TIPI

sono oggetti che:

- *sono assegnati* alle strutture del programma

- *servono a classificare* tali strutture per studiare la correttezza (semantica) della loro composizione

- *sono descritti* da opportune espressioni (sistema di tipi)

UN SISTEMA DI TIPI

definisce:

- le *espressione di tipo*
  (forma degli oggetti)

- *regole per* assegnare tipi alle strutture

Espressioni di tipo (una semplice struttura)

1) *tipi basici*

real, int, char, file, Null

2) *costruttori* (tipi strutturati o derivati)

array: I x T -> array(I,T)

prodotto: T1 x T2 -> T1 ✕ T2

record:

({i1}xT1)x..x({ik}xTk) -> record(i1:T1..ik:Tk)

enumerato: {v1,..,vk} -> (v1,..,vk)

pointer: T -> pointer(T)

funzione: TD x TC -> TD ➔ TC

procedura: TD -> TD ➔ Null

3) *identificatori*

4) *variabili*

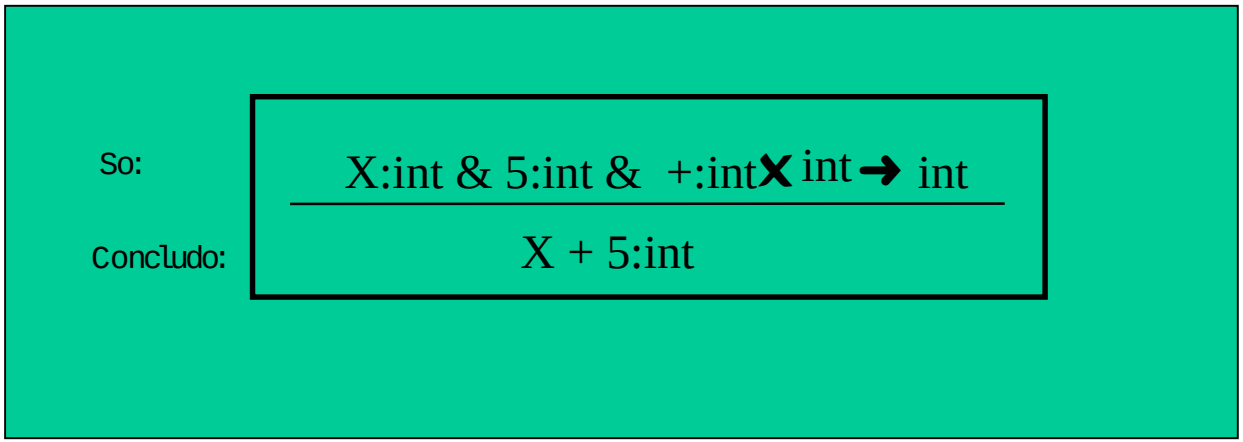
Regole per assegnare espressioni di tipo

dipendono dal linguaggio:

- struttura sintattica
- semantica

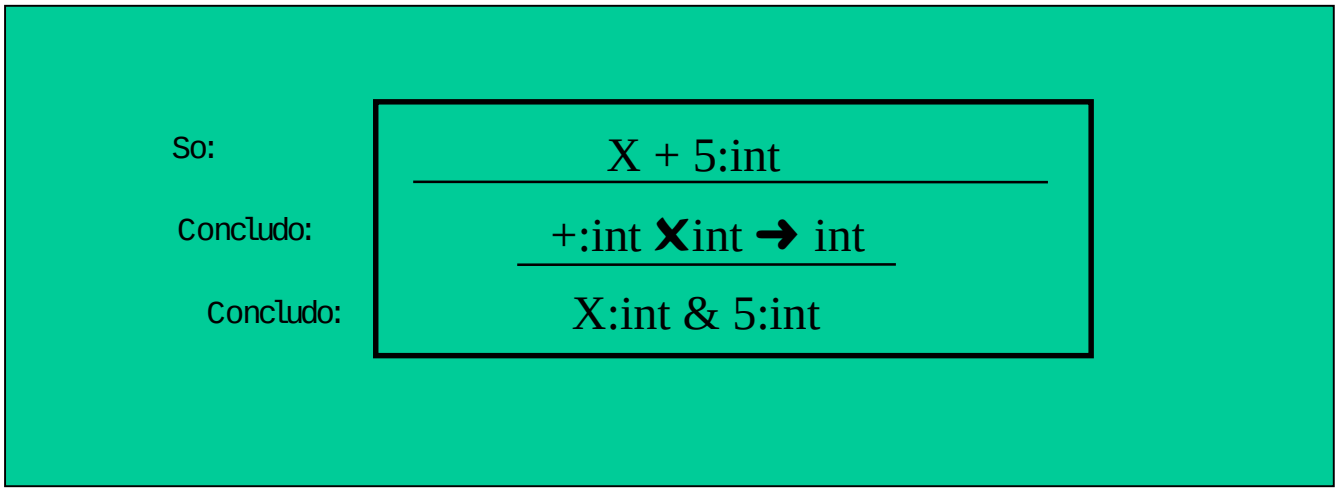
analisi di tipi

forward reasoning



Inferenza di tipi

backward reasoning



Type Checking (1)

- Estendiamo il linguaggio con tipi basici: una grammatica
- Pianifichiamo un’analisi dei tipi: symbol-table
- Definizione degli attributi:ereditiamo la lista degli identificatori
- Usiamo un’altra grammatica:
  - Ereditiamo il tipo: ma non e’ l-attributata
  - Usiamo solo sintetizzati

Una grammatica LL(1) per il Semplice linguaggio di programmazione al quale aggiungiamo i tipi *basici*

[0]**P**rogram= **D**eclaration**s** **C**omman**d**s | [1]**C**omman**d**s

[2]**Ds**::= **Var Dt**yped**s**

[3]**Dts**::= **Dt Dts'**

[4]**Dts'** ::= **; Dt Dts'** | [5] ε

[6]**Dt**::= **ide O**theridentifiers

[7]**O**::= , **ide O** | [8] **:** t**Y**pe

[9]**Cs**::= **;** Command **Cs** | [10] ε
[11]**C**::= **A**ssign | [12] **W**hile
[13]**A**::= **ide := E**xpression
[14]**W**::= **while E do C Cs endw**hile

[15]**E**::= **F E'**
[16]**E'**::= **op-l**ower **F E'** | [17] ε
[18]**F**::= **T**erm **F'**
[19]**F'**::= **op-h**ight **T F'** | [20] ε
[21]**T**::= **num** | [22] **ide** | [23] **(E)**

[24]**Y**::= **boolean** | [25] **integer** | [26] **file** | ...

Associamo nella Sym-Table tipi alle variabili

attributi: entry=locazione in sym-table
t=espressione di tipo di**Y**
ty=lista variabilii

entry=sintetizzato (scanner) solo per **ide**
t=sintetizzato solo per **Y**
ty=ereditato solo per dichiarazioni

operiamo effetti laterali (SDD)

operazioni primitive su Sym-Table

Addtype: ide X type

[0]**P**::= **Ds Cs**	
[1]**P**::= **Cs**	
[2]**Ds**::= **Var Dts**	
[3]**Dts**::=**Dt Dts'**	
[4]**Dts'** ::= **; Dt Dts'**	
[5] **Dts'** ::= ε	
[6]**Dt**::=**ide O**	**O**.ty:= cons(**ide**.entry,**Dt**.ty)
[7]O_1::= , **ide** O_2	O_2.ty:=cons(**ide**.entry,**Dt**.ty)

[8]**O**::= **: Y**	temp:=**O**.ty;
	repeat {addtype(car(temp),**Y**.t);
	temp:=cdr(temp)}
	until isempty(temp)

[9]Cs_1::= **; C** Cs_2 [10]**Cs**::= ε [11]**C**::= **A** [12]**C**::=**W** [13]**A**::= **ide := E** [14]**W**::=**while E do C Cs edw** [15]**E**::= **F E'** [16]**E'**::= **op-l F E'** [17]**E'**::= ε [18]**F**::= **T F'** [19]**F'**::= **op-h T F'** [201]**F'**::= ε [21]**T**::= **num** [22]**T**::= **ide** [23]**T**::= **(E)**	
[24]**Y**::= **boolean**	**Y**.t:= boolean
[25]**Y**::= **integer**	**Y**.t:= integer
[26]**Y**::= **file**	**Y**.t:= file

Usiamo un’altra grammatica

attributi: entry=locazione in sym-table
t=espressione di tipo di**Y**
ty=espressione di tipo

entry=sintetizzato (scanner) solo per **ide**
t=sintetizzato solo per **Y**
ty=ereditato solo per dichiarazioni

[0]**P**::= **Ds Cs**	
[1]**P**::= **Cs**	
[2]**Ds**::= **Var Dts**	
[3]**Dts**::=**Dt : Y Dts'**	**Dt**.ty:= **Y**.t
[4]**Dts'** ::= **; Dt : Y Dts'**	**Dt**.ty:=**Y**.t
[5] **Dts'** ::= ε	
[6]**Dt**::=**ide O**	**O**.ty:= **Dt**.ty, addtype(**ide**.entry,**Dt**.ty)
[7]$\mathbf{O_1}$::= , **ide** $\mathbf{O_2}$	$\mathbf{O_2}$.ty:=$\mathbf{O_1}$.ty, addtype(**ide**.entry,$\mathbf{O_1}$.ty)
[8]**O**::= ε [9]$\mathbf{Cs_1}$::= **; C** $\mathbf{Cs_2}$ [10]**Cs**::= ε [11]**C**::= **A** [12]**C**::=**W** [13]**A**::= **ide := E** [14]**W**::=**while E do C Cs edw** [15]**E**::= **F E'** [16]**E'**::= **op-l F E'** [17]**E'**::= ε [18]**F**::= **T F'** [19]**F'**::= **op-h T F'** [201]**F'**::= ε [21]**T**::= **num** [22]**T**::= **ide** [23]**T**::= **(E)**	
[24]**Y**::= **boolean**	**Y**.t:= boolean
[25]**Y**::= **integer**	**Y**.t:= integer
[26]**Y**::= **file**	**Y**.t:= file

La precedente grammatica non L-attributata

perche’ nella produzione [4]
Dt eredita dal fratello destro **Y**

Possiamo usare attributi in modo differente

attributi: entry=lista delle locazioni in sym-table
t=espressione di tipo di**Y**

entry=sintetizzato (scanner) solo per **Ide** e per **Dt**
t=sintetizzato solo per **Y**

[0]**P**::= **Ds Cs**	
[1]**P**::= **Cs**	?
[2]**Ds**::= **Var Dts**	
[3]**Dts**::=**Dt : Y Dts'**	addtype-set(**Dt**.entry,**Y**.t)
[4]**Dts'** ::= **; Dt : Y Dts'**	addtype-set(**Dt**.entry,**Y**.t)
[5] **Dts'** ::= ε	
[6]**Dt**::=**ide O**	**Dt**.entry:= cons(**ide**.entry,**O**.entry)
[7]O_1::= , **ide** O_2	O_1.entry:=cons(**ide**.entry,O_2.entry)
[8]**O**::= ε	**O**.entry:=emptylist
[9]Cs_1::= **; C** Cs_2 [10]**Cs**::= ε [11]**C**::= **A** [12]**C**::=**W** [13]**A**::= **ide := E** [14]**W**::=**while E do C Cs edw** [15]**E**::= **F E'** [16]**E'**::= **op-l F E'** [17]**E'**::= ε [18]**F**::= **T F'** [19]**F'**::= **op-h T F'** [20]**F'**::= ε [21]**T**::= **num** [22]**T**::= **ide** [23]**T**::= **(E)**	
[24]**Y**::= **boolean**	**Y**.t:= boolean
[25]**Y**::= **integer**	**Y**.t:= integer
[26]**Y**::= **file**	**Y**.t:= file

Nel precedente piano:

usiamo solo sintetizzati

la grammatica ci consente di farlo con poca fatica

Anche la prima grammatica consentiva di usare solo sintetizzati. Un differente piano

attributi: entry=locazione in sym-table
t=espressione di tipo di **Y**

entry=sintetizzato (scanner) solo per **ide**
t=sintetizzato solo per **Y**,**O**

Una differente definizione degli attributi

Dt::=**ide O** addtype(**ide**.entry,**O**.t)

Completare con il calcolo del sintetizzato .t di O

Conclusioni (dall'esempio):

Scegliere la gramm. con attenzione

grammatiche differenti conducono a piani di attributi differenti e con attributi differenti per tipo (ered., sint.) e per numero

Scegliere il piano con attenzione

alcune grammatiche consentono piu' piani di attributi di differente difficolta'

per una grammatica LL (LR) si possono trovare anche piani non L-attributati

Associamo espressioni di tipo alle strutture di un programma del linguaggio Semplice

regole di inferenza:

$$\frac{i:t \quad e:t}{i:=e:N} \qquad \frac{e:bool \quad c_S:N}{\text{while e do } c_S:N} \qquad \frac{c:N \quad c_S:N}{c;c_S:N}$$

$$\frac{e_1:t_1 \quad e_2:t_2 \quad op: t_1 \times t_2 \rightarrow t}{op\ e_1\ e_2 : t}$$

attributi: type=tipo delle espressioni
in=tipo sottoespressione sinistra
r=tipo comandi

type=sintetizzato
in=ereditato sottoespressione sinistra
r=sintetizzato (true=N, false=error)

[0]**P**::= **Ds Cs**	**P**.r:= **Cs** .r
[1]**P**::= **Cs**	**P**.r:= **Cs** .r
[2]**Ds**::= **Var Dts**	
[3]**Dts**::=**Dt : Y Dts'**	addtype-set(**Dt**.entry,**Y**.t)
[4]**Dts'** ::= **; Dt : Y Dts'**	addtype-set(**Dt**.entry,**Y**.t)
[5] **Dts'** ::= ε	
[6]**Dt**::=**ide O**	**Dt**.entry:= cons(**ide**.entry,**O**.entry)
[7]O_1::= , **ide** O_2	O_1.entry:=cons(**ide**.entry,O_2.entry)
[8]**O**::= ε	**O**.entry:=emptylist
[9]Cs_1::= **; C** Cs_2	Cs_1.r::= **C**.r & Cs_2.r
[10]**Cs**::= ε	**Cs**.r::= true
[11]**C**::= **A**	**Cs**.r::= **A**.r
[12]**C**::=**W**	**Cs**.r::= **W**.r
[13]**A**::= **ide := E**	**A**.r:= (**ide**.type = **E**.type), **E**.in:=N
[14]**W**::= **while E do C Cs edw**	**W**.r:= (**E**.type=boolean)&**C**.r & **Cs**.r **E**.in:=N

[16] **E** ::= **F E'**	**E**.type:=**E'**.type, **F**.in:=**E**.in, **E'**.in:=**F**.type
[17] **E'**$_1$::= **op-l F E'**$_2$	**F**.in:=N let **op-l**.type = $t_1 \times t_2 \rightarrow t$ in if (t_1=**E'**$_1$.in) & (t_2=**F**.type) then begin **E'**$_2$.in:=t; **E'**$_1$.type:=**E'**$_2$.type end else **E'**$_1$.type:= error
[18] **E'** ::= ε	**E'**.type:=**E'**.in
[25] **Y** ::= **boolean**	**Y**.t:= bool
[26] **integer**	**Y**.t:= integer
[27] **file**	**Y**.t:= file

Osservazioni:

l’attributo .in e’ereditato dai simboli: **E**, **E’**, **F**, **F’**

solo le produzioni [17], di E’, e [19] di F’, richiedono tale attributo

allora, **E** ed **F** potrebbero non avere attributo .in e la grammatica semplificata