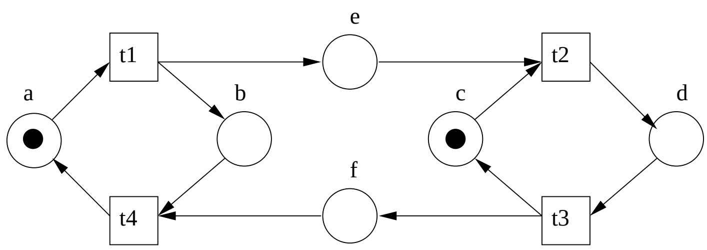
Technische Universität München
Fakultät für Informatik
Dr. M. Schneider, G. Bauer,
Dr. M. Pizka, Dr. B. Rumpe,
Dr. R. Steinbrüggen

WS 2002/2003
Lösungsvorschläge zu Blatt 5
15. November 2002

Übungen zur Vorlesung Einführung in die Informatik III

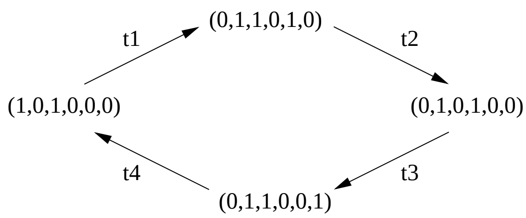
Aufgabe 16 Petrinetze, Invarianten



(a) Der Zustandsraum des Petrinetzes N ist die Menge

$$\{(1,0,1,0,0,0),(0,1,1,0,1,0),(0,1,0,1,0,0),(0,1,1,0,0,1),\}$$

und es ergibt sich das folgende Zustandsübergangsdiagramm:



(b) Sei B die Menge aller möglichen Belegungen des Petrinetzes N. Die beiden Prädikate

$$q1 : B \to \mathbb{B}, \quad q1(m) =_{def} m(a) + m(b) = 1$$

$$q2 : B \to \mathbb{B}, \quad q2(m) =_{def} m(c) + m(d) = 1$$

Invarianten von N mit der Anfangsbelegung m_0.

Wir wenden das folgende Beweisprinzip an:

Um zu beweisen, dass q eine Invariante ist, genügt es zu zeigen, dass die beiden folgenden Bedingungen gelten:

(1) Es gilt $q(m_0)$.

(2) Für alle Belegungen m und m' und für alle Transitionen t_i des Petrinetzes N gilt: falls $q(m)$ und $m \xrightarrow{t_i} m'$ dann gilt $q(m')$.

Wir müssen also zeigen:

(1) Es gilt $m_0(a) + m_0(b) = 1$.

(2) Falls für eine Belegung m die Bedingung $m(a)+m(b)=1$ gilt und $m \xrightarrow{t_1} m'$, dann ist $m'(a)+m'(b)=m(a)-1+m(b)+1=m(a)+m(b)=1$.

Falls für eine Belegung m die Bedingung $m(a)+m(b)=1$ gilt und $m \xrightarrow{t_2} m'$ oder $m \xrightarrow{t_3} m'$, dann ist $m'(a)+m'(b)=m(a)+m(b)=1$.

Falls für eine Belegung die Bedingung m $m(a)+m(b)=1$ gilt und $m \xrightarrow{t_4} m'$, dann ist $m'(a)+m'(b)=m(a)+1+m(b)-1=m(a)+m(b)=1$.

analog für $q2(m)$.

(c) Um nachzuweisen, dass t_1 und t_3 gegenseitig ausgeschlossen sind, müssen wir zeigen, dass nicht gleichzeitig die Stelle a und die Stelle d markiert sind, also $m(a)+m(d)<2$ für alle erreichbaren Markierungen m.

Diese Eigenschaft lässt sich allerdings nicht direkt mit dem oben angewendeten Beweisprinzip für Invarianten nachweisen. Beispielsweise folgt aus $m(a)+m(d)<2$ und $m \xrightarrow{t_4} m'$ nicht unbedingt $m'(a)+m'(d)=(m(a)+1)+m(d)<2$. Wir suchen daher eine stärkere Eigenschaft, deren Invarianz wir nachweisen können und aus der wir $m(a)+m(d)<2$ folgern können.

Eine solche Invariante des Petrinetzes ist

$$q_3 : B \to \mathbb{B}, \quad q_3(m) =_{def} 2m(a)+m(b)+m(c)+2m(d)+m(e)+m(f)=3.$$

q_3 ist eine Invariante des Petrinetzes N, denn:

(1) Es gilt $2m_0(a)+m_0(b)+m_0(c)+2m_0(d)+m_0(e)+m_0(f)=2m_0(a)+m_0(c)=3$.

(2) Für eine Belegung gelte m die Bedingung
$2m(a)+m(b)+m(c)+2m(d)+m(e)+m(f)=3$.

Falls $m \xrightarrow{t_1} m'$,
dann ist $2m'(a)+m'(b)+m'(c)+2m'(d)+m'(e)+m'(f)=$
$2(m(a)-1)+(m(b)+1)+m(c)+2m(d)+(m(e)+1)+m(f)=$
$2m(a)+m(b)+m(c)+2m(d)+m(e)+m(f)=3$

Falls $m \xrightarrow{t_2} m'$,
dann ist $2m'(a)+m'(b)+m'(c)+2m'(d)+m'(e)+m'(f)=$
$2m(a)+m(b)+(m(c)-1)+2(m(d)+1)+(m(e)-1)+m(f)=$
$2m(a)+m(b)+m(c)+2m(d)+m(e)+m(f)=3$

Falls $m \xrightarrow{t_3} m'$,
dann ist $2m'(a)+m'(b)+m'(c)+2m'(d)+m'(e)+m'(f)=$
$2m(a)+m(b)+(m(c)+1)+2(m(d)-1)+m(e)+(m(f)+1)=$
$2m(a)+m(b)+m(c)+2m(d)+m(e)+m(f)=3$

Falls $m \xrightarrow{t_4} m'$,
dann ist $2m'(a)+m'(b)+m'(c)+2m'(d)+m'(e)+m'(f)=$
$2(m(a)+1)+(m(b)-1)+m(c)+2m(d)+m(e)+(m(f)-1)=$
$2m(a)+m(b)+m(c)+2m(d)+m(e)+m(f)=3$

Da q_3 eine Invariante des Petrinetzes N mit der Anfangsbelegung m_0 ist, gilt für alle erreichbaren Markierungen m: $2m(a)+m(b)+m(c)+2m(d)+m(e)+m(f)=3$.
Mit $m(a)\in\mathbb{N}$ und $m(d)\in\mathbb{N}$ folgt $m(a)+m(d)<2$: Wäre $m(a)+m(d)\geq 2$, dann wäre $2m(a)+m(b)+m(c)+2m(d)+m(e)+m(f)\geq 4$.
Es gibt also keine erreichbare Belegung bei der sowohl t_1 als auch t_3 schaltbereit ist. t_1 und t_3 befinden sich im gegenseitigen Ausschluss.

Aufgabe 17 **Abläufe von Agenten**

Es seien a, b, c Aktionen.

a) Für welche Prozesse p_0 gilt $\mathtt{ispar}(p_0,p_1,p_2,\emptyset)$, wenn p_1 und p_2 Abläufe des Agenten a sind?

1) Es gilt offenbar $\mathtt{ispar}(p_e,p_e,p_\emptyset,\emptyset)$.

2) Die Aktion a besitzt die Abläufe $p_i=(\{e_i\},\leq_i,\alpha_i)$ mit $\leq_i=\{(e_i,e_i)\}$, $\alpha_i(e_i)=a$ für $i=1,2$.

Für einen Ablauf $p_0=(E_0,\leq_0,\alpha_0)$ gilt $\mathtt{ispar}(p_0,p_1,p_2,\emptyset)$ genau dann, wenn

$E_0 = E_1 \cup E_2 = \{e_1, e_2\}$ und
$E_1 \cap E_2 = \{e \in E_0: \alpha_0(e) \in \emptyset\} = \emptyset$
d.h. $e_1 \neq e_2$ und
$\leq_0$ ist die reflexiv-transitive Hülle von $\leq_1 \cup \leq_2$
d.h. $\leq_0 = \{(e_1, e_1), (e_2, e_2)\}$ und
$\alpha_0(e_1) = \alpha_0(e_2) = a$.

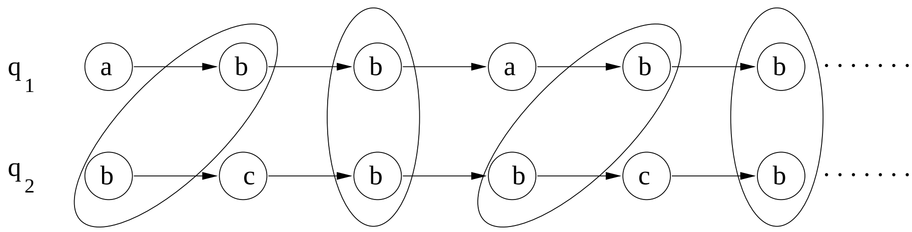
Dies ist der erwartete Prozess mit zwei kausal unabhängigen Aktionen a.

b) Geben Sie mindestens zwei nichtleere Abläufe für den Agenten $a||a$ nach der Definition der Vorlesung an. Wenn ein Ablauf unvollständig ist, so geben Sie eine Fortsetzung an. Unter Verwendung der Ergebnisse und Bezeichnungen aus a) gilt nach den Regeln über die parallele Komposition von Agenten:

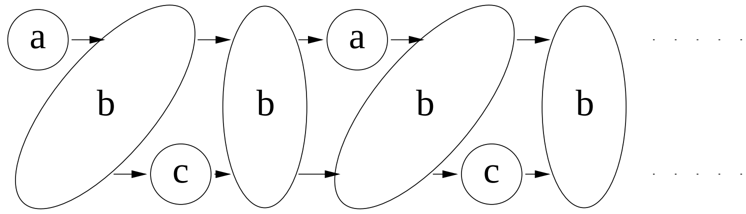
1) $a||a \xrightarrow{p_e} \mathtt{skip}||a \xrightarrow{p_e} \mathtt{skip}$

2) $a||a \xrightarrow{p_0} \mathtt{skip}$

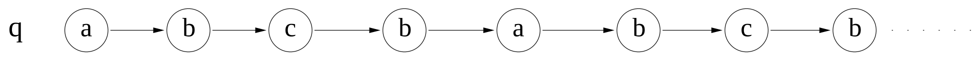
c) Geben Sie graphisch einen vollständigen Ablauf q_1 des Agenten $t_1=(x::a;b;b;x)$ sowie einen vollständigen Ablauf q_2 des Agenten $t_2=(y::b;c;b;y)$ und außerdem einen vollständigen Ablauf q von $t_1||_{\{b\}}t_2$ an. Zeichnen Sie ein, welche Ereignisse von q_1 und q_2 zu identifizieren sind, um q zu bekommen?



Das ispar-Prädikat verlangt, dass die Kausalitätsrelation $\leq$ von q die reflexiv-transitive Hülle der Vereinigung von $\leq_1$ und $\leq_2$, den Kausalitätsrelationen von q_1 und q_2, ist. Die Pfeile im neuen Graphen müssen also verträglich sein mit denen in den beiden Ausgangsgraphen.



Die graphische Darstellung eines Prozesses enthält keine Kanten, die aufgrund des Transitivitätsgesetzes der Kausalitätsordnung erschlossen werden können. Läßt man also die Kanten von einem mit `b` markierten Knoten zum nächsten weg, weil es noch den Pfad über einen Zwischenknoten gibt, so erhält man eine Kette. Parallele Ereignisse treten daher nicht auf:



Aufgabe 18 Parallele Programme

Ein Programmstück, das in einem gegebenen Feld **var** `[1:n]`**int** `a` ganzer Zahlen jedes Element durch seinen Absolutbetrag ersetzt: Die Bearbeitung des Feldes erfolgt durch zwei parallele ablaufende Programme. Jedes Feldelement wird nur einmal behandelt. Parallele Zugriffe auf verschiedene Feldelemente ist vorgesehen. Lösungsidee: Die beiden Programme greifen unter wechselseitigem Ausschluss auf eine Variable `i` zu, die den Index des nächsten Feldelements angibt. Die beiden parallelen Programme sind Aufrufe einer parameterlosen Prozedur `update`.

```
proc update = ():
  ⌈var nat j;
   await true then j,i:=i,i+1 endwait;
   if j <= n then a[j]:=abs(a[j]); update() else nop fi ;⌋
var nat i:=1;
⟦update() || update()⟧
```

Das Konstrukt **await** `E` **then** `S` **endwait** kennzeichnet den Abschnitt `S` als „bewachten kritischen Abschnitt“. Er kann nur betreten werden, wenn der boolesche Ausdruck `E` bei seiner Auswertung **true** ergibt (was hier natürlich stets der Fall ist) und kein anderes, parallel ablaufendes Programm in einem kritischen Abschnitt ist. In der Zwischenzeit muss das Programm an dieser Stelle warten. Im unserem Fall soll erreicht werden, dass jeder Index `i` einmal aber nicht mehrmals benutzt wird.

Aufgabe 19 (P) Java Threads

Im folgenden Beispielprogramm wird für jeden Parameter des Programmaufrufs ein Thread erzeugt, der den Parameter zehn mal auf den Bildschirm schreibt. Zum Beispiel führt die Ausführung des Programms mit dem Aufruf `java ParPrint "Hello " "World! "` dazu, dass zwei Threads nebenläufig je zehn mal die Zeichenketten „Hello “ und „World! “ auf den Bildschirm ausgeben. Da die Ausgaben der beiden Threads unsynchronisiert schreiben, ist das beobachtete Verhalten nichtdeterministisch; es wird eine beliebige Sequenz, die je zehn mal die Zeichenfolgen „Hello “ und „World! “ enthält ausgegeben.

```
class ParPrint extends Thread {
    String message;

    ParPrint(String s) {
      message = s;
    }

    public void run() {
      for(int k=0; k < 10; k++)
          System.out.print(message);
    }
```

```
    static public void main(String args[]) {
        ParPrint p[] = new ParPrint[args.length];
        for (int i = 0; i < args.length; i++) {
            p[i] = new ParPrint(args[i]);
            p[i].start();
        }
    }
}
```