SoSe 2017

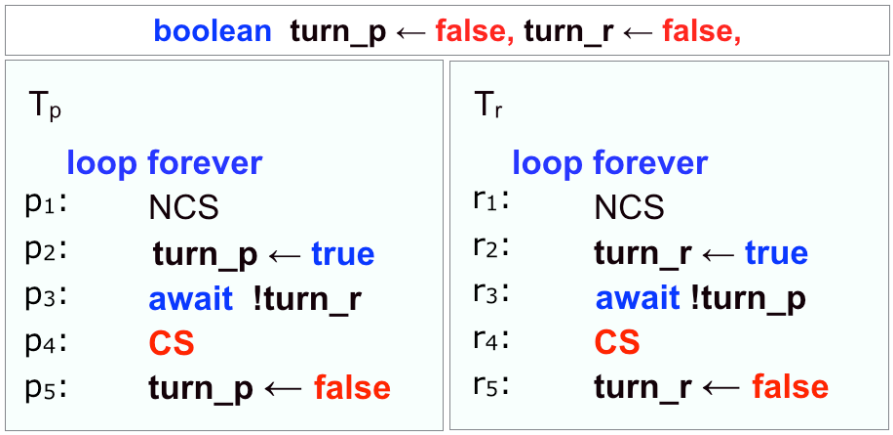
Nichtsequentielle und verteilte Programmierung

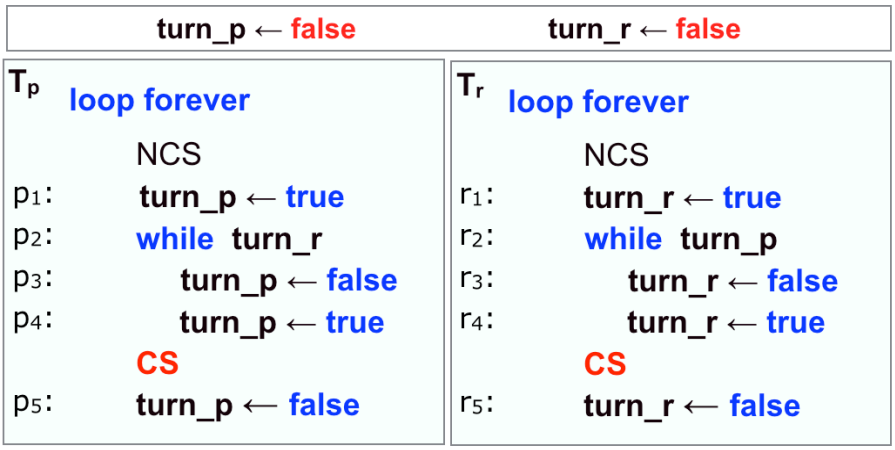
2. Übungsblatt

Prof. Dr. Margarita Esponda

1. Aufgabe (12 Punkte)

Zeigen Sie mit Hilfe von Zustandsdiagrammen, dass in folgenden zwei Algorithmen der Wechselseitige Ausschluss gewährleistet ist. Verwenden Sie dabei die abgekürzte Version des Algorithmus, indem Sie die CS- und NCS-Zeilen ignorieren.





2. Aufgabe (8 Punkte)

Wir haben in der Vorlesung gelernt, dass in modernen Rechnern das Testen und Verändern von Variablen in einem atomaren Maschinenbefehl (TSL) ausgeführt werden kann und damit eine einfache Lösung für den Wechselseitigen Ausschluss möglich ist.

Betrachten Sie folgende Lösung in Pseudocode, die genau das gleiche tut:

Die TSL-Anweisung kopiert die *Lock*-Variable in ein lokales Register/Variable des Threads (*reg_p* oder *reg_r*) und setzt dann die Lock-Variable auf true.

Beweisen Sie mit Hilfe eines Zustandsdiagramms, dass diese Lösung für zwei Prozesse korrekt ist. D.h., dass der Wechselzeitige Ausschuss gewährleistet wird und das kein Deadlock entstehen kann.

```
boolean lock ← false
```

T_p		T_r	
	boolean local_p ← true		boolean local_r ← true
	loop forever		loop forever
p_1:	NCS	r_1:	NCS
	repeat		repeat
p_2:	TSL (lock, local_p)	r_2:	TSL (lock, local_r)
p_3:	until !local_p	r_3:	until !local_r
p_4:	CS	r_4:	CS
p_5:	lock ← false	r_5:	lock ← false

3. Aufgabe (14 Punkte)

Betrachten Sie folgende Version des Peterson-Algorithmus für zwei Prozesse.

a) Zeige mit Hilfe eines Zustandsdiagramms, dass der Algorithmus korrekt ist. Sie können dabei den abgekürzten Algorithmus (ohne die NSC und CS Zeilen) verwenden.

```
boolean ready[ false, false ]
integer turn ← 0
```

T_0		T_1	
	loop forever		loop forever
p_1:	NCS	r_1:	NCS
p_2:	ready[0] ← true	r_2:	ready[1] ← true
p_3:	turn ← 1	r_3:	turn ← 0
p_4:	await !ready[1] or turn = 0	r_4:	await !ready[0] or turn = 1
p_5:	CS	r_5:	CS
p_6:	ready[0] ← false	r_6:	ready[1] ← false

b) Zeigen Sie, dass folgende Invarianten gelten:

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

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

4. Aufgabe (Freiwillige Aufgabe)

Antworten Sie folgende Fragen über das Frösche-Rätsel aus der Vorlesung:

- Gibt es immer eine Lösung für n weibliche und n männliche Frösche?
- Was ist die minimale?/maximale? Anzahl von Bewegungen, die zur Lösung führen?
- Welche Regeln müssen beachtet werden, damit keine Blockierung stattfindet?
- Wie sieht der Algorithmus aus, der die Verklemmungen vermeidet?

Allgemeine Wichtige Hinweise zur Übungsabgabe:

1. Es muss der Quellcode hochgeladen werden (.java-Dateien), nicht der kompilierte Bytecode (.class-Dateien).
2. Zusätzlich zur Online-Abgabe muss der Quellcode vollständig ausgedruckt werden. Der ausgedruckte Quellcode muss lesbar sein, insbesondere ist auf automatische Zeilenumbrüche zu achten.
3. Für die gute Verständlichkeit des Quellcodes sind sinnvolle Bezeichnernamen zu wählen und der Code ausreichend zu kommentieren.
4. Das Programm muss Testläufe enthalten, die die Aufgabe vollständig abdecken. Das heißt, zum Testen sollen keine Änderungen am Code durch die Tutorin/den Tutor mehr notwendig sein. Die Testläufe müssen in einer eigenen Klasse Main enthalten sein.
5. Das Programm muss mit Java 8 kompatibel sein.
6. Alle Übungsaufgaben, die nicht Programmieraufgaben sind, müssen ebenfalls digital und nicht handschriftlich sowohl online als auch ausgedruckt zu dem im KVV angegebenen Datum (in der Regel Dienstag um 11:55) abgegeben werden. Zu spät abgegebene Lösungen werden mit 0 Punkten bewertet.