PROGRAMMAZIONE – VIII Appello del 24/09/2003 — Ascoli

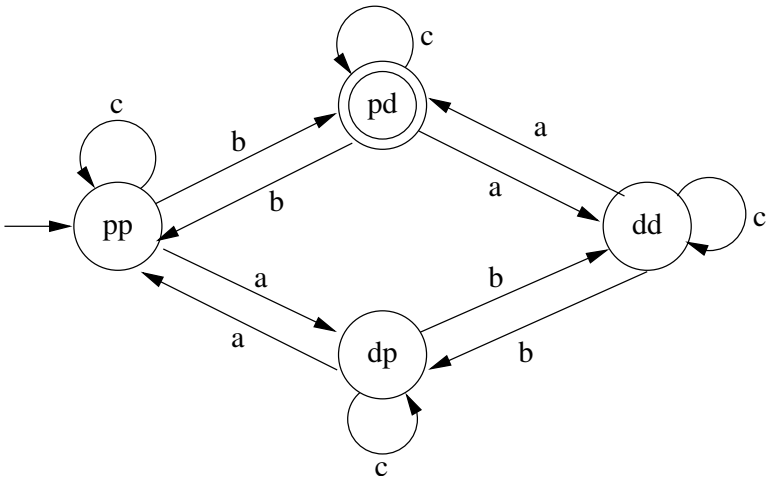
Scrivere **in stampatello** COGNOME, NOME e NUMERO DI MATRICOLA (se conosciuto) su ogni foglio consegnato e sul testo, che va consegnato insieme al compito.

ESERCIZIO 1 (6 punti)

Sia $\Lambda = \{a, b, c\}$. Si disegni un automa che accetti tutte e sole le parole di Λ^* che contengono un numero pari di `a` e un numero dispari di `b`. Il numero delle `c` non importa. Ovviamente zero è da considerarsi numero pari.

SOLUZIONE

Definiamo un automa con quattro stati in cui ogni stato conserva l'informazione che vogliamo, cioè se il numero di `a` e `b` lette è pari o dispari. Ci sono quattro possibilità: `pp` (`a` pari e `b` pari), `pd` (`a` pari, `b` dispari), `dp` e `dd`.



ESERCIZIO 2 (6 punti)

Si descriva formalmente il linguaggio generato dalla seguente grammatica:

```
<S> ::= p<A>q | a<B>b | x<S>y
<A> ::= 1<A> | epsilon
<B> ::= <B>0 | 0
```

SOLUZIONE

$$L(<\texttt{S}>) = \{x^n\ p\ 1^m\ q\ y^n \mid n \geq 0,\ m \geq 0\} \cup \{x^n\ a\ 0^m\ b\ y^n \mid n \geq 0,\ m > 0\}$$

ESERCIZIO 3 (6 punti)

Si supponga di estendere la sintassi dei comandi con la seguente produzione:
`Com = alt while (Exp) do Com, Com tla`
Un ciclo `alt while (E) do C1, C2 tla` si comporta come un ciclo while alternante. Esso cicla fintantoché la condizione `E` è vera. Alla prima iterazione, che è considerata una iterazione dispari, viene eseguito il comando `C1` e poi viene fatto ripartire il ciclo. Se il ciclo riparte, allora siamo alla seconda iterazione, che è considerata pari, e quindi il comando da eseguire è C2, dopodiché il ciclo viene fatto ripartire. Fino a quando la condizione diventa falsa, il ciclo si comporta nello stesso modo, cioè esegue `C1` nelle iterazioni dispari (la terza, la quinta e così via) e `C2` in quelle pari (la quarta, la sesta e così via).
Definire le regole di semantica operazionale per il while alternante. Si consiglia di utilizzare un sottosistema di transizione apposito in cui nello stato viene aggiunto qualcosa che indica se il ciclo si trova in una iterazione pari oppure dispari.

SOLUZIONE

Definiamo un sottosistema $\rightarrow_{altw}$ del sistema $\rightarrow_{com}$. Le configurazioni del nuovo sistema sono quelle dell'insieme $\{\langle \texttt{alt while (E) do C1 , C2 tla}, n, \sigma\rangle \mid \texttt{E} \in <\texttt{Exp}>,\ \texttt{C1}, \texttt{C2} \in <\texttt{Com}>,\ n \in \{0,1\},\ \sigma \in \Sigma\} \cup \Sigma$. Il valore zero per il numero n nella configurazione sta ad indicare che il sistema si trova attualmente ad effettuare una iterazione pari, mentre il numero uno, una iterazione dispari. La prima regola da scrivere è quella per far partire il sottosistema non appena si deve eseguire un while alternante. Siccome la prima iterazione è dispari, inseriamo il numero 1 nella configurazione di partenza.

$$com_{altw} \quad \frac{\langle \texttt{alt while (E) do C1 , C2 tla}, 1, \sigma\rangle \rightarrow_{altw} \sigma'}{\langle \texttt{alt while (E) do C1 , C2 tla}, \sigma\rangle \rightarrow_{com} \sigma'}$$

La regola seguente vale sia per le iterazioni pari che per quelle dispari. Infatti se la guardia è falsa il while alternante esce in tutti e due i casi senza modificare lo stato.

$$altw_{ff} \quad \frac{n \in \{0,1\}, \quad \langle \texttt{E}, \sigma\rangle \rightarrow_{exp} \underline{ff}}{\langle \texttt{alt while (E) do C1 , C2 tla}, n, \sigma\rangle \rightarrow_{altw} \sigma}$$

Se invece la guardia è vera il comportamento cambia a seconda della iterazione. Ad esempio, se la iterazione è dispari ($n = 1$) allora si deve eseguire il comando `C1` e richiamare ricorsivamente tutto il while sul nuovo stato ottenuto e indicando che la prossima è una iterazione pari ($n = 0$).

$$altw_{tt-dis} \quad \frac{\langle \texttt{E}, \sigma\rangle \rightarrow_{exp} \underline{tt}, \quad \langle \texttt{C1}, \sigma\rangle \rightarrow_{com} \sigma', \quad \langle \texttt{alt while (E) do C1 , C2 tla}, 0, \sigma'\rangle \rightarrow_{altw} \sigma''}{\langle \texttt{alt while (E) do C1 , C2 tla}, 1, \sigma\rangle \rightarrow_{altw} \sigma''}$$

$$altw_{tt-pari} \quad \frac{\langle \texttt{E}, \sigma\rangle \rightarrow_{exp} \underline{tt}, \quad \langle \texttt{C2}, \sigma\rangle \rightarrow_{com} \sigma', \quad \langle \texttt{alt while (E) do C1 , C2 tla}, 1, \sigma'\rangle \rightarrow_{altw} \sigma''}{\langle \texttt{alt while (E) do C1 , C2 tla}, 0, \sigma\rangle \rightarrow_{altw} \sigma''}$$

ESERCIZIO 4 (6 punti)

Si scriva il codice del metodo `public boolean m(int [] a, int[] b)` che riceve in ingresso due puntatori ad array di interi `a` e `b`. Il metodo deve trovare l'elemento minimo nell'array `a` e andare a cercare se questo si trova nell'array `b`. In caso positivo il metodo deve ritornare `true` e in caso contrario `false`. Il metodo deve rispondere `false` anche nel caso in cui l'array `a` sia vuoto e quindi non se ne possa trovare il minimo.

SOLUZIONE

```
public boolean m(int [] a, int [] b) {
  int i;
  int min;
  if (a.length != 0)
    min = a[0];
  else return false; // Caso particolare
  // Minimo di a:
  for (i=1; i < a.length; i++)
    if (a[i] < min)
      min = a[i];
  //Ricerca lineare incerta di min in b:
  boolean trovato = false;
  i = 0;
  while (i < b.length && !trovato)
    if (b[i] == min)
      trovato = true;
    else
      i++;
  return trovato;
}
```

ESERCIZIO 5 (6 punti)

Si consideri il seguente programma:

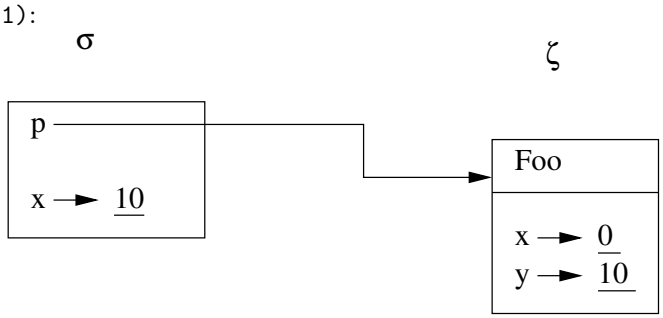
```
prog {
class Foo {
  public int x;
  public int y;

  public void m1(int m) {
    if (m == 0)
      this.x = 0;
    else
      this.x = m*m;
  }

  public int calc() {
    this.y = 0;
```

```
    while (this.x > 0) {
      this.y = this.y + this.x;
      this.x--;
    }
    return this.y;
  }
}
{
  Foo p = new Foo;
  p.m1(2);
  int x = p.calc(); (1)
  Foo q = new Foo;
  q.x = x;
  int y = q.calc(); (2)
}
}
```

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

SOLUZIONE

Al punto `(1)`:



Al punto `(2)`: