Technische Universität München **SoSe 2012**
Institut für Informatik
Prof. Dr. M. Broy, Dr. M. Spichkova **31. Mai 2012**
J. Mund, P. Neubeck **Übungsblatt 3**

Übungen zu "Modellierung verteilter Systeme"

Aufgabe 1 **Paralleles Rechnen**

Gegeben sei die Implementierung einer Funktion $f\colon \mathbf{Nat} \to \mathbf{Nat}$. Mit dieser soll die Summe $S_n = \sum_{i=0}^{n} f(i)$ für ein beliebiges $n \in \mathbf{Nat}$ bestimmt wird. Da die Berechnung von $f(i)$ zum Teil sehr lange dauert, soll sie auf zwei Prozesse verteilt und parallel bearbeitet werden. Die Prozesse können durch einen gegebenen Algorithmus für wechselseitigen Ausschluß koordiniert:

```
do Q then [nicht kritischer Abschnitt]
          Zugang
          [kritischer Abschnitt]
          Austritt
od
```

(a) Definieren Sie zwei Prozesse, die in Zusammenarbeit die Summe S_n berechnen, indem Sie jeweils den kritischen und nicht kritischen Abschnitt sowie geeignete Schleifenbedingungen angeben.

(b) Wie geeignet ist Ihr Verfahren, wenn (1.) die Berechnungen $f(i)$ in beiden Prozessen und für alle i ungefähr gleich lange dauern oder (2.) die Berechnungen sehr verschieden und unbekannt lange dauern?

Aufgabe 2 **Petersons Algorithmus**

In der Vorlesung wurde der Dekker-Algorithmus zur Gewährleisung des *wechselseitigen Ausschlusses* in nebenläufigen Programmen mit gemeinsamen Variablen besprochen. Eine alternative Lösung ist der Peterson-Algorithmus:

```
begin                          proc P_i :
  turn := 0                      do true then fl_i := true
  fl_0, fl_1 := false                         turn := j
  ⌈P_0 ‖ P_1⌋                                 do fl_j ∧ turn = j then nop od
end                                           crit_i
                                              fl_i := false
                                 od
```

Prüfen Sie formal (Zusicherungsbeweis, Inferenzfreiheit), ob der Algorithmus folgende Eigenschaften gewährleistet:

Wechselseitiger Ausschluss (*Mutual Exclusion*) Zu jeder Zeit befindet sich höchstens ein Prozess innerhalb des kritischen Abschnittes.

Keine Verklemmung (*Deadlock-freedom*) Die Prozesse blockieren sich nicht gegenseitig. Wenn mehrere Prozesse versuchen, in den kritischen Bereich zu gelangen, erreicht irgendwann ein Prozeß den kritischen Bereich.

Kein Verhungern (*Starvation-freedom*) Wenn ein Prozeß versucht, in den kritischen Bereich zu gelangen, erreicht er diesen auch irgendwann.