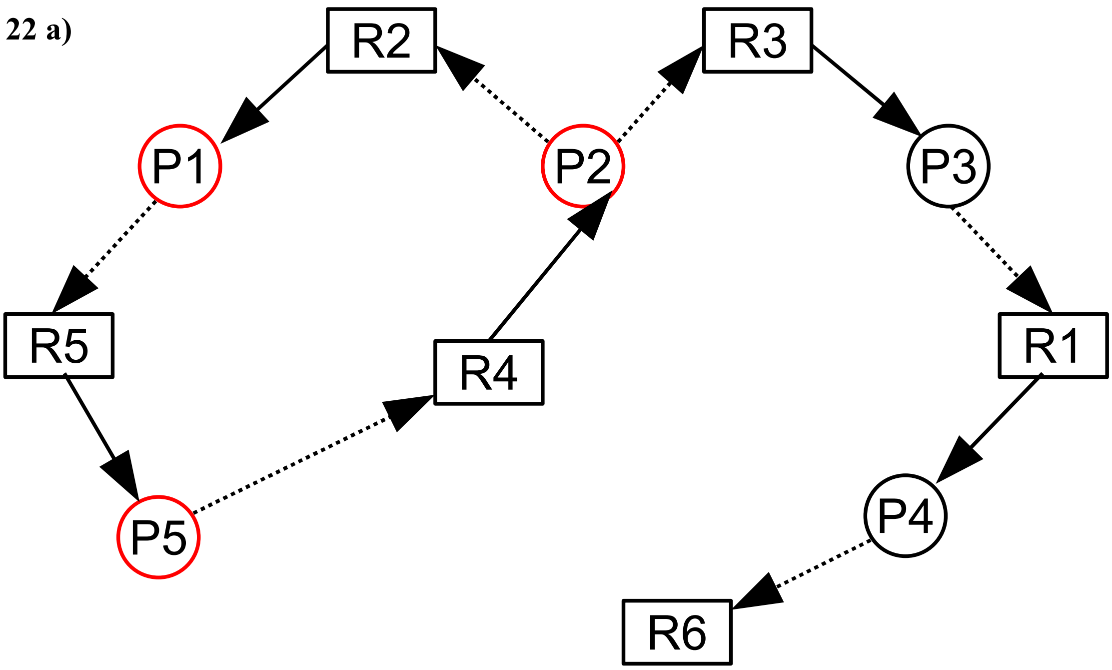
Übung 11

22 a)



Es ergibt sich ein (gerichteter!) Kreis mit den Prozessen P1, P2 und P5 (sowie den beteiligten Ressourcen R2, R4, R5), das System ist also im Deadlock-Zustand.

22 b)

Alle Ressourcen sind nur 1x da: E = (1 1 1 1 1 1)
Belegung: nur R6 ist frei, also A = (0 0 0 0 0 1)

Aktuelle Belegung:

C =		R =		
0 1 0 0 0 0	(P1 hat R2)	0 0 0 0 1 0	(P1 braucht R5)	
0 0 0 1 0 0	(P2 hat R4)	0 1 1 0 0 0	(P2 braucht R2, R3)	
0 0 1 0 0 0	(P3 hat R3)	1 0 0 0 0 0	(P3 braucht R1)	
~~1 0 0 0 0 0~~	~~(P4 hat R1)~~	~~0 0 0 0 0 1~~	~~(P4 braucht R6)~~	diese Zeile streichen
0 0 0 0 1 0	(P5 hat R5)	0 0 0 1 0 0	(P5 braucht R4)	

E =	A =
1 1 1 1 1 1	1 0 0 0 0 1
C =	R =
0 1 0 0 0 0	0 0 0 0 1 0
0 0 0 1 0 0	0 1 1 0 0 0
~~0 0 1 0 0 0~~	~~1 0 0 0 0 0~~
~~1 0 0 0 0 0~~	~~0 0 0 0 0 1~~
0 0 0 0 1 0	0 0 0 1 0 0

E =	A =
1 1 1 1 1 1	1 0 1 0 0 1
C =	R =
0 1 0 0 0 0	0 0 0 0 1 0
0 0 0 1 0 0	0 1 1 0 0 0
~~0 0 1 0 0 0~~	~~1 0 0 0 0 0~~
~~1 0 0 0 0 0~~	~~0 0 0 0 0 1~~
0 0 0 0 1 0	0 0 0 1 0 0

kein weiteres Streichen möglich, es bleiben P1, P2, P5 (nicht durchgestrichen) übrig, diese sind Teil des Deadlocks. Das 2. Verfahren bestätigt also das Ergebnis aus dem ersten Verfahren.

23 a)

Wait-for-Graph...

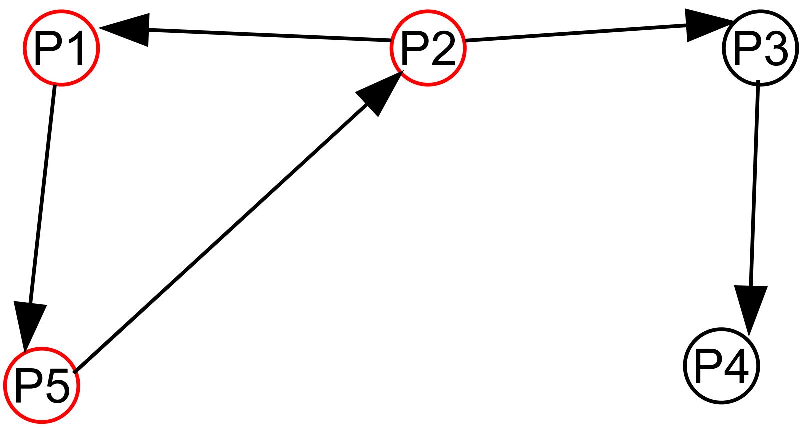
Beweis, 1. Richtung: Kreis im Ressourcenzuordnungsgraph **R** → Kreis im Wait-for-Graph **W**:

Sei $P_1 \rightarrow R_1 \rightarrow P_2 \rightarrow R_2 \dots \rightarrow P_n \rightarrow R_n \rightarrow P_1$ der Kreis in **R**.
Dann ist wegen $P_1 \rightarrow R_1 \rightarrow P_2$ in **R** die Kante $P_1 \rightarrow P_2$ in **W**.
Analog gilt $P_2 \rightarrow P_3$ in **W**, $P_3 \rightarrow P_4$ in **W** etc., bis $P_{n-1} \rightarrow P_n$ in **W** und $P_n \rightarrow P_1$ in **W**.
Also gibt es einen Kreis $P_1 \rightarrow P_2 \rightarrow \dots P_n \rightarrow P_1$ in **W**.

Beweis, 2. Richtung: Kreis im Wait-for-Graph → Kreis im Ressourcenzuordnungsgraph

Sei $P_1 \rightarrow P_2 \rightarrow \dots P_n \rightarrow P_1$ ein Kreis in **W**.
Nach Definition des Wait-for-Graphen kann nur $P_1 \rightarrow P_2$ in **W** sein, wenn es eine Ressource R_1 gibt, so dass $P_1 \rightarrow R_1 \rightarrow P_2$ in **R** liegt. Analog muss es weitere Ressourcen $R_2, \dots, R_n$ geben, so dass insgesamt $P_1 \rightarrow R_1 \rightarrow P_2 \rightarrow R_2 \dots \rightarrow P_n \rightarrow R_n \rightarrow P_1$ in **R** liegt, also ein Kreis in **R** vorhanden ist.

23 b)
Erstellt man zu Aufgabe 22) den Wait-for-Graph, ergibt sich folgendes Bild:



Auch hier ist der Kreis sichtbar, der P1, P2 und P5 enthält.

24 a)

E = 4 9 7	A = 1 3 1		E = 4 9 7	A = 1 3 5		E = 4 9 7	A = 2 6 5
C = 2 0 1	R = 2 0 6	→	C = 2 0 1	R = 2 0 6	→	C = 2 0 1	R = 2 0 6
0 2 2	3 4 3		0 2 2	3 4 3		0 2 2	3 4 3
1 3 0	1 2 2		~~1 3 0~~	~~1 2 2~~		~~1 3 0~~	~~1 2 2~~
~~0 0 4~~	~~0 1 0~~		~~0 0 4~~	~~0 1 0~~		~~0 0 4~~	~~0 1 0~~

Nach diesen Schritten geht es nicht weiter, weil weder der erste noch der zweite Prozess ihre Anforderungen mit A = (2 6 5) erfüllen können. Es befinden sich also P1 und P2 im Deadlock.

24 b)
Bei der Graphreduzierung werden alle Kanten zum Prozess und vom Prozess entfernt.

- Das Entfernen einer solchen Kante, die einen Ressourcenbedarf darstellt (Pfeil P → R), bedeutet praktisch, dass dieser Prozess nicht weiter beachtet werden muss, und entspricht dem Streichen der Zeile in unserem Algorithmus
- Das Entfernen einer Kante, die eine Ressourcenbelegung darstellt (Pfeil R → P), bedeutet, dass der Prozess seine gehaltenen Ressourcen freigibt, und entspricht dem Aktualisieren von **A** in unserem Algorithmus (Addieren der frei werdenden Ressourcen aus **C** zu **A**).

Damit sind beide Verfahren gleichwertig.

24 c)
Zwei Möglichkeiten, mit mehrfachen Ressourcen umzugehen, sind:

1. mehrere Pfeile zwischen Prozessen und Ressourcen
2. Pfeile zwischen Prozessen und Ressourcen mit „Vielfachheit“ beschriften