Gruppo Studio 16 Maggio

$A \to \overset{\checkmark}{A}\gamma_1$

$A \to \delta_1$

$\longrightarrow$

$A \to \delta_1 A$

$A' \to \gamma_1 A'$

$A' \to \varepsilon$

$\text{Exp} : \overset{A\ \gamma_1}{\text{Exp} + \text{Exp}} \mid \overset{A\ \gamma_2}{\text{Exp} - \text{Exp}} \mid \overset{\delta_1}{X} \mid \overset{\delta_2}{N}$

$\Downarrow$

$\text{Exp} : X\,\text{Exp}' \mid N\,\text{Exp}'$

$\text{Exp}' : +\text{Exp}\,\text{Exp}' \mid -\text{Exp}\,\text{Exp}' \mid \varepsilon$

Corso di Laurea Magistrale in Informatica

Compito di Compilatori e Interpreti

20 Luglio 2020

Nota Bene. Quando avete terminato, fare una foto a tutto il compito col cellulare usando una applicazione che esegue scansioni, tipo CamScanner, e inviarla per email a `cosimo.laneve@unibo.it` .

Si consideri la seguente grammatica (scritta in ANTLR)

```
prg :  'let' dec 'in' stm ;
dec :  ('int' Id ';')+ ;
exp :  Integers | Id | exp '+' exp ;
stm :  (Id '=' exp ';')+
```

dove

- gli `Integers` sono sequenze non vuote di cifre prefissate dal segno + o -;
- gli `Id` sono gli identificatori (sequenze non vuote di caratteri);

Esercizi

1. (**punti 2**) completare l'input di ANTLR con le regole per l'analizzatore lessicale che riguardano `Integers` e `Id`;
2. (**punti 9**) dare tutte le regole di inferenza per verificare l'uso di identificatori non inizializzati. Ad esempio `let int x; int y; in x = 3 + y ;` è un programma erroneo secondo l'analisi semantica. L'analisi semantica ritorna anche informazioni sull'offset degli identificatori (vedi punto 4);
3. (**punti 4**) verificare, scrivendo l'albero di prova, che il programma seguente sia correttamente tipato:
   `let int x; int y; in y = 5 ; x = 3 + y ;`

1.

Integers:= Sign Number

Sign:= + | -

Number:= 0 | [1..9][0..9]*

Id := [a..z | A..Z]+

2.

$$\frac{\emptyset\cdot[\,]\vdash dec : \Gamma, n \quad \Gamma, n \vdash stm}{\emptyset, 0 \vdash \text{let } dec \text{ in } stm}\ [prog]$$

$$\frac{\Gamma, n \vdash d : \Gamma', n' \quad \Gamma', n' \vdash D : \Gamma'', n''}{\Gamma, n \vdash d D : \Gamma'', n''}$$

$$\frac{x \notin dom(top(\Gamma))}{\Gamma, n \vdash \text{int } x; : \Gamma', n'}\ \ \overset{n+1}{\Gamma'}$$

$[x \to dec, n]$

$$\frac{+ : int \times int \to int \quad \Gamma, n \vdash e_1 : int \quad \Gamma \vdash e_2 : int}{\Gamma, n \vdash e_1 + e_2 : int}$$

3

$$\frac{}{\Gamma, n \vdash num : int}$$

$$\frac{\Gamma(x) = T}{\Gamma, n \vdash x : T}$$

$$\frac{\Gamma, n \vdash s : \Gamma', n \quad \Gamma', n \vdash S : \Gamma'', n}{\Gamma, n \vdash s S : \boxed{\Gamma'', n}}$$

stm $\left\{ \dfrac{\Gamma \vdash x : dec \quad \Gamma \vdash e : int}{\Gamma, n \vdash x = e ; \Gamma', n} \right.$

$[x \to int, n]$
$x \to dec$

[int x; [x → dec]

[x = 3 [x → int]

y = x + 1 ✓

[x → dec]

$\Gamma \cdot [y \to dec]$

Γ''
$\begin{bmatrix} x \to dec \\ y \to int \end{bmatrix}$

$$\cfrac{\cfrac{\cfrac{x \notin dom(top(\emptyset\cdot[\,]))}{\emptyset\cdot[\,]\vdash \text{int } x : \Gamma, n}\ \overset{\Gamma}{[x \to dec]} \quad \cfrac{y \notin dom(top(\Gamma))}{\Gamma \vdash \text{int } y : \Gamma', n'}\ \overset{\Gamma'}{\begin{bmatrix} x \to dec \\ y \to dec \end{bmatrix}}}{\emptyset\cdot[\,]\vdash \text{int } x; \text{int } y : \Gamma', n'} \to \cfrac{\cfrac{\cfrac{\Gamma'(y) = dec}{\Gamma', n \vdash y : dec} \quad \cfrac{}{\Gamma', n' \vdash 5 : int}}{\Gamma', n' \vdash y = 5 : \Gamma'', n'} \to \cfrac{\cfrac{\Gamma''(x) = dec}{\Gamma'', n' \vdash x : dec} \quad \cfrac{\cfrac{}{\Gamma'', n' \vdash 3 : int} \quad \cfrac{\Gamma''(y) = int}{\Gamma'', n' \vdash y : int}}{\Gamma'', n' \vdash 3 + y : int}}{\Gamma'', n' \vdash x = 3 + y ; \Gamma''', n'}\ [SeqS]}{\Gamma', n' \vdash y = 5 ; x = 3 + y}}{\emptyset \vdash \text{let int } x ; \text{int } y ; \text{ in } y = 5 ; x = 3 + y}\ [prog]$$

$\begin{bmatrix} x \to int \\ y \to int \end{bmatrix}$

21 Giugno 2022

Si consideri il linguaggio di programmazione la cui sintassi in ANTLR è la seguente:

```
prog : '{' (dec)? (stm)+ '}' ;
dec  : 'int' (ID ',')* ID ';';
stm  : ID '=' exp ';' | 'if' '(' exp ')' stm 'else' stm | block  ;
block: '{' (dec)? (stm)+ '}' ;
exp  : ID | NUM | exp ('+' | '-') exp ;
ID   : ('a'..'z')+ ;
NUM  : ('0'..'9')+ ;
```

$\Gamma \to \Gamma'$ $\Gamma \to \Gamma''$

dove la guardia del condizionale è vera quando `exp` è diverso da 0, falsa altrimenti.

Un identificatore `ID` è considerato "*costante*" quando viene assegnato una sola volta (si trova una sola volta come left-hand side expression di un assegnamento) e gli eventuali identificatori della right-hand side expression sono anch'essi costanti. [Immaginare che se un identificatore è costante nel tempo, allora è possibile rimuoverlo rimpiazzandolo con il valore costante.]

Esercizi

1. (**punti 9**) definire tutte le regole di inferenza per verificare gli identificatori costanti e per gestire gli offset nella generazione di codice. In particolare
   - definire il dominio astratto e le operazioni su di esso da usare nelle regole semantiche e dimostrare che le operazioni sono monotone;
   - per quanto riguarda gli offset, la memoria per eseguire il codice dovrà essere tutta in un unico frame (non si dovranno usare liste di frames);
2. (**punti 6**) scrivere gli alberi di prova per i seguenti programmi:

```
{ int x, y, z ; x = 1; y = 2 ; if (x + y) { z= x+y ;} else { z = 4 ;} x = z+y ; }

{ int x, y, z ; x = 1; y = 2 ; if (x - x) { z= x+y ;} else { z = 3 ;} x = x+1 ; }

{ int x, y ; if (x + y) { x = 1; y = 2 ; } else { x = y +1 ; } }
```

1.

(prog)

$$\frac{\emptyset\cdot[\,], 0 \vdash dec : \Gamma, n \quad \Gamma, n \vdash stm}{\emptyset, 0 \vdash \{dec \ stm\}}\ (prog)$$

$$\frac{\emptyset\cdot[\,], 0 \vdash stm}{\emptyset, 0 \vdash \{stm\}}\ (prog2)$$

$$\frac{x \notin dom(top(\Gamma))}{\Gamma, n \vdash \text{int } x, : \Gamma', n'}$$

$[x \to dec]$

bl

$$\frac{\Gamma, n \vdash d : \Gamma', n' \quad \Gamma', n \vdash D : \Gamma'', n''}{\Gamma, n \vdash d D; : \Gamma'', n''}$$

$$\frac{x \notin dom(top(\Gamma))}{\overset{n''}{\Gamma}, n \vdash \text{int } x; : \Gamma', n'}$$

$(x \to dec)$

zero o cost o int

$choose(T, \Gamma', \Gamma'') = \Gamma''', n'$

$$\frac{\Gamma, n \vdash e : T \qquad \overset{ock}{\Gamma, n \vdash S_1 : \Gamma', n'} \quad \overset{block}{\Gamma, n \vdash S_2 : \Gamma''}}{\Gamma, n \vdash \text{if } (e) \text{ then } S_1 \text{ else } S_2 : \Gamma''', n'}$$

$$\frac{\Gamma\cdot[\,], n \vdash dec : \Gamma', n' \quad \Gamma', n' \vdash stm}{\Gamma, n \vdash \{dec \ stm\}}$$

$$\frac{\Gamma, n \vdash stm}{\Gamma, n \vdash \{stm\}}$$

$$\frac{\Gamma(x) = T}{\Gamma, n \vdash x : T}$$

$$\frac{}{\Gamma, n \vdash num : int}$$

$$\frac{\underset{(-)}{+} : int \times int \to int \quad \Gamma, n \vdash e_1 : int \quad \Gamma, n \vdash e_2 : int}{\Gamma, n \vdash e_1 \underset{(-)}{+} e_2 : int}$$

$$\frac{}{\Gamma, n \vdash 0 : zero}$$

$$\frac{T \neq zero \quad \Gamma \vdash x : dec \quad \Gamma \vdash e : T}{\Gamma, n \vdash x = e : [x \to cost], n}$$

$$\frac{T \neq zero \quad \Gamma \vdash x : cost \quad \Gamma \vdash e : T}{\Gamma, n \vdash x = e : [x \to T], n}$$

$$\frac{\Gamma \vdash x : cost \quad \Gamma \vdash e : cost}{\Gamma, n \vdash x = e : \Gamma}$$

$$\frac{\Gamma, n \vdash x : T \quad \Gamma \vdash e : zero}{\Gamma, n \vdash x = e : [x \to zero], n}$$

$$\frac{T_2 \neq zero \quad \Gamma, x : T_1 \quad \Gamma \vdash e : T_2 \quad T_1 = T_2}{\Gamma, n \vdash x = e : \Gamma, n}$$

28/05/21

Compito di Compilatori e Interpreti

28 Maggio 2021

Nota Bene. Quando avete terminato, fare una foto a tutto il compito col cellulare usando una applicazione che esegue scansioni, tipo CamScanner, e inviarla per email a `cosimo.laneve@unibo.it` .

I programmi di un linguaggio di programmazione sono blocchi `Dec Stm` dove

- `Dec` sono sequenze di dichiarazioni di identificatori interi (int);
- `Stm` sono sequenze di comandi che possono essere
  - assegnamenti di una espressione `Exp` a una variabile;
  - iterazioni while (la guardia del condizionale è una espressione intera, la semantica è quella di `C`).
- `Exp` possono essere costanti intere, identificatori o espressioni con somma.

Esercizi

1. (**punti 6**) definire l'input *completo* di `ANTLR` per la grammatica del linguaggio di sopra;
2. (**punti 9**) dare tutte le regole di inferenza per verificare il corretto uso degli identificatori (identificatori non dichiarati o di dichiarazioni multiple) e per gestire gli offset nella generazione di codice.

Prg : {Dec Stm}

Dec : (type ID';') +

Stm : (ID'='exp | 'while' '('exp')' '{' stm '}'

type : 'int'

exp : Id | $\overset{\text{assegno}}{\text{Int}}$ | exp + exp

2.

$$\frac{\emptyset, 0 \vdash dec : \Gamma, n \quad \Gamma, n \vdash stm}{\emptyset, 0 \vdash \{Dec \ Stm\}}$$

$$\frac{x \notin dom(top(\Gamma))}{\Gamma, n \vdash \text{int } x; : \Gamma', n'}$$

$[x \to int]$

$$\frac{\Gamma, n \vdash d : \Gamma', n' \quad \Gamma', n' \vdash D : \Gamma'', n''}{\Gamma, n \vdash d D : \Gamma'', n''}$$

$$\frac{T1 = T2 \quad \Gamma, n \vdash x : T1 \quad \Gamma, n \vdash e : T2}{\Gamma, n \vdash x = e}$$

$$\frac{\Gamma, n \vdash e : int \quad \Gamma, n \vdash stm : \Gamma, n}{\Gamma, n \vdash \text{while}(e)\{ stm \} : \Gamma, n}$$

$$\frac{\Gamma(x) = T}{\Gamma, n \vdash x : T}$$

$$\frac{}{\Gamma, n \vdash num : int}$$

$$\frac{\Gamma, n \vdash e_1 : int \quad \Gamma, n \vdash e_2 : int \quad + : int \times int \to int}{\Gamma, n \vdash e_1 + e_2 : int}$$