PROGRAMMAZIONE – VIII Appello del 23/09/2003 — Camerino

Scrivere **in stampatello** COGNOME, NOME e NUMERO DI MATRICOLA (se conosciuto) su ogni foglio consegnato e sul testo, che va consegnato insieme al compito.

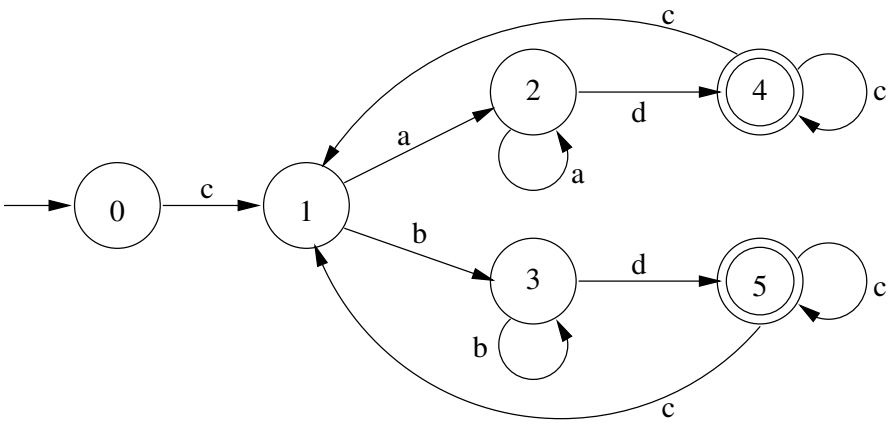
ESERCIZIO 1 (6 punti)

Siano $L_1 = \{a^n\ d\ c^m \mid n > 0,\ m \geq 0\}$ e $L_2 = \{b^n\ d\ c^m \mid n > 0,\ m \geq 0\}$ due linguaggi su $\{a, b, c, d\}^*$. Disegnare un automa che accetti il seguente linguaggio:

$$L = \{c\ s_1\ c\ s_2\ \cdots\ c\ s_k \mid k > 0,\ s_i \in L_1 \cup L_2 \text{ per } i = 1, 2, \ldots, k\}$$

SOLUZIONE

Si tratta di fare gli automi per L_1 ed L_2 e poi fonderli nella maniera indicata.



ESERCIZIO 2 (10 punti)

Si supponga di voler estendere il linguaggio delle espressioni aritmetiche visto a lezione con le *espressioni condizionali multiple* . Una espressione condizionale multipla è una lista di una o più espressioni con guardia. Una espressione con guardia è una espressione preceduta da una espressione booleana e da due punti. Le espressioni con guardia sono separate da virgole, cominciano con la parola `cond` e terminano con la parola `dnoc`. Ad esempio:

`cond x > 1 : y + 7 * 9, p && q : pippo + 4, z == 0 : z+1 dnoc`

è una espressione condizionale multipla di 3 espressioni con guardia. La prima espressione con guardia è `x > 1 : y + 7 * 9` dove `x > 1` è la guardia e `y + 7 * 9` è l'espressione.

(1) Estendere il linguaggio dalla categoria sintattica `Exp` facendo in modo che generi anche le espressioni condizionali multiple.

(2) Definire un sottosistema di transizioni del sistema $\rightarrow_{exp}$ per valutare le espressioni condizionali multiple. Il valore di una espressione condizionale multipla deve essere il valore dell'espressione della prima espressione con guardia la cui guardia è vera. Le espressioni con guardia devono essere considerate dalla

prima a partire da sinistra. Se nessuna guardia è vera, allora il valore deve essere quello dell'espressione dell'ultima espressione con guardia.

SOLUZIONE

La grammatica migliore è quella che genera le espressioni con guardia della lista a destra. In questo modo l'ordine di valutazione imposto dalla semantica si può seguire più agevolmente.

```
<Exp> ::= ... | cond <CondExpList> dnoc
<CondExpList> ::= <GuardExp> | <GuardExp> , <CondExpList>
<GuardExp> ::= <Exp> : <Exp>
```

Per la semantica definiamo un sottosistema $\rightarrow_{condexplist}$ che ha come insieme di configurazioni $\{\langle \mathtt{L}, \sigma\rangle \mid \mathtt{L} \in <\mathtt{CondExpList}>,\ \sigma \in \Sigma\} \cup \{\underline{v} \mid \underline{v} \in \mathbb{N} \cup \{\underline{tt}, \underline{ff}\}\}$. La prima regola è quella per far partire il sottosistema non appena si deve valutare una espressione condizionale con guardia:

$$exp_{condexplist} \quad \frac{\langle \mathtt{L}, \sigma\rangle \rightarrow_{condexplist} \underline{v}}{\langle \mathtt{cond\ L\ dnoc}, \sigma\rangle \rightarrow_{exp} \underline{v}}$$

Le seguenti tre regole definiscono il sottosistema. Esse usano la ricorsione a destra della sintassi per definire ricorsivamente la semantica. Si noti che il sistema principale e il sottosistema sono mutualmente ricorsivi, cioè ognuno chiama l'altro. La terminazione è garantita dal fatto che ogni chiamata viene effettuata sempre su una espressione con un albero di derivazione più piccolo.

$$condexplist_{tt} \quad \frac{\langle \mathtt{E1}, \sigma\rangle \rightarrow_{exp} \underline{tt}, \quad \langle \mathtt{E2}, \sigma\rangle \rightarrow_{exp} \underline{v}}{\langle \mathtt{E1\ :\ E2\ ,\ L}, \sigma\rangle \rightarrow_{condexplist} \underline{v}}$$

$$condexplist_{ff} \quad \frac{\langle \mathtt{E1}, \sigma\rangle \rightarrow_{exp} \underline{ff}, \quad \langle \mathtt{L}, \sigma\rangle \rightarrow_{condexplist} \underline{v}}{\langle \mathtt{E1\ :\ E2\ ,\ L}, \sigma\rangle \rightarrow_{condexplist} \underline{v}}$$

$$condexplist_{last} \quad \frac{\langle \mathtt{E2}, \sigma\rangle \rightarrow_{exp} \underline{v}}{\langle \mathtt{E1\ :\ E2}, \sigma\rangle \rightarrow_{condexplist} \underline{v}}$$

ESERCIZIO 3 (7 punti)

Si scriva il codice del metodo `public boolean pred(int [] a, int [] b)` che riceve in ingresso come parametri due puntatori ad array di interi `a` e `b` e restituisce `true` o `false`. Il metodo deve controllare se tutti gli elementi dell'array `a` sono minori di tutti gli elementi dell'array `b` e in tal caso restituire `true`. Nel caso contrario la risposta deve essere `false`.

SOLUZIONE

```
public boolean pred(int [] a, int [] b) {
  // Per fare il controllo mi basta trovare il massimo dell'array a
  // ed il minimo dell'array b e poi confrontare questi.
```

```
  int i;
  int max;
  int min;
  if (a.length != 0)
    max = a[0];
  else return true; // la condizione e' banalmente vera
  if (b.length != 0)
    min = b[0];
  else return true; // idem
  // Massimo di a:
  for (i=1; i < a.length; i++)
    if (a[i] > max)
      max = a[i];
  //Minimo di b:
  for (i=1; i < b.length; i++)
    if (b[i] < min)
      min = b[i];
  // Controllo finale:
  if (max < min)
    return true;
  else
    return false;
}
```

ESERCIZIO 4 (7 punti)

Si consideri il seguente programma:

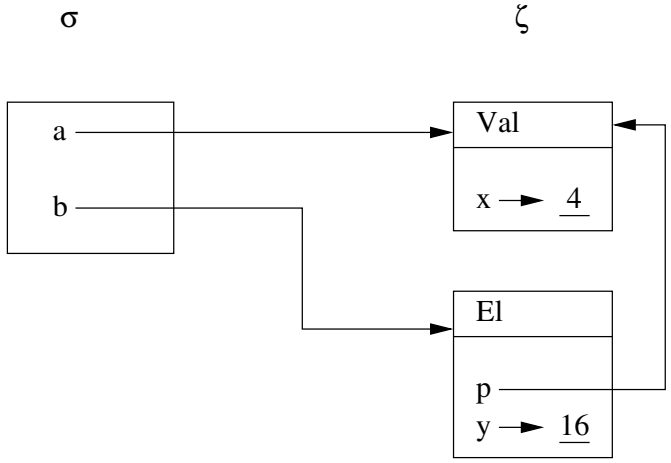
```
prog {
class Val {
  public int x;

  public int inc() {
    this.x++;
  }
}

class El {
  public Val p;
  public int y;

  public void calc(int q) {
    if (p != null) {
      p.inc();
      this.y = p.x * q;
    } else this.y = 0;
  }
}
{
  Val a = new Val;
```

```
  a.x = 3;
  El b = new El;
  b.p = a;
  b.calc(4); (1)
  a = null;
  El c = b;
  b.p.inc();
  c.calc(2); (2)
}
}
```

Si disegni lo stato (pila di frame e heap) nei punti `(1)` e `(2)` del programma.

SOLUZIONE

Al punto `(1)`:



Al punto `(2)`: