Corso di Laurea Magistrale in Informatica

Compito di Compilatori e Interpreti

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')+ ;
```

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 ; } }
   ```

3. (**punti 9**) definire il codice intermedio *per tutti i costrutti del linguaggio*, in particolare tenendo conto del vincolo che non ci dovranno essere più frames.