Universität Stuttgart — Christoph Dibak
Institut für Parallele und Verteilte Systeme — Jonathan Falk
Abteilung Verteilte Systeme — Henriette Röger

Systemkonzepte und -programmierung

Aufgabenblatt 1

Veröffentlicht:	25.10.2017
Abgabe bis:	01.11.2017, 12:00
Besprechung:	03., 08. & 10.11.2017
Gesamtpunktzahl:	60 Punkte

Die Abgabe Ihrer Lösung erfolgt in Dreiergruppen online über eine Übungsgruppe des ILIAS-Kurses[1]. Wenn Sie sich in Campus für die SKP-Vorlesung registriert haben, sollten Sie automatisch Zugriff auf den ILIAS-Kurs bekommen. Im ILIAS-Kurs stehen Ihnen auch Foren für Fragen zu den Übungen und zur Suche nach Übungspartnern für die Gruppenabgaben zur Verfügung.

Bei organisatorischen Fragen wenden Sie sich bitte an
Christoph Dibak, Raum 2.404, christoph.dibak@ipvs.uni-stuttgart.de,
Jonathan Falk, Raum 2.352, jonathan.falk@ipvs.uni-stuttgart.de oder
Henriette Röger, Raum 2.342, henriette.roeger@ipvs.uni-stuttgart.de.
Bitte beachten Sie die Zeiten für die Sprechstunden.

Aufgabe 1: Kinokarten-Problem — 19 Punkte

In dieser Aufgabe behandeln wir das Kinokarten-Problem aus der Vorlesung: Sie sind Kinobesitzer und verkaufen die Sitzplätze für die kommende Kinoveranstaltung. Für jeden Kunden bestimmt das Reservierungssystem den nächstbesten verfügbaren Platz und reserviert diesen. Die Reservierung wird über die Datei "seats.dat" vollzogen, in der für jeden Kinoplatz abgespeichert wird, ob dieser bereits gebucht ist. Im ILIAS-System stehen vordefinierte Klassen zum Download bereit.

(a) [1 Punkt] Laden Sie das Eclipse-Projekt als zip-Datei aus dem ILIAS-System, importieren Sie die zip-Datei als existierendes Projekt in Ihren Eclipse-Workspace und führen Sie die Klasse **ResetSeats** aus. Sie finden nun in Ihrem Projektordner die Datei "seats.dat". Die erste Spalte repräsentiert die Sitznummer, die zweite Spalte ist *0*, wenn der Platz noch frei und *1*, wenn er reserviert ist. Welcher Wert steht anfänglich in der zweiten Spalte?

(b) [4 Punkte] Führen Sie jetzt die Klasse **Main** aus. Wie viele Sitze wurden durch das Reservierungssystem reserviert? Wie lange braucht das Reservierungssystem hierfür?

(c) [6 Punkte] Ihre Kunden beschweren sich nun, da die Wartezeiten an dem Kinoschalter sehr lange sind. Deshalb beschließen Sie, die Kartenreservierung zu parallelisieren. Dazu öffnen Sie einen zweiten Kinoschalter. Dieser liest und reserviert

[1] `https://ilias3.uni-stuttgart.de/goto_Uni_Stuttgart_crs_1322194.html`

die Kinoplätze über eine gemeinsame Ressource: die Datei *seats.dat*. Führen Sie die Klasse **ResetSeats** aus um alle Plätze wieder auf frei zu setzen. Danach führen Sie die Klasse **Main** zweimal aus, indem Sie zweimal unmittelbar nacheinander auf den *run*-Button drücken. Beide Programme laufen nun parallel und versuchen Sitzplätze zu reservieren. Sie können beide Ausgaben beobachten, indem Sie in dem Konsole-Tab die jeweilige Konsolenausgabe ändern. Welche Zeit benötigte die parallele Reservierung? Ist diese schneller als die sequenzielle? Wie viele Sitze wurden insgesamt reserviert?

(d) [8 Punkte] Welche der folgenden Sicherheitseigenschaften sind in der parallelen Variante erfüllt und welche nicht erfüllt? Begründen Sie Ihre Antwort!

i. Es werden maximal 200 Sitze vergeben.

ii. Jeder Sitz wird höchstens einmal vergeben.

iii. Jeder Sitz wird mindestens einmal vergeben.

iv. Eine Anfrage wird nur abgelehnt, wenn es keine freien Karten gibt.

Aufgabe 2: Atomarität 24 Punkte

(a) In Abbildung 1 ist ein Programm $\pi = \{p, q\}$ dargestellt.

Programm π	
integer $n \leftarrow 5$	
p	q
$n \leftarrow n + 2$	$n \leftarrow n * 2$

Figure 1: Programm π

i. [7 Punkte] Geben Sie für π eine Abbildung auf eine Stapelmaschine und eine Registermaschine an. Verwenden Sie hierfür die atomaren Operationen `Load`, `Store`, `Push`, `Pop`, `Add`, sowie die Register $R1$ und $R2$. Außerdem gibt es eine Operation `Mult`, die bei der Registermaschine als Parameter zwei Register nimmt und das Ergebnis in das erste Register schreibt. Alle Operation der Stapelmaschine und der Registermaschine sind atomar.

ii. [2 Punkte] Finden Sie sowohl für das Registermaschinen- als auch für das Stapelmaschinenprogramm eine Ausführungssequenz mit Endzustand $n = 7$.

iii. [2 Punkte] Gibt es mehrere mögliche Ausführungssequenzen mit Endzustand $n = 7$ für das Programm mit der Registermaschine? Wie viele mögliche Ausführungssequenzen gib es? Begründen Sie.

iv. [2 Punkte] Finden Sie für beide Programme jeweils eine **weitere** Ausführungssequenz, bei welcher das Lost-Update Problem auftritt. Finden Sie außerdem jeweils eine Sequenz bei welcher es nicht auftritt.

v. [1 Punkt] Welche Sequenz von Befehlen muss atomar ausgeführt werden, damit das Lost Update Problem garantiert nicht auftritt?

(b) In Abbildung 2 ist ein Programm, bestehend aus atomaren Befehlen, für eine Stapelmaschine gegeben.

integer $n \leftarrow 1$	
p	q
p1: Push n	q1: Push 4
p2: Push n	q2: Push n
p3: Add	q3: Sub
p4: Pop n	q4: Pop n

Figure 2: Programm für eine Stapelmaschine

i. [4 Punkte] Geben Sie jeweils eine Ausführungssequenz an, für welche n im Endzustand die Werte $4, 3$ oder 2 annimmt. Gibt es einen Endzustand in dem n einen anderen Wert annimmt?

ii. [3 Punkte] Angenommen, die Sequenz $q1, q2$ wird atomar ausgeführt. Kann dann n nach der Ausführung die Werte $4, 3$ oder 2 annehmen? Geben Sie jeweils die Ausführungssequenz an oder begründen Sie.

iii. [3 Punkte] Angenommen die Sequenzen $p1, p2, p3$ und $q1, q2, q3$ werden atomar ausgeführt. Welche Werte kann n annehmen? Geben Sie jeweils eine Ausführungssequenz an.

Aufgabe 3: Korrektheitsanalyse 17 Punkte

In Abbildung 3 sind zwei Prozesse p und q gegeben, welche beide regelmäßig auf eine gemeinsame Resource R zugreifen möchten. Beispielsweise wollen beide Prozesse auf einen Drucker oder eine Datei zugreifen.

Hinweis: Falls nicht anders angegeben gehen wir in der Vorlesung und den Übungen immer von einer schwach-fairen Ausführung aus.

Gemeinsamer Zugriff auf R	
integer $n \leftarrow 0$	
p	q
loop	**loop**
p1: **if** n **mod** $2 = 0$ **then**	q1: **if** n **mod** $2 = 1$ **then**
p2: access R	q2: access R
p3: $n \leftarrow n + 1$	q3: $n \leftarrow n + 1$

Figure 3: Gemeinsamer Zugriff auf R.

(a) Wir definieren eine Sicherheitseigenschaft und eine Lebendigkeitseigenschaft für die Problemstellung:

1. Beide Prozesse dürfen nicht gleichzeitig auf R zugreifen.
2. In jedem Zustand gibt es eine Fortsetzung, in der ein Prozess auf R zugreifen kann.

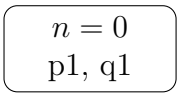
i. [1 Punkt] In welchem Zustand ist die erste Eigenschaft nicht erfüllt?

ii. [2 Punkte] Sind dies Eigenschaften über die Lebendigkeit oder die Sicherheit? Begründen Sie.

(b) [9 Punkte] Bei dem Programm in Abbildung 3 wird jede Zeile als atomar angenommen. Erstellen Sie ein Zustandsdiagramm dieses Programms für alle Zustände mit

$n < 2$. Zeigen Sie damit, ob die in (a) genannten Eigenschaften erfüllt sind. Begründen Sie ihre Antwort.

Hinweis: Gehen Sie von folgendem Initialzustand aus:



(c) [5 Punkte] In Abbildung 4 wurde das Programm leicht abgeändert. Geben Sie auch für dieses Programm ein Zustandsdiagramm für $n < 2$ an. Zeigen Sie außerdem, ob die unter (a) genannten Eigenschaften erfüllt sind.

Gemeinsamer Zugriff auf R	
integer $n \leftarrow 0$	
p	q
loop	**loop**
p1: **if** n **mod** $2 = 0$ **then**	q1: **if** n **mod** $2 = 1$ **then**
p2: access R	q2: access R
p3: $n \leftarrow n + 1$	q3: $n \leftarrow n + 1$

Figure 4: Abgeänderte Version des Programms zum gemeinsame Zugriff auf R