Freie Universität Berlin

Prof. Dr. Margarita Esponda

Nichtsequentielle Programmierung, SoeSe 2017

Project 1

TutorIn: Lilli Walter
Tutorium Tutorium 6

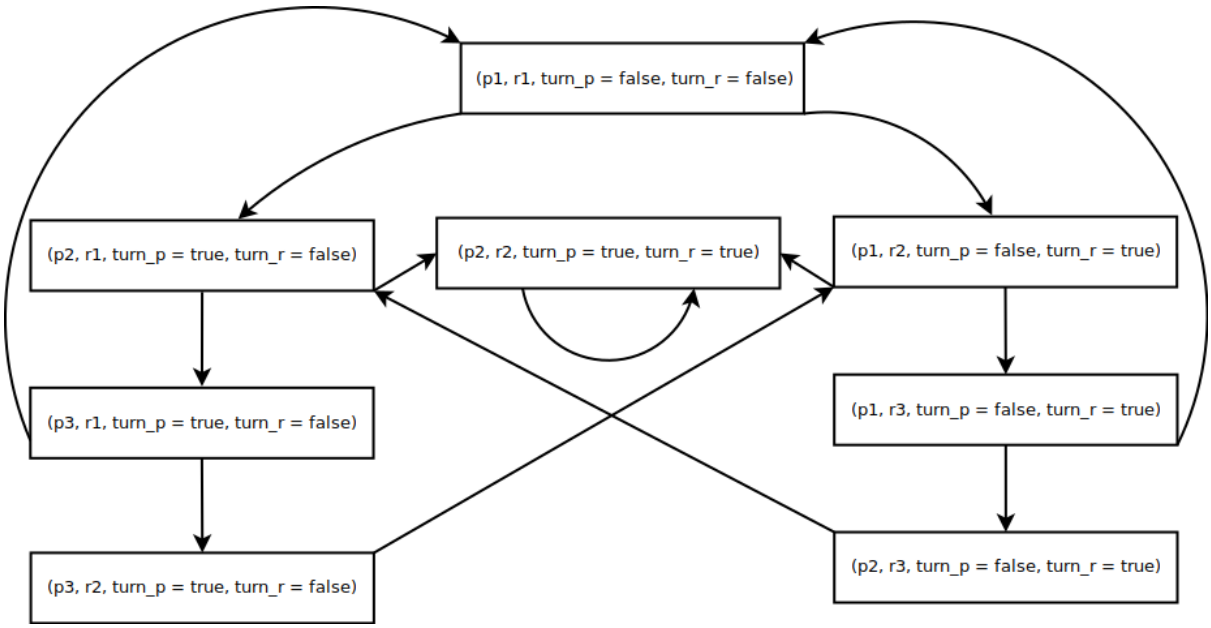
Boyan Hristov

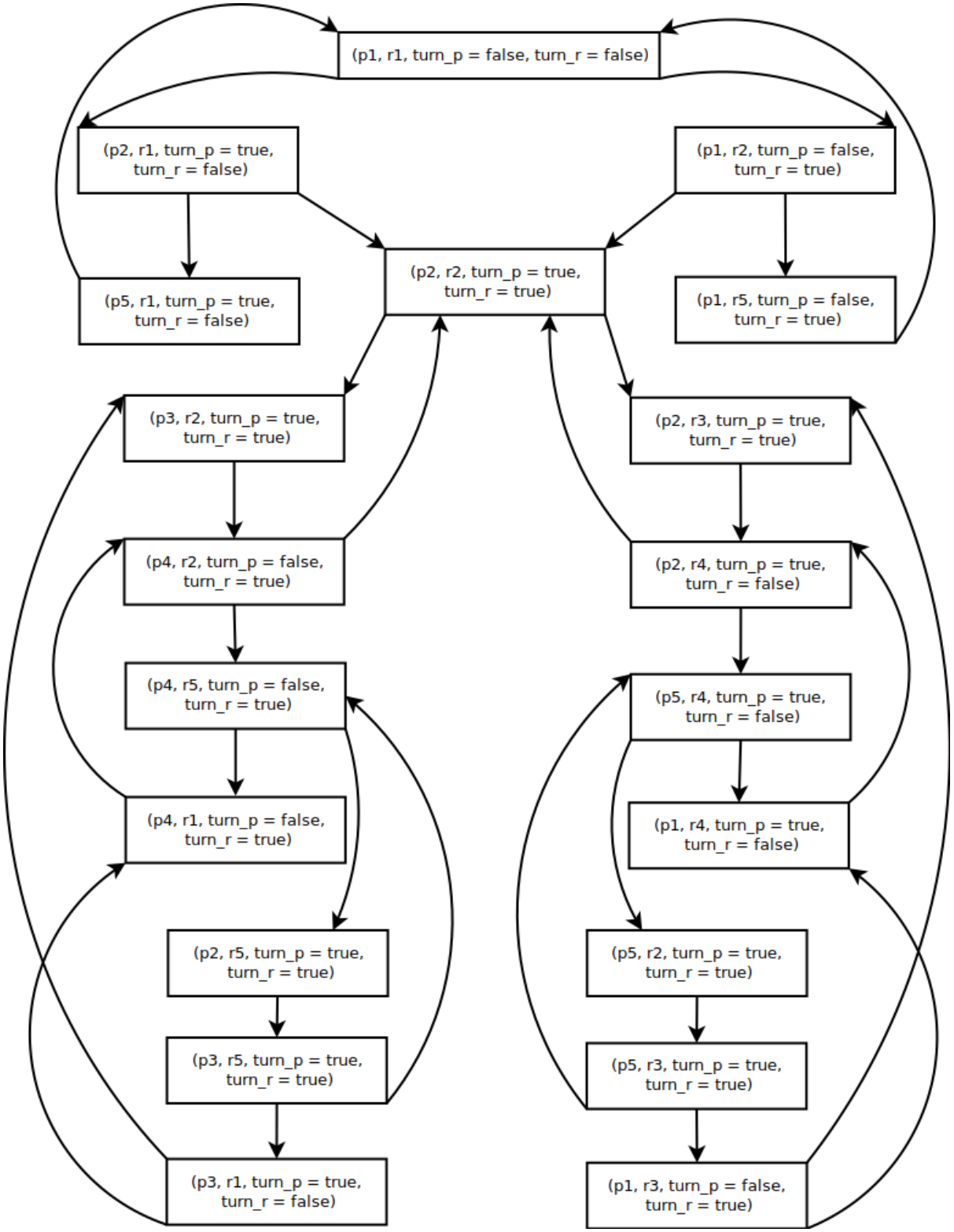
16. Mai 2017

Link zum Git Repository: https://github.com/BoyanH/FU-Berlin-ALP4/tree/master/Solutions/Homework2

1 Aufgabe

Man kann erkenne, dass Wechselseitige Ausschluss gewährleistet ist, da es keine Zustände (p3, r3) für den 1. Algorithmus und (p5, r5) für den 2. Algorithmus existieren.

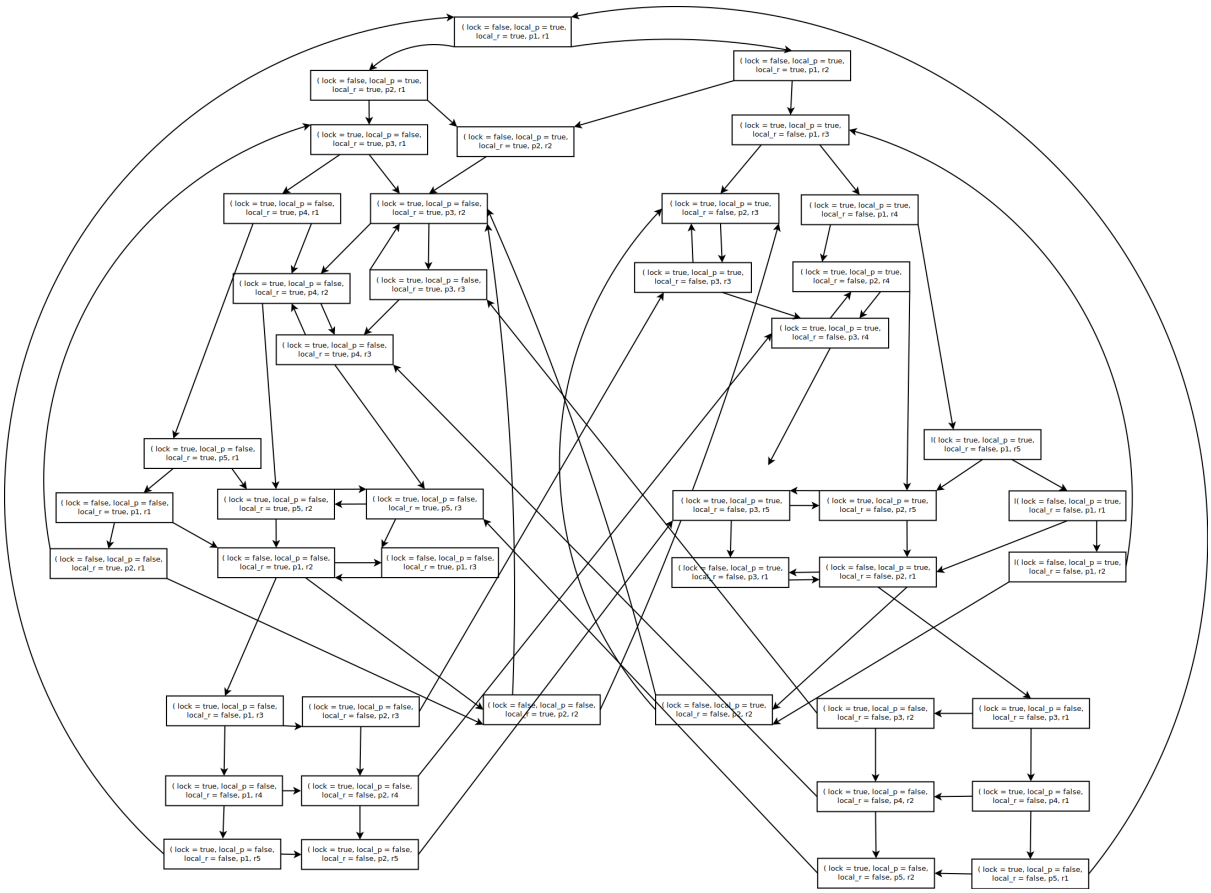



(p1, r1, turn_p = false, turn_r = false)
(p2, r1, turn_p = true, turn_r = false)
(p1, r2, turn_p = false, turn_r = true)
(p5, r1, turn_p = true, turn_r = false)
(p2, r2, turn_p = true, turn_r = true)
(p1, r5, turn_p = false, turn_r = true)
(p3, r2, turn_p = true, turn_r = true)
(p2, r3, turn_p = true, turn_r = true)
(p4, r2, turn_p = false, turn_r = true)
(p2, r4, turn_p = true, turn_r = false)
(p4, r5, turn_p = false, turn_r = true)
(p5, r4, turn_p = true, turn_r = false)
(p4, r1, turn_p = false, turn_r = true)
(p1, r4, turn_p = true, turn_r = false)
(p2, r5, turn_p = true, turn_r = true)
(p5, r2, turn_p = true, turn_r = true)
(p3, r5, turn_p = true, turn_r = true)
(p5, r3, turn_p = true, turn_r = true)
(p3, r1, turn_p = true, turn_r = false)
(p1, r3, turn_p = false, turn_r = true)

2 Aufgabe

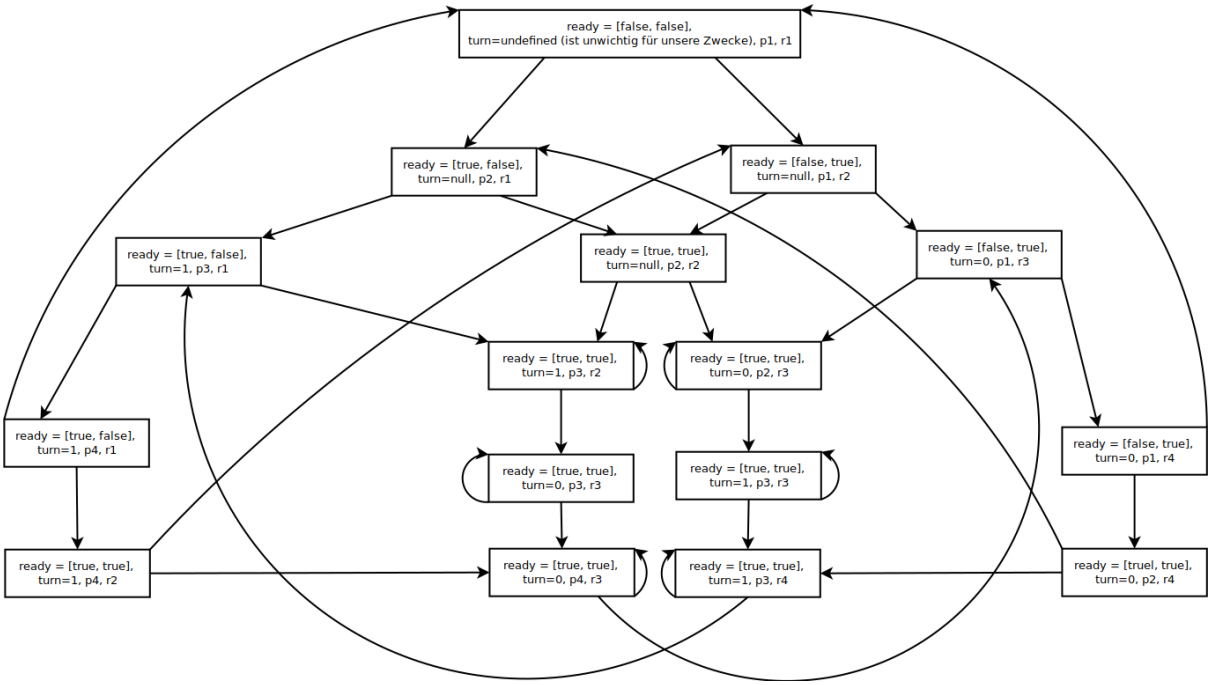
Wir haben irgendwann erkannt, dass die Idee die Aufgabe nicht solche ist, da diese nur 8 Punkte gibt. Eigentlich ist aber in Assembler so, dass man erstmal ein Exchange machen kann, und dann noch Compare + JE oder JNE. Wir haben deswegen gemeint, dass 'until !local_r' eine Art von 'If true, jump to r2/p2, else continue to r4/p4'. Wir dachten dass es trotzdem Sinn macht diese Variante zu zeigen, wollten auch unsere Arbeit nicht wegschmeißen.

Man erkennt, dass es keine Deadlocks gibt, da es keine Zustände gibt mit nur eingehende Pfeile. Wechselseitige Ausschluss ist auch gewährleistet, da die verbotene Zustände (diese, die p4 und r4 gleichzeitig enthalten) nicht existieren.



3 Aufgabe

a Wir haben schon bei Aufgabe 2 gezeigt, dass lokale Variablen, die am Anfang der endlosen Schleifen gesetzt werden für den Zustandsdiagramm unwichtig sind, deswegen haben wir diese uns diesmal gespart. Man kann sehen, dass es keine Zustände mit nur eingehende Pfeile gibt (Deadlocks) und keine verbotene Zustände (p4, r4), also Wechselseitige Ausschluss ist auch gewährleistet.



b

$p_4 \wedge r_5 \equiv p_3 \wedge r_4$ (In verkürzte Variante)
$\Rightarrow$ Man sieht am Diagramm, dass der einzige Zustand im Diagramm, der diese Bedingung erfüllt, erfüllt auch die rechte Seite der Implikation

$p_5 \wedge r_4 \equiv p_4 \wedge r_3$ (In verkürzte Variante)
$\Rightarrow$ Man sieht am Diagramm, dass der einzige Zustand im Diagramm, der diese Bedingung erfüllt, erfüllt auch die rechte Seite der Implikation

Alternativ, ready ist im 3...6 immer True für beide Threads. (offensichtlich). Der Prozess, der als zweites die dritte Zeile ausführt kommt in den Critical Section rein (5. Zeile) und das zweite bleibt stehen an der 4. Zeile (man kann das aus dem Zustandsdiagramm ablesen).

Weitere Beweise sind nicht nötwendig, da im mathematischen Sinne ist die Abdeckung aller möglichen Fälle (Zustandsiagramm) schon ausreichend.