{G} \vdash < E, h_0 > \rightarrow_e < [(z+z) = (y+1)] \text{ and } z, h_0 >$

②:

$\mathcal{G} \vdash < 1+z, h_0 > \rightarrow_e < 3, h_0 >$

$\mathcal{G} \vdash < (1+z) = (y+1), h_0 > \rightarrow_e < 3 = (y+1), h_0 >$

$\mathcal{P} \vdash < [(1+z) = (y+1)] \text{ and } z, h_0 > \rightarrow_e < [3 = (y+1)] \text{ and } z, h_0 >$

③  $\mathcal{G} \vdash < y, h_0 > \rightarrow_e < z, h_0 >$

$\mathcal{G} \not\vdash < y+1, h_0 > \rightarrow_e < z+z, h_0 >$

$\mathcal{G} \vdash < [3 = (y+1)], h_0 > \rightarrow_e < [3 = (z+z)], h_0 >$

$\mathcal{P} \vdash < [3 = (y+1)] \text{ and } z, h_0 > \rightarrow_e < [3 = (z+z)] \text{ and } z, h_0 >$

④ : v. pete  $\vdash_{\text{so}} (8+1)$

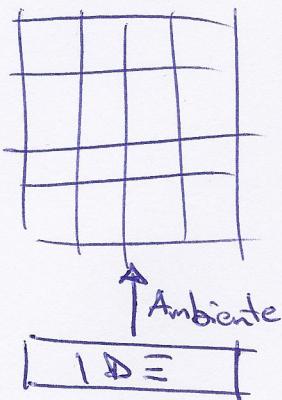
⑤ :  $\vdash \vdash \langle 3 = 3, h_0 \rangle \rightarrow_c \langle \#, h_0 \rangle$

$\vdash \vdash \langle 3 = 3 \rangle \text{and} \# \rightarrow \langle \# \text{ and } \#, h_0 \rangle$

⑥ :  $\vdash \vdash \langle ?, h_0 \rangle \rightarrow_c \langle \#, h_0 \rangle$

$\vdash \vdash \langle \# \text{ and } ?, h_0 \rangle \rightarrow_c \langle \# \text{ and } \#, h_0 \rangle$

110/03/2011



memoria della macchina astratta.  
 E:  $\text{Uloeg} \rightarrow \text{SVal} \cup \{\text{?}, \perp\}$

$$E: \text{Obj}_{\mathcal{E}} \rightarrow \text{SVal} \cup \{\text{?}, \perp\}$$

A diagram illustrating pointer assignment. It shows two memory locations. The first location contains the value "allocato" and is labeled "ma var ancora init". An arrow points from this location to the second location, which contains the value "non allocato re init".

$$\wp : \text{ID}\Xi \rightarrow \text{Loc}$$

$\Delta : IDE \rightarrow TIPI$

$\varphi \circ \Delta$  è se  $\varphi : I \rightarrow X$ ,  $\Delta : I \times I \rightarrow I$

$$E := k \mid id \mid E_0 \ bop \ E_1 \mid \lambda x. E$$

$E_{\text{val}} : E \times \text{Store} \rightarrow \text{Const}$

$$E_{\text{val}}(E, \mathbb{E}) = k \Leftrightarrow \langle E, \mathbb{E} \rangle \xrightarrow{e^*} \langle k, \mathbb{E}_0 \rangle$$

$$\langle E, \mathbb{E} \rangle \equiv \langle E', \mathbb{E}' \rangle \Leftrightarrow \forall \mathbb{E}. E_{\text{val}}(E, \mathbb{E}) = E_{\text{val}}(E', \mathbb{E}')$$

$$E \equiv_s E' \Leftrightarrow \forall \mathbb{E}. \langle E, \mathbb{E} \rangle \equiv_m \langle E', \mathbb{E}' \rangle$$

Dimostrare il Teorema di Subject Reduction per le espressioni

$$\frac{\begin{array}{c} \textcircled{1} \\ \Delta \vdash_I E : t \end{array}}{\Delta \vdash_I E : t} \quad \frac{\begin{array}{c} \textcircled{2} \\ g \vdash_\Delta \langle E, \mathbb{E} \rangle \rightarrow_e \langle E', \mathbb{E}' \rangle \Rightarrow \Delta \vdash_I E', \mathbb{E}' \end{array}}{\Delta \vdash_I E : t}$$

Dim:  $\text{BASE} = \{k, id\}$

k) Verifica  $\textcircled{1} \circ \textcircled{2}$

$$\text{id}) \Delta \vdash_I x : \mathcal{C}, \Delta(x) = \mathcal{C}$$

$$g \vdash \langle x, \mathbb{E} \rangle \rightarrow_e \langle k, \mathbb{E} \rangle, k = \mathbb{E}(g(x))$$

$$g : \Delta [\text{compatibile con}] \Rightarrow \lambda x. (\Delta(x) = \mathcal{C} \Leftrightarrow g(x) \in \text{loc } \mathcal{C}).$$

$$\downarrow \\ \mathbb{E}(g(x)) \in \mathcal{C}$$

HP IND:  $E_0, E_1$  soddisfano la Subject Reduction

$\exists \text{ goal}, \exists \text{ goal}, \exists \text{ Thm} \models \exists$

PASSO IND: Dimostrare Subject Reduction da  $E_0 \text{ bop } E_1$

$$\frac{\Delta \vdash E_0 : \tilde{x}_0, \Delta \vdash E_1 : \tilde{x}_1}{\Delta \vdash E_0 \text{ bop } E_1 : R_{\text{bop}}(\tilde{x}_0, \tilde{x}_1)}$$

$$\Delta \vdash E_0 : \tilde{x}_0, \Delta \vdash E_1 : \tilde{x}_1$$

$$\frac{P \vdash \langle E_0, \tilde{s} \rangle \rightarrow_e^c \langle E_0', \tilde{s}' \rangle}{P \vdash \langle E_0 \text{ bop } E_1, \tilde{s} \rangle \rightarrow_e^c \langle E_0' \text{ bop } E_1, \tilde{s}' \rangle}$$

$$P \vdash \langle E_0 \text{ bop } E_1, \tilde{s} \rangle \rightarrow_e^c \langle E_0' \text{ bop } E_1, \tilde{s}' \rangle$$

$$\frac{P \vdash \langle E_1, \tilde{s} \rangle \rightarrow_e^c \langle E_1', \tilde{s}' \rangle}{P \vdash \langle E_1, \tilde{s} \rangle \rightarrow_e^c \langle E_1' \text{ bop } E_0, \tilde{s}' \rangle}$$

$$P \vdash \langle E_1, \tilde{s} \rangle \rightarrow_e^c \langle E_1' \text{ bop } E_0, \tilde{s}' \rangle$$

$$\frac{P \vdash \langle K_0 \text{ bop } K_1, \tilde{s} \rangle \rightarrow_e^c \langle K, \tilde{s}' \rangle, K = k_0 \text{ bop } k_1}{P \vdash \langle K_0, \tilde{s} \rangle \rightarrow_e^c \langle K_1, \tilde{s}' \rangle}$$

$$\boxed{R_{\text{bop}}(\tilde{x}_0, \tilde{x}_1)}$$

$$\langle (\lambda x. x) z - x, \langle z, \lambda x. x \rangle z \rightarrow \langle z, x \rangle z \rangle \vdash$$

$$\langle (\lambda x. x) z - x, \langle z, \lambda x. x \rangle z \rightarrow \langle z, x \rangle z \rangle \vdash$$

$$\langle (\lambda x. x) z - x, \langle z, \lambda x. x \rangle z \rightarrow \langle z, x \rangle z \rangle \vdash$$

$E ::= k \mid id \mid \bar{E}_0 \text{ bop } E_1 \mid \bar{E} \text{ uop } \bar{E}$

estenderlo con: if  $\bar{E}_0$  then  $E_1$  else  $E_2$

- 2) semantica statica
- 3) semantica dinamica
- 4) Commentare.
- ?

$\Delta \vdash E_0 : \text{bool}, \Delta \vdash E_1, E_2 : \mathbb{K}$

$\Delta \vdash \text{if } \bar{E}_0 \text{ then } \bar{E}_1 \text{ else } \bar{E}_2 : \mathcal{C}$

3)

$\mathcal{P} \vdash \langle \bar{E}_0, L \rangle \rightarrow_e \langle \bar{E}'_0, L \rangle$

$\mathcal{P} \vdash \text{if } \bar{E}'_0 \text{ then } \bar{E}_1 \text{ else } \bar{E}_2, L \rangle \rightarrow_e \langle \text{if } \bar{E}'_0 \text{ then } \bar{E}'_1 \text{ else } \bar{E}'_2, L \rangle$

essendo corretto ② mi troverò:

$\mathcal{P} \vdash \langle \text{if } \# \text{ then } \bar{E}_1 \text{ else } \bar{E}_2, L \rangle \rightarrow_e \langle \bar{E}_1, L \rangle$

$\mathcal{P} \vdash \langle \text{if } \# \text{ then } \bar{E}_1 \text{ else } \bar{E}_2, L \rangle \rightarrow_e \langle \bar{E}_2, L \rangle$

Var  $\text{X} \oplus \text{int} = 1$   
 $X := 2 + Y$

posizione :  $\parallel$   
 istante o  
 occorrente

Procedure  $P(\text{var } Z: \text{int})$

var  $\text{X} \oplus \text{int} = 3$   
 $X := Z + 1$

Procedure  $reg(\text{var } W: \text{int})$

$X := W + 1$

var  $\text{Y} \oplus \text{int} = 2$

$q(3)$

$P(2)$   
 $X = 10$   
 $Y = 3$

Se esistono variabili  
 libere sicuramente  
 ci sono degli errori  
 di esecuzione



- legate

L'usato dopo  
 essere definiti  
 riesco a determinare  
 la ~~f~~ definizione  
 corrispondente

- libere

contrario delle  
 legate  $\Rightarrow$  non  
 riesco a trovare la  
 definizione

$$E ::= K \mid id \mid E_0 \text{ bop } E_1 \mid \text{loop } E$$

$$FI(K) = \emptyset$$

$$FI(id) = \{id\}$$

$$FI(E_0 \text{ bop } E_1) = FI(E_0) \cup FI(E_1)$$

$$FI(\text{loop } E) = FI(E)$$

Per capire se do  $E$  posso avere errori, devo controllare se ho delle variabili libere in  $E$ , tradotto:

$$FI(E) \subseteq I \Rightarrow \Delta \vdash_I E : \tau$$

$\exists x. x \in FI(E) \wedge x \notin I$  |  $I$  è l'insieme delle dichiarazioni

DIM.: per induzione sulla struttura della sintassi di  $E$

BASE:

K)  $FI(K) = \emptyset$

id)  $FI(id) = \{id\}$ , id  $\notin I$

Hp IND: vale per  $E_0, E_1 \in E$

$\Delta \vdash_I id : \tau, \Delta(id) = \perp$  |  $\begin{array}{l} \text{non} \\ \text{riesco ad} \\ \text{assegnare un} \\ \text{tipo ad id.} \end{array}$

P. IND:  $FI(E_0 \text{ bop } E_1) = FI(E_0) \cup FI(E_1)$

$\exists x \in FI(E_0 \text{ bop } E_1), x \notin I$

$\exists x \in FI(E_0) \Rightarrow$  Hp del teorema su  $E_0$ , per hp induttiva  
 V(ov) il teorema vale  $\Rightarrow$  non riesco ad assegnare un tipo ad  $=_0$

equivale a dire  $\exists x \in FI(E_1)$

HP IND: rate per  $\bar{e}$

$$\text{P IND: } \text{FI}(\cup_{\bar{e}} \bar{e}) = \text{FI}(\bar{e})$$

$$\underbrace{\exists x \in \text{FI}(\bar{e})}_{\downarrow =} \wedge x \notin I$$

$$\exists x \in \text{FI}(\bar{e}), x \notin I$$

$$\text{FI}(\bar{e}) \notin I \Rightarrow \Delta \not\models \bar{e} : \bar{e}$$