Technische Universität München
Fakultät für Informatik
Prof. Dr. M. Broy

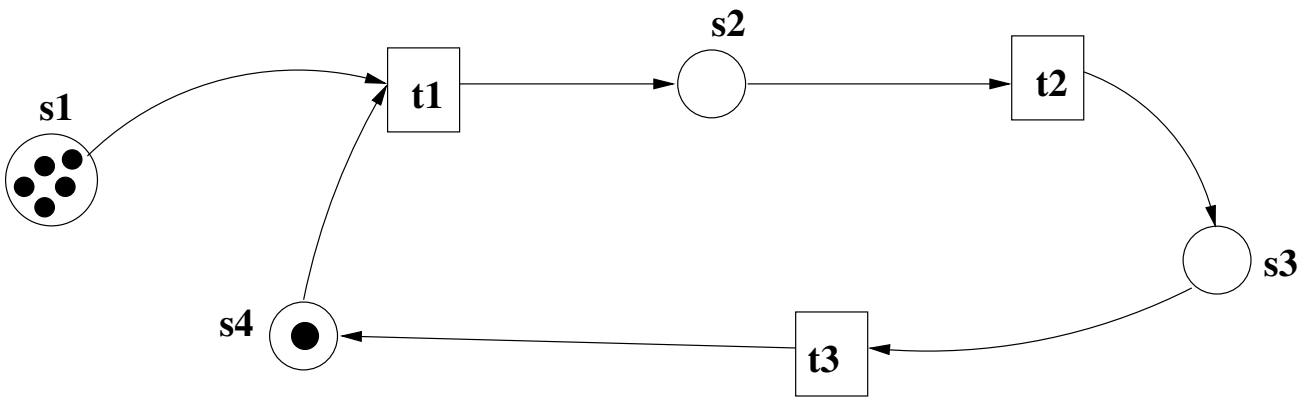
WS 2002/2003
4. April 2003

Lösungsvorschläge der Klausur zu Einführung in die Informatik III

A

Aufgabe 1 Modellieren mit Hilfe von Petrinetzen (5 Punkte)

Es handelt sich hierbei um einen sequentiellen Prozess. Folgendes ist ein mögliches Petri-Netz:

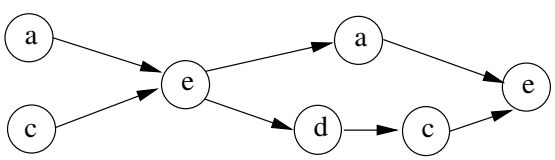


Den Stellen kann folgende Bedeutung zugeordnet werden:
s1: (ganzzahliger Semaphor) N Kunden warten vor dem Schalter
s2: (boolescher Semaphor) Ein Kunde ist am Schalter
s3: (boolescher Semaphor) Die Bearbeitung des Anliegens ist abgeschlossen
s4: (boolescher Semaphor) Der Schalter ist frei

Die Transitionen haben dann folgende Bedeutung:
t1: Ein Kunde geht an den Schalter
t2: Der Angestellte bearbeitet das Anliegen
t3: Der Kunde verlässt die Bank

Aufgabe 2 (2+2+3=7 Punkte)

(a) Gegeben sei das folgende Aktionsdiagramm.



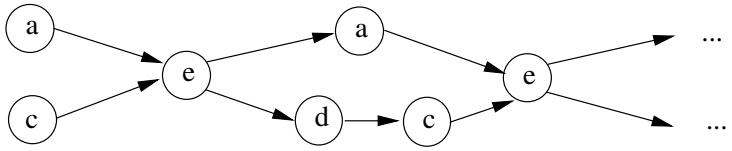
Geben Sie einen Agenten an, der genau dieses Aktionsdiagramm beschreibt.

$$a;e;a;e \,||_e\, c;e;d;c;e$$

(b) Geben Sie alle vollständigen Abläufe dieses Agenten an.

$$< a,c,e,a,d,c,e >$$
$$< a,c,e,d,a,c,e >$$
$$< a,c,e,d,c,a,e >$$
$$< c,a,e,a,d,c,e >$$
$$< c,a,e,d,a,c,e >$$
$$< c,a,e,d,c,a,e >$$

(c) Gegeben sei das folgende Aktionsdiagramm.



Geben Sie einen Agenten an, der genau dieses Aktionsdiagramm beschreibt.

$$(x :: a;e;x)||_e(x :: c;e;d;x)$$

Aufgabe 3 **(10 Punkte)**

Eine mögliche Implementierung der Schreiberprozedur auf der MI lautet:

```
prog:   SEG                          -- Programmsstart
        MOVE    W I H'10000', SP     -- Initial. des Stacks
        JUMP    writer

ws:     DD      W 0                  -- ws := false
rs:     DD      W 0                  -- rs := 0
x:      DD      W 7                  -- Belegung von x
mutex:  DD      B 1                  -- mutex := true

writer: MOVE    W x, -!SP            -- Proz.-par. auf den Stack
        CALL    prod                 -- Unterprogrammaufruf: produziere
await1: JBCCI   I 7, mutex, await1   -- Zutritt in gesch. Bereich
                                     -- möglich ? -> mutex := false
        CMP     W ws, I 1            -- ws == true ?
        JEQ     v_mut                -- dann verlasse await und -> V(mutex)
        CMP     W rs, I 0            -- rs == 0 ?
        JNE     v_mut                -- wenn nicht, dann verlasse
                                     -- await und -> V(mutex)
        MOVE    W I 1, ws            -- ws := true
        MOVE    B I 1,mutex          -- verlasse await und V(mutex)
        JUMP    cont

v_mut:  MOVE    B mutex, I 1         -- V(mutex)
```

```
        JUMP    await1
cont:   MOVE W x, -!SP            -- Unterprogrammaufruf: schreibe
        CALL    write

await2: JBCCI   I 7, mutex, await2   -- Zutritt in gesch. Bereich
                                     -- möglich ? -> mutex := false
        MOVE    W I 0, ws            -- ws := false
        MOVE    B I 1, mutex         -- V(mutex)

        JUMP    writer               -- Endlosschleife
ende:   HALT

prod:   PUSHR
        -- Hier stehen geeignete MI- Befehle
        POPR
        RET
write:  PUSHR
        -- Hier stehen geeignete MI- Befehle
        POPR
        RET

        END
```

Aufgabe 4 **(3+5=8 Punkte)**

(a)

$$Need = Max - Allocation = \begin{pmatrix} 3 & 0 & 4 \\ 1 & 2 & 0 \\ 0 & 9 & 2 \\ 1 & 4 & 7 \end{pmatrix}$$

$$Available = (0,3,8)$$

Das System kann die maximalen Anforderungen keines Prozesses erfüllen. Der Zustand ist nicht sicher. Das System kann nicht in diesen Zustand gelangen.

(b) In diesem Fall würde das System in den neuen Zustand

$$Allocation' = \begin{pmatrix} 1 & 2 & 1 \\ 1 & 3 & 1 \\ 1 & 1 & \mathbf{2} \\ 1 & 2 & 2 \end{pmatrix}$$

übergehen.

$$Need = Max - Allocation' = \begin{pmatrix} 2 & 0 & 4 \\ 1 & 0 & 0 \\ 4 & 8 & \mathbf{0} \\ 0 & 6 & 7 \end{pmatrix}$$

$$Available = (2,2,6).$$

Die maximalen Anforderungen von Prozess 1 und 2 können erfüllt werden. Nach Rückgabe der von Prozess 1 und 2 belegten Betriebsmittel ist

$$Available = (4, 7, 8).$$

Die maximalen Anforderungen von Prozess 4 können erfüllt werden. Nach Rückgabe der von Prozess 4 belegten Betriebsmittel ist

$$Available = (5, 8, 10).$$

Die maximalen Anforderungen von Prozess 3 können erfüllt werden. Nach Rückgabe der von Prozess 3 belegten Betriebsmittel ist

$$Available = (6, 10, 12).$$

Der Zustand

$$Allocation'$$

ist also sicher. Das Betriebssystem kann das angeforderte Betriebsmittel zuteilen.

Aufgabe 5 Parallele Suche durch Bisektion in Java (Lösungsvorschlag) (10 Punkte)

(a)
```
class GemBool {                                          1/2
    private boolean b;                                   1/2

    public synchronized void set(boolean b) {            1/2
        this.b = b;                                      1/2
    }
    public synchronized boolean get() {                  1/2
        return b;                                        1/2
    }
}
```

(b)
```
class PSuche extends Thread {                             1
    private int                              // korrekte Attribute
    int lowbound;                            // 1 Punkt
    int highbound;
    int i;
    GemBool z;

    PSuche(int f[], int lowbound, int highbound, int i, GemBool z) { 1/2
        this.f = f;                          // korrekte Uebernahme
        this.lowbound = lowbound;            // der Argumente in
        this.highbound = highbound;          // die Attribute
        this.i = i;                          // 1/2 Punkt
        this.z = z;
    }
}
```

(c)
```
public void run() {
   int middle = (lowbound+highbound)/2;
   if (lowbound <= highbound) {
      if (f[middle] == i)                          // Fallunterscheidung
         z.set(true);                              // und z.set(true)     1/2
      else {
         if (!z.get()) {                                                   1/2
             PSuche lsearch = new PSuche(f,lowbound,middle-1,i,z);         1/2
             PSuche rsearch = new PSuche(f,middle+1,highbound,i,z);        1/2
             lsearch.start();                                              1/2
             rsearch.start();                                              1/2
             try { lsearch.join(); } catch(Exception e) {};                1
             try { rsearch.join(); } catch(Exception e) {};                1
         }
      }
   }
}
```