Programmazione A. A. 2004/2005

V° Appello del 8/07/2005

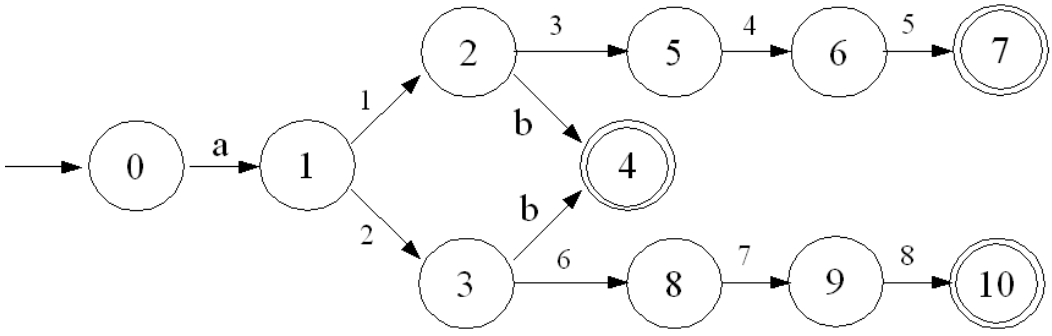
ISTRUZIONI: Scrivere **in stampatello** COGNOME e NOME su ogni foglio. Non occorre consegnare la brutta copia e il testo. Coloro che non vogliono consegnare possono andarsene, consegnando il testo, dopo un'ora dall'inizio del compito ed entro 15 minuti dalla scadenza del tempo.

ESERCIZIO 1 (5 punti)

Un timbratore automatico non deterministico quando viene azzerato è pronto per stampare una stringa codice. Per prima cosa stampa una a, poi o il numero 1 o il numero 2. A questo punto può stampare una b e azzerarsi, oppure stampare in sequenza i numeri 3 4 e 5, se precedentemente era stato stampato il numero 1, o i numeri 6, 7 e 8, se precedentemente era stato stampato il numero 2. Dopo aver stampato l'ultimo numero si azzera.

Scrivere un automa che accetti tutte e sole le stringhe stampabili dallo stampatore fra un azzeramento e l'altro.

SOLUZIONE



ESERCIZIO 2 (11 punti)

Si supponga di estendere la sintassi dei comandi del linguaggio didattico come segue:

```
Com ::= restore Ide after (L);
L ::= Com -> L | Com -> Com
```

L'esecuzione di un comando `restore x after (List);` consiste nell'eseguire **in sequenza** i comandi di `List` e poi ripristinare il valore della variabile `x` con quello che aveva inizialmente. Se `x` non aveva un valore allora il sistema si deve bloccare. Lo stato ottenuto è quello successivo all'esecuzione della lista con il ripristino finale.

Definire un sottosistema di transizioni $\rightarrow_{list}$ per l'esecuzione della lista `List` di comandi, con le relative regole guidate dalla sintassi. Definire poi le regole di semantica operazionale per il nuovo comando utilizzando il sottosistema definito. Si faccia riferimento al modello in cui lo stato è composto da una pila di frame.

SOLUZIONE

Il sottosistema di transizioni ha come insieme di configurazioni $\Gamma = \{\langle \texttt{List}, \sigma\rangle \mid \texttt{List} \in \texttt{L}, \sigma \in \Sigma\} \cup \Sigma$. Le configurazioni terminali sono l'insieme Σ di tutti gli stati possibili. Le regole che definiscono la relazione di transizione sono nel seguito. Esse sono scritte secondo lo stile big step e la loro definizione è guidata dalle produzioni per la categoria sintattica `L`.

$$list_{list} \quad \frac{\langle \texttt{C}, \sigma\rangle \rightarrow_{com} \sigma', \quad \langle \texttt{List}, \sigma'\rangle \rightarrow_{list} \sigma''}{\langle \texttt{C -> List}, \sigma\rangle \rightarrow_{list} \sigma''}$$

$$list_{com} \quad \frac{\langle \texttt{C1}, \sigma\rangle \rightarrow_{com} \sigma', \quad \langle \texttt{C2}, \sigma'\rangle \rightarrow_{com} \sigma''}{\langle \texttt{C1 -> C2}, \sigma\rangle \rightarrow_{list} \sigma''}$$

Per quanto riguarda il comando basta semplicemente scrivere una nuova regola per $\rightarrow_{com}$ che chiami il sottosistema preoccupandosi di fare il ripristino del valore della variabile indicata. La premessa $\sigma(\texttt{x}) = \underline{\texttt{v}}$ impone che la variabile `x` abbia un valore definito. In caso contrario la regola non può essere applicata e la configurazione risulta quindi bloccata come richiesto.

$$com_{restore} \quad \frac{\sigma(\texttt{x}) = \underline{\texttt{v}} \quad \langle \texttt{List}, \sigma\rangle \rightarrow_{list} \sigma' \quad \sigma'' = \sigma'[\underline{\texttt{v}}/\texttt{x}]}{\langle \texttt{restore x after (List);}, \sigma\rangle \rightarrow_{com} \sigma''}$$

ESERCIZIO 3 (7 punti)

Si scriva l'implementazione del seguente metodo:

```
\** Restituisce il numero di elementi uguali in posizioni
    uguali di due array
 @param a un array di interi
 @param b un array di interi
 @return quanti elementi di a e b sono nella stessa posizione
         e sono lo stesso valore. Se gli array sono di
         dimensioni diverse si guardano solo le posizioni comuni
 */
public int m(int[] a, int[] b) {...}
```

SOLUZIONE

```
public int m(int[] a, int[] b) {
  if (a == null || b == null) return 0;
  int conta = 0;
  int stop;
  if (a.length > b.length)
    stop = b.length;
  else stop = a.length;
  for (int i = 0; i < stop; i++)
    if (a[i]==b[i]) conta++;
  return conta;
}
```

ESERCIZIO 4 - PER GLI STUDENTI DA 5 CFU (7 punti)

Si modifichi la porzione di grammatica introdotta nell'ESERCIZIO 2 in modo da poter inserire:

- la parola 'as' seguita da una dichiarazione con inizializzazione di una variabile di tipo base, il tutto opzionale, prima della parola `after`
- una lista di dichiarazioni di variabili di tipo base:
  - con o senza inizializzazione
  - separate da spazi
  - la lista deve essere racchiusa da parentesi tonde
  - la lista può anche essere vuota (ma le parentesi vanno comunque inserite)
  - la lista va inserita dopo la parola '`after`'

Si assumano come definite le categorie sintattiche `Decl` ed `Exp` viste a lezione.

SOLUZIONE

```
Com ::= restore Ide A
A ::= as T Ide = Exp; B | B
B ::= after (C) (L)
C ::= Decl C | epsilon
L ::= Com -> L | Com -> Com
T ::= int | boolean
```

ESERCIZIO 4 - PER GLI STUDENTI DA 6 CFU (7 punti)

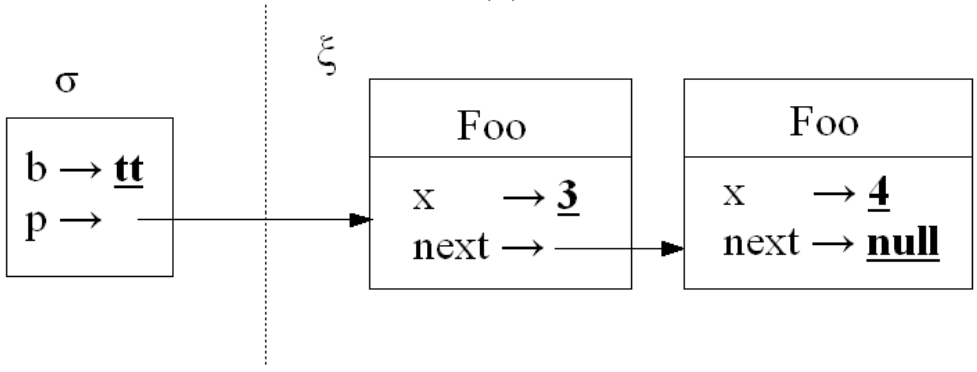
Si consideri la seguente definizione di classe e il successivo blocco main:

```
class Foo {
  public int x;
  public Foo next;
  public void m(int x) {
      this.x = this.x * x;
  }
  public boolean add(int x) {
    if (next==null) {
      this.next = new Foo();
      this.next.x = x;
      this.next.next = null;
      return true;
    } else return false;
  }
}{
Foo p = new Foo; p.x = 3; p.next = null;
boolean b = p.add(4); (1)
Foo r = p.next;
boolean c = p.add(3);
if (c) b = r.add(2);
else r.m(5);          (2)
if (b) r.add(6);
r.next.m(r.x);        (3)
}
```

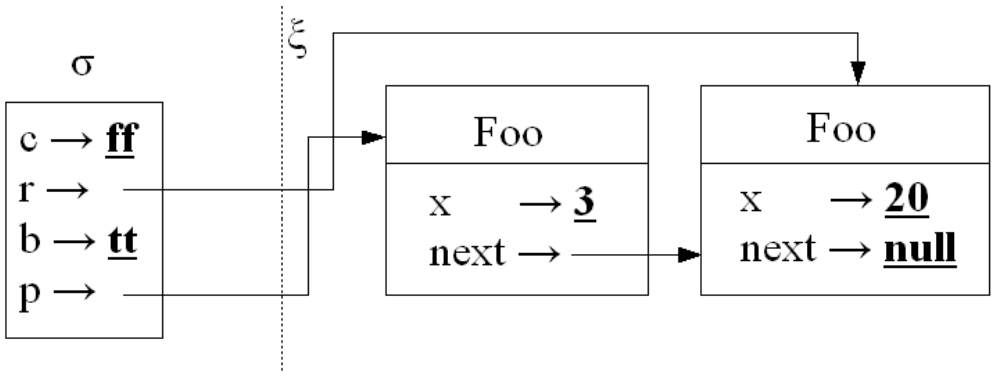
Si disegni lo stato, composto dalla pila di frame e dallo heap, nei punti del programma (1), (2) e (3).

SOLUZIONE

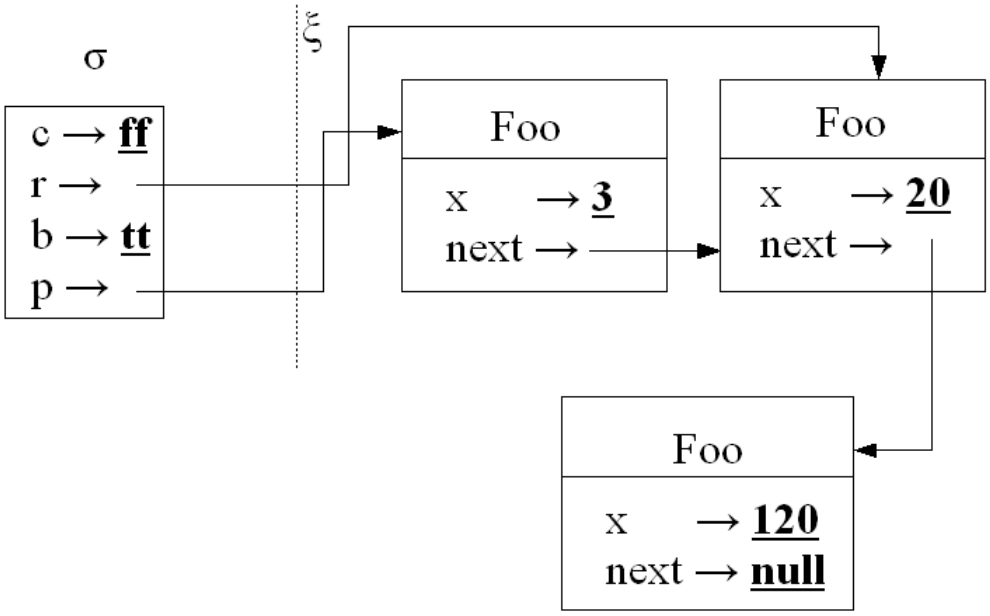
Punto (1)



Punto (2)


σ
c → ff
r →
b → tt
p →
Foo
x → 3
next →
Foo
x → 20
next → null

Punto (3)


σ
c → ff
r →
b → tt
p →
Foo
x → 3
next →
Foo
x → 20
next →
Foo
x → 120
next → null