

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:

$$\Box A \Rightarrow \Diamond B \text{ und } \Diamond \Box A \text{ impliziert } \Diamond B$$

3. Aufgabe (6 P.)

Zeigen Sie, dass folgende distributive Regeln gelten:

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

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

4. Aufgabe (10 P.)

Betrachten Sie den folgenden Pseudocode des Peterson-Algorithmus:

| | |
|--|--|
| <pre>boolean ready[false, false] integer turn ← 0</pre> | |
| <p>T₀</p> <pre>loop forever p1: NCS p2: ready[0] ← true p3: turn ← 1 p4: await !ready[1] or turn = 0 p5: CS p6: ready[0] ← false</pre> | <p>T₁</p> <pre>loop forever r1: NCS r2: ready[1] ← true r3: turn ← 0 r4: await !ready[0] or turn = 1 r5: CS r6: ready[1] ← false</pre> |

Unter der Annahme, dass folgende 2 Invarianten wahr sind,

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

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

beweisen Sie, dass folgende Formeln gelten:

- 1) $(p_4 \wedge \Box \neg p_5) \rightarrow \Box \Diamond (\text{ready}[1] \wedge (\text{turn}=1))$
- 2) $\Diamond \Box (\neg \text{ready}[1]) \vee \Diamond (\text{turn}=0)$
- 3) $p_4 \wedge \Box \neg p_5 \wedge \Diamond (\text{turn}=0) \rightarrow \Diamond \Box (\text{turn}=0)$

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 | |
|--|---|
| T_p | T_r |
| integer temp \leftarrow 0 | integer temp \leftarrow 0 |
| loop forever | loop forever |
| p1: NCS | r1: NCS |
| p2: temp \leftarrow ticket_r | r2: temp \leftarrow ticket_p |
| p3: ticket_p \leftarrow temp + 1 | r3: ticket_r \leftarrow temp + 1 |
| p4: await ticket_r = 0 or ticket_p \leq ticket_r | r4: await ticket_p = 0 or ticket_r < ticket_p |
| p5: CS | r5: CS |
| p6: ticket_p \leftarrow 0 | r6: ticket_r \leftarrow 0 |

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

* Die Aufgaben sind aus dem Buch von M. Ben-Ari entnommen und angepasst worden.