5. Betriebsmittelverwaltung

5.1 Einführung

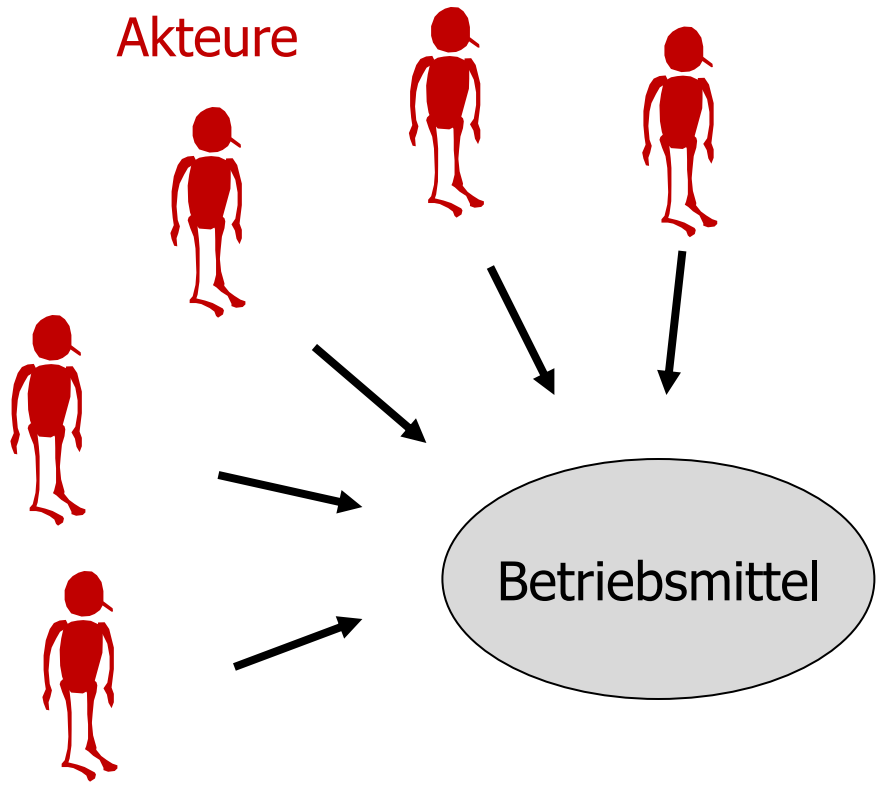
5.2 Auswahlstrategien

5.3 Verklemmungen

5.4 Verklemmungsvermeidung

5.1 Einführung

- Betriebsmittel (BM) oder Ressource (resource)
  - Alles, was ein Prozess als Aktivitätsträger in einem System zum Vorankommen benötigt
  - BM nur dann ein Problem, wenn sie nicht in beliebigem Umfang zur Verfügung stehen bzw. nicht simultan genutzt werden können



- Betriebsmittel sind knapp
- Benutzung erfolgt exklusiv
- Verwaltung ist sinnvoll

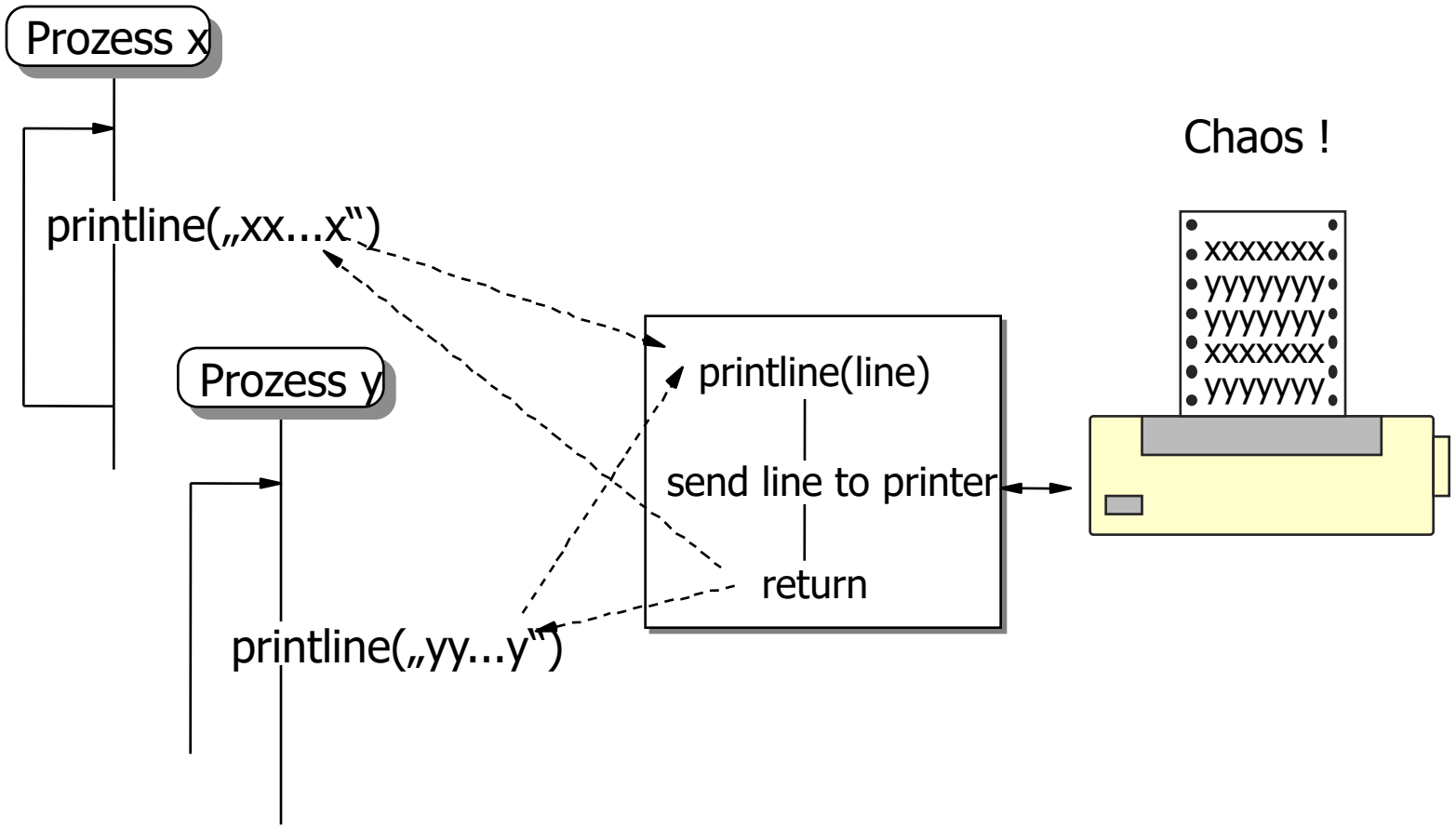
Akteure, z.B.
- Benutzer
- Prozess
- Thread

Betriebsmittel, z.B.
- Prozessor
- Speicher
- Bandbreite

- Datei
- Signal
- Nachricht
- Name
- Farbe

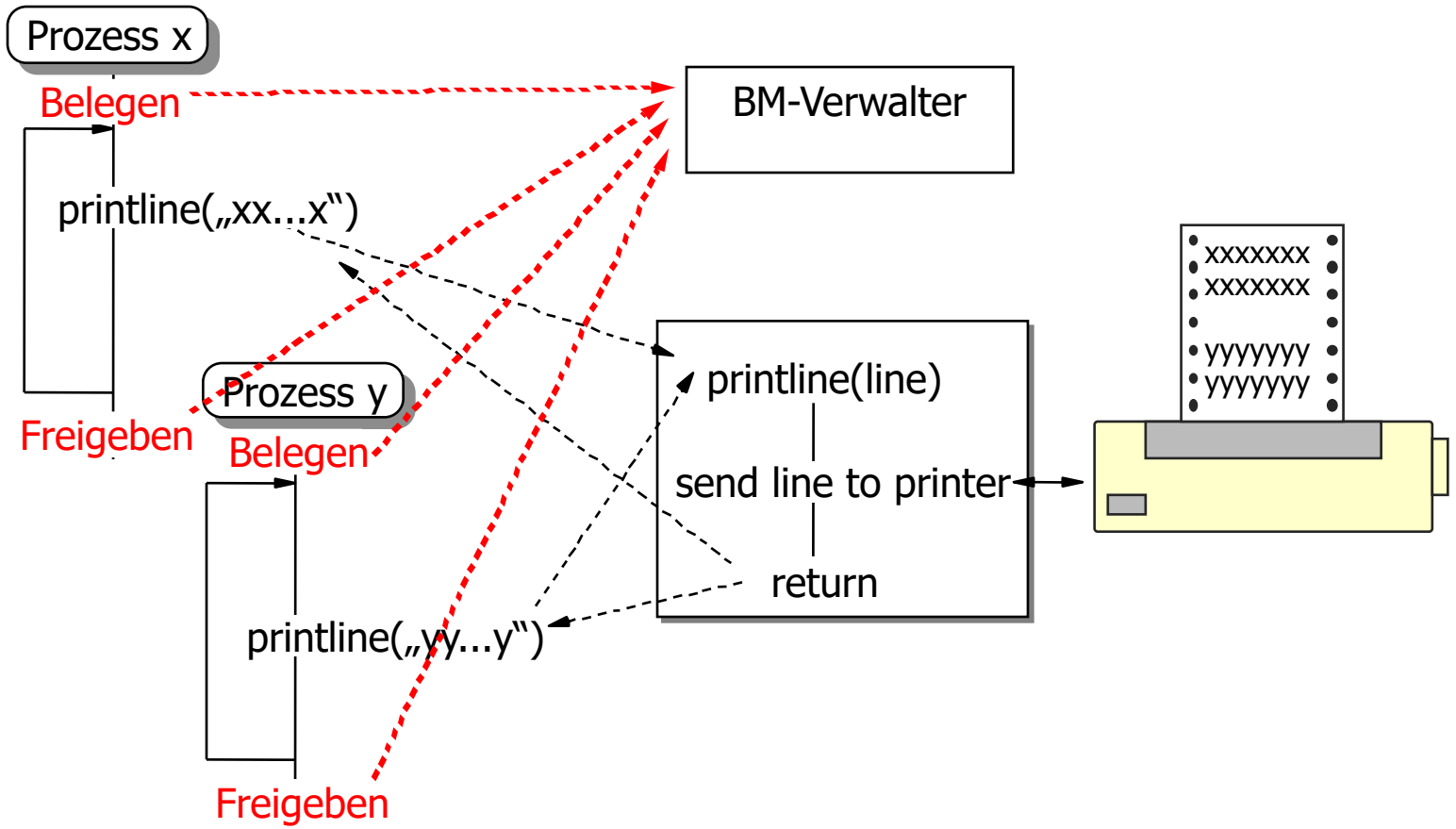
Unkoordinierte Nutzung

- Bei unkoordinierter Nutzung eines Betriebsmittels können unerwünschte Effekte auftreten



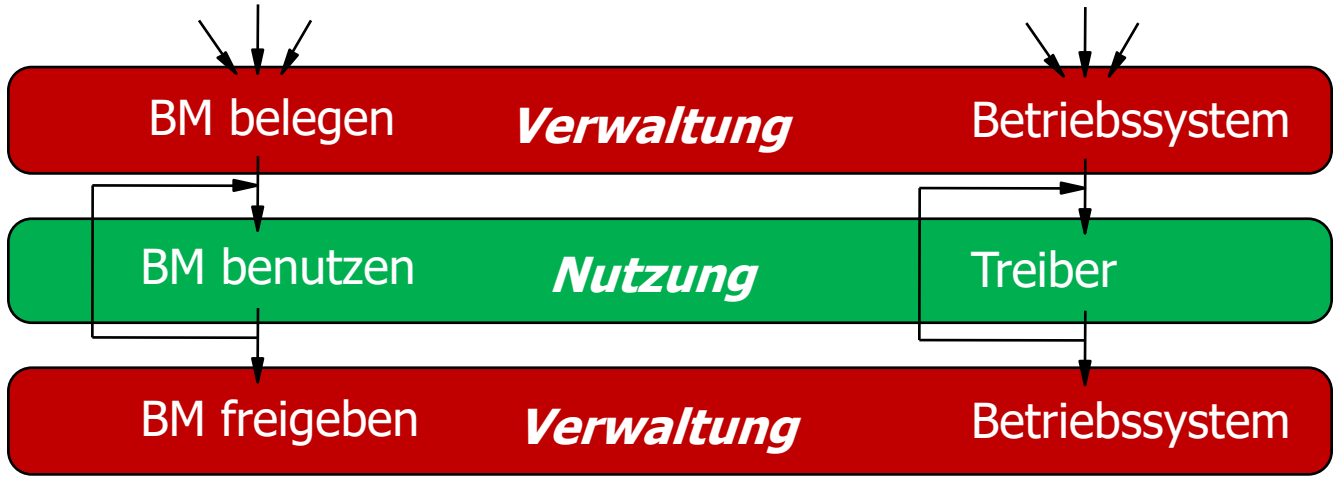
Koordination durch Verwaltungskomponente

- Durch den Einsatz eines BM-Verwalters kann z.B. eine exklusive Nutzung erreicht werden



Betriebsmittelverwaltung

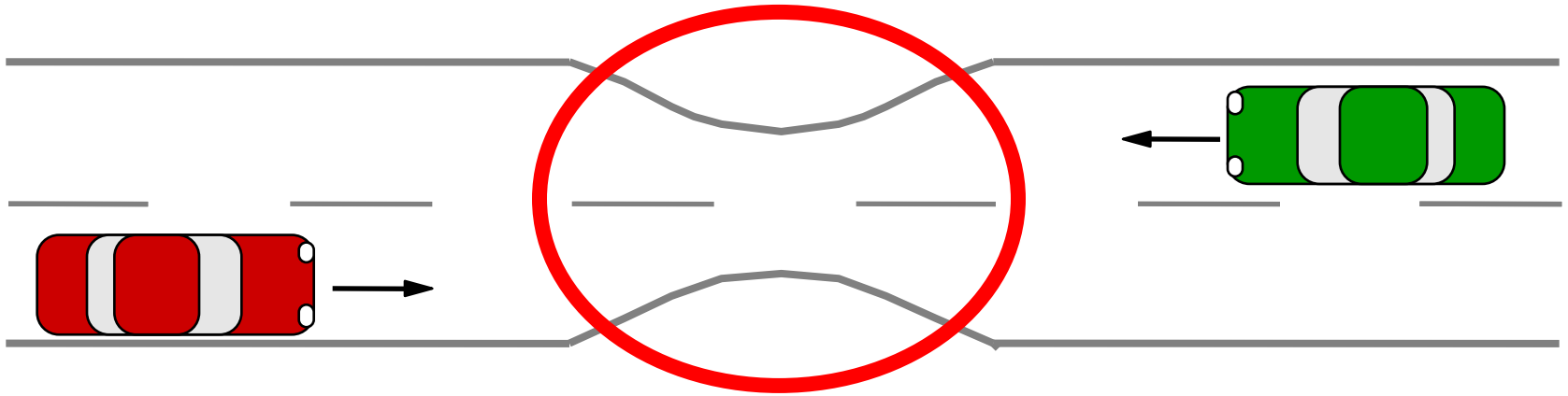
- Differenzierung zwischen Nutzung und Verwaltung!
- Die Nutzung wird von Verwaltungsoperationen geklammert!



Betriebsmittelprobleme im Alltag

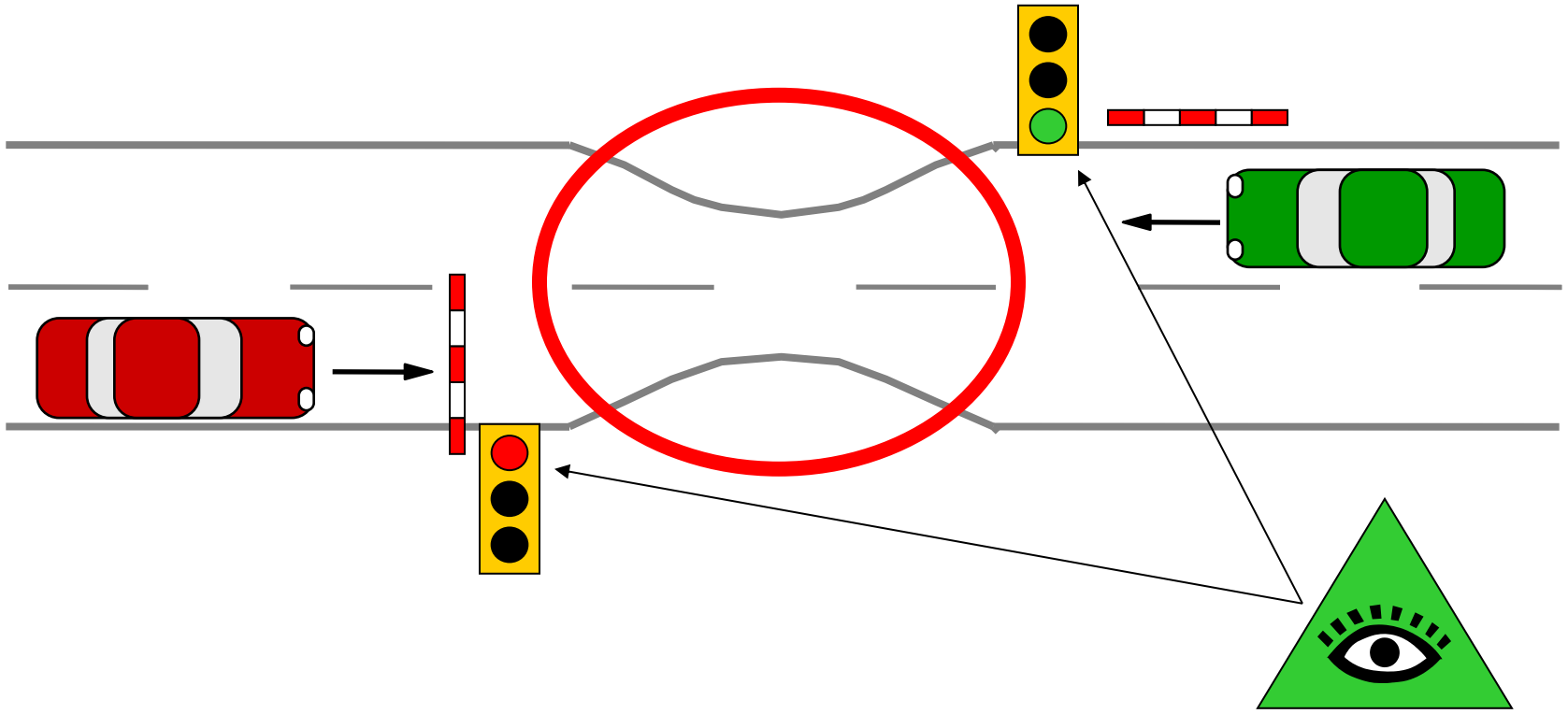
- Mögliche Lösungen?
  -
  -
  -

Ein Straßenengpass



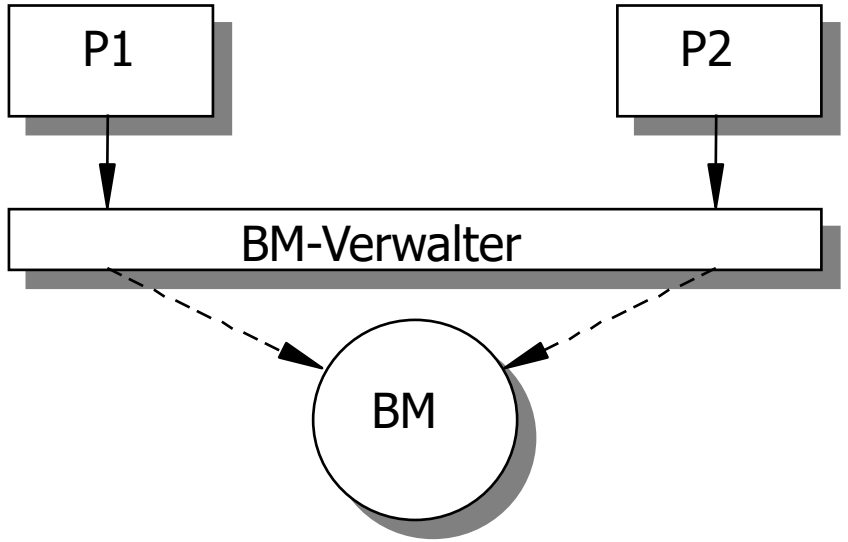
Betriebsmittelprobleme im Alltag

- Lösung 1
  - Zentrale Instanz entscheidet (Betriebsmittelverwalter)
  - Ampeln, Schranken



Lösung 1: Betriebsmittelverwalter

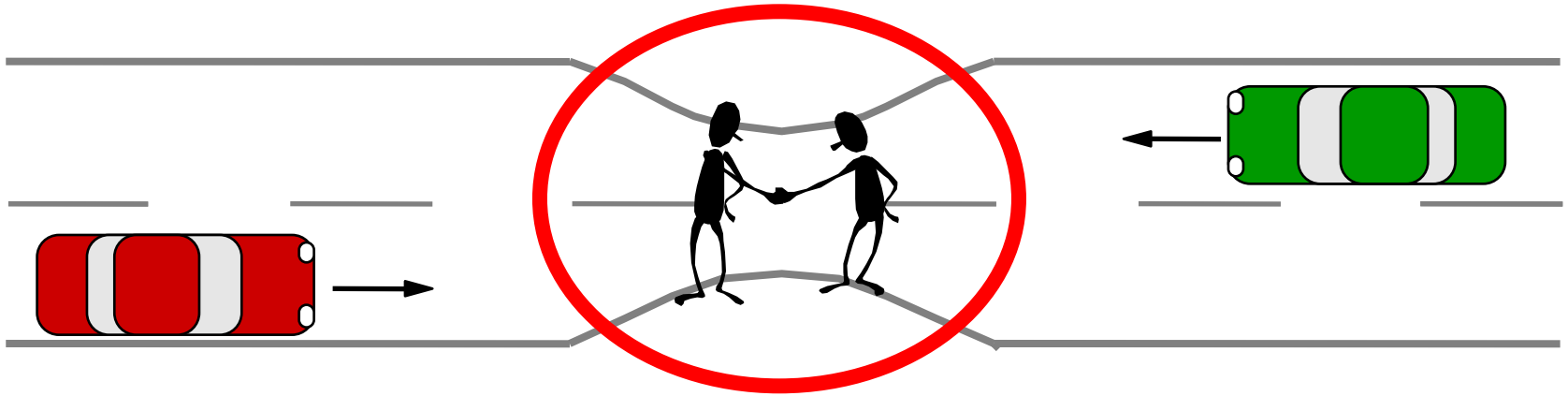
- Nutzung des BM nur nach vorheriger Belegung möglich
- Die vorherige Belegung wird durch eine zwischengeschaltete Instanz erzwungen



- Beispiele
  - Hauptspeicherverwaltung (Zugriffe sind nur innerhalb der zugewiesenen Speicherbereiche möglich)
  - Monitor (Aufruf einer Prozedur ist nur möglich, wenn der Monitor frei, also nicht von einem anderen Thread belegt ist.)
  - Drucken (Zugriff auf den Drucker ist nicht unmittelbar möglich, sondern nur über spezielle Software, den Treiber, der als Verwalter fungiert)

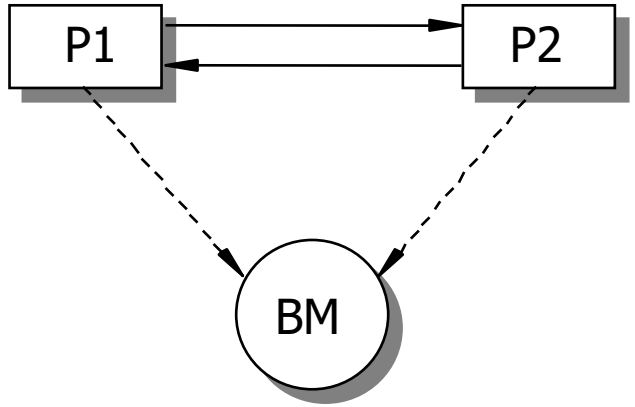
Betriebsmittelprobleme im Alltag

- Lösung 2
  - Verständigung der Teilnehmer (Regeln, Verhandlung, Protokoll)
  - Verkehrszeichen, „Berg- vor Talfahrt“, Handzeichen, Lichthupe



Lösung 2: Verständigung

- Die Bewerber um das Betriebsmittel stimmen sich ab (Protokoll)



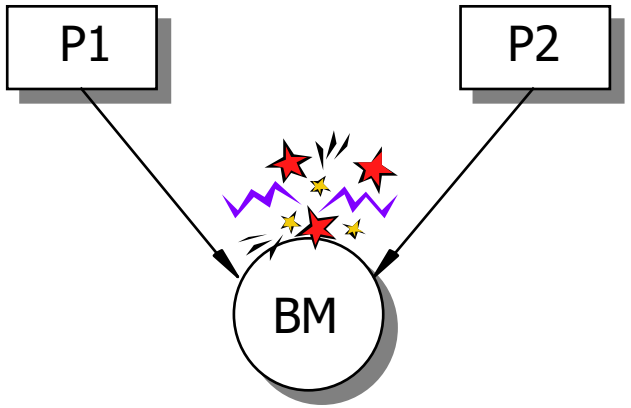
- Beispiele für Protokolle / Regeln
  - Kritischer Abschnitt
    - Vereinbarung, den kritischen Abschnitt durch Nutzung von Sperren unter gegenseitigen Ausschluss zu stellen
  - Dezentrale Bus-Arbitrierung
    - Sendewillige Komponenten (bus request) legen im speziellen Koordinationsprotokoll (Bus-Arbitrierung) fest, wer als nächster senden darf
  - Verteilte Systeme
    - Knoten melden per Broadcast Bedarf an, Abstimmung basierend auf logischer Zeit legt Zugriffsreihenfolge fest

Betriebsmittelprobleme im Alltag

- Lösung 3
  - Keinerlei Maßnahmen
  - Kollisionsgefahr

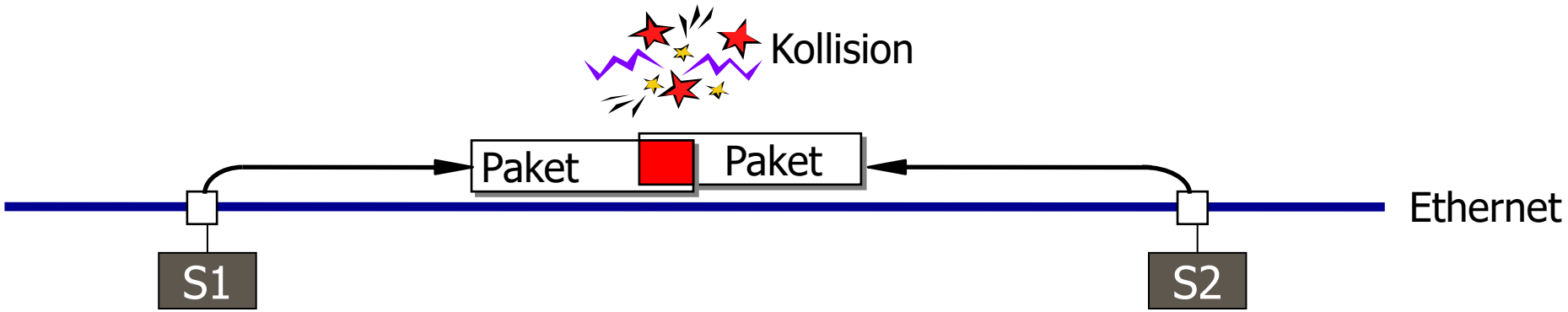
Lösung 3: Unkoordinierte Nutzung

- Ohne Abstimmung der Interessenten kann es zu Kollisionen kommen
  - ⇒Kollisionen müssen geeignet aufgelöst werden
- Aufwand für Kollisionsauflösung kann geringer sein als der permanente Aufwand für eine vorherige Abstimmung
- Einsatz dort, wo
  - ➢ eine Kollision unwahrscheinlich, d.h. selten ist und
  - ➢ der durch die Kollision entstandene „Schaden" „reparabel" ist.



Beispiele für unkoordinierte Nutzung

- Optimistische Synchronisation von Transaktionen (Validierung)
  - Transaktionen setzen keine Sperren, sondern greifen einfach zu
  - Zugriffe werden protokolliert (Log)
  - Am Ende (Commit) Überprüfung, ob Konflikte zu anderen aufgetreten sind (Validierung)
  - Falls ja, wird die Transaktion abgebrochen (und neu gestartet)
- Lokale Netze: Kollisionsbehaftete Protokolle (Ethernet)
  - Sendewillige Station sendet, nachdem sie vorher kurz die Leitung abgehört hatte
  - Senden zwei Stationen gleichzeitig, so kollidieren die Datenpakete und werden zerstört ⇒ beide Stationen müssen nach Wartezeit erneut die Daten schicken



Klassifikation von Betriebsmitteln

- Unterscheidung: real / logisch / virtuell
- Reale Betriebsmittel: physisch existent (z.B. Hauptspeicher, SSD, CPU, GPU)
- Logische Betriebsmittel: Abstraktion des realen BM (Verwaltet vom BS, oft Nutzer:in präsentiert)
  - Komfortablere, funktional angereicherte Schnittstelle im Vergleich zum realen BM
  - Beispiel: Datei = Abstraktion der Platte, Fenster = Abstraktion des Bildschirms
- Virtuelle Betriebsmittel: eine größere Anzahl eines BM als real vorhanden wird vorgespiegelt (Verwaltet vom BS)
  - Virtuelle BM werden nur für kurze Zeiten auf das reale BM abgebildet (Multiplexing)
  - Beispiel: Virtueller Speicher, Virtuelle Verbindung

Klassifikation von Betriebsmitteln (2)

- Persistenz
  - Wiederverwendbar: BM werden nach Nutzung freigegeben und können von anderen Prozessen genutzt werden
  - Verbrauchbar: Einige logische BM wie z.B. Signale, Nachrichten, Zeitstempel werden durch die Nutzung verbraucht
    - $\Rightarrow$BM werden erzeugt und sind nach einmaliger Nutzung nicht mehr vorhanden

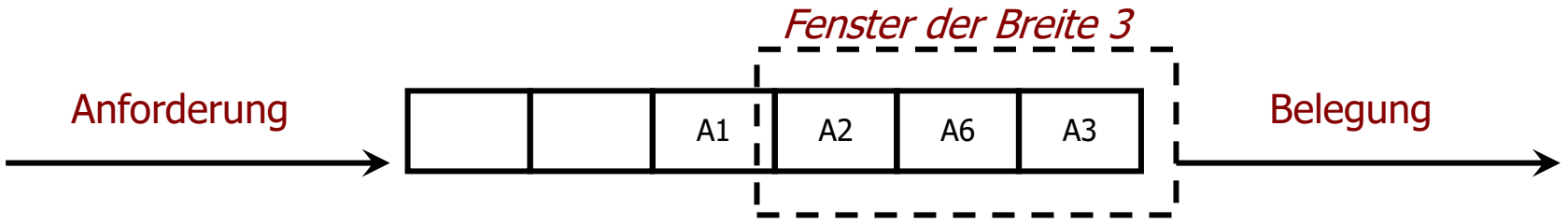
- Kapazität
  - Begrenzt: BM muss bewirtschaftet werden (explizites Belegen / Freigeben)
  - Unbegrenzt: BM-Verwaltung weitgehend verzichtbar, höchstens An-/Abmelden einer Nutzung

5.2 Auswahlstrategien

- Ziel: gute BM-Auslastung und faire Behandlung der Interessenten
  - nf(t): zum Zeitpunkt t freie BM-Einheiten, n(i): Prozess i geforderten BM-Einheiten
  - W(t): Warteschlange der angemeldeten Anforderungen / Prozesse
- Strategie First-Fit-Request: Durchsuche die Warteschlange (vorne beginnend), bis die erste erfüllbare Anforderung i gefunden ist, d.h. n(i) ≤ nf(t)
- Strategie Best-Fit-Request: Durchsuche die Warteschlange vollständig und finde die Anforderung i, welche die Restkapazität minimiert, d.h.

$$\min_{j \in W(t) \wedge n(j) \le n_f(t)} \{n_f(t) - n(j)\}$$

- Iterativ: Wende die Strategien an, bis keine Belegung mehr möglich

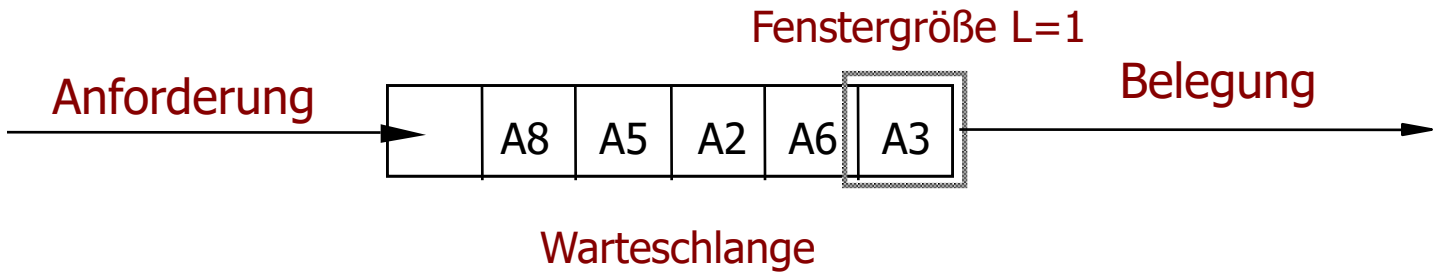


Problem des Verhungerns (Starvation)

- Bei First/Best-Fit besteht die Gefahr, dass Prozesse mit großen Anforderungen sehr lange warten müssen (Verhungern)
- Idee: Verwende Fenster dynamischer Größe
  - Initialbreite L_{max}
  - Nach jeder erfolgreichen Belegung wird die Fensterbreite folgendermaßen reduziert

$$L = \begin{cases} L - 1, \text{ falls } L > 1 \text{ und erste Anforderung nicht berücksichtigt} \\ L_{max}, \text{ sonst} \end{cases}$$

  - Nach spätestens L_{max-1} Zugriffe gilt Fenstergröße = 1, d.h. die vorderste Anforderung muss berücksichtigt werden



5.3 Verklemmung (Deadlock)

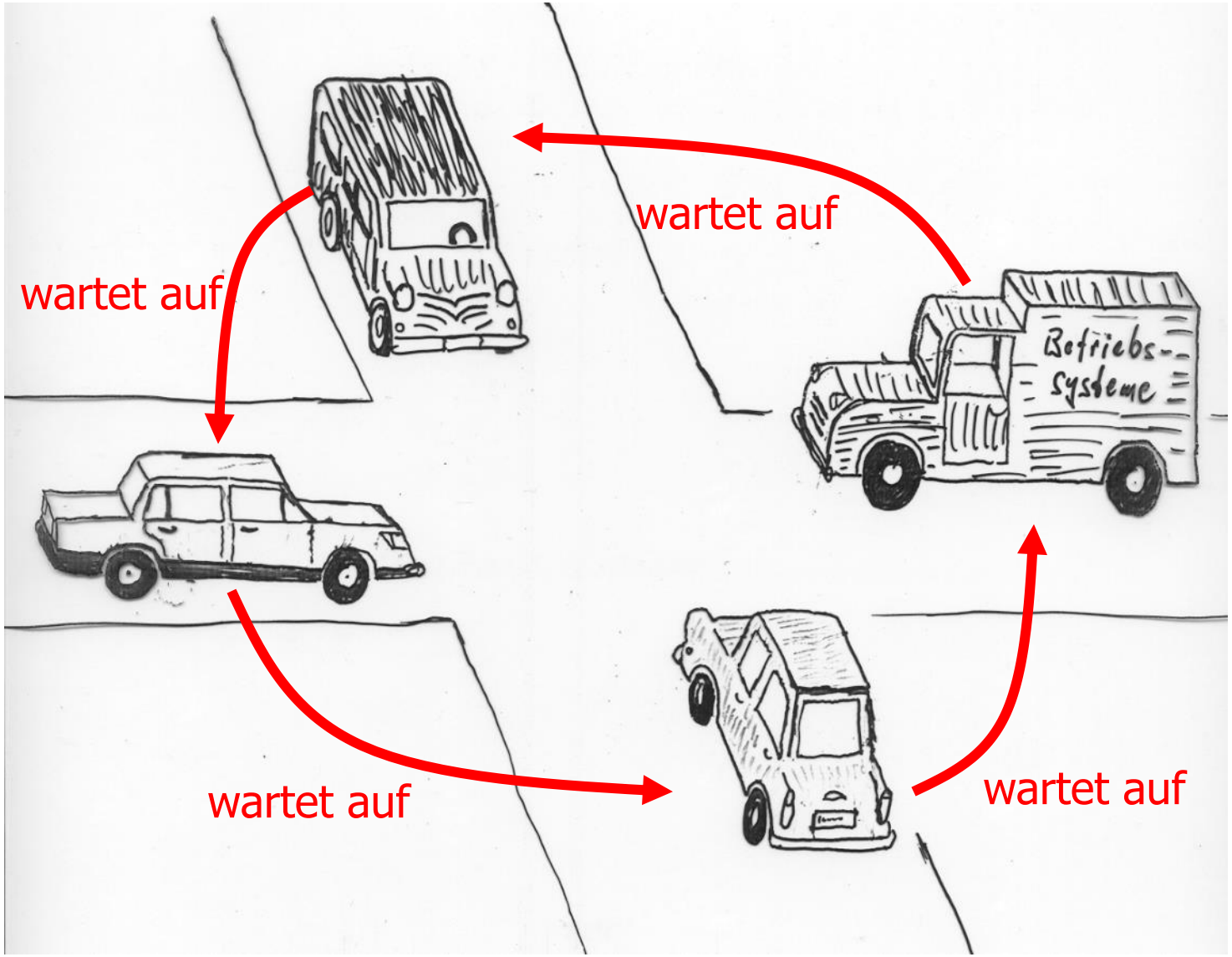
- Verklemmung = Situation, in der Prozesse sich gegenseitig behindern und blockieren und deshalb nicht weiter ausgeführt werden können
  - ⇒Abarbeitung bestimmter Operationen wird dauerhaft gestoppt
  - ⇒Kein Fortschritt im System

- Praxis: Blockierte Prozesse und Threads

- Erkennung und Behandlung: Vier notwendige und hinreichende Bedingungen überwachen

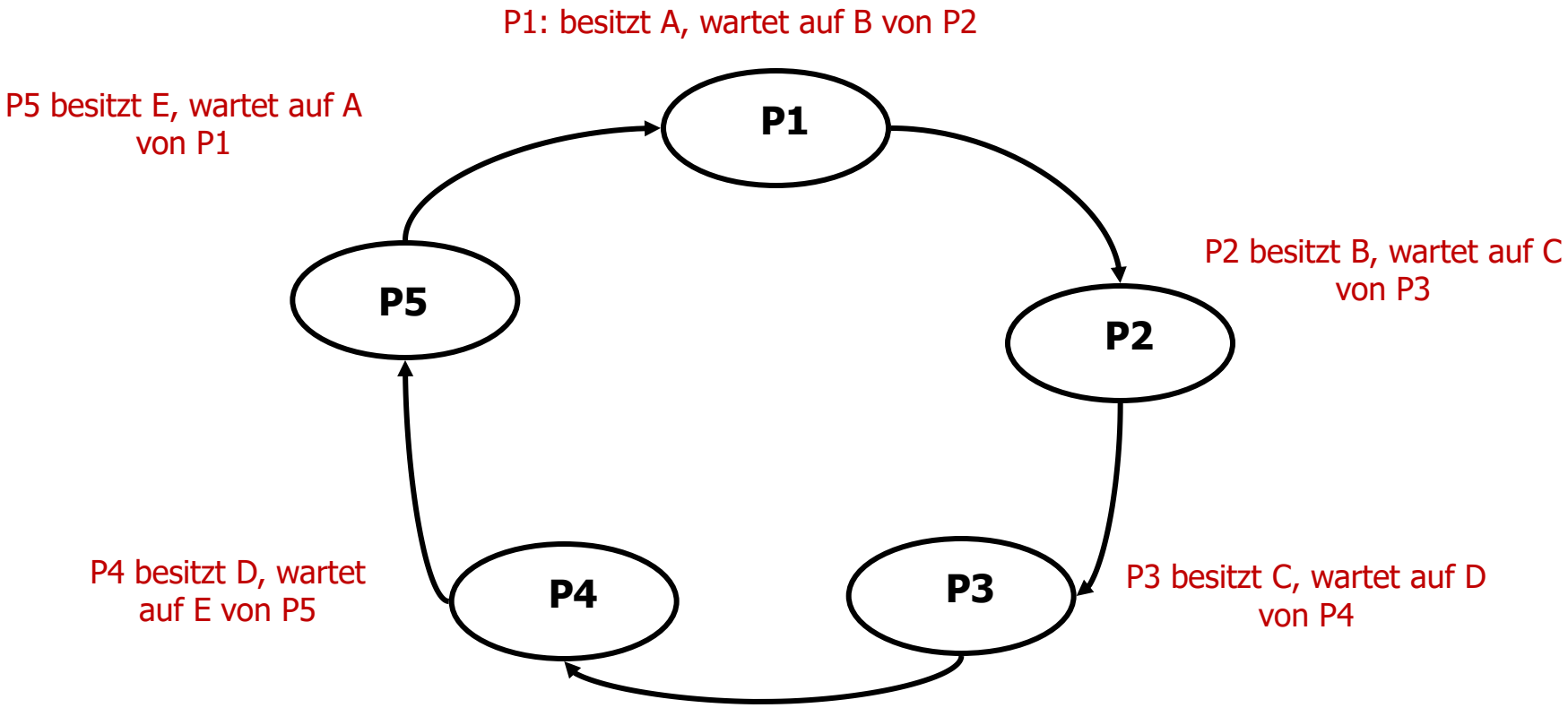
- Ziel: Vermeidung von Deadlocks durch einen Systementwurf, der eine Verklemmung erst gar nicht auftreten lässt

Verklemmung im Alltag



Wartegraph

- Wartegraph (wait-for graph) = gerichteter Graph mit den Prozessen als Knoten und Wartebeziehungen als Pfeile
- Verklemmung = charakterisiert durch Zyklus im Wartegraphen



Notwendige Bedingungen für Deadlocks

- Notwendige drei Bedingungen für das Auftreten einer Verklemmung:
  1. Beschränkte Belegung (mutual exclusion): Jedes involvierte BM ist entweder exklusiv belegt oder frei
  2. Zusätzliche Belegung (hold-and-wait): Die Prozesse haben bereits BM belegt, wollen zusätzlich weitere BM belegen und warten darauf, dass sie frei werden $\Rightarrow$ notwendige BM werden nicht auf einmal angefordert
  3. Keine vorzeitige Rückgabe (no pre-emption): Die bereits belegten BM können den Prozessen nicht wieder entzogen werden, sondern müssen von den Prozessen selbst explizit zurückgegeben werden
- Hinreichende Bedingung für Deadlocks
  4. Gegenseitiges Warten (circular wait): Eine geschlossene Kette von zwei oder mehr wartenden Prozessen muss existieren, wobei ein Prozess BM vom nächsten haben will, die dieser belegt hat und die deshalb nicht mehr frei sind

Beschreibung der BM-Situation

- Die Betriebsmittelsituation definiert den aktuellen Anforderungs- und Belegungszustand und ist vollständig beschrieben durch $(P, BM, \vec{v}, B, A)$

  P = Menge der m Prozesse und BM-Menge der BM-Typen, $|BM| = n$

  Vorhandene Betriebsmittel: $\vec{v} = (v_1, v_2, \ldots, v_n)$

Belegungen B

$$B := \begin{pmatrix} b_{11} & \cdots & b_{1n} \\ \vdots & \ddots & \vdots \\ b_{m1} & \cdots & b_{mn} \end{pmatrix}$$

Anforderungen A

$$A := \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix}$$

Gesamt/Maximalanforderungen G

$$G := \begin{pmatrix} g_{11} & \cdots & g_{1n} \\ \vdots & \ddots & \vdots \\ g_{m1} & \cdots & g_{mn} \end{pmatrix}$$

Restanforderungen R

$$R := \begin{pmatrix} r_{11} & \cdots & r_{1n} \\ \vdots & \ddots & \vdots \\ r_{m1} & \cdots & r_{mn} \end{pmatrix} = G - B$$

- Freie Betriebsmittel: $\vec{f} = (f_1, f_2, \ldots, f_n) = \vec{v} - \sum_{i=1}^{m} \vec{b}_i$ ← ***i*-ter Zeilenvektor aus B**

Bedingungen

- Es kann nicht mehr belegt sein, als vorhanden ist:

$$\forall j \in \{1, \dots, n\}: \sum_{i=1}^{m} b_{ij} \leq v_j$$

- Kein Prozess kann mehr anfordern als insgesamt vorhanden (wäre nie erfüllbar):

$$\forall i \in \{1, \dots, m\}, \forall j \in \{1, \dots, n\}: \quad g_{ij} \leq v_j$$
$$a_{ij} + b_{ij} \leq v_j$$

- Ein Prozess, der eine derzeit nicht erfüllbare Anforderung gestellt hat, wird bis zur Erfüllbarkeit blockiert

Schreibweise pro Prozess

- Nutzung von Zeilenvektoren

  Anforderung von Prozess i $\vec{a}_i = (a_{i1}, a_{i2}, \ldots, a_{in})$

  Belegung von Prozess i $\vec{b}_i = (b_{i1}, b_{i2}, \ldots, b_{in})$

  Gesamtanforderung von Prozess i $\vec{g}_i = (g_{i1}, g_{i2}, \ldots, g_{in})$

  Restanforderung von Prozess i $\vec{r}_i = (r_{i1}, r_{i2}, \ldots, r_{in})$

- Vergleichsoperatoren

  Anforderung x erfüllbar: $\vec{x} \leq \vec{y} \Leftrightarrow \forall k \in \{1, \ldots, n\}: x_k \leq y_k$

  Anforderung x nicht erfüllbar: $\vec{x} \not\leq \vec{y} \Leftrightarrow \exists k \in \{1, \ldots, n\}: x_k > y_k$

Definitionen

- Ein Prozess Pi heißt blockiert, wenn $\vec{a}_i \not\leq \vec{f}$, d.h. wenn seine aktuelle Anforderung derzeit nicht erfüllbar ist
- Eine Prozessmenge P = {P1, P2, ... , Pm} heißt verklemmt, genau dann wenn:

$$\exists I \subseteq \{1, \dots, m\}: \forall i \in I: \vec{a}_i \not\leq \vec{v} - \sum_{c \in I} \vec{b}_c$$

d.h. es gibt eine Teilmenge von Prozessen, deren Anforderungen nicht erfüllbar sind, auch nicht durch den Vorrat außerhalb der Prozesse in dieser Teilmenge

- Selbst wenn wir alle anderen Prozesse terminieren und Ressourcen freigeben, wären die Anforderungen der Prozesse in der Teilmenge nicht erfüllbar:

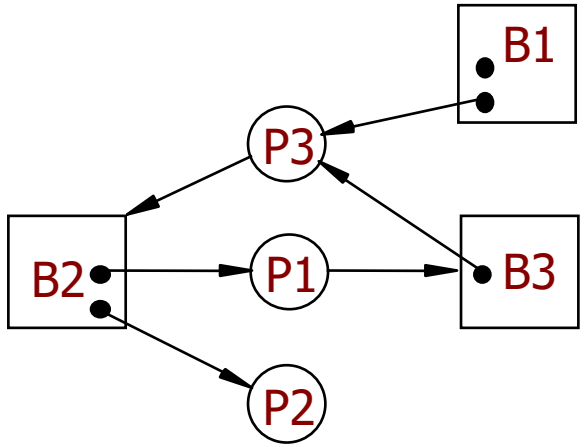
$$\exists I \subseteq \{1, \dots, m\}: \forall i \in I: \vec{a}_i \not\leq \vec{f} + \sum_{c \notin I} \vec{b}_c$$

Betriebsmittelgraph

- Formale Darstellung von Anforderungs- und Belegungssituationen
- Definition Betriebsmittelgraph
  - Sei P die Menge der Prozesse, BM die Menge der Betriebsmitteltypen
  - Ein gerichteter Graph (V, E) mit $V = P \cup BM$ und der folgenden Pfeilsemantik heißt Betriebsmittelgraph:

$(p,b) \in E \Leftrightarrow$ Prozess p fordert eine Einheit von BM-Typ b

$(b,p) \in E \Leftrightarrow$ Prozess p besitzt eine Einheit von BM-Typ b

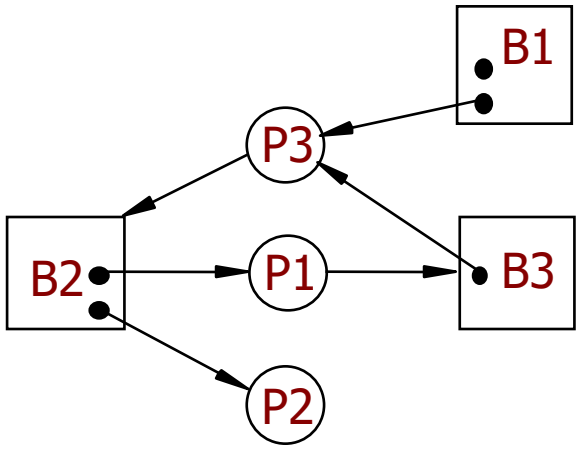


Betriebsmittelgraph: Eigenschaften und Operationen

- Der BM-Graph ist bezüglich der Knotenmengen P (Kreise) und BM (Rechtecke) bipartit
  - es gibt nur Kanten von P nach B oder umgekehrt
- Menge von Punkten im Knoten = Anzahl der insgesamt verfügbaren Einheiten eines Betriebsmitteltyps
  - bestimmt den maximalen Ausgangsgrad des BM-Knotens
- Ein Zyklus im BM-Graph weist auf eine potentielle Verklemmungssituation hin
  - nur eine notwendige, aber keine hinreichende Bedingung für die Existenz einer Verklemmung (siehe Beispiel)
- Jede Operation (Anfordern, Belegen, Freigeben) bedeutet eine Graphtransformation (Hinzufügen bzw. Entfernen von Kanten)
- Beendigung eines Prozesses = Freigabe aller belegter BM

Betriebsmittelgraph: Reduktionen

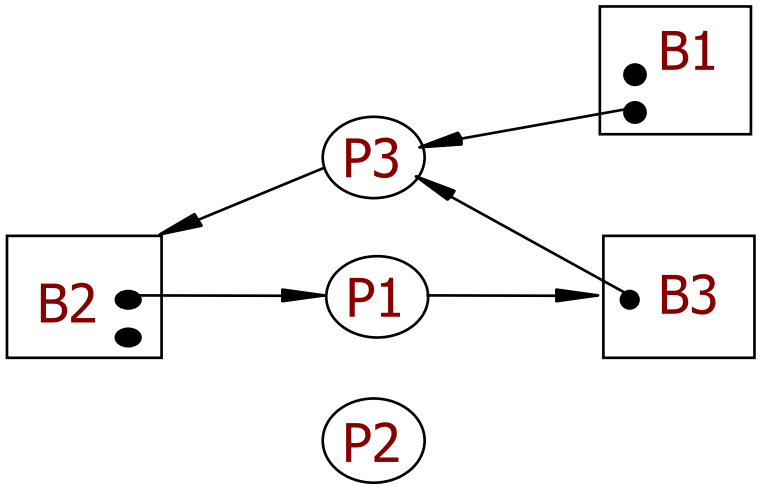
- BM-Graph reduzierbar ⇔ es gibt einen Prozess, dessen Anforderungen sofort erfüllbar sind und alle seine Kanten entfernt werden können
- BM-Graph vollständig reduzierbar ⇔ es gibt eine Folge von Reduktionen, so dass am Ende alle Kanten entfernt sind
- Verklemmungstheorem für BM-Graphen: Aktuelle BM-Situation verklemmt ⇔ dazugehöriger BM-Graph ist nicht vollständig reduzierbar
- Beispiel für eine Reduktion (Ausgangsgraph)



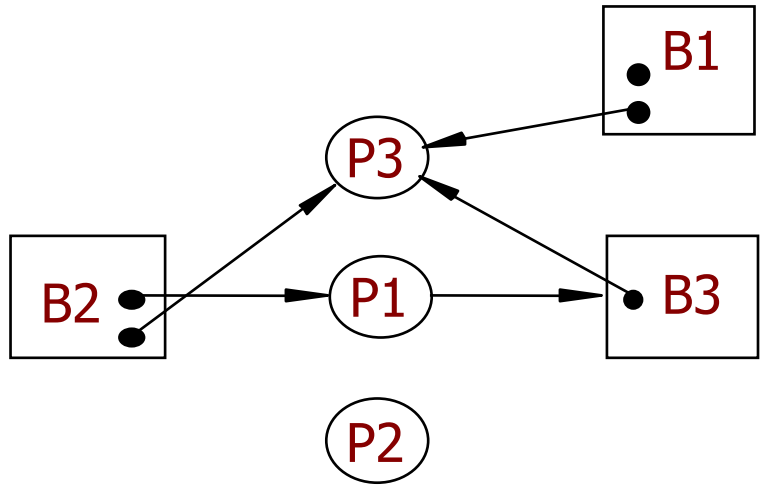
Ausgangssituation a)

Beispiel (Fortsetzung)

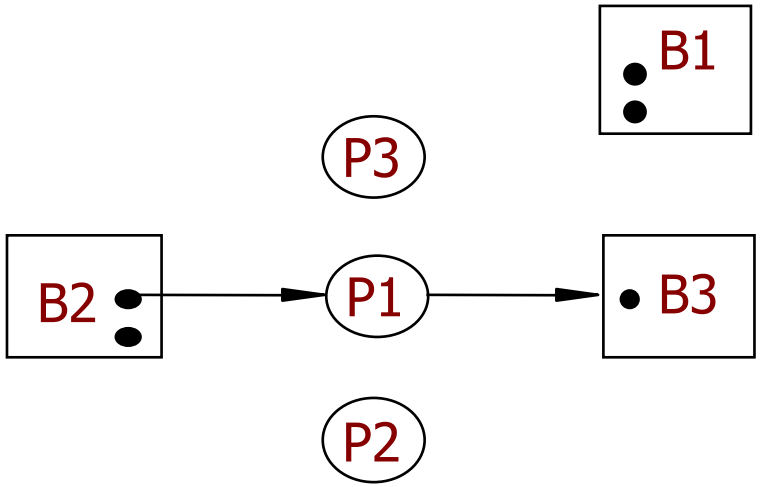
b)



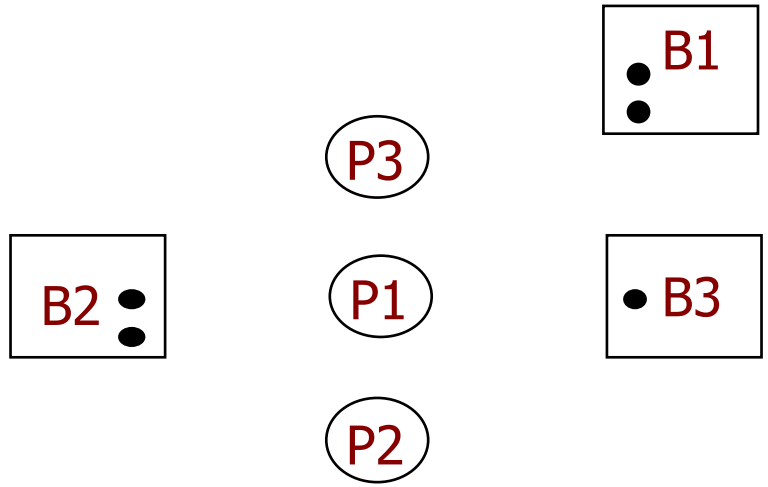
c)



d)



e)



5.4 Verklemmungsvermeidung (Theoretischer Ansatz)

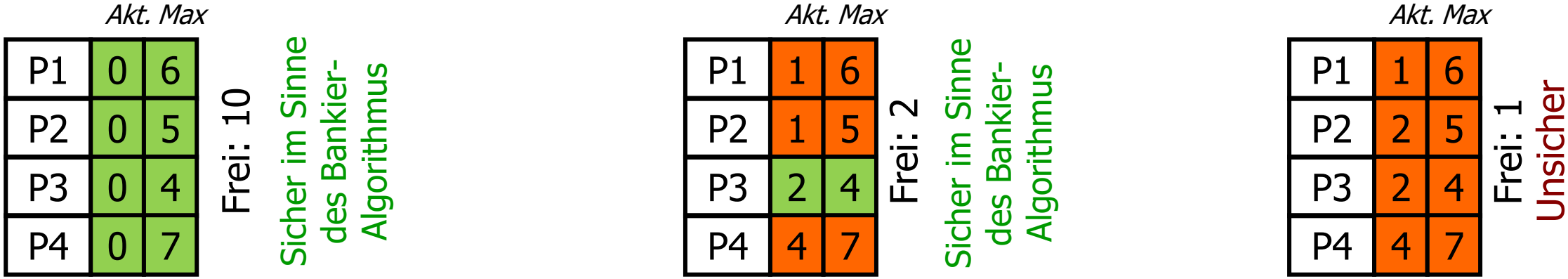
- Wie kann man eine Verklemmung vermeiden?
  - Aktuelle Situation kennen $\Rightarrow$ BM-Graph
  - Restanforderungen der Prozesse müssen bekannt sein
  - Eine Belegungsstrategie anwenden, so dass kein Wartezyklus entsteht

- Definition unsichere Betriebsmittelsituation
  - Teilmenge von Prozessen existiert, deren Restanforderungen nicht alle erfüllbar sind
  - Aktuelle Anforderungen mögen zwar noch erfüllt werden, aber bei ungünstiger Reihenfolge der Anforderungen und Freigaben kann es zur Verklemmung kommen

- Ziel: finde eine Beendigungsreihenfolge der Prozesse derart, dass jede Anforderung durch das erfüllt werden kann, was von früher beendeten Prozessen freigegeben wird

Ermitteln der Prozess-Sortierung: der Bankier-Algorithmus

- „Banker's Algorithm" von E. Dijkstra (1965!)
- Banker in Kleinstadt gewährt Kunden Kredit bis zu ihrem jeweiligen Kreditrahmen
  - Kredite werden nicht auf einmal angefordert ⇒ Bank hält nur einen Teil des notwendigen Geldes (hier 4 Kunden / 10 Einheiten)
  - Nur genug Geld zurückhalten, um einen Kunden voll zu bedienen
  - Dieser tätigt dann seine Geschäfte und zahlt den Kredit voll zurück
  - Mit der Rückzahlung kann dann der nächste Kunde bedient werden usw.



Bankier-Algorithmus (2)

- Bankier-Algorithmus zur Ermittlung des BM-Zustands:
  - Wähle eine Zeile aus der Matrix R so aus, deren
    - Prozess Pi noch nicht terminiert ist
    - Restforderungen kleiner/gleich der freien Ressourcen in f sind
  - Nimm an, der zugehörige Prozess wird bedient, terminiert dann und gibt schließlich seine gesamten belegten Ressourcen frei
    - aktualisiere den Vektor f (addiere bi)
  - Wiederhole die Schritte 1 und 2, bis:
    - alle Prozesse terminiert sind (sicherer Zustand)
    - oder kein Prozess mehr bedient werden kann (unsicherer Zustand)
- Anmerkungen
  - Sind mehrere Prozesse für die Auswahl geeignet, spielt die Reihenfolge keine Rolle, da die Anzahl freier Ressourcen schlimmstenfalls gleich bleibt $\Rightarrow$ beliebige Wahl

Beispiel

- Gegeben sei ein System mit vier Prozessen und zwei BM-Typen. Die aktuelle Situation sei wie folgt gegeben:

Belegungen: Gesamtanforderungen: Freie Betriebsmittel:

$$B = \begin{pmatrix} 0 & 4 \\ 1 & 0 \\ 3 & 0 \\ 5 & 4 \end{pmatrix} \qquad G = \begin{pmatrix} 4 & 6 \\ 2 & 4 \\ 3 & 1 \\ 10 & 6 \end{pmatrix} \qquad f = (3 \;\; 2)$$

Daraus ergeben sich die Restanforderungen R und die vorhandenen BM:

$$R = \begin{pmatrix} 4 & 2 \\ 1 & 4 \\ 0 & 1 \\ 5 & 2 \end{pmatrix} \qquad v = (12 \;\; 10)$$

Beispiel Banker-Algorithmus (4 Prozesse, 2 BM-Typen)

Vorhandene BM **v**

BM1	*BM2*
12	10

Belegungen **B**

	BM1	*BM2*
P1	0	4
P2	1	0
P3	3	0
P4	5	4

Gesamtforderung **G**

	BM1	*BM2*
P1	4	6
P2	2	4
P3	3	1
P4	10	6

Restforderung **R**

	BM1	*BM2*
P1	4	2
P2	1	4
P3	0	1
P4	5	2

Freie BM **f**

BM1	*BM2*
3	2

- Ist das System im sicheren Zustand?
  - P3 kann beendet werden: R3=(0 1) ≤ f = (3 2)
    f = f + B3 = (3 2)+(3 0) = (6 2)
  - P1 (oder P4) kann beendet werden: R1=(4 2) ≤ f = (6 2)
    f = f + B1 = (6 2)+(0 4) = (6 6)
  - P2 (oder P4) kann beendet werden: R2=(1 4) ≤ f = (6 6)
    f = f + B2 = (6 6)+(1 0) = (7 6)
  - P4 kann beendet werden: R4=(5 2) ≤ f = (7 6), f = (12 10)=v
- Sichere Reihenfolge (z. B.): P3 P1 P2 P4

Verklemmungsvermeidung: Banker-Algorithmus

```
enum state {safe, unsafe, undefined}
state deadlock_avoidance(set_of_processes P, vector v, matrix r, matrix b,
set_of_processes DP){
  vector f;
    state answer = undefined;
    DP = P;
          f: = v − ∑_{i=1}^{m} b_i ;
    while (answer == undefined) {
           if (∃ Pi ∈ DP: ri ≤ f){
                  DP = DP – {Pi};
                  f = f + bi;
           } else if (DP == ∅) answer= safe;
           else answer = unsafe;
    }
    return answer;
}
```

Praxis: Beseitigung von Verklemmungen

- Realistischerweise: Restanforderungen nicht bekannt, Deadlocks treten auf
- Reaktion: Erkennung Deadlock (meist durch Stillstand zahlreicher Prozesse) und Ausführung einer Aktion
  - Prozesse abbrechen: ein unbeteiligter Prozess, der das benötigte BM hat, wird anhand einer Strategie selektiert und abgebrochen
    - Hoffnung, alles arrangiert sich selbst wieder
    - Auswahlstrategien basierend auf Größe der Anforderung, Umfang belegter BM, Dringlichkeit, Restbedienzeit, Aufwand des Abbruchs
  - Prozesse zurücksetzen: Ein oder mehrere Prozesse werden auf den letzten gültigen Checkpoint zurückgesetzt und neu gestartet
  - BM entziehen: einem Prozess werden von anderen Prozessen benötigte BM entzogen $\Rightarrow$ nur für bestimmte BM-Arten
- Allerdings: mögliche Datenverluste und -inkonsistenzen