Corso di Laurea Magistrale in Informatica

Compito di Compilatori e Interpreti

19 Settembre 2019

Esercizio 1 (6 punti). Definire un analizzatore lessicale in `ANTLR` che accetta sequenze di token che a loro volta sono stringhe **non vuote** sull'alfabeto a, b, c, per cui le occorrenze di a (se ci sono) precedono le occorrenze di b e di c (se ci sono) e le occorrenze di b precedono quelle di c (se ci sono). Ad esempio `a abbc bcc c` è un input riconosciuto.

Esercizio 2 (9 punti). Si assuma di avere un linguaggio object oriented.

1. Definire la regola semantica per l'overriding, ossia la regola che permette di tipare

```
class A { ...
    T1 m (T2 x) { ...}
...}
class B extends A { ...
    T3 m (T4 x) { ...}
...}
```

   [Suggerimento: definire la regola di tipo per il costrutto `class B extends A`, assumendo, per semplicità che `B` abbia esattamente un metodo. Fare i due casi che il metodo sia sovrascritto oppure no]

2. Definire anche la funzione `checkDecs` che implementa la regola del punto precedente.

Esercizio 3 (9 punti). Definire la funzione `code_gen` per il comando

```
whileTre E do { C } { C' } od
```

che (1) calcola `E` e sia v il suo valore; (2) se $v = 0$ allora termina, altrimenti se v è un multiplo di 3 esegue `C`, altrimenti esegue `C'`.

Quindi applicare le regole di sopra al comando

```
whileTre (x+y) { y := y+1; x := x-2 } { x= x-1 } od
```

assumendo che la variabile `x` si trovi ad offset $+4$ del frame pointer `$fp`, mentre la variabile `y` si trova nell'ambiente statico immediatamente esterno all'ambiente corrente e a offset $+8$.