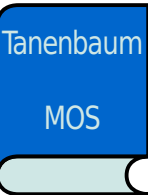


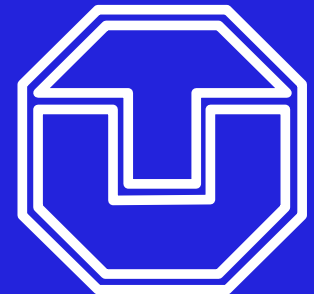
Verklemmungen

(grob nach Tanenbaum „Moderne Betriebssysteme“)



Betriebssysteme

Hermann Härtig
TU Dresden



Wegweiser

Begriff, Beispiele und
Modellierung

Maßnahmen

- Vermeiden per Konstruktion
- Entdecken und Beseitigen

Beispiel Verklemmungen

Banküberweisung

```
void transfer (Account from, Account to, unsigned Betrag) {  
  
    from.lock();  
    to.lock();  
  
    from.Stand -= Betrag;  
    to.Stand += Betrag;  
  
    to.unlock();  
    from.unlock();  
}
```

```
transfer (A, B, 100) || transfer (B, A, 50)
```

Definition Verklemmung (Deadlock)

Eine Menge von Prozessen heißt **verklemmt** (die Prozesse befinden sich in einem **Deadlock**), wenn jeder Prozess dieser Menge auf ein **Ereignis** wartet, das nur von einem anderen Prozess dieser Menge ausgelöst werden kann.

- Ereignis:
 - Botschaften
 - Semaphor wird frei;

- Ereignis-Ursache in der Regel: Freigeben von Betriebsmitteln

Betriebsmittel

CPU, Speicher, E/A-Geräte, ... Datenbank-Eintrag, ...

- 1 Benutzer pro BM zu einem Zeitpunkt
- Verdrängbar (preemptible):
 - ♦ CPU / Hauptspeicher können ohne Probleme entzogen werden
 - ♦ Nicht verdrängbar (non-preemptible):
 - ♦ Drucker/Datenbank-Eintrag kann erst nach Ende eines Auftrages/Transaktion entzogen werden

Sequenz

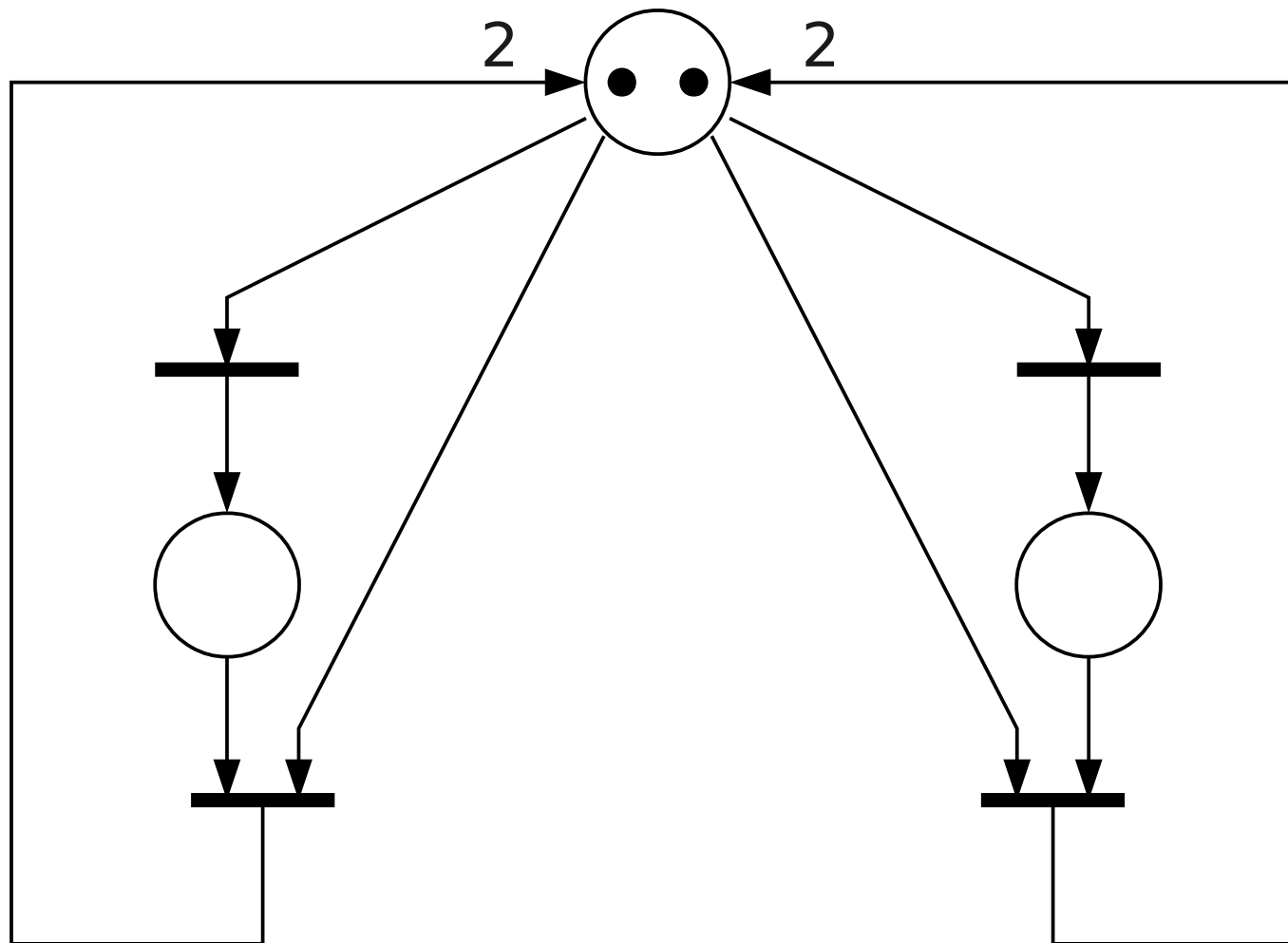
```
request ...  
use ...  
release ...
```

Zweck von Modellierung

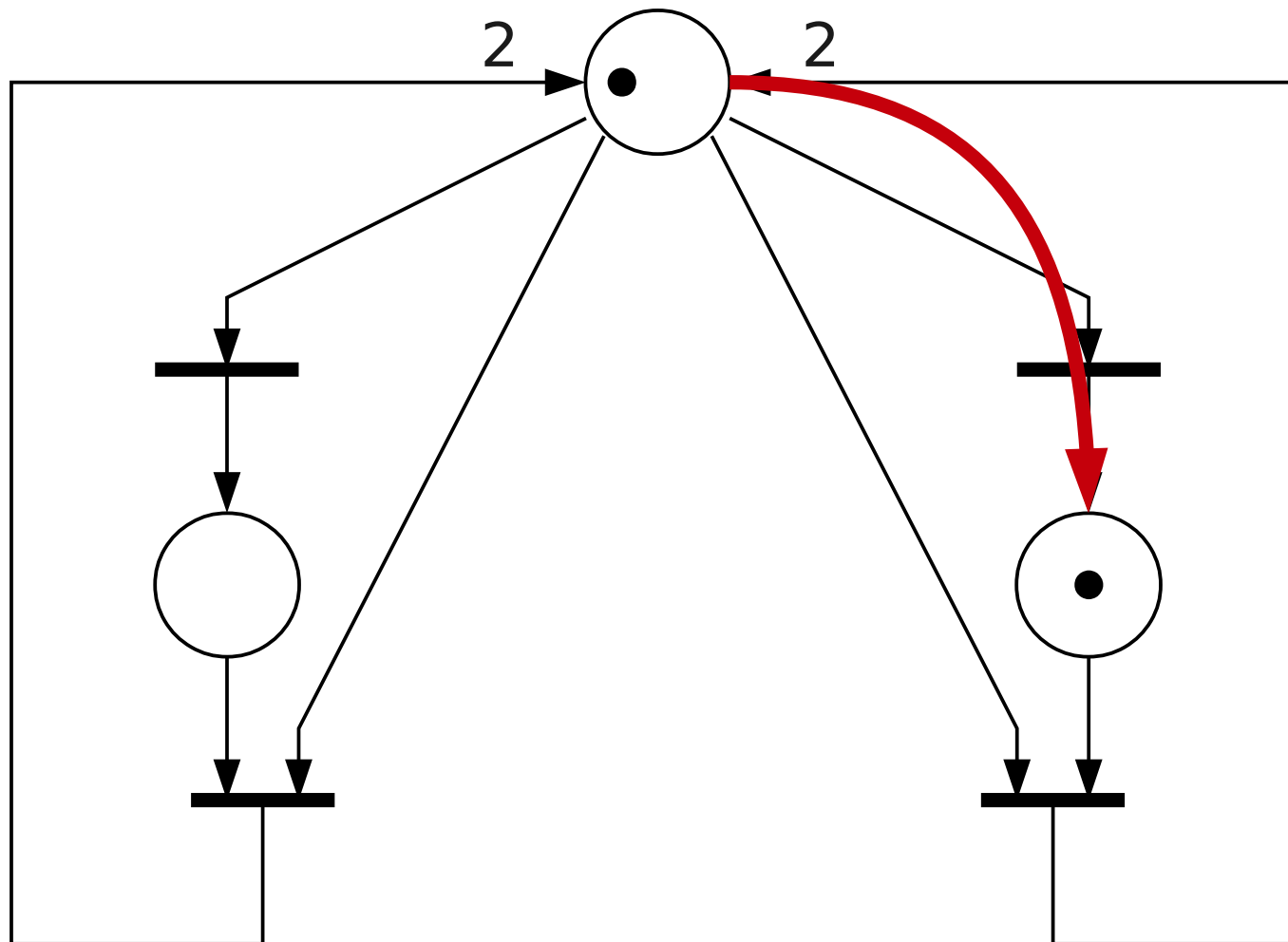
- Abstrakteres Verstehen des Sachverhalts
- Feststellen, ob es sich bei einer “anomalen” Situation um eine Verklemmung handelt
- Analyse, ob für ein gegebenes Verfahren oder eine gegebene Situation eine Anforderungsfolge zur Verklemmung führen kann

**Hier motivierende Rede zu
Modellierung halten!**

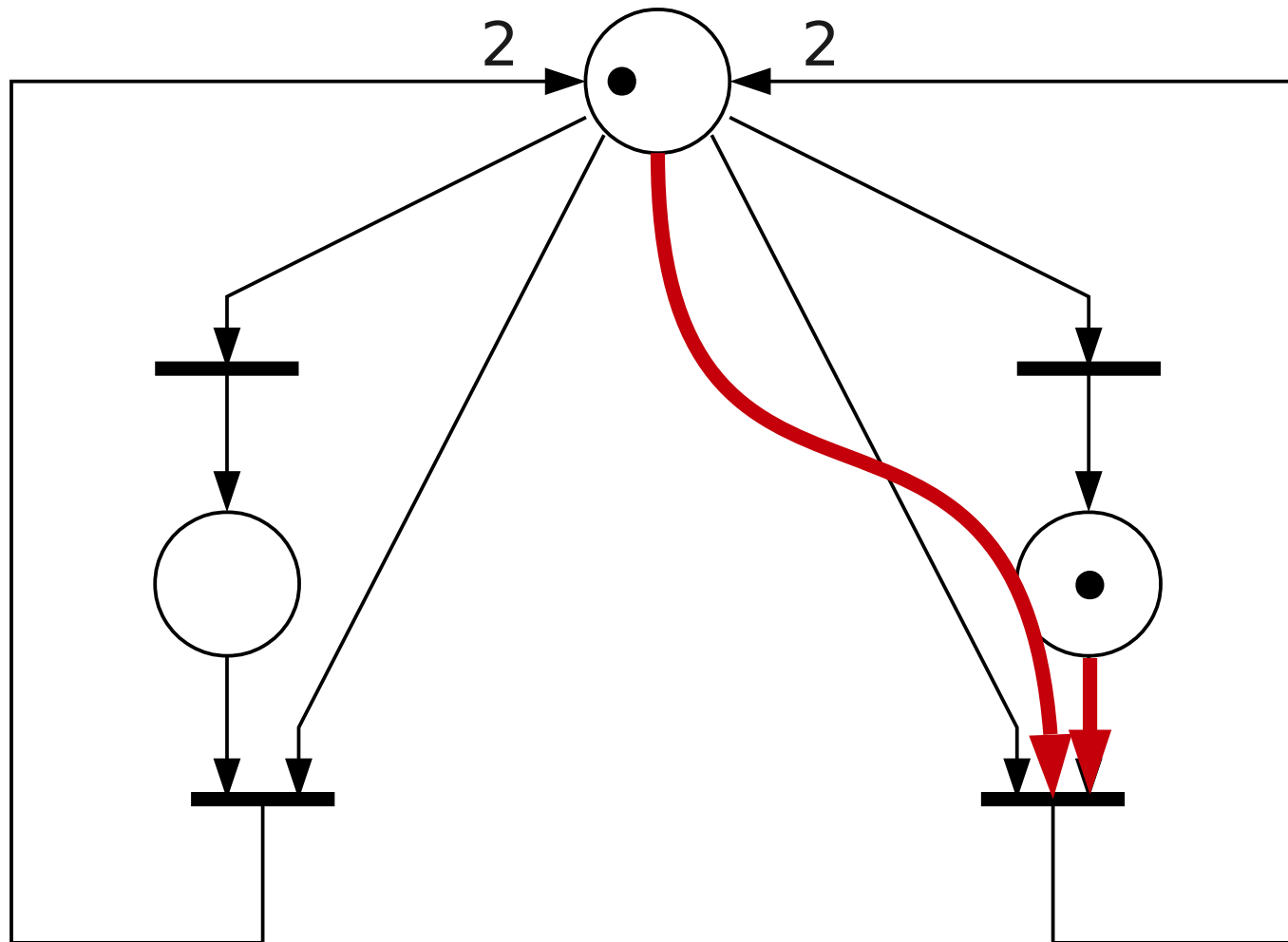
Modellierung mittels PETRI-Netzen



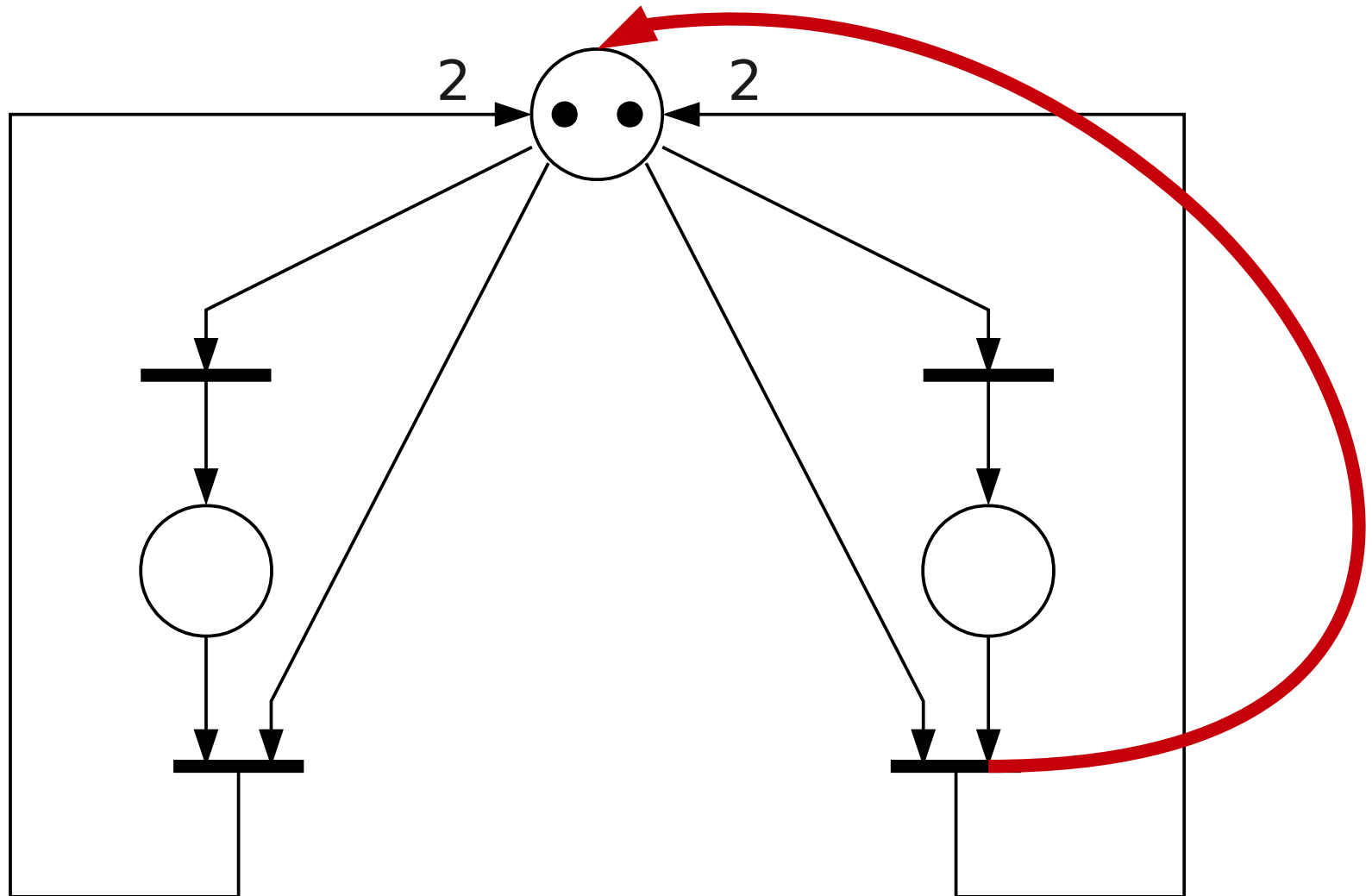
Modellierung mittels PETRI-Netzen



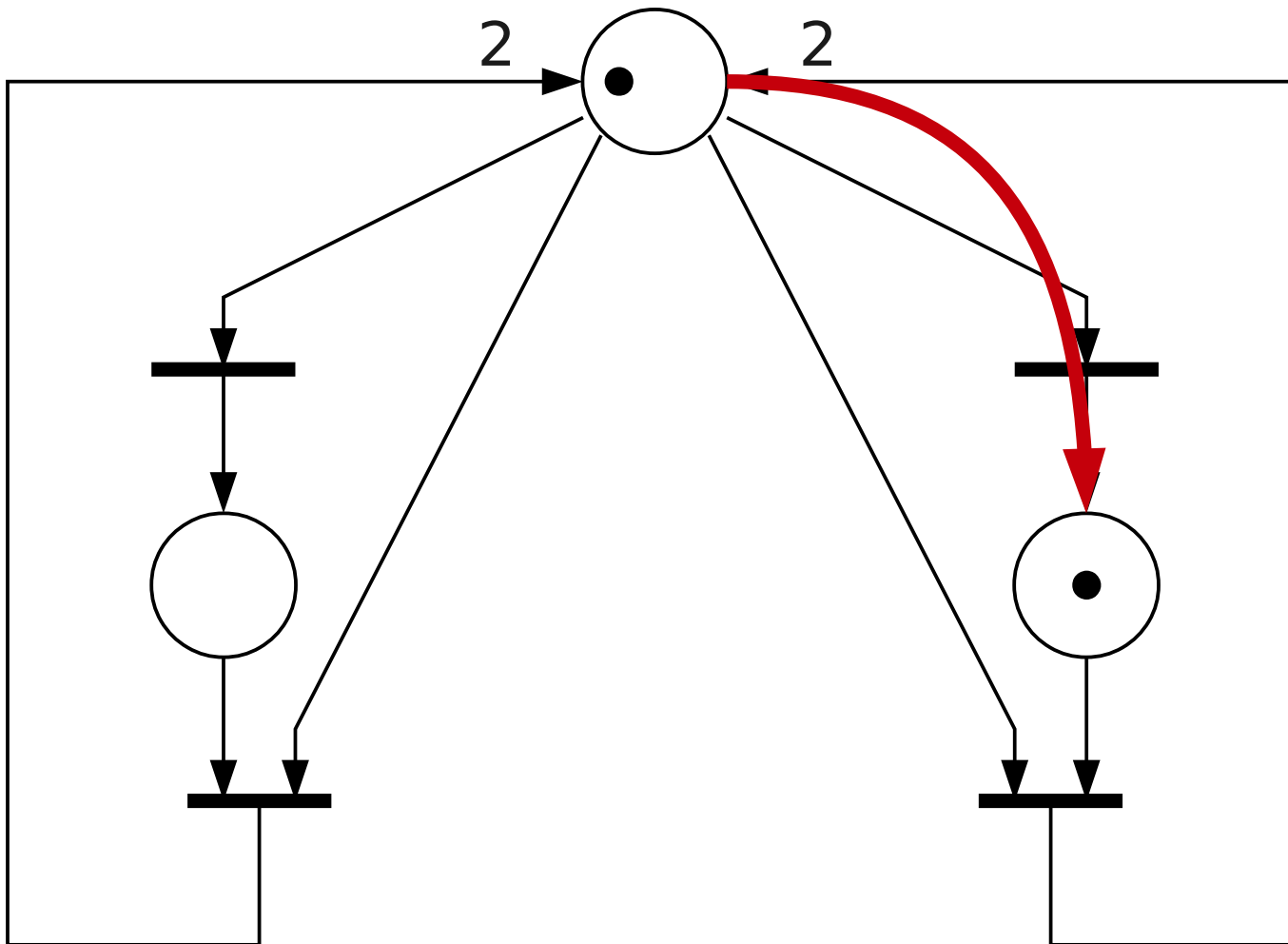
Modellierung mittels PETRI-Netzen



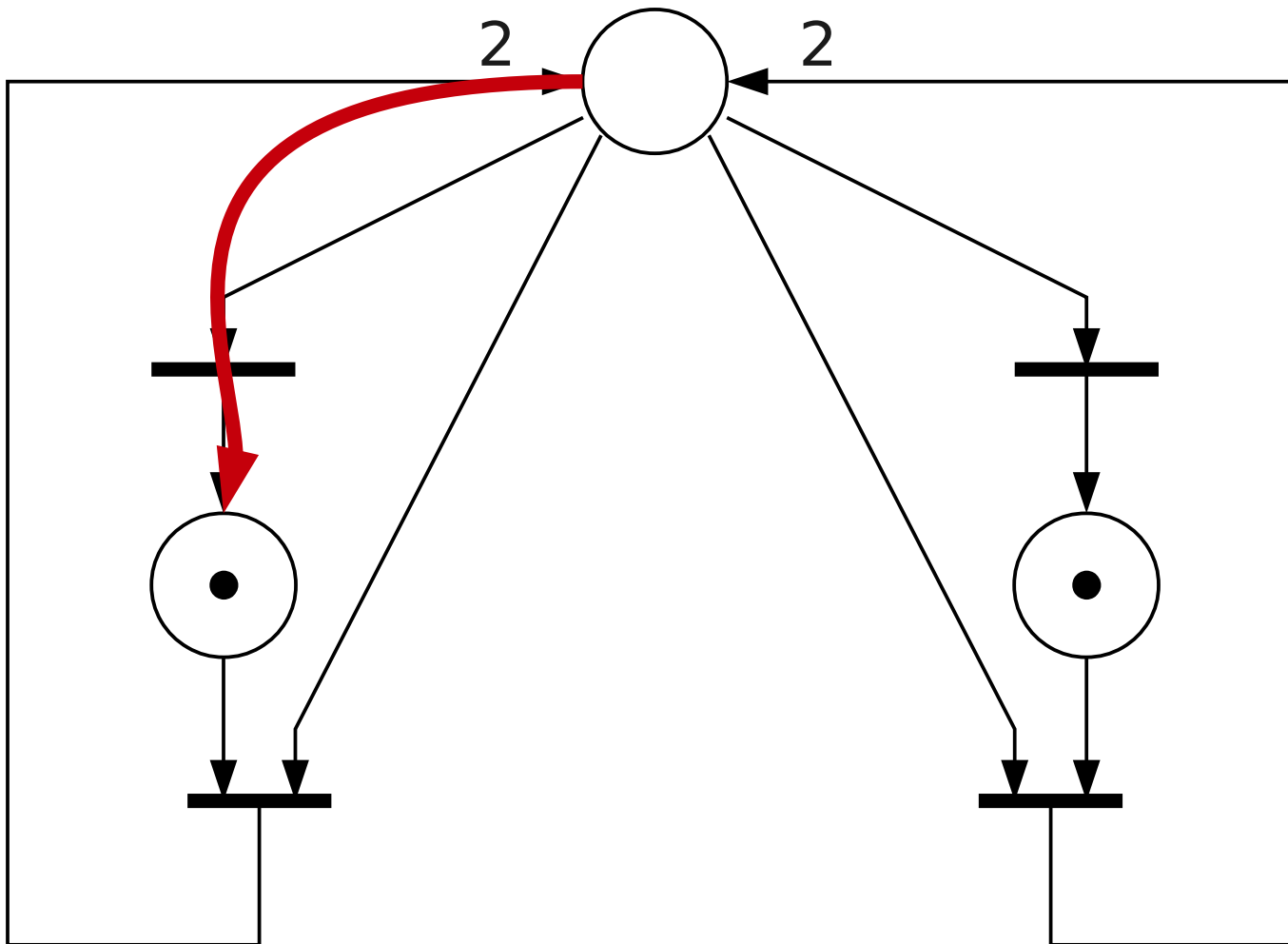
Modellierung mittels PETRI-Netzen



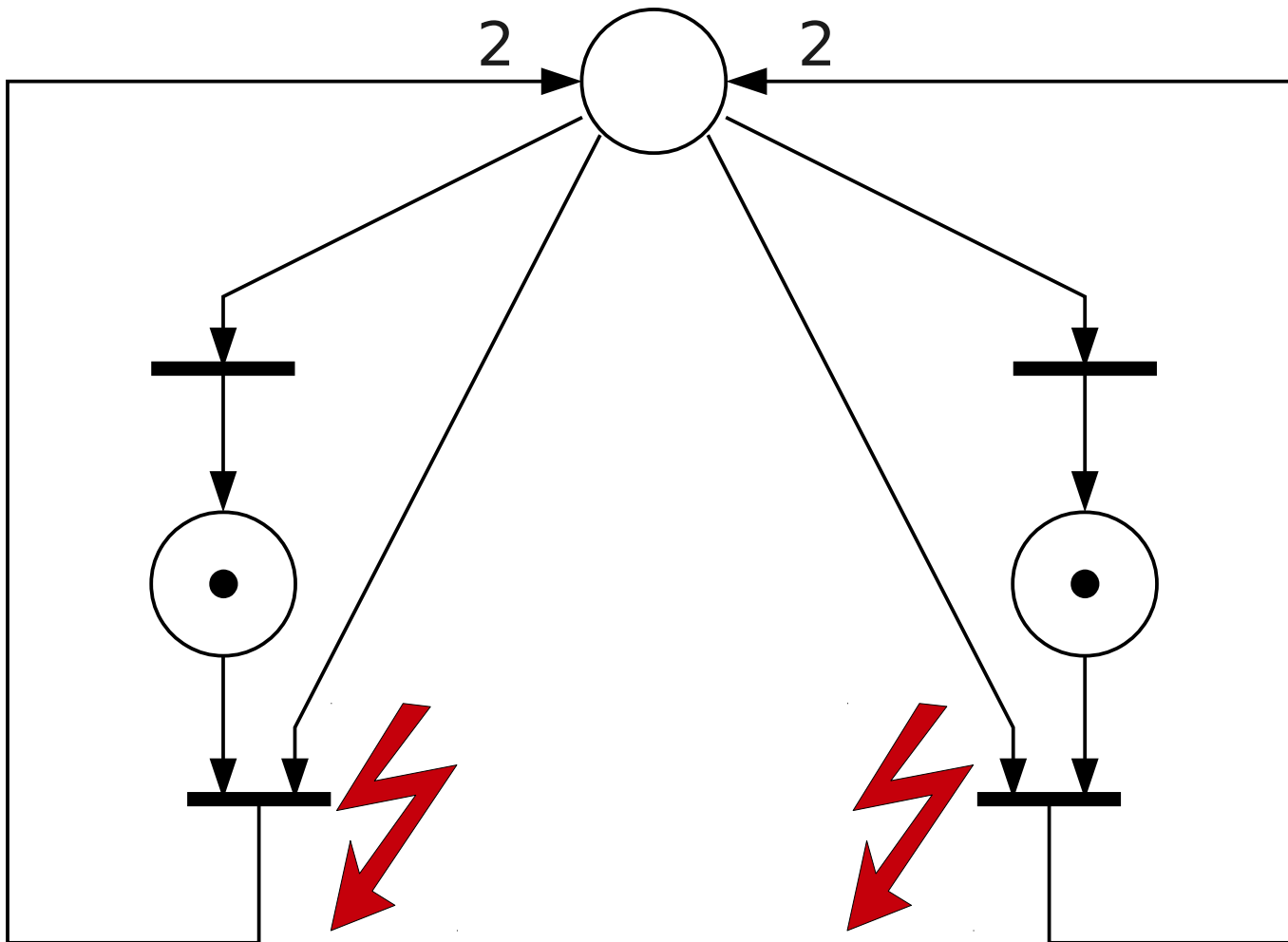
2. Versuch



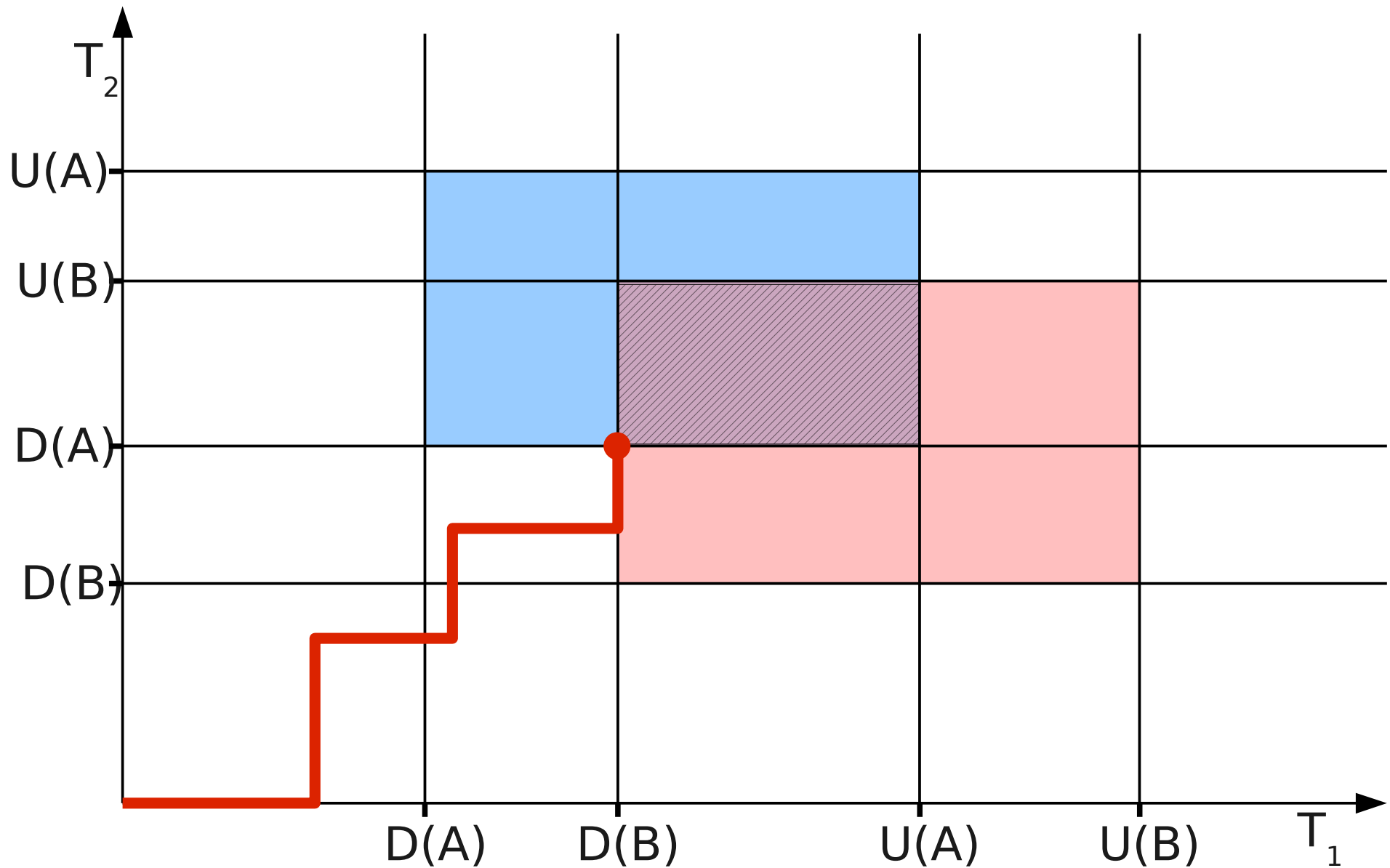
2. Versuch



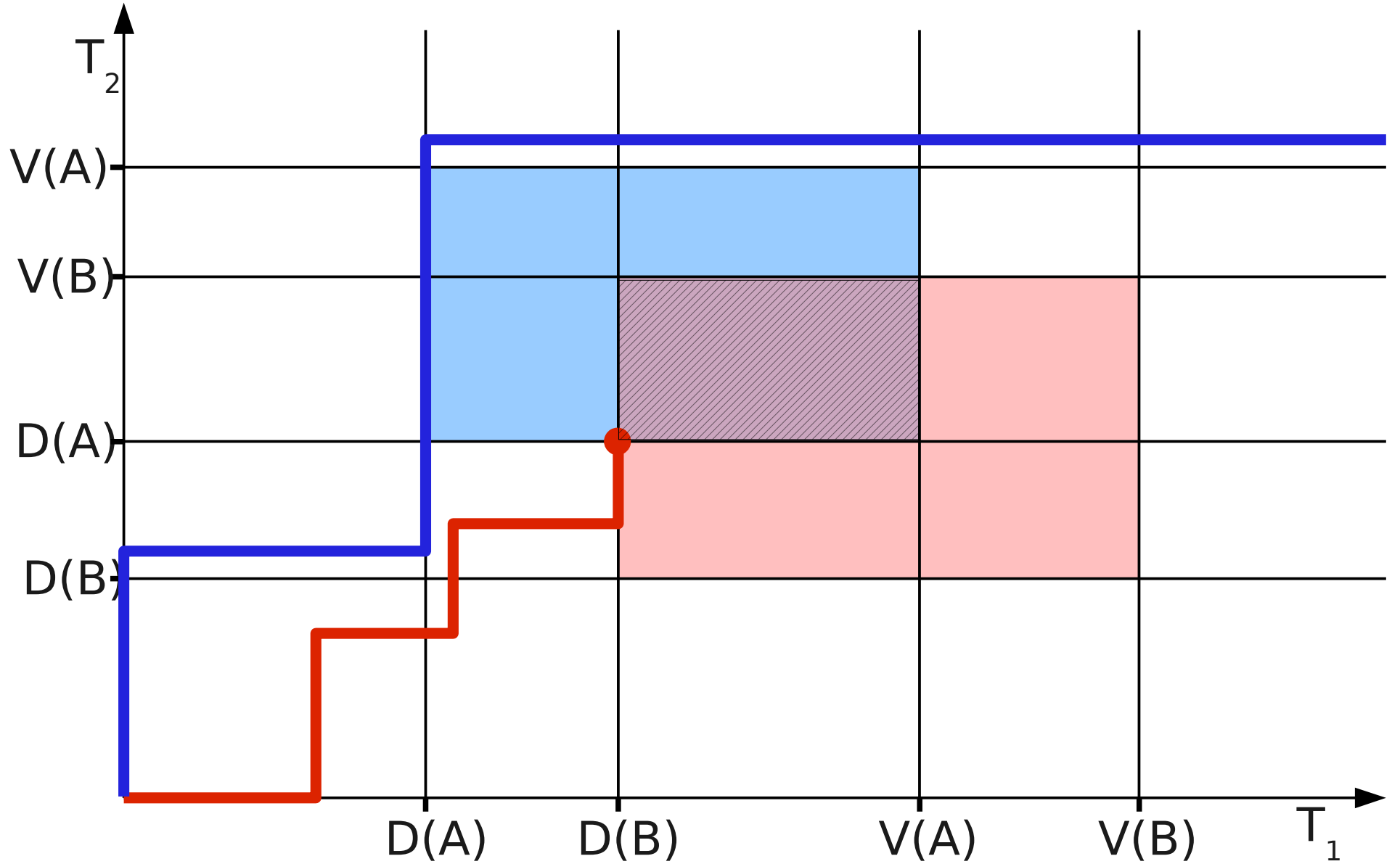
2. Versuch



Modellierung mittels Prozessfortschritts-Diagrammen



Modellierung mittels Prozessfortschritts-Diagrammen



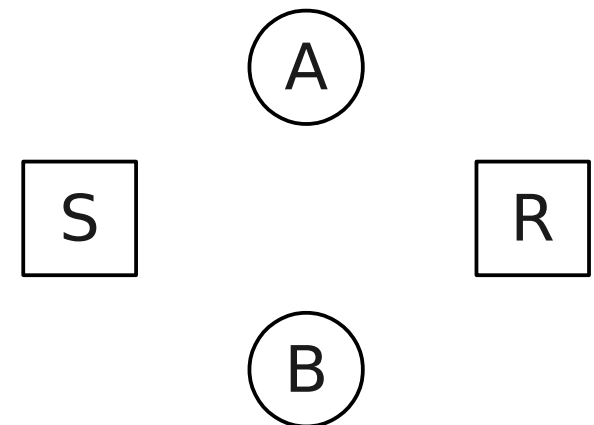
Modellierung mittels BM-Zuteilungsgraph

Betriebsmittel-Zuteilungsgraph (HOLT 1972 et al.)

Zyklus im Graphen → zyklische Wartesituation

Beispiel

- Prozess A fordert an BM S
- Prozess B fordert an BM R
- Prozess A fordert an BM R
- Prozess B fordert an BM S



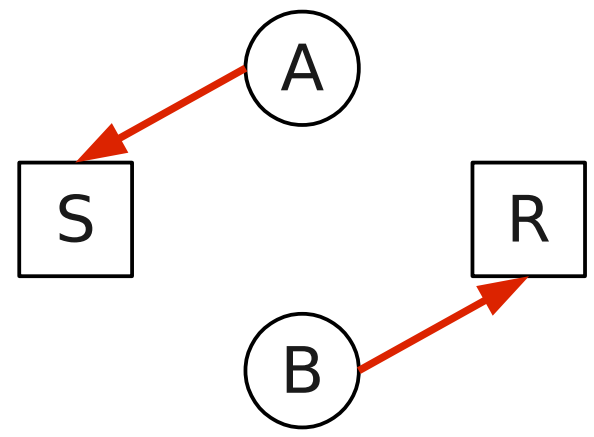
Modellierung mittels BM-Zuteilungsgraph

Betriebsmittel-Zuteilungsgraph (HOLT 1972 et al.)

Zyklus im Graphen → zyklische Wartesituation

Beispiel

- Prozess A **besitzt** BM S
- Prozess B **besitzt** BM R
- • Prozess A **fordert an** BM R
- Prozess B **fordert an** BM S



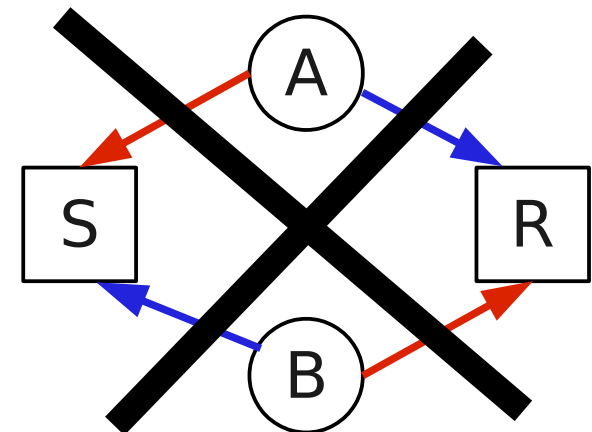
Modellierung mittels BM-Zuteilungsgraph

Betriebsmittel-Zuteilungsgraph (HOLT 1972 et al.)

Zyklus im Graphen → zyklische Wartesituation

Beispiel

- Prozess A **besitzt** BM S
- Prozess B **besitzt** BM R
- Prozess A fordert an BM R
- Prozess B fordert an BM S



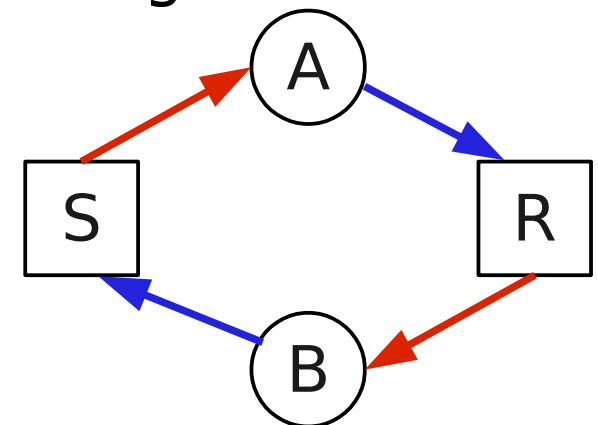
Modellierung mittels BM-Zuteilungsgraph

Betriebsmittel-Zuteilungsgraph (HOLT 1972 et al.)

Zyklus im Graphen → zyklische Wartesituation

Beispiel

- Prozess A **besitzt** BM → S wartet für Freigabe auf A
- Prozess B **besitzt** BM R → R wartet für Freigabe auf B
- Prozess A fordert an BM R → A wartet auf Freigabe von R
- Prozess B fordert an BM S → B wartet auf Freigabe von S



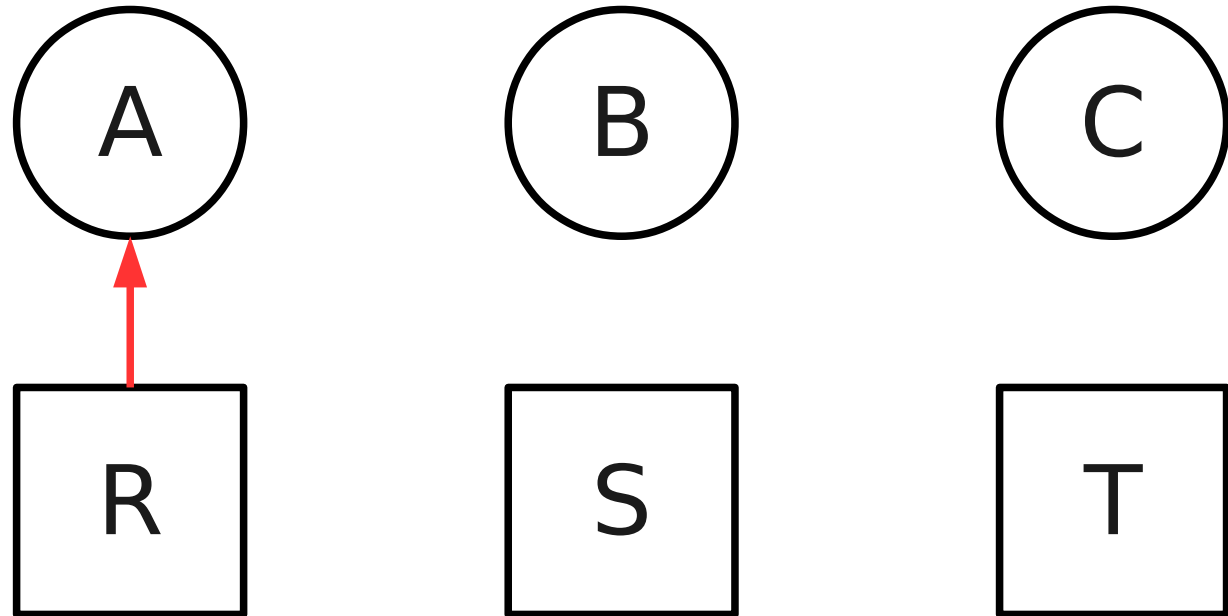
Beispiel (1)

A :
→ request (R) ;
request (S) ;
release (R) ;
release (S) ;

B :
→ request (S) ;
request (T) ;
release (S) ;
release (T) ;

C :
→ request (T) ;
request (R) ;
release (T) ;
release (R) ;

1. A fordert R



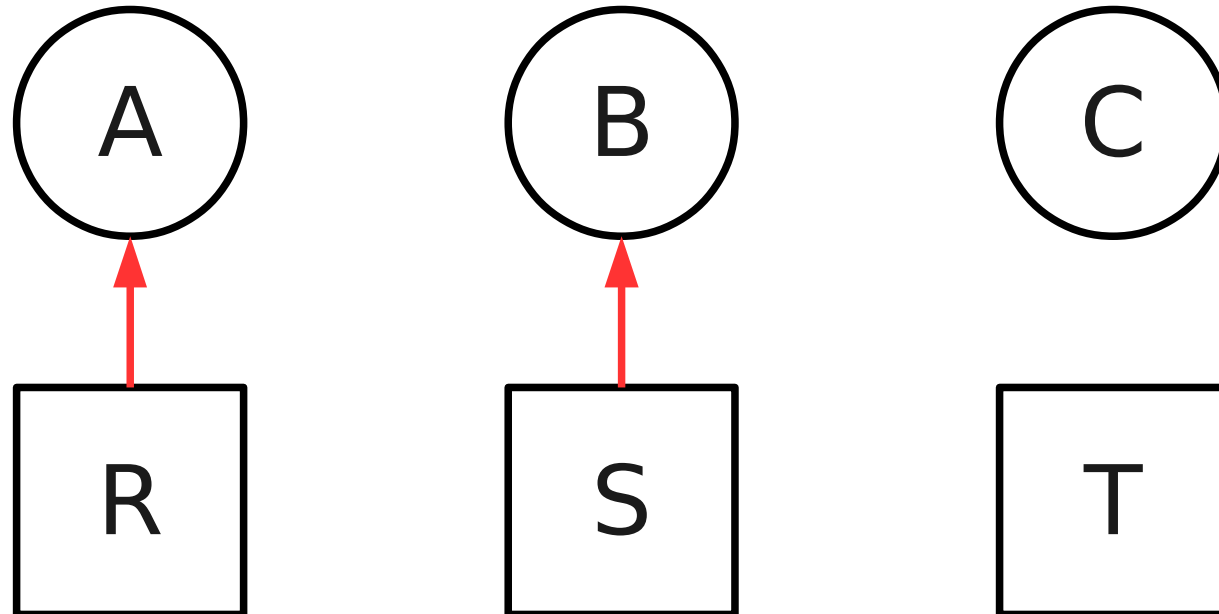
Beispiel (1)

A :
→ request (R) ;
request (S) ;
release (R) ;
release (S) ;

B :
→ request (S) ;
request (T) ;
release (S) ;
release (T) ;

C :
→ request (T) ;
request (R) ;
release (T) ;
release (R) ;

- 1. A fordert R
- 2. B fordert S



Beispiel (1)

A :

➔ request (R) ;
request (S) ;
release (R) ;
release (S) ;

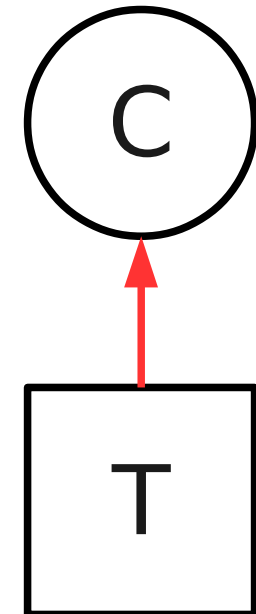
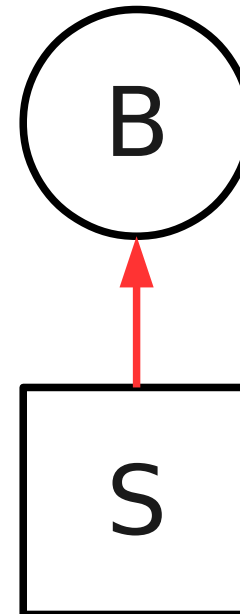
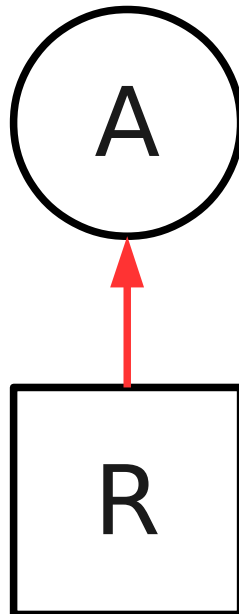
B :

➔ request (S) ;
request (T) ;
release (S) ;
release (T) ;

C :

➔ request (T) ;
request (R) ;
release (T) ;
release (R) ;

1. A fordert R
2. B fordert S
3. C fordert T



Beispiel (1)

A :

```
request (R) ;  
➔ request (S) ;  
release (R) ;  
release (S) ;
```

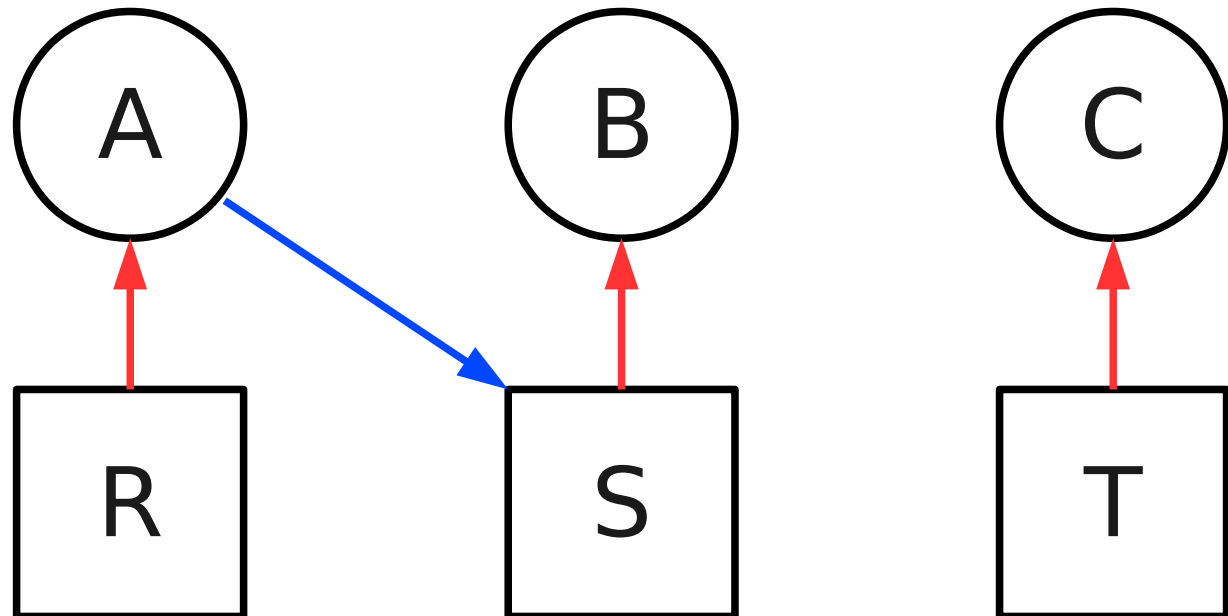
B :

```
➔ request (S) ;  
request (T) ;  
release (S) ;  
release (T) ;
```

C :

```
➔ request (T) ;  
request (R) ;  
release (T) ;  
release (R) ;
```

1. A fordert R
2. B fordert S
3. C fordert T
4. A fordert S



Beispiel (1)

A :

```
request (R) ;  
➔ request (S) ;  
release (R) ;  
release (S) ;
```

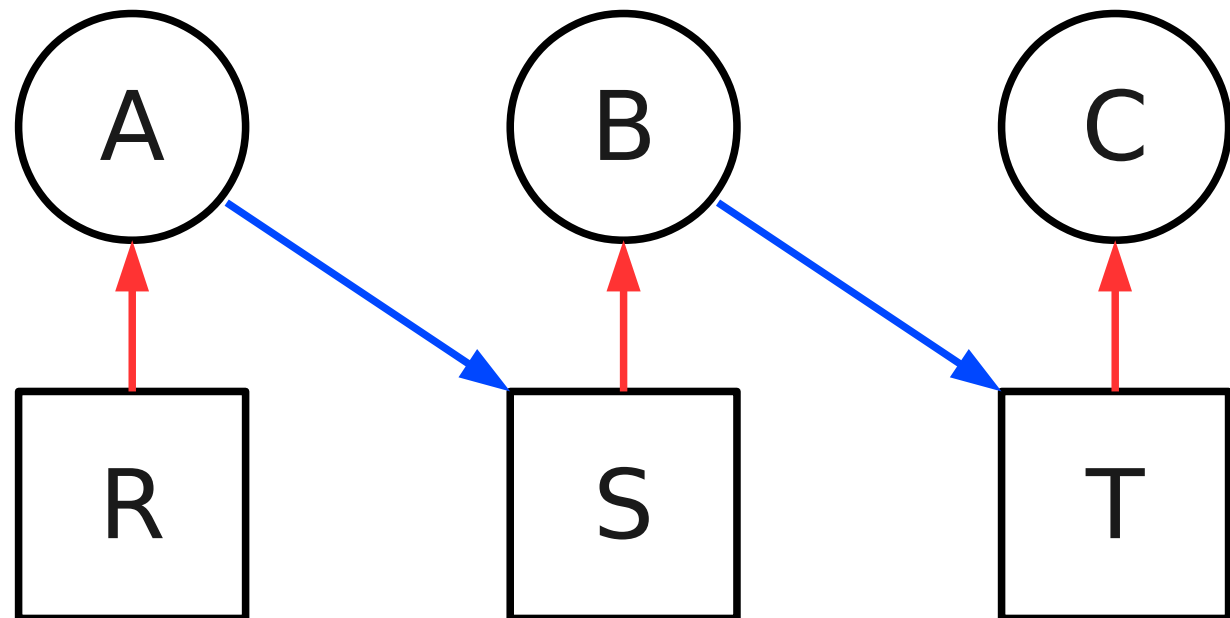
B :

```
request (S) ;  
➔ request (T) ;  
release (S) ;  
release (T) ;
```

C :

```
➔ request (T) ;  
request (R) ;  
release (T) ;  
release (R) ;
```

1. A fordert R
2. B fordert S
3. C fordert T
4. A fordert S
5. B fordert T



Beispiel (1)

A :

```
request (R) ;  
➔ request (S) ;  
release (R) ;  
release (S) ;
```

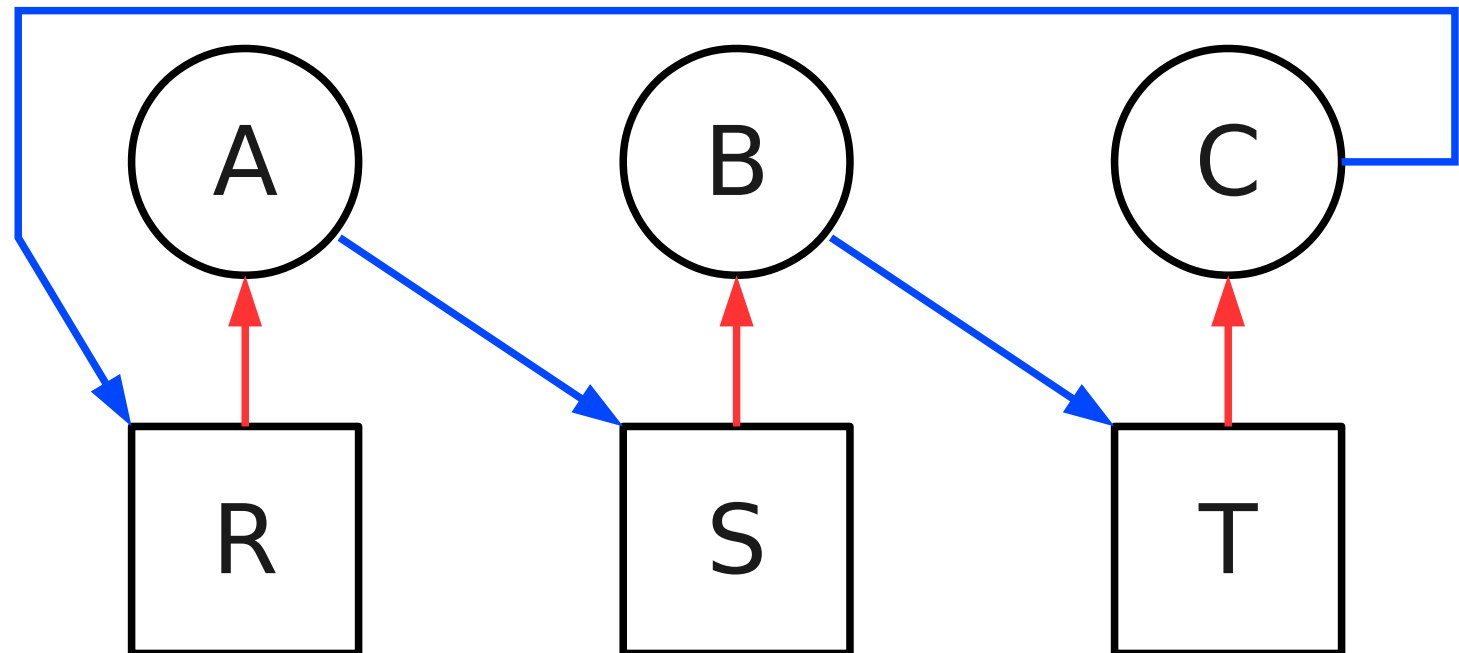
B :

```
request (S) ;  
➔ request (T) ;  
release (S) ;  
release (T) ;
```

C :

```
request (T) ;  
➔ request (R) ;  
release (T) ;  
release (R) ;
```

1. A fordert R
2. B fordert S
3. C fordert T
4. A fordert S
5. B fordert T
6. C fordert R



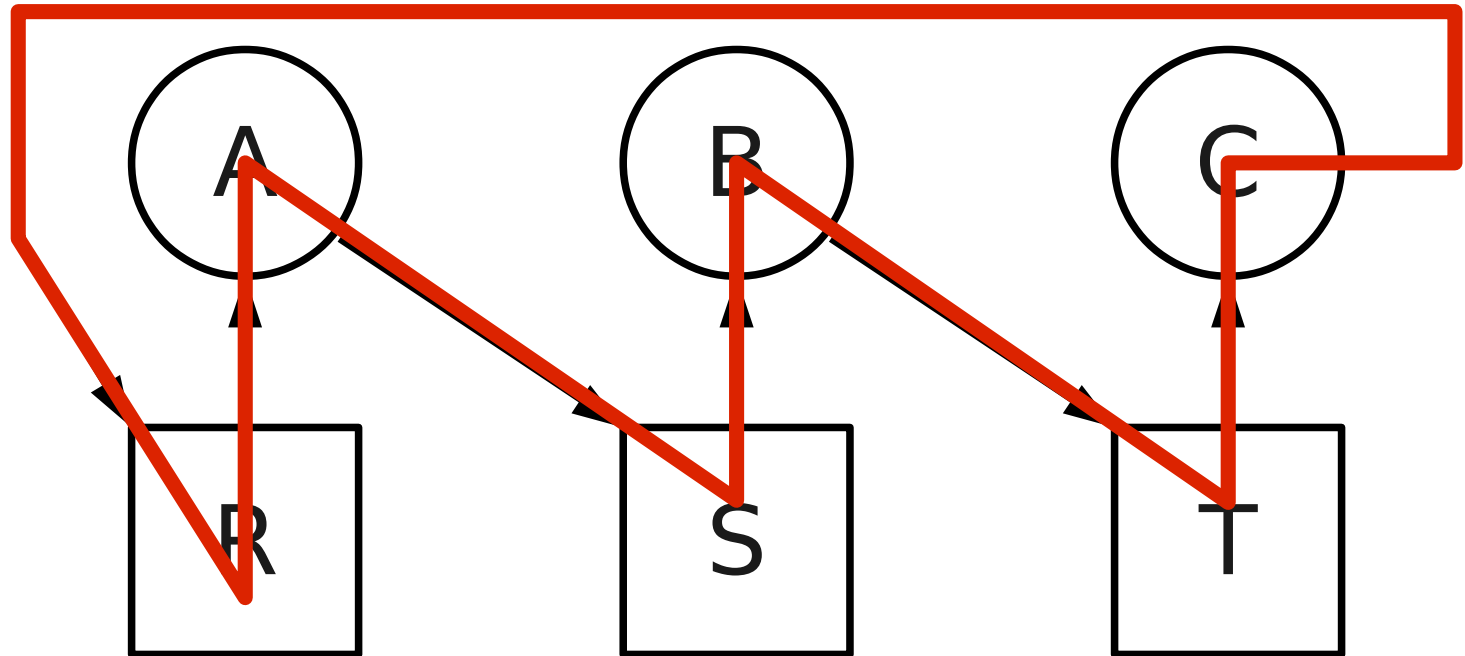
Beispiel (1)

A :
 request (R) ;
 → request (S) ;
 release (R) ;
 release (S) ;


B :
 request (S) ;
 → request (T) ;
 release (S) ;
 release (T) ;


C :
 request (T) ;
 → request (R) ;
 release (T) ;
 release (R) ;


1. A fordert R
2. B fordert S
3. C fordert T
4. A fordert S
5. B fordert T
6. C fordert R



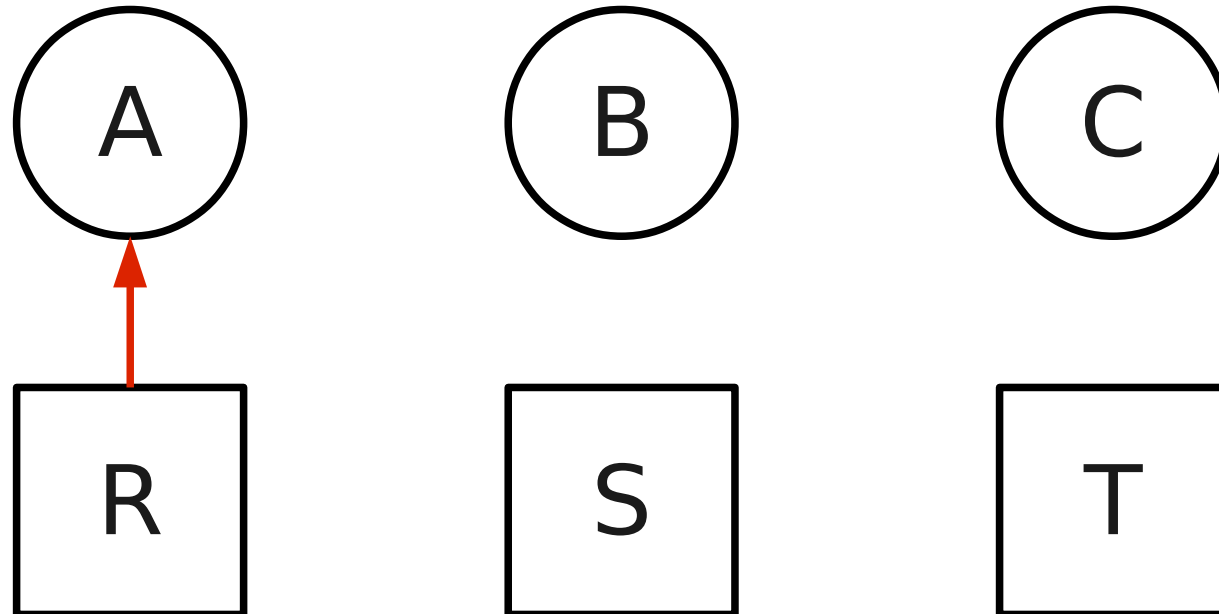
Beispiel (2)

A :
 request (R) ;
 request (S) ;
 release (R) ;
 release (S) ;

B :
 request (S) ;
 request (T) ;
 release (S) ;
 release (T) ;

C :
 request (T) ;
 request (R) ;
 release (T) ;
 release (R) ;

1. A fordert R



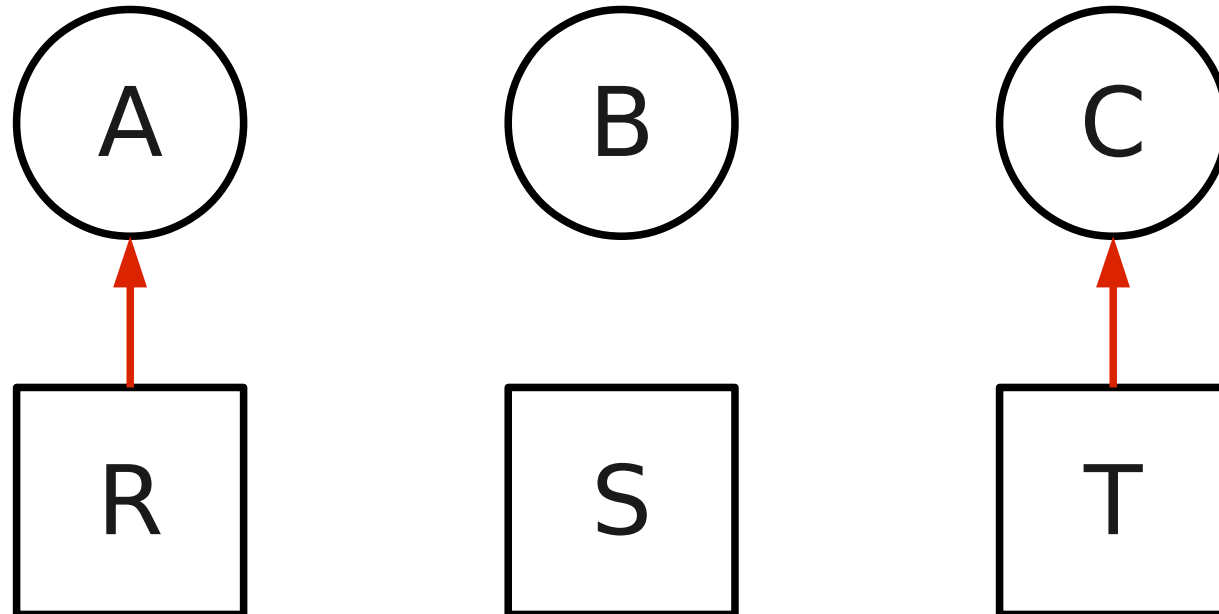
Beispiel (2)

A :
→ request (R) ;
request (S) ;
release (R) ;
release (S) ;

B :
→ request (S) ;
request (T) ;
release (S) ;
release (T) ;

C :
→ request (T) ;
request (R) ;
release (T) ;
release (R) ;

- 1. A fordert R
- 2. C fordert T



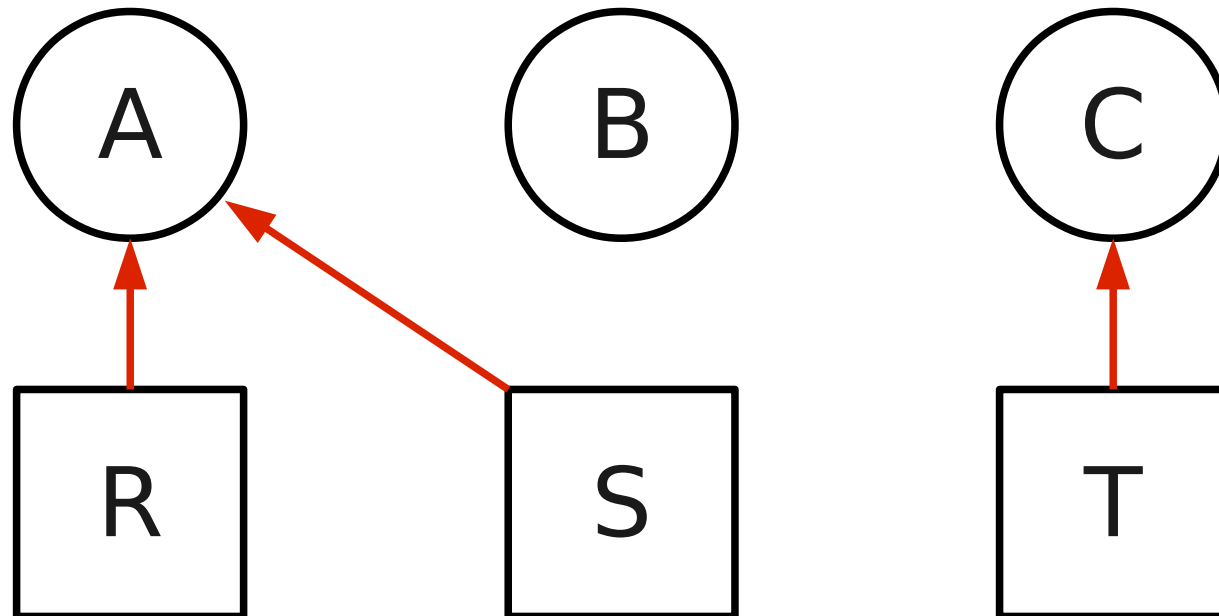
Beispiel (2)

A :
request (R) ;
request (S) ;
release (R) ;
release (S) ;

B :
request (S) ;
request (T) ;
release (S) ;
release (T) ;

C :
request (T) ;
request (R) ;
release (T) ;
release (R) ;

- 1. A fordert R
- 2. C fordert T
- 3. A fordert S



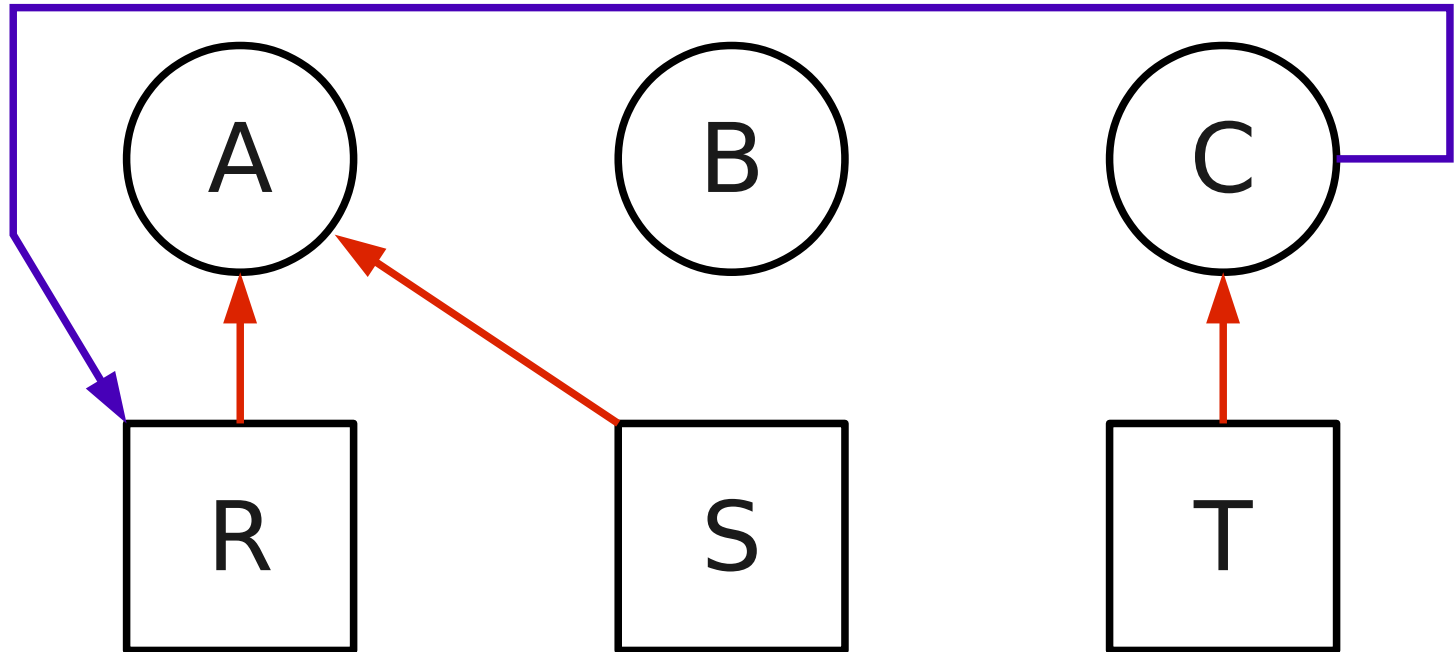
Beispiel (2)

A :
request (R) ;
request (S) ;
release (R) ;
release (S) ;

B :
request (S) ;
request (T) ;
release (S) ;
release (T) ;

C :
request (T) ;
request (R) ;
release (T) ;
release (R) ;

- 1. A fordert R
- 2. C fordert T
- 3. A fordert S
- 4. C fordert R



Beispiel (2)

A :

```
request (R) ;  
request (S) ;  
→ release (R) ;  
release (S) ;
```

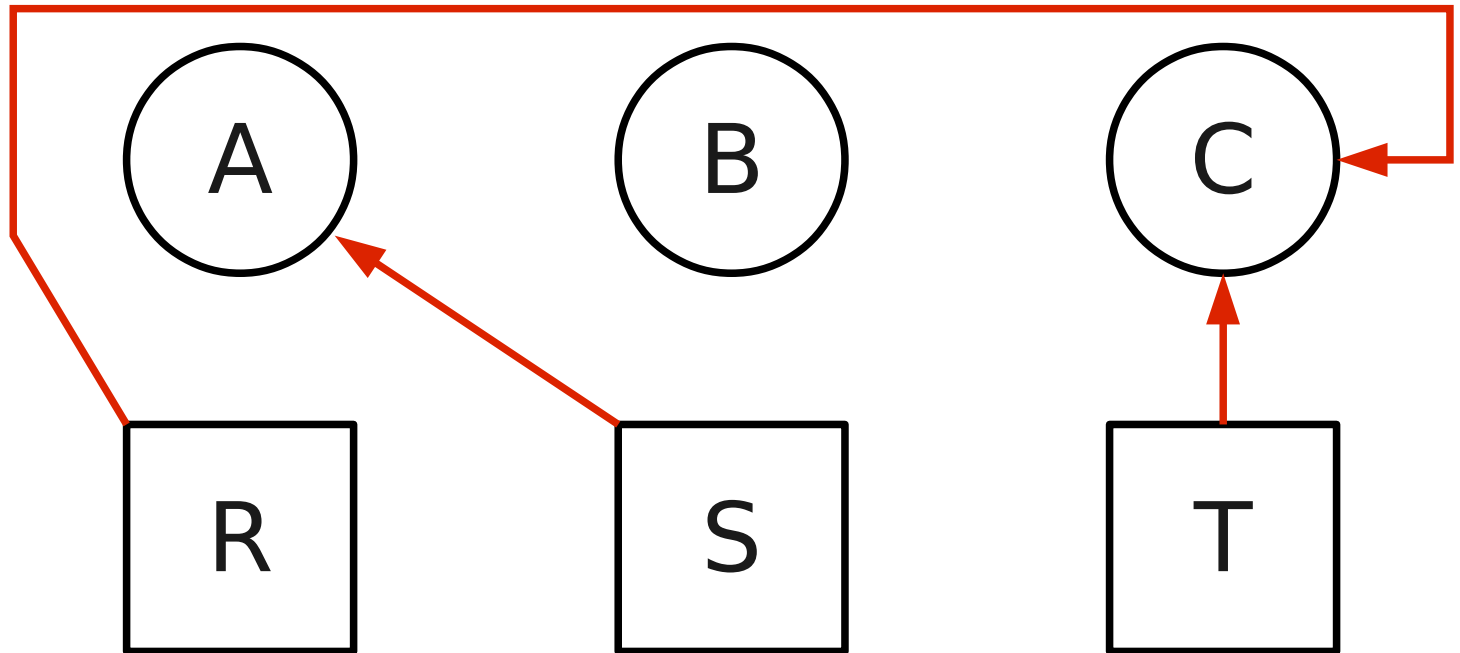
B :

```
→ request (S) ;  
request (T) ;  
release (S) ;  
release (T) ;
```

C :

```
request (T) ;  
→ request (R) ;  
release (T) ;  
release (R) ;
```

1. A fordert R
2. C fordert T
3. A fordert S
4. C fordert R
5. A gibt frei R



Beispiel (2)

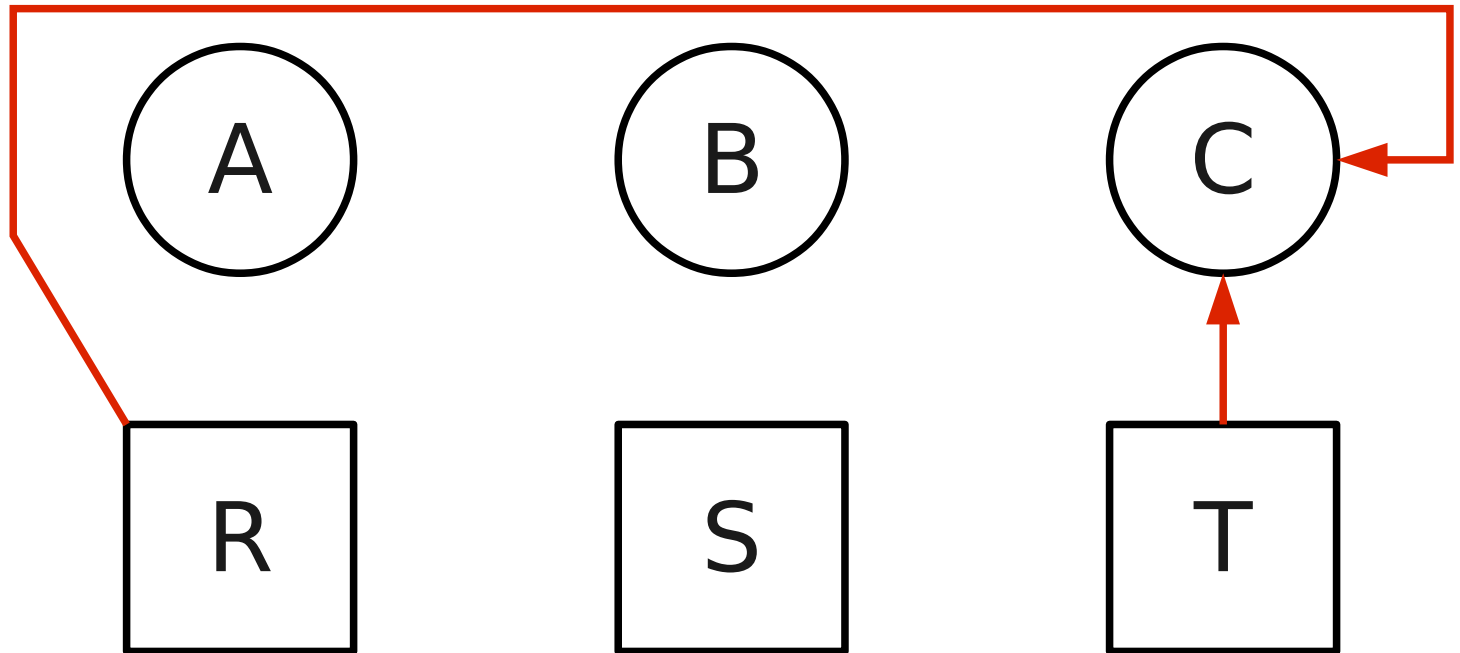
A :

request (R) ;
request (S) ;
release (R) ;
release (S) ;

B :
request (S) ;
request (T) ;
release (S) ;
release (T) ;

C :
request (T) ;
request (R) ;
release (T) ;
release (R) ;

1. A fordert R
2. C fordert T
3. A fordert S
4. C fordert R
5. A gibt frei R
6. A gibt frei S



Charakterisierende Bedingungen

Notwendige und hinreichende Bedingungen für das Auftreten von Verklemmungen bei Ein-Exemplar-BM
COFFMAN (1971)

Notwendig ist einzeln jede der Bedingungen:

- (1) Gegenseitiger Ausschluss bei Benutzung von BM
- (2) Nachfordern von BM (hold and wait)
- (3) Keine Verdrängung (Entzug der BM) möglich
- (4) Zyklische Wartesituation

Charakterisierende Bedingungen

Notwendige und hinreichende Bedingungen für das Auftreten von Verklemmungen bei Ein-Exemplar-BM
COFFMAN (1971)

Notwendig ist einzeln jede der Bedingungen:

- (1) Gegenseitiger Ausschluss bei Benutzung von BM
- (2) Nachfordern von BM (hold and wait)
- (3) Keine Verdrängung (Entzug der BM) möglich
- (4) Zyklische Wartesituation

Hinreichend ist:

$$(1) \wedge (2) \wedge (3) \wedge (4)$$

Wegweiser

Begriff, Beispiele und
Modellierung

Maßnahmen

- Vermeiden per Konstruktion
- Entdecken und Beseitigen

Ausschließen der notwendigen Bedingungen

1. Gegenseitiger Ausschluss (Mutual Exclusion)
 - z. B. Spooling → nur ein Prozess (Daemon) erhält BM
2. Nachfordern
 - Postulat:
 - ♦ alle BM müssen bei Start angefordert werden
 - ♦ Freigabe aller BM bei Neuforderung
 - Problem:
 - ♦ Dynamik
 - ♦ BM-Auslastung
3. Nicht-Verdrängbarkeit
 - bei logischen Betriebsmitteln (→ Transaktionen, später)
4. Zyklische Wartebedingung
 - sorgfältige Betriebsmittelzuteilung (**BMZ**, nächste Folien)

Sorgfältige BMZ(1): Globale Reihenfolge einhalten

- Anforderung der BM in festgelegter Reihenfolge
- Zahlreiche Einzelsituationen in Betriebssystemen
- Beispiel:
 - Jedes Mutex im Betriebssystem wird mit einer Priorität versehen.
 - Mutex darf nur gegriffen werden, wenn keine Mutexe mit höherer Priorität gehalten werden.
 - Beim Anfordern des Mutex:
 - Prüfung ob $\text{Priorität des Mutex} \leq \text{max. Priorität der Mutexe für den aktuellen Thread}$
 - Problem: Laufzeit-Kosten

Sorgfältige BMZ(2): Banker's Algorithm

Bank-Algorithmus für 1 Betriebsmittel-Typ (HABERMANN 1969)

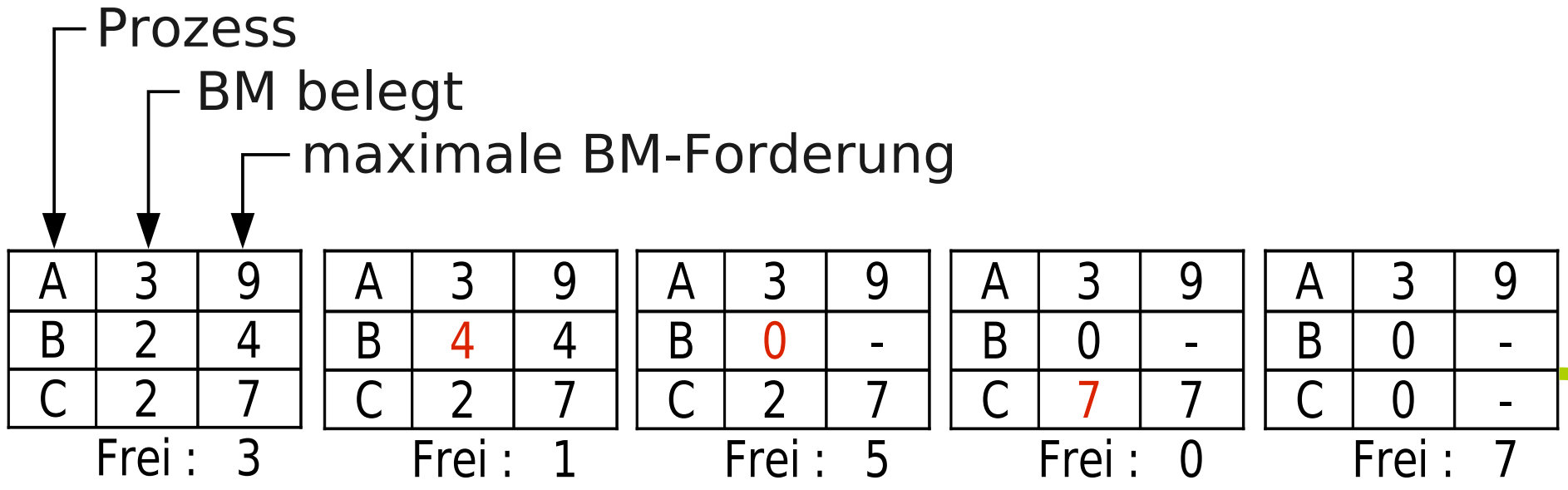
- Gegeben:
 - ♦ Prozesssystem P
 - ♦ maximale Anzahl G von BM-Exemplaren
 - ♦ maximale Anzahl K_p von BM-Exemplaren, die Prozess $P_i \in P$ im Laufe seiner Existenz benötigt mit $K_p \leq G$
- Zustand:
 - ♦ Vektor der momentan den Prozessen zugeteilten BM-Exemplare

Sicherer Zustand

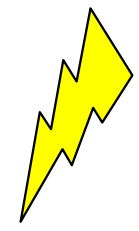
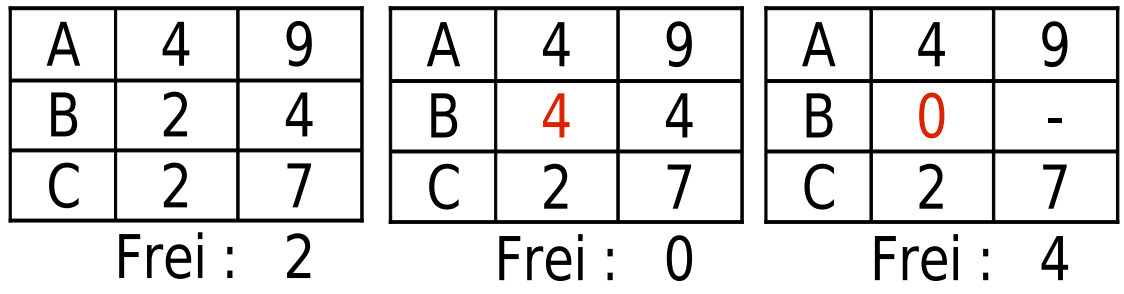
- Ein Zustand heißt **sicher**, wenn
 - ♦ keine Verklemmung vorliegt
 - ♦ es eine Reihenfolge der Erfüllung aller Anforderungen gibt, ohne daß eine Verklemmung auftritt.
- Andernfalls heißt der Zustand **unsicher**.
- Idee des Bank-Algorithmus:
 - ♦ Zuteilung nur vornehmen, wenn Nachfolgezustand immer noch sicher ist, andernfalls muss Prozess warten.

Beispiel (nach Tanenbaum)

→ 10 Betriebsmittel-Exemplare eines Typs verfügbar
(Mehrere Typen → Lehrbuch)



Sicherer Zustand



Bank-Algorithmus – Probleme

- Komplexität: $O(n^2)$
 n : Anzahl der Prozesse
- Mehrere BM-Typen: mehrere „Währungen“
- Basis: maximale Anzahl genutzter BM-Exemplare
 - ♦ Eintreten dieser Situation („pessimistischer Algorithmus“)
 - ♦ Kenntnis
- Blockierdauer nicht kalkulierbar

Entdecken (und Beseitigen)

- **Symptom:**

Operation dauert wesentlich länger als erwartet

z. B. TIMEOUT als zusätzlicher Parameter blockierender Operationen:

- ♦ receive(message, TIMEOUT)
- ♦ Sema.P(TIMEOUT)

- **Diagnose:**

Feststellen ob zyklische Wartebedingung vorliegt

POSIX.4 :

int sem_wait(sem_t *s): EDEADLK

als Rückgabewert bei Gefahr eines Deadlocks

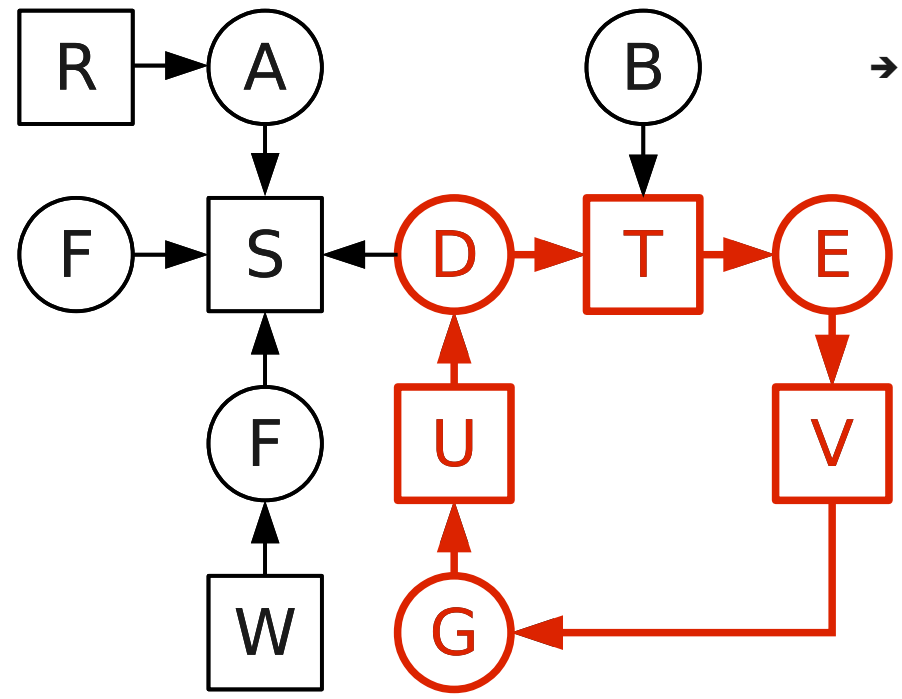
- **Therapie:**

Verdrängung von Betriebsmitteln

Entdecken (und Beseitigen): Diagnose

<u>Prozess</u>	<u>belegt</u>	<u>fordert</u>
A	R	S
B	nichts	T
C	nichts	S
D	U	S und T
E	T	V
F	W	S
G	V	U

→ ein BM pro Typ



(Entdecken und) Beseitigen: Therapie

- Verdrängen
 - ♦ „von Hand“
- Entfernen von Prozessen
- DIMMUNIX: „Merken“ von Deadlock-Situationen und späteres Vermeiden
- Zurücksetzen
 - ♦ regelmäßiges Erstellen von Rücksetzpunkten
 - ♦ Zurücksetzen vor die verursachende BM-Anforderung
 - ♦ Außenwelt-Problem
(Terminal, Drucker, Datenbankeinträge, ...)
 - ♦ → Transaktionen!

Transaktionen

Zwei-Phasen-Technik (two phase locking)

Datenbanken, Transaktionssysteme:

- Aufsammeln der „Locks“, BM nutzen und freigeben am Ende der Transaktion
- falls ein „Lock“ nicht erworben werden kann (timeout), alle erworbenen BM freigeben und auf alten Zustand zurücksetzen

```
Begin Transaction
  request_and_lock(      , timeout);    use( ... );
  request_and_lock(      , timeout);    use( ... );

  ON Timeout :
    Abort Transaction //release all locks
Commit Transaction     //release all locks
```

Entdeckung : Timeout

Auflösen: Rücksetzen

Verwandt: Aushungern (Starvation)

Ausgangspunkt

BM werden nach Gesichtspunkten vergeben, die im Normalfall gut funktionieren, aber im Einzelfall zum „Verhungern“ eines Prozesses führen können

Beispiel

- Scheduling: shortest job next
- Drucker: shortest job next

Zusammenfassung Verklemmungen

- Vermeiden per Konstruktion
- Wenn das zu aufwendig:
Erkennen und Beseitigen
- Virtualisierung von Betriebsmitteln