

IDENTIFICATORI → nomi usati per denotare oggetti

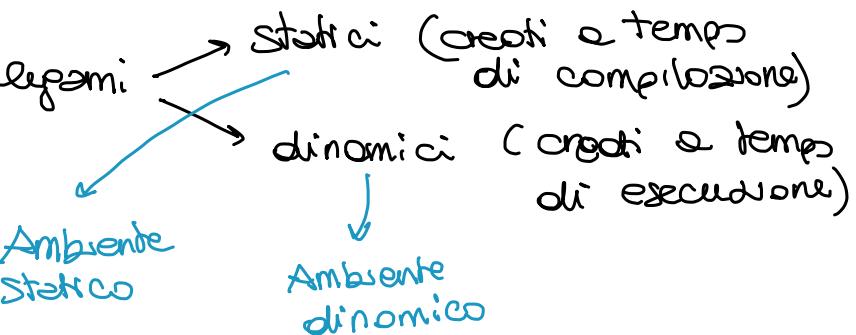
LEGAMI

⇒ esiste una associazione / legame tra identificatore e oggetto denotato

AMBIENTE

→ insieme di legami:

Ambiente Statico



VALORE DENOTABILE → valori che possono essere inferiti mediante identificatore

DVal insieme dei valori denotabili

Occorrente di Identificatori

libere

(non sono nel scope di alcuna definizione)

Uso - Definizione (occorrenza che definisce il significato)

↪ (occorrenza nel scope di alcuna di una definizione)

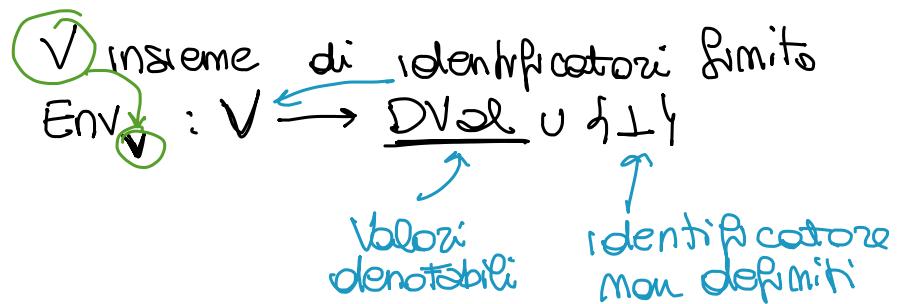
Se un programma non contiene identificatori → è un termine ground (Exp senza id)

Se un programma contiene identificatori ma non in posizione libera → è un termine

chiuso (ovvero ha significato anche bisogna chi altre informaz.)

Se un programma ha identificatori liberi non necessita di un ambiente per essere eseguito

AMBIENTE (dinamico)
 $\rho \in \text{Env}$



Associa ad ogni identificatore in V
 un valore in $D\text{Val} \cup \{\perp\}$

$\text{Env} = \bigcup_{V \subseteq_f \text{Id}} \text{Env}_V$
 Unione degli ambienti definiti su insiemi finiti di identificatori

$V = \{x, y\}$ Env_V funzione che associa a x e a y
 il valore denotato

Exp :

$E \rightarrow A \mid B$
 $A \rightarrow x \mid m \mid A \text{ op } A$ $\text{op} \in \{+, *, -\} \Rightarrow N \subseteq D\text{Val}$
 $B \rightarrow x \mid \text{true} \mid \text{false} \mid B \text{ or } B \mid \text{not } B \mid A = A$
 $\Rightarrow B = \{\text{true}, \text{false}\} \subseteq D\text{Val}$
 (identificatori costanti)

\Rightarrow Le grammatica Exp genera solo espressioni con identificatori liberi

$\boxed{p \vdash e_1 \rightarrow_e e_2}$
 Ambiente

Nell'ambiente p l'espressione e_1 viene valutata in e_2

Sintassi / Simboli	Semantica / significato
$\vdash m \text{ op } n \rightarrow p$	$p = m \text{ op } n$
$\vdash \text{not } t_0 \rightarrow t$	$t = \text{not } t_0$
$\vdash t_0 \text{ or } t_1 \rightarrow t$	$t = t_0 \text{ or } t_1$
$\vdash m = n \rightarrow t$	$\begin{cases} t = \text{true} & \text{se } m = n \\ t = \text{false} & \text{se } m \neq n \end{cases}$

$$\boxed{\vdash x \rightarrow k \quad \text{se } p(x) = k \in \text{DVal}}$$

$$\frac{\vdash e_1 \rightarrow e'_1}{\vdash e_1 \text{ op } e_2 \rightarrow e'_1 \text{ op } e_2}$$

$$\frac{\vdash e_2 \rightarrow e'_2}{\vdash m \text{ op } e_2 \rightarrow m \text{ op } e'_2}$$

$$\frac{\vdash e_1 \rightarrow e'_1}{\vdash e_1 \text{ bop } e_2 \rightarrow e'_1 \text{ bop } e_2}$$

$$\frac{\vdash e_2 \rightarrow e'_2}{\begin{aligned} &\vdash m = e_2 \rightarrow m = e'_2 \\ &\vdash t \text{ or } e_2 \rightarrow t \text{ or } e'_2 \end{aligned}}$$

$$\frac{\vdash e_0 \rightarrow e'_0}{\vdash \text{not } e_0 \rightarrow \text{not } e'_0}$$

Tipo \Rightarrow è l'insieme dei valori (e quindi delle operazioni) che possiamo associare ad un identificatore

Questi valori devono condividere un proprio strutturale

e le tipi deve rappresentare tutti i valori che hanno quelle proprietà strutturali

$\{m \mid m \text{ non }\}$

Non è un tipo

è tipo:

non è tipo:

Integer / int
bool
String
Int → Bool
:

if true, 5

Print

$$: f(x) = x^2 + x$$

A livello di progettazione → permettono di organizzare l'informazione

A livello di programmazione → permettono di prevenire errori (errori di tipo)

→ "controllo dimensionale"

$$B + C \neq 0$$

A livello di implementazione → permettono ottimizzazioni dell'uso della memoria

Abbiamo bisogno di creare legami id ↔ tipo
type binding (legame di tipo)

↪ se i legami per le dimensioni sono espliciti
allora i legami di tipo sono statici (gestiti a tempo di compilazione)

↪ Altrimenti i legami di tipo sono dinamici (gestiti a tempo di esecuzione)

$$\begin{array}{ccc} \rightarrow & = & \\ \rightarrow & \text{or} & \end{array}$$

$$\begin{array}{l} x = y \\ x \text{ or } y \end{array}$$

SEMANTICA STATICA

Ambiente statico osserva od opui identificatore il suo tipo

\underline{DTyp} = insieme dei tipi dei valori denotabili

$$DTyp = \{ \text{int}, \text{bool} \}$$

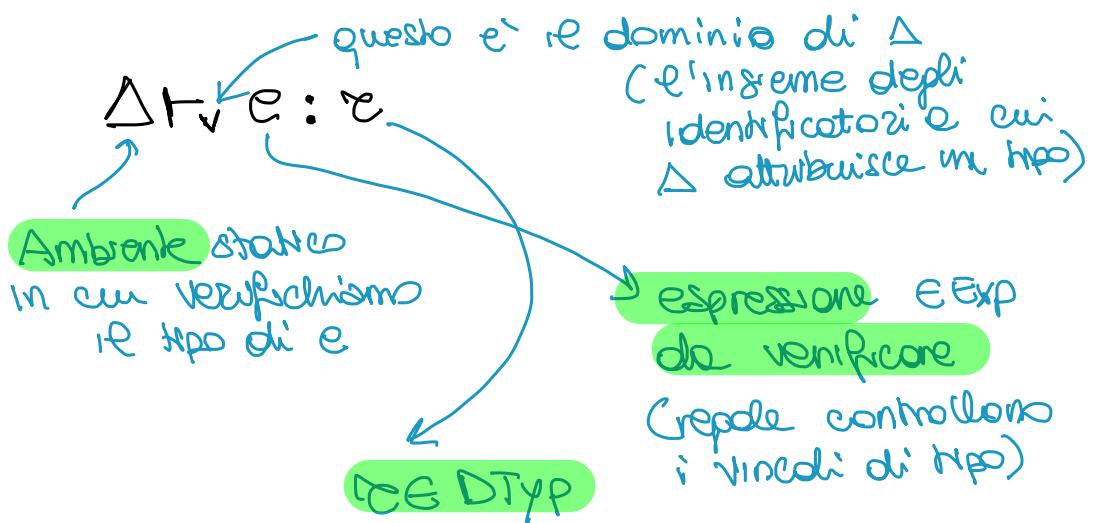
$V \subseteq_f Id$ insieme finito di identificatori

Δ metarawable per ambienti statici

$$\boxed{TEnv_V : V \rightarrow DTyp} \quad \left. \begin{array}{l} \text{essere od opui identificatore} \\ \text{in } V \text{ un tipo denotabile} \end{array} \right\}$$

$$\Delta \in TEnv = \bigcup_{V \subseteq_f Id} TEnv_V$$

↗ Semantica statica di Exp :



$$\Delta \vdash m : \text{int} \quad \text{MEN}$$

$$\Delta \vdash t : \text{bool} \quad \text{tGB}$$

$$\Delta \vdash x : \tau \quad \Delta(x) = \tau$$

$$\frac{\Delta \vdash e_1 : \text{bool} \quad \Delta \vdash e_2 : \text{bool}}{\Delta \vdash e_1 \text{ or } e_2 : \text{bool}}$$

$$\frac{\Delta \vdash e_0 : \text{bool}}{\Delta \vdash \text{not } e_0 : \text{bool}}$$

$$\frac{\Delta \vdash e_1 : \text{int} \quad \Delta \vdash e_2 : \text{int}}{\Delta \vdash e_1 \text{ op } e_2 : \text{int}}$$

$\text{op} \in \{+, -, *\}$

$$\frac{\Delta \vdash e_1 : \text{int} \quad \Delta \vdash e_2 : \text{int}}{\Delta \vdash e_1 = e_2 : \text{bool}}$$

Semantica dinamico usa $\rho \in \bar{\text{Env}}$ $V \rightarrow D\text{Val}$

Semantica statica usa $\Delta \in \bar{\text{Env}}$ $V \rightarrow D\text{Typ}$

Ambienti combuibili (ρ e Δ) :

- Δ e ρ sono compatibili se $\exists V \subseteq_p \text{Id}$ t.c. $\Delta : V \rightarrow D\text{Typ}$
- $\rho : V \rightarrow D\text{Val}$

$\rho \vdash_{\Delta}$

e $\forall id \in V$. se $\Delta(id) = \infty$

allora $\rho(id) \in \infty$

valore di tipo ∞

valutazione nell'ambiente dinamico ρ compatibile con l'ambiente statico Δ

DICHIARAZIONI

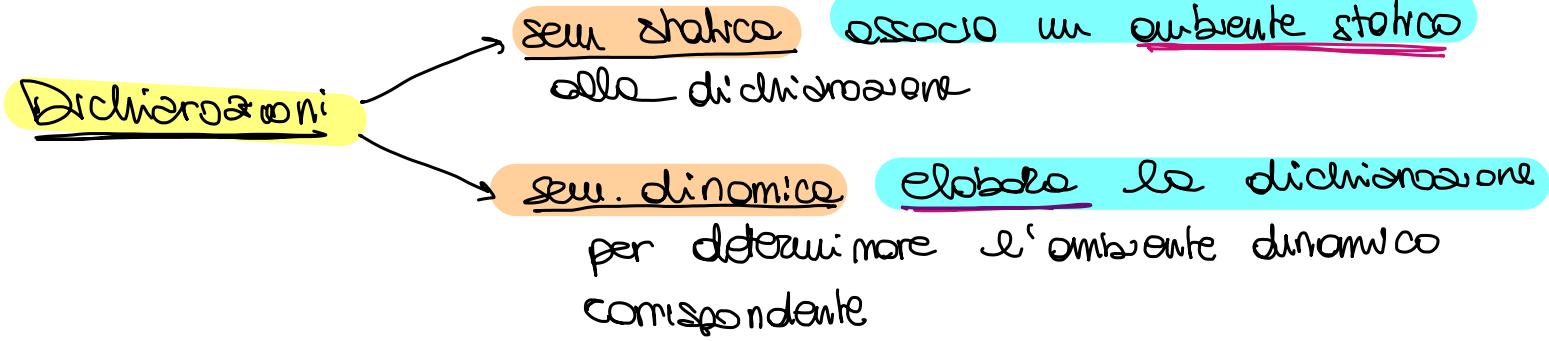
↪ è lo strumento sintattico per creare e manipolare legami, ovvero ambienti

sem. dinamico valuta l'espressione per determinare il valore associato

Espressioni

sem statico associa un tipo all'espressione
(se non contiene errori)

(non riesce ad associare un tipo altrimenti)



Dichiarazioni in IMP

Dec $D \rightarrow \text{nil} \mid \text{const } x : c = e \mid D ; D \mid D \text{ in } D \mid P$

componibile con le dichiarazioni di x , costante di tipo c e di valore rappresentato da e

composizione sequenziale
comp. privata

pe Env