Ludwig-Maximilians-Universität München
Institut für Informatik
Lehrstuhl für Mobile und Verteilte Systeme
Prof. Dr. Claudia Linnhoff-Popien

mobile and distributed systems group

Betriebssysteme im Wintersemester 2018/2019
Übungsblatt 7

Abgabetermin: 10.12.2018, 18:00 Uhr

Besprechung: Besprechung der T-Aufgaben in den Tutorien vom 03. – 07. Dezember 2018
Besprechung der H-Aufgaben in den Tutorien vom 10. – 14. Dezember 2018

Aufgabe 31: (T) Deadlock Prevention

(– Pkt.)

Eine Methode, um Deadlocks zu vermeiden, ist es, eine der Bedingungen für das Entstehen von Deadlocks im Vorhinein auszuschließen.

a. Geben sie die vier Voraussetzungen für die Entstehung eines Deadlocks an.

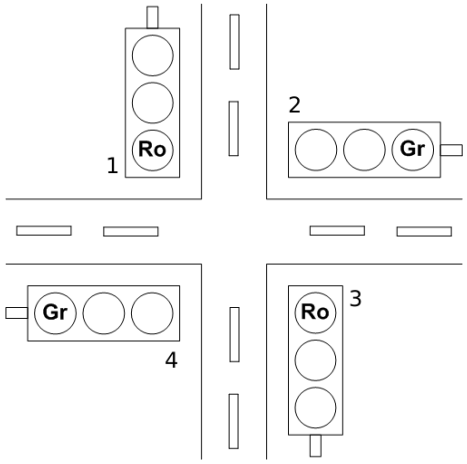
b. Beschreiben Sie, wie durch eine Ordnung der Ressourcen bei geeigneter Reservierungsstrategie Deadlocks vermieden werden können.

   Tipp: Ordnung der Ressourcen bedeutet: Ob ein Prozess ein Betriebsmittel anfordern darf, hängt nicht nur davon ab, ob dieses gerade frei ist, sondern auch davon, welche Betriebsmittel der Prozess zuvor schon angefordert hat. Welche der vier Deadlock-Bedingungen könnte man mit diesem Ansatz ausschließen?

Aufgabe 32: (T) Petri-Netze: Modellierung einer Ampelanlage

(– Pkt.)

Wir betrachten im Folgenden die Ampelschaltung einer Kreuzung wie sie hier zu sehen ist.



Gehen Sie von folgenden Randbedingungen aus:

- Zwei sich gegenüberliegende Ampeln schalten ihr Signal stets synchron.
- Eine Ampel schaltet stets in der Reichenfolge **rot** → **rotgelb** → **grün** → **gelb** → **rot** → ...
- Wenn eines der sich gegenüberliegenden Ampelpaare *nicht* auf **rot** steht, dann *muss* das andere Ampelpaars auf rot stehen.

Bearbeiten Sie unter Berücksichtigung der Randbedingungen folgenden Aufgaben:

a. Modellieren Sie eine einzelne Ampel durch ein Petrinetz. Wählen Sie dabei aussagekräftige Bezeichner für die verschiedenen Stellen Ihres Petrinetzes. Verwenden Sie dabei eine minimale Anzahl an Stellen, Marken, Kanten und Transitionen.

b. Modellieren Sie nun das Schaltverhalten der beiden Ampeln **1** und **2** (also das von zwei Ampeln, die sich **nicht** gegenüberliegen) durch ein Petrinetz, so dass die genannten Randbedingungen erfüllt sind. Wählen Sie dabei wieder aussagekräftige Bezeichner für die verschiedenen Stellen Ihres Petrinetzes. Verwenden Sie dabei wieder eine minimale Anzahl an Stellen, Marken, Kanten und Transitionen.

c. Modellieren Sie nun das Schaltverhalten der beiden Amplen **1** und **3** (also von zwei sich gegenüberliegenden Ampeln) durch ein Petrinetz, so dass die genannten Randbedingungen erfüllt sind. Wählen Sie dabei wieder aussagekräftige Bezeichner für die verschiedenen Stellen Ihres Petrinetzes. Verwenden Sie wieder eine minimale Anzahl an Stellen, Marken, Kanten und Transitionen.

d. Modellieren Sie nun alle vier Ampeln (also die gesamte Ampelschaltung) durch ein Petrinetz, so dass die genannten Randbedingungen erfüllt sind. Wählen Sie dabei wieder aussagekräftige Bezeichner für die verschiedenen Stellen Ihres Petrinetzes. Verwenden Sie wieder eine minimale Anzahl an Stellen, Marken, Kanten und Transitionen.

Aufgabe 33: (T) Race Condition

(– Pkt.)

Race Conditions treten dann auf, wenn mehrere Prozesse (bzw. Threads) auf Datenelemente (lesend bzw. schreibend) zugreifen und somit das Endergebnis von der Reihenfolge der Befehlsausführung abhängig ist (Stallings, W.: Operating Systems, 2015). In anderen Worten ist das Ergebnis somit nicht deterministisch und abhängig von der Ausführungsreihenfolge der Prozesse (bzw. Threads).
Gehen Sie davon aus, dass mehrere Threads auf den `Example*`-Klassen arbeiten. Dies könnte für `Example1` wie folgt aussehen:

```
1  public class RaceConditionThread extends Thread {
2      Example1 example;
3
4      RaceConditionThread(Example1 ex) {
5          example = ex;
6      }
7
8      public void run() {
9          while (true) {
10             example.setTrue();
11             example.setFalse();
12         }
13     }
14 }
```

```
1  public class Main {
2      public static void main(String[] args) {
3          Example1 example = new Example1();
4
5          RaceConditionThread t1 = new RaceConditionThread(example);
6          RaceConditionThread t2 = new RaceConditionThread(example);
7          RaceConditionThread t3 = new RaceConditionThread(example);
8          t1.start();
9          t2.start();
10         t3.start();
11     }
12 }
```

Entscheiden Sie, ob die folgenden Aussagen zutreffen. Begründen Sie Ihre Antwort und geben Sie im Falle einer möglichen Race Condition einen beispielhaften Ablauf und eine Strategie zur Fehlervermeidung an.

a. Der Text „`Race Condition`“ wird niemals ausgegeben.

```
1  public class Example1 {
2     private boolean flag;
3     public void setTrue() {
4        flag = true;
5        if(!flag) {
6           System.out.println("Race Condition");
7        }
8     }
9     public void setFalse() {
10       flag = false;
11       if(flag) {
12          System.out.println("Race Condition");
13       }
14    }
15 }
```

b. Es wird stets der richtig inkrementierte Wert von `count` zurückgegeben.

```
1  public class Example2 {
2     private int count;
3
4     public Example2(){
5         count = 0;
6     }
7
8     public int increment() {
9        return ++count;
10    }
11 }
```

Aufgabe 34: (H) Prozessfortschrittsdiagramm

(8 Pkt.)

Gegeben seien zwei Prozesse A und B, die neben anderen Prozessen auf einem Einprozessorsystem ausgeführt werden sollen. A benötigt zu seiner Ausführung 12 Zeiteinheiten, B 10 Zeiteinheiten. Es stehen 3 Betriebsmittel (BM) zur Verfügung, die von den Prozessen während ihrer Ausführung benötigt werden.

A benötigt	Prozess B benötigt
BM1 im Zeitraum $]2-6[$	BM1 im Zeitraum $]5-8[$
BM2 im Zeitraum $]4-7[$	BM2 im Zeitraum $]4-7[$
BM3 im Zeitraum $]8-10[$	BM3 im Zeitraum $]1-3[$

a. Skizzieren Sie das Prozessfortschrittsdiagramm, bei welchem **Prozess** A **auf der x-Achse** und **Prozess** B **auf der y-Achse** abgetragen ist! Zeichnen Sie alle unmöglichen und unsicheren Bereiche ein und beschriften Sie diese entsprechend! Treffen Sie zudem eine Aussage darüber, wo genau unter Umständen ein Deadlock eintritt!

b. Zeichnen Sie einen Ausführungspfad in das Diagramm aus Teilaufgabe a) ein, bei welchem die Prozesse A und B ordnungsgemäß ausgeführt werden und terminieren.

c. Kann es bei nicht-preemptivem Scheduling zu einem Deadlock kommen? Begründen Sie Ihre Antwort und zeichnen Sie für nicht-preemptives Scheduling alle *prinzipiell* verschiedenen Möglichkeiten der Abarbeitung von Prozess A und B in ihrer Abbildung aus Aufgabe a) ein! Sie können dabei den diskreten Zeitpunkt, an dem ein Prozesswechsel zwischen den Prozessen A und B erfolgt vernachlässigen.

d. Vorausgesetzt, es kommt nun ein preemptiver Scheduling-Algorithmus zum Einsatz: Kann man dann die Anzahl an verschiedenen Scheduling-Abläufen bestimmen, um die Prozesse A und B erfolgreich terminieren zu lassen? Begründen Sie Ihre Antwort!

Aufgabe 35: (H) Einfachauswahlaufgabe: Multiprocessing

(5 Pkt.)

Für jede der folgenden Fragen ist eine korrekte Antwort auszuwählen („1 aus n“). Nennen Sie dazu in Ihrer Abgabe explizit die jeweils ausgewählte Antwortnummer ((i), (ii), (iii) oder (iv)). Eine korrekte Antwort ergibt jeweils einen Punkt. Mehrfache Antworten oder eine falsche Antwort werden mit 0 Punkten bewertet.

a) Was ist nach dem Vorlesungsskript keine Voraussetzung für einen Deadlock?			
(i) Free Running	(ii) Mutual Exclusion	(iii) Hold and Wait	(iv) No Preemption
b) Wie bezeichnet man den nicht-preemptiven Scheduling-Algorithmus, bei welchem jeweils der Auftrag ausgewählt wird, bei dem die kürzeste Abarbeitungszeit erwartet wird?			
(i) Shortest Job First	(ii) First Come First Served	(iii) Shortest Remaining Processing Time	(iv) Round Robin
c) Was beschreibt die Deadlock-Situation beim 2 Philosophenproblem? Bei diesem kann Philosoph A bzw. B wahlweise denken oder essen. Auf dem Tisch werden 2 Stäbchen zur Verfügung gestellt, die ein Philosoph beide benötigt, um zu essen.			
(i) Philosoph A nimmt beide Stäbchen auf	(ii) Philosoph B nimmt beide Stäbchen auf	(iii) Philosoph A und B nehmen jeweils das (von ihnen aus) rechte Stäbchen auf	(iv) beide Stäbchen liegen auf dem Tisch
d) Welche allgemeine Aussage bezüglich der Kanten eines Petri-Netzes zur Prozessmodellierung ist korrekt?			
(i) Sie befinden sich zwischen je zwei Stellen.	(ii) Eine Kante besteht zwischen genau einer Stelle s und einer Transition t, entweder von s nach t oder t nach s.	(iii) Sie befinden sich zwischen je zwei Transitionen.	(iv) Die Kanten sind ungerichtet.
e) Wodurch wird die Dynamik eines Systems im Bezug auf Petri-Netze zur Prozessmodellierung beschrieben?			
(i) durch das Schalten von Stellen	(ii) durch das Schalten von Kantengewichten	(iii) durch das Schalten von Kapazitäten	(iv) durch das Schalten von Transitionen