13. Deadlock – einfach verfügbare Ressourcen

Es gebe fünf Prozesse P_1, P_2, P_3, P_4 und P_5 sowie sechs Ressourcen R_1, R_2, R_3, R_4, R_5, R_6. Es gelte dabei:

- P_1 hat R_2 belegt und fordert R_5 an.
- P_2 hat R_4 belegt und fordert R_2 und R_3 an.
- P_3 hat R_3 belegt und fordert R_1 an.
- P_4 hat R_1 belegt und fordert R_6 an.
- P_5 hat R_5 belegt und fordert R_4 an.

a) Zeichnen Sie den Ressourcen-Zuordnungsgraph für dieses Szenario und leiten Sie daraus ab, ob sich die fünf Prozesse im Deadlock-Zustand befinden. Begründen Sie Ihre Antwort.

b) Überprüfen Sie Ihr Ergebnis, indem Sie den Deadlock-Erkennungs-Algorithmus (mit den Belegungs- und Anforderungsmatrizen) durchführen.

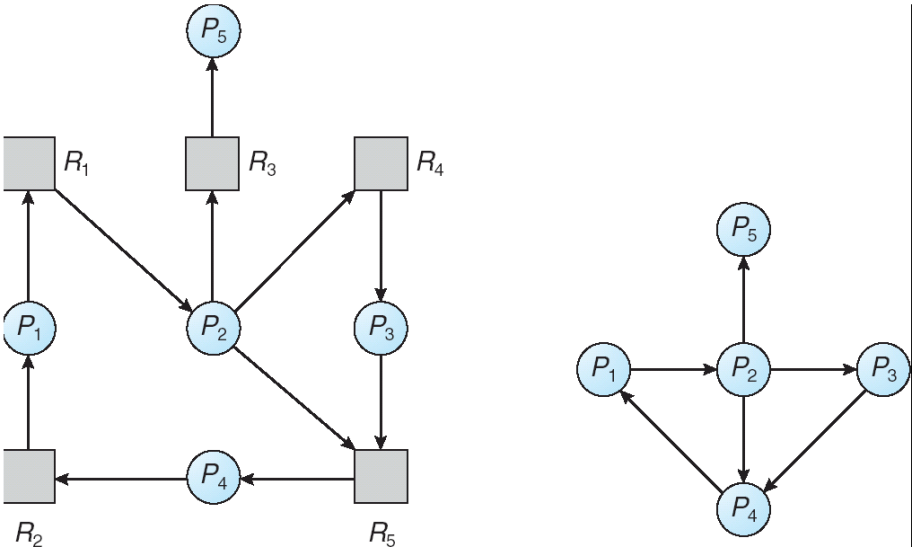
14. Wait-for-Graph

Mit dem folgenden Algorithmus können Sie jeden Ressourcen-Zuordnungs-Graph in einen so genannten **Wait-for-Graph** umwandeln:

Seien P_1, ... P_n alle Prozess-Knoten und R_1, ... R_m alle Ressourcen-Knoten des Graphs.

1. Der neue Graph enthält alle Prozess-Knoten P_1, ..., P_n, aber keine Ressourcen-Knoten.
2. Für jede Kombination P -> R -> P' im Ressourcen-Zuordnungs-Graph zeichnen Sie einen gerichtete Kante P -> P' in den Wait-for-Graph

Beispiel:



a) Beweisen Sie, dass es genau dann einen Kreis im Wait-for-Graph gibt, wenn es auch einen Kreis im zugrundeliegenden Ressourcen-Zuordnungs-Graph gibt. (Ein solche „genau dann wenn“-Beweis besteht immer aus zwei Richtungen.)

   Ersatzweise argumentieren Sie anschaulich, warum diese Behauptung gilt.

b) Erzeugen Sie zum Graph aus Aufgabe 1) den Wait-for-Graph und untersuchen Sie diesen auf Kreise.

15. Banker-Algorithmus

Für drei Ressourcen-Klassen A, B und C sowie vier Prozesse P_1, P_2, P_3, P_4 seien folgende Ressourcenbelegungen (**C**) und Maximalanforderungen (**Max**) gegeben:

$$\mathbf{C} = \begin{pmatrix} 0 & 0 & 2 \\ 2 & 1 & 0 \\ 0 & 2 & 2 \\ 1 & 3 & 0 \end{pmatrix} \quad \mathbf{Max} = \begin{pmatrix} 0 & 1 & 2 \\ 2 & 9 & 3 \\ 0 & 7 & 5 \\ 1 & 5 & 2 \end{pmatrix}$$

$$E = (4\ 9\ 5)$$
$$A = (1\ 3\ 1)$$

A gibt die aktuell noch verfügbaren Ressourcen an, E die Gesamtressourcen.

a) Ist dieses System in einem sicheren Zustand? Falls ja, geben Sie eine Ausführreihenfolge der Prozesse an, in der jeder Prozess zunächst seine maximalen Ressourcenforderungen stellt und anschließend alle belegten Ressourcen freigibt.

b) Prozess P2 stellt die Anfrage (0 1 1) an das System. Prüfen Sie mit dem Banker-Algorithmus, ob das System diese Anfrage bedienen darf, wenn es Deadlocks vermeiden will.

16. Viele Threads, viele Ressourcen

Ein Programm besteht aus mehreren Threads, und es gibt mehrere Ressourcen, die jeweils durch einen Mutex geschützt sind. Einer der Threads benötigt regelmäßig eine Reihe von Ressourcen, wobei die Zusammensetzung der gerade benötigten Ressourcen ständig wechselt. Er fordert sie dabei in zufälliger Reihenfolge an. Alle übrigen Threads benötigen stets nur eine Ressource, die sie anfordern, dann nutzen und wieder freigeben.

Kann es in diesem Programm zu einem Deadlock kommen?

Lösungsblatt – Übung 7

Punkte:

	Name	**Matrikelnummer**
Teilnehmer 1		
Teilnehmer 2		