Type system del linguaggio Toy3

Prof. Gennaro Costagliola
Corso di Compilatori A.A. 2024/25

December 11, 2024

Le seguenti regole di tipo **coprono solo parzialmente** l'intero linguaggio.

Le regole sono formate dall'*antecedente*, che è situato nella parte alta della regola e che può anche non essere presente, ed il *conseguente* che è la parte bassa della regola.

Ci sono due modi per leggere le regole: dichiarativo e operativo. Si prenda ad esempio la regola per l'identificatore qui di sotto.

dichiarativo o top-down : **se** l'operazione di lookup di id nell'ambiente O restituisce il tipo τ **allora** è dimostrato che id è ben tipato in O ed ha tipo τ in O;

operativo o bottom-up : per verificare che id sia ben tipato e calcolare il suo tipo nel type environment O bisogna fare il lookup di id in O e, se presente, usare il tipo restituito τ come suo tipo di id.

Quando si usano le regole per sviluppare l'analizzatore semantico conviene leggere le regole dal basso verso l'alto, dal *conseguente* all'*antecedente*, in modo operativo. Nel conseguente (parte bassa), il costrutto fra i simboli ` e :, deve sempre fare riferimento ad un nodo dell'albero sintattico.

Identificatore

$$\frac{O(id) = \tau}{O \vdash id : \tau}$$

Costanti

$O \vdash double\_const : double$
$O \vdash int\_const : integer$
$O \vdash string\_const : string$
$O \vdash char\_const : string$
$O \vdash true : boolean$
$O \vdash false : boolean$

Lista di istruzioni

$$\frac{O \grave{} \mathit{stmt}_1 : \mathit{notype} \quad O \grave{} \mathit{stmt}_2 : \mathit{notype}}{O \grave{} \mathit{stmt}_1; \mathit{stmt}_2 : \mathit{notype}}$$

Lettura operativa: per vedere se la sequenza di statements $\mathit{stmt}_1; \mathit{stmt}_2$ (in basso fra ` e :) è ben tipata in O, bisogna vedere se entrambi stmt_1 e stmt_2 sono ben tipati in O. Nel caso lo siano, poichè sono istruzioni e quindi non hanno tipo di ritorno, alla sequenza verrà assegnato il tipo fittizio *notype*.

Chiamata a funzione

$$\frac{O \grave{} f : \tau_1 \times \ldots \times \tau_n \to \sigma \quad O \grave{} e_i : \tau_i^{i \in 1 \ldots n}}{O \grave{} f(e_1, \ldots, e_n) : \sigma}$$

Chiamata a procedura senza controllo del parametro ref

$$\frac{O \grave{} f : \tau_1 \times \ldots \times \tau_n \to \mathit{notype} \quad O \grave{} e_i : \tau_i^{i \in 1 \ldots n}}{O \grave{} f(e_1, \ldots, e_n) : \mathit{notype}}$$

Assegnazione multipla senza controllo per assenza di chiamate a funzioni

$$\frac{O(id_i) : \tau_i^{i \in 1 \ldots n} \quad O \grave{} e_i : \tau_i^{i \in 1 \ldots n}}{O \grave{} id_1, \ldots, id_n := e_1, \ldots, e_n : \mathit{notype}}$$

Blocco dichiarazione-istruzione
(il type environment dell'istruzione stmt viene esteso con la dichiarazione di id, prima di fare il controllo di tipo dell'istruzione stmt):

$$\frac{O[id \mapsto \tau] \grave{} \mathit{stmt} : \mathit{notype}}{O \grave{} \tau \ id; \mathit{stmt} : \mathit{notype}}$$

Istruzione while

$$\frac{O \grave{} e : \mathit{boolean} \quad O \grave{} \mathit{body} : \mathit{notype}}{O \grave{} \textbf{while } (e) \textbf{ do } \mathit{body} \ : \mathit{notype}}$$

Istruzione if-then-else

$$\frac{O \grave{} e : \mathit{boolean} \quad O \grave{} \mathit{body}_1 : \mathit{notype} \quad O \grave{} \mathit{body}_2 : \mathit{notype}}{O \grave{} \textbf{if } (e) \textbf{ then } \mathit{body}_1 \textbf{ else } \mathit{body}_2 : \mathit{notype}}$$

Operatori unari (vedi Table 1):

$$\frac{O \vdash e : \tau_1 \quad optype1(op_1, \tau_1) = \tau}{O \vdash op1\ e : \tau}$$

Operatori binari (vedi Table 2):

$$\frac{O \vdash e_1 : \tau_1 \quad O \vdash e_2 : \tau_2 \quad optype2(op_2, \tau_1, \tau_2) = \tau}{O \vdash e_1\ op_2\ e_2 : \tau}$$

Tabelle per gli operatori:

op1	operando	risultato
MINUS	integer	integer
MINUS	double	double
NOT	boolean	boolean

Table 1: Tabella per *optype1*. Esempio di uso: *optype1*(NOT, boolean) = boolean

op	operando1	operando2	risultato
PLUS, TIMES, ...	integer	integer	integer
PLUS, TIMES, ...	integer	double	double
PLUS, TIMES, ...	double	integer	double
PLUS, TIMES, ...	double	double	double
PLUS	string	string	string
AND, OR	boolean	boolean	boolean
LT, LE, GT, ...	integer	integer	boolean
LT, LE, GT, ...	double	integer	boolean
LT, LE, GT, ...	integer	double	boolean
LT, LE, GT, ...	double	double	boolean

Table 2: Tabella per *optype2*. Esempio di uso: *optype2*(TIMES, double, integer) = double