Christian-Albrechts-Universität zu Kiel
Institut für Informatik
Prof. Dr. W.-P. de Roever
Jens Schönborn, Christian Motika und Tim Fenten

Nebenläufiges Programmieren

Sommersemester 2008 26.5.2008

Serie 8
Mittsemestertest

Ausgabetermin: 26.5.2008

Abgabe: 6.6.2008 (11:00)

Diese Serie soll in Einzelarbeit gelöst werden!

Aufgabe 1 (3 Punkte) Betrachte folgendes Programm:

```
1 int x = 0;
2 int y = 0;
3 co
4   x = x + 1;
5   x = x + 2;
6   //
7   x = x + 2;
8   y = y - x;
9 oc
```

1. Angenommen jedes assignment statement wird atomar ausgeführt. Wie viele mögliche histories gibt es? (vgl. Buch Seite 42,43) Was sind die möglichen Werte von x und y nach der Ausführung?

2. Angenommen jedes assignment statement wird durch drei atomare Aktionen, ein Register laden, addieren oder subtrahieren eines Wertes zu oder von einem Register und ein Register schreiben, ausgeführt. Wie viele mögliche histories gibt es? Was sind die möglichen Werte von x und y nach der Ausführung?

Aufgabe 2 (5 Punkte) Nehme an, dass es n Worker-Prozesse (1 bis n) gibt. Nehme ebenfalls an, dass der verwendete Computer über eine atomare Inkrement-Instruktion verfügt. Betrachte dabei den folgenden Code für eine n-Prozess-Barriere, welche wiederverwendbar sein soll:

```
1   int count = 0; go = 0;      #shared variables
2
3   code executed by Worker[1]:
4   ...
5   <await (count == n -1);>
6   count = 0;
7   go = 1;
8   ...
9
10  code executed by Worker[2:n]:
11  ...
12  <count++;>
13  <await (go == 1);>
14  ...
```

(a) Erläutere was an obiger Lösung falsch ist.

(b) Ändere den Code so ab, dass er korrekt arbeitet. Benutze dabei keine weiteren *shared variables*. Neue lokale Variablen dürfen eingeführt werden. Es dürfen ebenfalls nur await-Statements mit leerem Anweisungsblock (`<await (B);>`) verwendet werden, die gem. AMO-Property durch While-Schleifen implementierbar sind.

(c) Angenommen der obige Code wäre korrekt und alle Prozesse kämen gleichzeitig an der Barriere an. Wie viele Zeiteinheiten werden benötigt, bis jeder Prozess die Barriere verlassen hat?
Zähle dabei jedes *assignment statement* und auch jedes *await statement* sobald die Bedingung wahr wird als eine Zeiteinheit.

Aufgabe 3 (5 Punkte) Schreibe ein mpd Programm, dass die Lösung des readers/writers Problems in Fig. 4.13 auf Seite 176 mit wechselnder Präferenz implementiert. Als Basis dafür findest du eine Implementierung von 4.13 auf der Veranstaltungsseite.
Mache mittels Kommentaren eindeutig klar, welche Ergänzungen oder Änderungen vorgenommen wurden. Wenn Zeilen ersetzt oder entfernt werden, kommentiere die Originalzeile bitte nur aus.
Argumentiere, warum die Lösung korrekt ist und erkläre auf welche Weise sie die wechselseitige Präferenz realisiert .