SoSe 2017

# Nichtsequentielle und verteilte Programmierung

## 3. Übungsblatt

Prof. Dr. Margarita Esponda

### 1. Aufgabe (2 P.)

Beweisen oder widerlegen Sie folgende Äquivalenz:

$$\Diamond \Box A \leftrightarrow \Box \Diamond A$$

### 2. Aufgabe (4 P.)

Zeigen Sie, dass folgendes gilt:

$$\square A \Rightarrow \lozenge B$$
 und  $\lozenge \square A$  impliziert  $\lozenge B$ 

### 3. Aufgabe (6 P.)

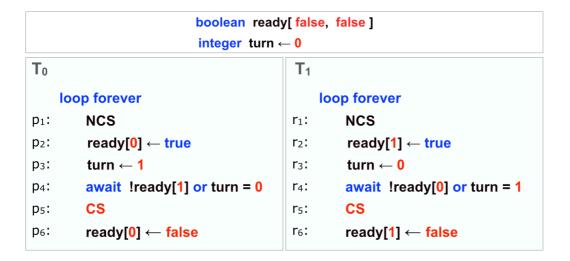
Zeigen Sie, dass folgende distributive Regeln gelten:

$$(\Box A \land \Box B) \leftrightarrow \Box (A \land B)$$

$$(\Diamond A \lor \Diamond B) \leftrightarrow \Diamond (A \lor B)$$

#### 4. Aufgabe (10 P.)

Betrachten Sie den folgenden Pseudocode des Peterson-Algorithmus:



Unter der Annahme, dass folgende 2 Invarianten wahr sind,

$$(p_4 \wedge r_5) \rightarrow (ready[1] \wedge turn = 1)$$

$$(p_5 \wedge r_4) \rightarrow (ready[0] \wedge turn = 0)$$

beweisen Sie, dass folgende Formeln gelten:

```
1) (p_4 \land \Box \neg p_5) \rightarrow \Box \diamondsuit (ready[1] \land (turn=1))
2) \diamondsuit \Box (\neg ready[1]) \lor \diamondsuit (turn=0)
3) p_4 \land \Box \neg p_5 \land \diamondsuit (turn=0) \rightarrow \diamondsuit \Box (turn=0)
```

### **5. Aufgabe** (6 P.)

Betrachten Sie die Formeln aus Aufgabe 4 und zeigen Sie durch Widerspruch, dass in dem vorgegebenen Peterson-Algorithmus die Prozesse nicht verhungern.

### **6. Aufgabe** (4 P.)

In der Variante des Bäckerei-Algorithmus für zwei Prozesse sind wir davon ausgegangen, dass die Zuweisungen  $ticket_p \leftarrow ticket_r + 1$  und  $ticket_r \leftarrow ticket_p + 1$  unteilbar sind. Das ist nicht immer der Fall, weil  $ticket_r$  und  $ticket_p$  globale Variablen sind, die zuerst lokal gelesen werden müssen.

Betrachten Sie folgende Version des Bäckerei-Algorithmus:

```
integer ticket_p \leftarrow 0, ticket_r \leftarrow 0
                                                                        \mathsf{T}_\mathsf{r}
T_p
      integer temp \leftarrow 0
                                                                               integer temp \leftarrow 0
                                                                               loop forever
      loop forever
                                                                        r<sub>1</sub>:
                                                                                    NCS
           NCS
p<sub>1</sub>:
                                                                        r<sub>2</sub>:
                                                                                    temp ← ticket_p
           temp \leftarrow ticket\_r
p<sub>2</sub>:
                                                                                    ticket_r \leftarrow temp + 1
                                                                        r<sub>3</sub>:
p<sub>3</sub>:
           ticket_p ← temp + 1
                                                                        r4:
                                                                                    await ticket_p = 0 or ticket_r < ticket_p
           await ticket_r = 0 or ticket_p \leq ticket_r
p<sub>4</sub>:
                                                                                    CS
                                                                        r5:
p<sub>5</sub>:
                                                                                    ticket r \leftarrow 0
                                                                        re:
           ticket_p \leftarrow 0
p<sub>6</sub>:
```

Zeigen Sie anhand eines Beispiels, dass der wechselseitige Ausschluss nicht mehr gewährleistet wird.

<sup>\*</sup> Die Aufgaben sind aus dem Buch von M. Ben-Ari entnommen und angepasst worden.