2. Prozeßverklemmungen

2.1. Beschreibung mittels PETRI-Netzen

2.1.1. Einführung

- ***Ausgangspunkte***

Realisierbarkeit von Automaten durch Schaltwerke (PETRI 1962)

Mängel des Automatenbegriffs

- ***Vorteile***

Anschaulichkeit – unterschiedliche Abstraktionsstufen –

ausgefeilte Theorie – Weiterentwicklungen – Software

- ***Literatur***

PETRI, E.A.: Kommunikation mit Automaten. Diss. Univ. Bonn, 1962.

STARKE, P.H.: Petri-Netze. Berlin 1980.

STARKE, P.H.: Analyse von Petri-Netz-Modellen. Teubner, 1990.

REISIG, W.: Petri-Netze. Eine Einführung. Springer, 1990.

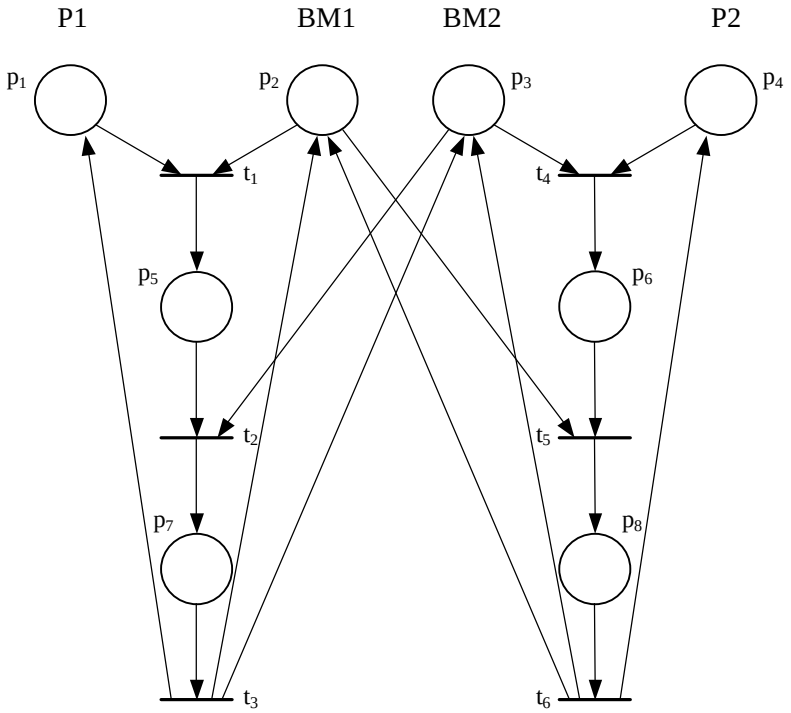
BAUMGARTEN, B.: Petri-Netze. Grundlagen und Anwendungen. Wissenschaftsverlag Mannheim, 1990.

KÖNIG/QUÄCK: Petri-Netze in der Steuerungstechnik. Verlag Technik, Berlin, 1988.

SONNENSCHEIN, M.: Petri-Netze: Einführende Übersicht. Schriften zur Informatik 136, Aachen, 1988.

2.1.2. Beispiel

PETRI-Netz P_V zur Verklemmung zweier Prozesse P1, P2 mit zwei Betriebsmitteln BM1, BM2



2.1.3. PETRI-Netze – Grundbegriffe

- ***Elemente:*** Ereignisse (Transitionen, Übergänge) t $\rightarrow|\rightarrow$
  Bedingungen (Plätze, Stellen) p $\rightarrow\bigcirc\rightarrow$

  Zusammenhang:

  $p \bigcirc\rightarrow \overset{t}{|}\rightarrow\bigcirc\ p'$ Die Bedingung p ist Vorbedingung des Ereignisses t.
  Die Bedingung p' ist Nachbedingung des Ereignisses t.

- ***Definition 1.***

  N = (P,T,F,m) heißt *markiertes PETRI-Netz* (kurz: PN), wenn gilt:

  (1) P, T sind endliche Mengen (*Plätze, Transitionen*);

  (2) $P \cap T = \emptyset$, $P \cup T \neq \emptyset$;

  (3) $F \subseteq (P \times T) \cup (T \times P)$ *Flußrelation* mit
  $dom(F) \cup codom(F) = P \cup T$;

  (4) $m: P \rightarrow \mathbb{N}$ *Markierung*.

  Weiter sei

  $\cdot t := \{p \in P \mid (p,t) \in F\}$ Menge der *Vorplätze* für $t \in T$,

  $t\cdot := \{p \in P \mid (t,p) \in F\}$ Menge der *Nachplätze* für $t \in T$.

  Analog $\cdot p$, $p\cdot$ Menge der *Vor-/Nachtransitionen*.

- ***Bemerkungen***

  bipartiter gerichteter Graph, $\bigcirc \in P$, $| \in T$, ohne isolierte Knoten, keine „hängenden Kanten“.

  Marke: •

  Vielfachheit der Kanten: $V: F \rightarrow \mathbb{N}^+$.

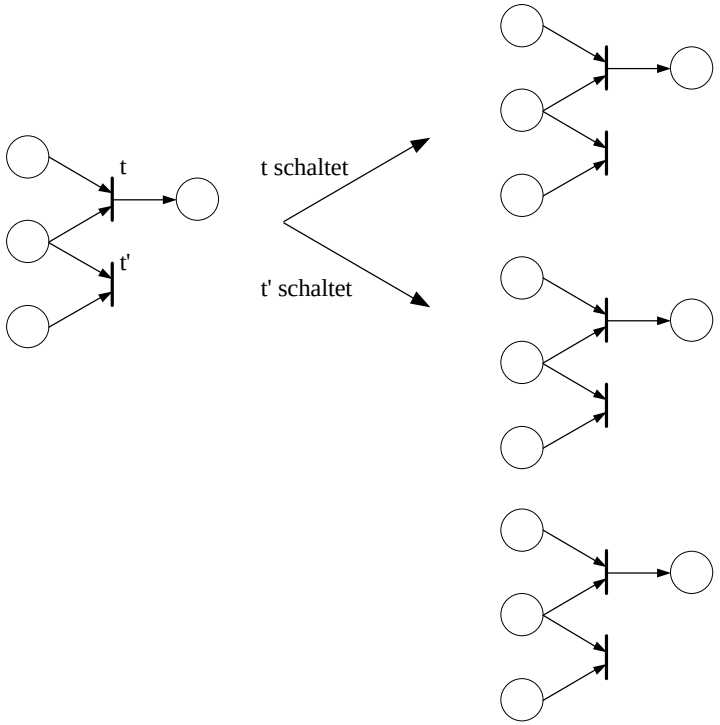
- ***Transitionsregel***

  Die Transition t kann höchstens dann *schalten*, wenn alle ihre Vorplätze markiert (alle ihre Vorbedingungen erfüllt) sind.

  Wenn t schaltet, werden alle Vorbedingungen von t beendet, und alle Nachbedingungen sind erfüllt.

  Sprechweise: t *hat Konzession*, kann feuern.

- ***Bsp. 1.***



- ***Erreichbarkeit***

Sei $N = (P,T,F,m_0)$ ein PN; T* bezeichne die Worthalbgruppe über T. Dann wird induktiv eine Relation $\rightarrow^q$ bzw. $\rightarrow^*$ über $\mathbb{N}^P$ für $q \in T^*$ definiert:

Sei $m, m' \in \mathbb{N}^P$, $q \in T^*$, $t \in T$.

(1) $m \rightarrow^\varepsilon m' :\Leftrightarrow m = m'$.

(2) $m \rightarrow^{qt} m' :\Leftrightarrow \exists m'': m \rightarrow^q m'' \wedge m'' \rightarrow^t m'$.

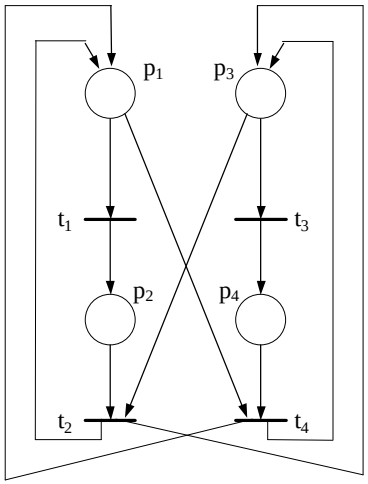
(3) $m \rightarrow^* m' :\Leftrightarrow \exists q \in T^*: m \rightarrow^q m'$. *Erreichbarkeitsrelation*

Ferner sei $R(m) := \{m' \mid m \rightarrow^* m'\}$

$EG(N) := [R(m_0), B]$ *Erreichbarkeitsgraph*

$$B := \{(m,m') \mid m,m' \in R(m_0) \wedge \exists t \in T: m \rightarrow^t m'.\}$$

- ***Bsp. 2.***



- ***Beschränktheit***

Ein PN $N = (P,T,F,m_0)$ heißt *beschränkt* bei

$\forall p \in P \ \exists k = k(p) \in \mathbb{N} \ \forall m \in R(m_0): m(p) \leq k$.

Ist $k = 1 \ \forall p \in P$, so heißt N *sicher*.

Es gilt: N beschränkt $\Leftrightarrow$ $R(m_0)$ endlich.

- ***Bemerkungen***

– Der Erreichbarkeitsgraph eines endlichen PN muß nicht endlich sein.

– Der Erreichbarkeitsgraph ist algorithmisch konstruierbar.

– Die Zeitkompliziertheit des Problems, den Erreichbarkeitsgraphen eines beschränkten PN zu berechnen, ist überexponentiell.

Zeitkompl.: Anzahl der Schritte bis zum Abbruch als Funktion von der Größe des PN.

Größe des PN: $|P| + |T| + |F| + S$, S: Summe aller Marken.

Schritt: Berechnung einer Markierung.

– Erreichbarkeitsproblem ist von gleicher Kompliziertheit.

2.1.4. Lebendigkeit

- ***Definition 2.*** Sei $N = (P,T,F,m_0)$ ein PN, $M = \mathbb{N}^P$.

  (1) $m \in M$ heißt *tot*, wenn kein $t \in T$ Konzession bei m hat.

  (2) $t \in T$ heißt *tot bei* $m \hat{I} M$ (bzw. *in N*), wenn von m (bzw. m_0) aus keine Markierung erreichbar ist, bei der t Konzession hat.

  (3) N heißt *schwach lebendig* oder *verklemmungsfrei*, wenn in N keine tote Markierung erreichbar ist.

  (4) $t \in T$ heißt *lebendig (bei* $m \hat{I} M$*)*, wenn t bei keiner von m_0 (von m) aus erreichbaren Markierung tot ist.

  (5) $m \in M$ heißt *lebendig*, wenn alle $t \in T$ lebendig bei m sind.

  (6) N heißt *lebendig*, wenn m_0 lebendig ist:

  $\forall t \forall m(t \in T \wedge m \in R(m_0) \Rightarrow \exists q \in T^* \exists m' \in R(m): m \rightarrow^{qt} m')$.

- ***Es gilt:***

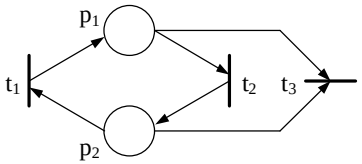
– Ist N lebendig, so ist N verklemmungsfrei.

– Ist t lebendig (bzw. tot) bei m, dann ist t lebendig (bzw. tot) bei allen von m aus erreichbaren Markierungen.

– Ist t tot, dann ist t nicht lebendig.

– Ist N nicht verklemmungsfrei, dann besitzt N keine lebendige Transition.

- ***Bsp. 3.***



a) m = (2 0)

b) m = (3 0)

- ***Weiter gilt:***

  – Es ist entscheidbar, ob eine Transition tot ist.

  – Verklemmungsfreiheit ist äquivalent mit Erreichbarkeit.

  – Verklemmungsfreiheit ist entscheidbar.

2.1.5. Struktureigenschaften

- ***Definition 3.*** Ein PN $N = (P,T,F,m_0)$ heißt

  (1) FC-Netz (free choice):

  $(p,t) \in F \Rightarrow p^{\bullet} = \{t\} \vee {}^{\bullet}t = \{p\}$ $\quad \forall p \in P \, \forall t \in T;$

  (2) EFC-Netz (extended f.c.):

  $(p,t), (p,t') \in F \Rightarrow {}^{\bullet}t = {}^{\bullet}t'$ $\quad \forall p \in P \, \forall t,t' \in T;$

  (3) ES-Netz (extended simple):

  $p^{\bullet} \cap q^{\bullet} \neq \varnothing \Rightarrow p^{\bullet} \subseteq q^{\bullet} \vee q^{\bullet} \subseteq p^{\bullet}$ $\quad \forall p,q \in P;$

  (4) Zustandsmaschine: $|{}^{\bullet}t| = |t^{\bullet}| = 1$ $\quad \forall t \in T;$

  (5) Synchronisationsgraph: $|{}^{\bullet}p| = |p^{\bullet}| = 1$ $\quad \forall p \in P.$

- ***Bsp. 4.***

- ***Es gilt:***

– FC $\Rightarrow$ EFC $\Rightarrow$ ES.

– Jeder Synchronisationsgraph und jede Zustandsmaschine sind FC-Netze.

– Jeder Synchronisationsgraph ist strukturell konfliktfrei.

– Jeder Synchronisationsgraph besitzt eine lebendige Markierung, jede Zustandsmaschine eine sichere.

- ***Deadlocks und Fallen***

Für $Q \subseteq P$ sei $Q^- := \{t \in T \mid \exists p \in Q: (p,t) \in F\}$,

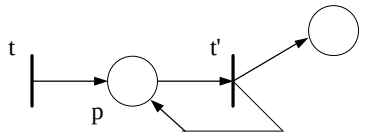
$Q^+ := \{t \in T \mid \exists p \in Q: (t,p) \in F\}$.

Definition 4. $Q \neq \emptyset$ heißt *Falle* (trap) bei $Q^- \subseteq Q^+$;

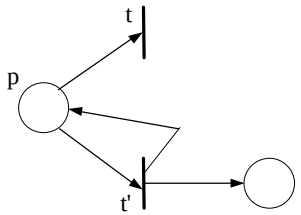
Q heißt *Deadlock* bei $Q^+ \subseteq Q^-$.

- ***Bsp. 5.***

a)



b)



- ***Definition 4.*** $Q \subseteq P$ heißt *sauber* bei $m \in M$, wenn die Plätze von Q keine Marken enthalten.

- ***Es gilt:***

Sei $N = (P,T,F,m_0)$ ein PN, $m \in \mathbb{N}^P$.

– Ist D ein bei der Markierung m sauberer Deadlock von N, dann ist D bei jeder vom m aus erreichbaren Markierung sauber, und alle Transitionen aus D^- sind tot bei m.

– Ist S eine bei m markierte Falle in N und $m' \in R(m)$, so ist S auch bei m' markiert.

– Besitzt N keinen Deadlock, so ist N lebendig.

- ***Definition Deadlock-Falle-Eigenschaft***

Das PN $N = (P,T,F,m_0)$ hat die Deadlock-Falle-Eigenschaft (DF-Eigenschaft), wenn jeder Deadlock von N eine bei m_0 markierte Falle enthält.

2.1.6. Charakterisierung lebendiger PETRI-Netze

Satz 1. Sei N ein EFC-Netz. Dann ist N genau dann lebendig, wenn N die DF-Eigenschaft besitzt.

Satz 2. Ist N ein ES-Netz mit DF-Eigenschaft, so ist N lebendig.

Satz 3. Besitzt $N = (P,T,F,m_0)$ die DF-Eigenschaft, so ist N verklemmungsfrei.

Beweis. Angenommen, N sei nicht verklemmungsfrei, d.h.:

Es wird betrachtet

$D := \{p \in P \mid m(p) = 0\}$.

Dann ist $D \neq \emptyset$, denn sonst gälte

$m(p) > 0 \quad \forall p \in P$,

d.h., alle t hätten Konzession –

D ist Deadlock: Sei $t \in D^+$. Da m tot, hat t keine Konzession, also:

$\exists p \in {}^\bullet t: \; m(p) = 0$,

d.h. $p \in D$ und damit $t \in D^-$.

D enthält aber keine bei m markierte Falle (nach Konstruktion). Dies ist WS zu DF-Eigenschaft. ¦

2.2. Systeme mit seriell wiederverwendbaren Betriebsmitteln

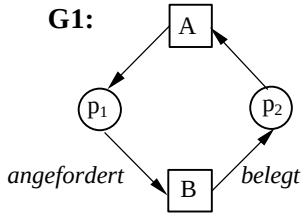
2.2.1. Beispiele

- Gemeinsames Benutzen von seriell wiederverwendbaren Einexemplar-Betriebsmitteln

p_1: anfordern (A)
anfordern (B)
...
freigeben (B)
freigeben (A)

p_2: anfordern (B)
anfordern (A)
...
freigeben (A)
freigeben (B)

G1:



- Gemeinsames Benutzen eines seriell wiederverwendbaren Mehrexemplar-Betriebsmittels B in m Exemplaren,

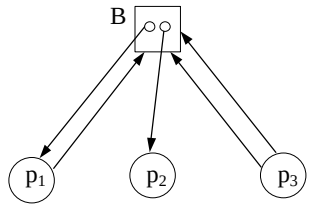
mehrfach benutzbar durch n Prozesse $(2 \leq m \leq n)$:

p_i: anfordern(B)
...
anfordern(B)
} k - mal, $k \leq m$

...

freigeben(B)
...
freigeben(B)
} k - mal

Bsp. m = 2, n = 3: **G2:**



- Nicht wiederverwendbare BM
- Aushungern

2.2.2. System-Modell

- ***Begriffe***
  Zustand: Zuweisungszustand für Betriebsmittel (frei – zugewiesen)
  Prozeßaktion: Betriebsmittel anfordern – belegen – freigeben.

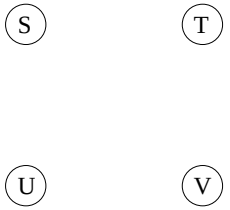
- ***Definition 1.*** Ein *System* $\Sigma = [\sigma, \pi]$ besteht aus einer Menge von Systemzuständen σ und einer Menge von Prozessen π. Dabei ist ein *Prozeß* eine partielle nichtdeterministische Funktion von σ nach σ: p: $\sigma \to \mathbf{P}(\sigma)$.

Notation und Sprechweise für $S,T \in \sigma$, $p \in \pi$, $T \in p(S)$:
$S \to^p T$: p kann S in T überführen
$S \to^* T$: es gibt eine endliche Folge von Prozessen, die S in T überführen kann.

- ***Beispiel 1.*** $\sigma = \{S,T,U,V\}$, $\pi = \{p_1, p_2\}$

p_1: $S \mapsto T,U$
$U \mapsto V$
$V \mapsto U$

graphisch:



p_2: $S \mapsto U$
$T \mapsto S,V$

mögl. Zustandsfolgen: $S \to_{p_1} U$, $T \to_{p_2} V$, $S \to^* V$.

- ***Definition 2.***

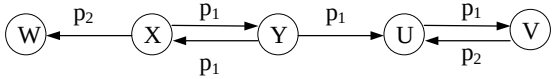
(1) Ein Prozeß p ist im Zustand S *blockiert*, wenn gilt: $p(S) = \emptyset$.

(2) $p \in \pi$ heißt *verklemmt* im Zustand S, falls für alle T mit $S \to^* T$ gilt: p ist im Zustand T blockiert.

(3) $S \in \sigma$ heißt *Verklemmungszustand*, wenn es einen Prozeß p gibt, der in S veklemmt ist.

(4) $S \in \sigma$ heißt *sicherer* Zustand, falls kein T mit $S \to^* T$ Verklemmungszustand ist.

- ***Beispiel 2.***



p_1 blockiert in
p_1 verklemmt in
p_2 blockiert in
p_2 verklemmt in
Verklemmungszustände:
sichere Zustände:
nicht sichere Zustände:

- ***Beispiel 3.*** Prozesse p_1, p_2; Betriebsmittel A, B.

– *Prozeßzustände:*

Zust.	p_1
0	besitzt keine BM
1	fordert A
2	besitzt A
3	besitzt A, fordert B
4	besitzt A, B
5	besitzt A nach Freigabe von B

Zust.	p_2
0	besitzt keine BM
1	fordert B
2	besitzt B
3	besitzt B, fordert A
4	besitzt A, B
5	besitzt B nach Freigabe von A

– *Systemzustände:*

S_{ij}: p_1 in Zustand i, p_2 in Zustand j (i,j = 0,..,5)

– *Zustandsdiagramm:*

p_1 \ p_2	0	1	2	3	4	5
0	.	.	.	.	.	.
1	.	.	.	.	.	.
2	.	.	.	.	.	.
3	.	.	.	.	.	.
4	.	.	.	.	.	.
5	.	.	.	.	.	.

2.2.3. Verklemmungen bei seriell wiederverwendbaren Betriebsmitteln

- ***Seriell wiederverwendbare Betriebsmittel***

endliche Menge von identischen Einheiten mit

– Anzahl der Einheiten konstant;

– jede Einheit ist entweder verfügbar oder genau einem Prozeß zugeteilt;

– ein Prozeß kann eine Einheit nur dann wieder freigeben, wenn sie ihm vorher zugeteilt wurde.

- ***Definition 3.*** *Betriebsmittel-Graph:*

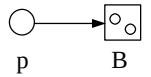
G = [P, K] bipartit mit

(1) $P = \pi \cup \rho, \quad \pi \cap \rho = \varnothing$.

π: Prozeßpunkte ○ ρ: Betriebsmittelpunkte □

(2) Jedes $B \in \rho$ besteht aus t_B Einheiten ○ B  $t_B = 3$

(3) $K \subseteq (\pi \times \rho) \cup (\rho \times \pi)$

k = (p, B): Anforderungskante 

k = (B, p): Zuweisungskante 

(4) a) $\sum_{p \in p} |(B, p)| \le t_B \quad \forall B \in r$;

b) $|(B, p)| + |(p, B)| \le t_B \quad \forall B \in \rho \; \forall p \in \pi$.

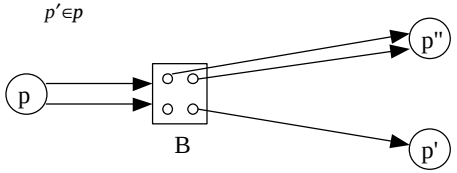
Bezeichnung: G(S): der zum Zustand S gehörige Betriebsmittel-Graph (Graph wiederverwendbarer Betriebsmittel).

- ***Zustandsänderungen***

– *Anforderung:* Hat p im Zustand S keine noch ausstehenden Forderungen, so kann p jede Zahl von Betriebsmitteln fordern bei Einhaltung von (4). System gelangt in Zustand T, der sich von S genau durch die entsprechenden Anforderungskanten (im Graphen) unterscheidet.

– *Belegung (Zuweisung):* Es gilt $S \rightarrow^{p} T$ durch Zuweisung genau dann, wenn p ausstehende Forderungen hat und diese alle gleichzeitig erfüllt werden. Dabei muß gelten:

(5) $|(p,B)| + \sum_{p' \in p} |(B,p')| \le t_B \quad \forall B: (p,B) \in K$.



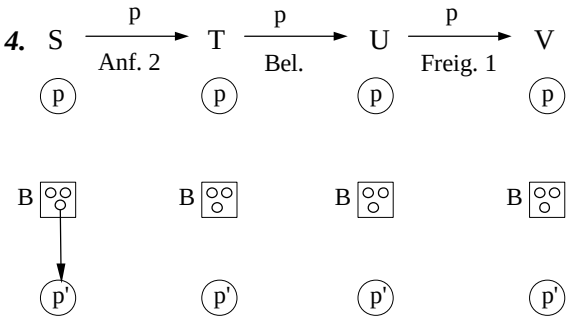
Betriebsmittel-Graph: G(T) ergibt sich aus G(S) durch Umkehrung aller Anforderungskanten von p nach B.

– *Freigabe:* $S \rightarrow^{p} T$ gilt genau dann, wenn p keine ausstehenden Forderungen, aber mindestens 1 Zuweisung hat, d.h.

$[\forall B \in \rho: |(p,B)| = 0] \wedge [\exists B \in \rho: |(B,p)| > 0]$.

G(T) ergibt sich aus G(S) durch Löschen der entsprechenden Kanten.

- ***Beispiel 4.***



2.2.4. Entdecken von Verklemmungen

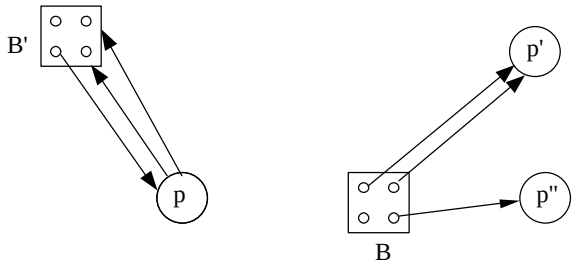
- ***Grundidee***

Ein nicht blockierter Prozeß belegt evtl. weitere geforderte Betriebsmittel, gibt dann alle seine Betriebsmittel frei. Die freigegebenen können anderen blockierten Prozessen zugeteilt werden usw. Sind am Ende dieses Vorgehens noch Prozesse blockiert, so war der ursprüngliche Zustand ein Verklemmungszustand, sonst nicht.

Formale Charakterisierung eines blockierten Prozesses:

p blockiert genau dann, wenn

(6) $\exists B \in \rho: (p,B) \in K \wedge |(p,B)| + \sum_{p' \in \pi} |(B,p')| > t_B$.



- ***Reduktion eines Graphen wiederverwendbarer Betriebsmittel***

  (1) Ein Graph wird durch einen Prozeß p, der weder blockiert noch isoliert ist, *reduziert*, indem alle Kanten von und nach p gelöscht werden. p wird dann zu isoliertem Punkt.

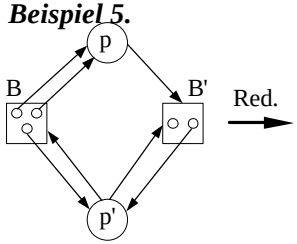
  (2) Ein Graph heißt *nicht reduzierbar*, wenn er durch keinen Prozeß reduziert werden kann.

  (3) Ein Graph heißt *vollständig reduzierbar*, wenn es eine Folge von Reduktionen gibt, so daß alle Kanten gelöscht werden können.

  (4) Eine Reduktionsfolge heißt *maximal*, wenn sie zu einem nicht reduzierbaren Graphen führt.

  Bemerkung. Die Elemente einer Reduktionsfolge sind paarweise verschiedene Elemente von π.

- ***Beispiel 5.***



- ***Lemma.*** Alle maximalen Reduktionsfolgen eines Graphen G wiederverwendbarer Betriebsmittel führen zu demselben Graphen.

  Beweis.

  a) Für 2 Reduktionsfolgen r_1, r_2 sei $r_1 \prec r_2$ („r_2 nach r_1“) die Folge, die entsteht (Operation!), indem an r_1 alle Elemente von r_2 angehängt werden, die nicht in r_1 enthalten sind.
  Beispiel. $123 \prec 43561 =$ $43561 \prec 123 =$
  $124 \prec 41 =$ $41 \prec 124 =$

  b) Seien r_1, r_2 zwei Reduktionsfolgen für G. Dann läßt sich G auch durch $r_1 \prec r_2$ und $r_2 \prec r_1$ reduzieren.
  Denn da durch jeden Reduktionsschritt die Menge der verfügbaren Betriebsmittel größer wird, bleibt ein nicht blockierter Prozeß nicht blockiert, kann also zur weiteren Reduktion verwendet werden.

  Mithin: Gibt es für G mehrere maximale Reduktionsfolgen, so sind dies Permutationen voneinander.

  c) Sind r_1, r_2 maximale Reduktionsfolgen von G, so sind die dadurch reduzierten Graphen gleich.
  Denn bei der Reduktion durch einen Prozeß p werden jeweils alle mit p verbundenen Kanten entfernt, keine hinzugefügt; somit werden (da r_1 und r_2 Permutationen voneinander sind) durch die Folgen pp' und p'p die gleichen disjunkten Kantenmengen entfernt.

- ***Satz 1. Verklemmungstheorem.***

  S ist Verklemmungszustand genau dann, wenn der Graph G(S) nicht vollständig reduzierbar ist.

  Beweis.

  a) Sei S Verklemmungszustand. Dann gibt es einen Prozeß p, der in S verklemmt ist. Dies bedeutet für alle T mit $S \rightarrow^{*} T$: p ist im Zustand T blockiert.

     Angenommen, G(S) wäre vollständig reduzierbar. Dann müßte p in einer Reduktionsfolge enthalten sein, dürfte also in irgendeinem erreichbaren Zustand nicht blockiert sein – Widerspruch.

  b) Sei G(S) nicht vollständig reduzierbar. Dann gibt es einen Prozeß p, der in allen möglichen Reduktionsfolgen blockiert bleibt (wegen Lemma 1). Da der reduzierte Graph durch eine maximale Reduktionsfolge erreicht wird, bei der alle freigebbaren Betriebsmittel auch wirklich freigegeben werden, kann kein einziges weiteres Betriebsmittel freiwerden, p bleibt mithin immer blockiert und damit verklemmt in S. ¦

- ***Beispiel 6.***

- ***Folgerung.*** Ist S Verklemmungszustand, so gibt es mindestens zwei Prozesse, die in S verklemmt sind.

2.2.5. Ergänzungen

- ***Satz 2. Zyklustheorem.*** Für eine Verklemmung ist ein Zyklus im Graphen notwendig, aber nicht hinreichend.

  Notwendig o.B.; nicht hinreichend.

- ***Satz 3.*** Ist S kein Verklemmungszustand und gilt $S \rightarrow^{p} T$, so ist T genau dann Verklemmungszustand, wenn p eine Anforderungsoperation darstellt und p im Zustand T verklemmt ist.

- ***Sofortige Zuweisung***
  Zuteilungsstrategie, bei der in einem Systemzustand alle erfüllbaren Forderungen auch sofort erfüllt werden.

  „Zweckdienlicher Zustand“: Zustand, in dem alle Prozesse mit ausgehenden Kanten blockiert sind.

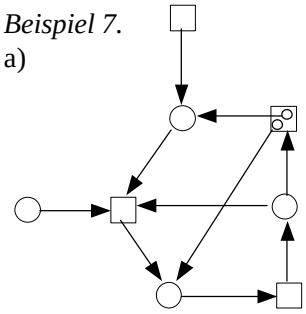
  „Klumpen“: Teilgraph, bei dem jeder Punkt von jedem aus erreichbar ist, aber kein Punkt außerhalb des Graphen ist erreichbar.

„Senke“: Punkt eines Graphen,
von dem keine Kanten ausgehen.

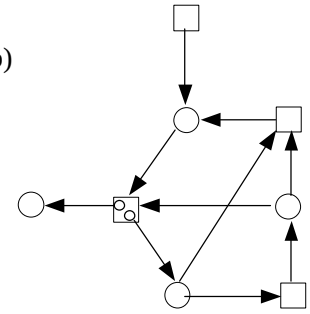
Es gilt: Ist ein Zustand zweckdienlich, dann ist die Existenz eines Klumpens hinreichend für eine Verklemmung.

Beispiel 7.

a)



b)



- ***Einexemplar-Betriebsmittel*** $t_B = 1 \quad \forall B \in \rho$.

Satz 4. Ein Zustand S ist genau dann Verklemmungszustand, wenn der BM-Graph G(S) einen Zyklus enthält.

Beweis. Notwendig: Satz 2.

Hinreichend: Sei p ein bel. Prozeß des Zyklus Z. Dann hat p eine zu einem Betriebsmittel aus Z gehende Kante. Ferner führt von jedem Betriebsmittel aus Z genau eine Kante zu einem Prozeß aus Z. Also gilt für alle $p \in Z$:

$$\exists B \in Z: \ |(p, B)| + \Sigma|(B, p')| \geq 2 > 1 = t_B,$$

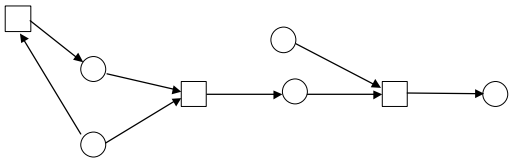
damit gilt (6).

Mithin ist jeder Prozeß des Zyklus blockiert, also sind von S aus nur Zustände T erreichbar, die auch blockiert sind. •

Algorithmus:

Schrittweises Löschen von Kanten, die zu Senken führen. Können alle Kanten gelöscht werden, so hatte der ursprüngliche Graph keine Zyklen.

Beispiel 8.

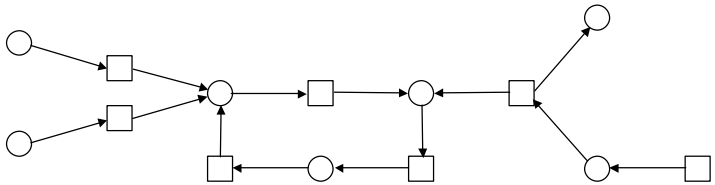


- ***Anfordern von nur einer einzigen Einheit eines Betriebsmittels***

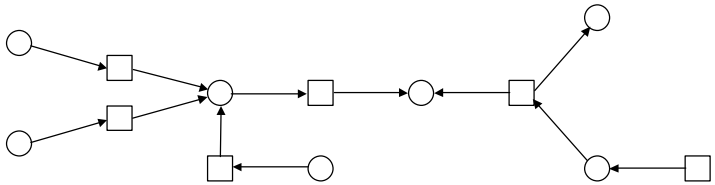
Es gilt: Genau dann ist ein Zustand, in dem alle Prozesse mit ausstehenden Anforderungen blockiert sind, ein Verklemmungszustand, wenn der zugehörige Graph einen Klumpen enthält (o.B.).

Beispiel 9.

a)



b)



Algorithmus:

- Alle Senken bestimmen $\rightarrow$ Menge S
- Iterativ alle Punkte zu S hinzunehmen, die Kanten nach S haben
- Ist am Ende S = P, so besitzt G(S) keinen Klumpen, andernfalls doch.

- ***Wiederherstellung eines Zustandes ohne Verklemmung***

Prozeßabbruch – Betriebsmittel-Entzug

- ***Verhinderung von Verklemmungen***

Hierarchische Betriebsmittel-Vergabe – Maximale Ansprüche

- ***Nicht wiederverwendbare Betriebsmittel***