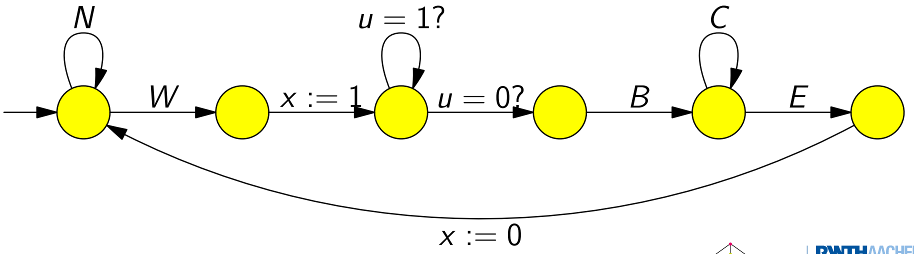
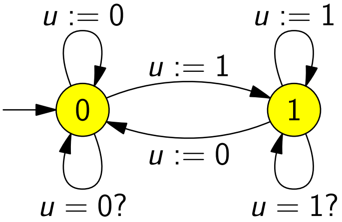
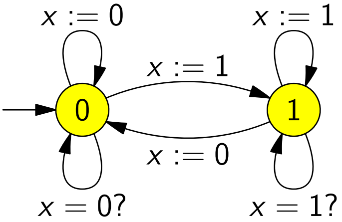
Modellierung des Prozesses P_1:

```
while(true) {
  /* non critical */
  x:=1;
  while(u=1) { }
  /* critical section */
  x:=0;
}
```



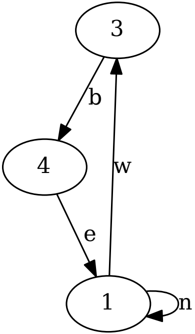
Modellierung der booleschen Variablen (B_u und B_x):

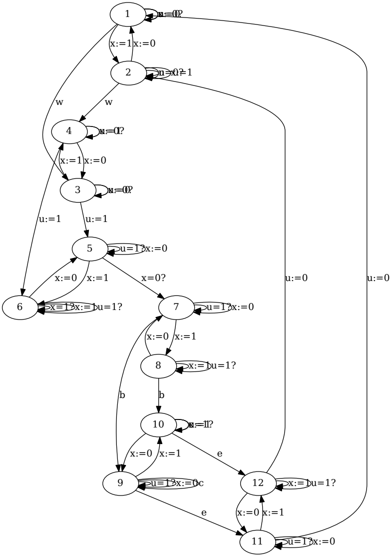




Was passiert, wenn nur P_0 läuft?

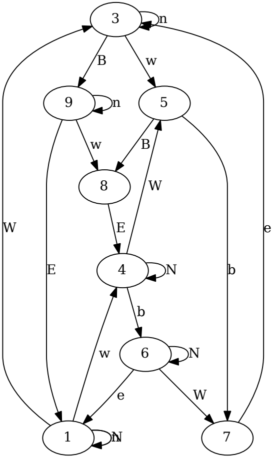
Betrachte $P_0 \circ B_u \circ B_x$.

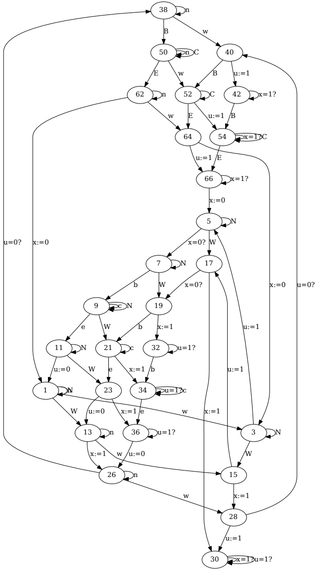


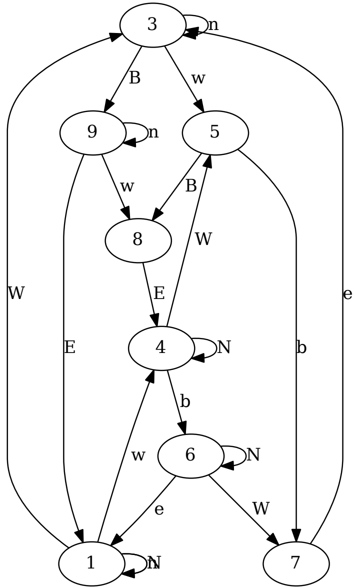


Was passiert, wenn P_0 und P_1 laufen?

Betrachte $(P_0 \sqcup P_1) \circ B_u \circ B_x$.







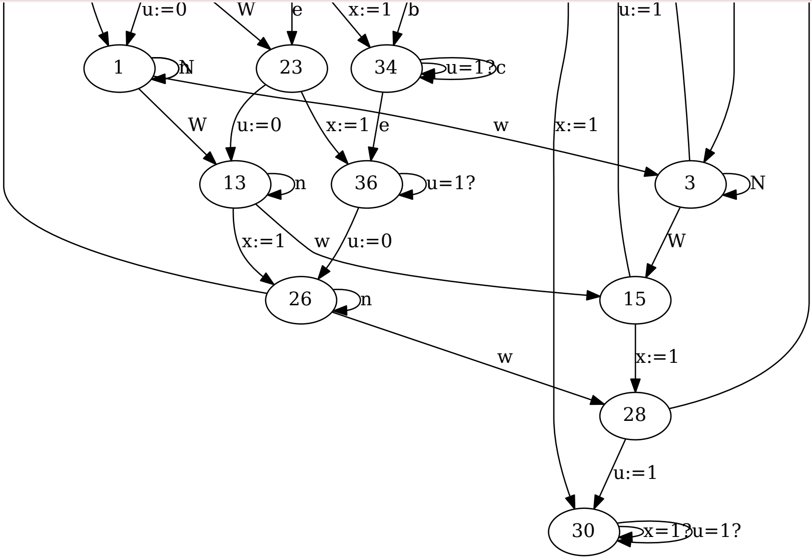
h ist ein Homomorphismus, der alle Symbole außer W, B, E, N, w, b, e und n löscht.

$L' = h(L(M))$ mit
$M = (P_0 \sqcup P_1) \circ B_u \circ B_x$.

$$L' \cap \Sigma^* b(\Sigma \setminus \{e\})^* B\Sigma^* = \emptyset$$

$$L' \cap \Sigma^* B(\Sigma \setminus \{E\})^* b\Sigma^* = \emptyset$$

Also kann sich nur ein Prozeß im kritischen Bereich befinden.



Problem: Deadlock!

Das Verfahren von Peterson

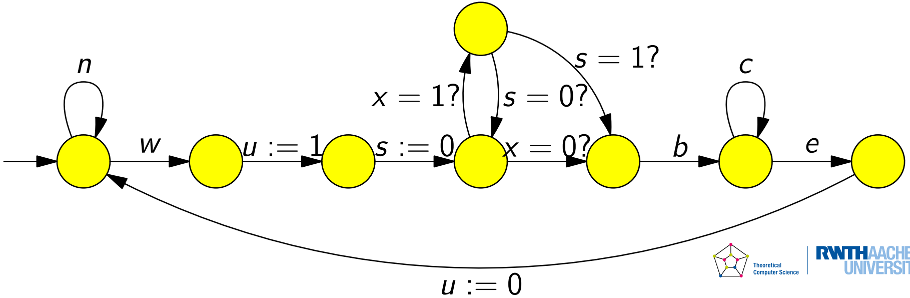
```
while(true) {
  /* non critical */
  u:=1;
  s:=0;
  while(x=1 ∧ s=0) { }
  /* critical section */
  u:=0;
}
```

```
while(true) {
  /* non critical */
  x:=1;
  s:=1;
  while(u=1 ∧ s=1) { }
  /* critical section */
  x:=0;
}
```

Nur ein Programm darf sich in der *critical section* befinden.

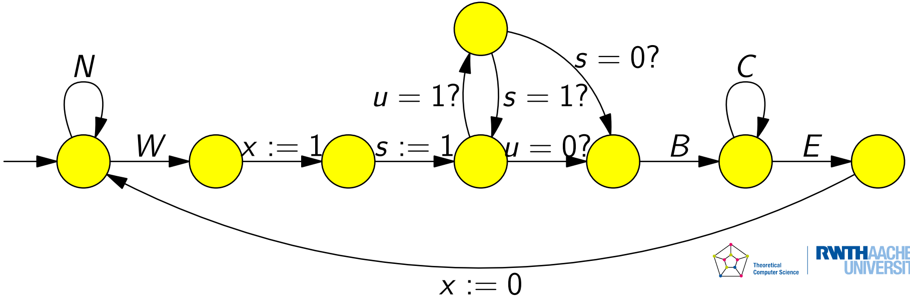
Modellierung des Prozesses P_0:

```
while(true) {
  /* non critical */
  u:=1;
  s:=0;
  while(x=1 ∧ s=0) { }
  /* critical section */
  u:=0;
}
```



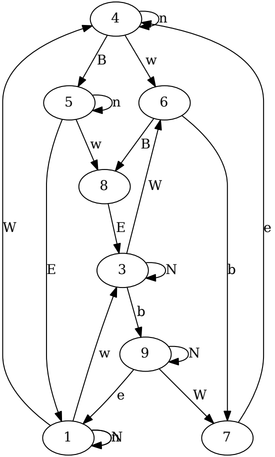
Modellierung des Prozesses P_1:

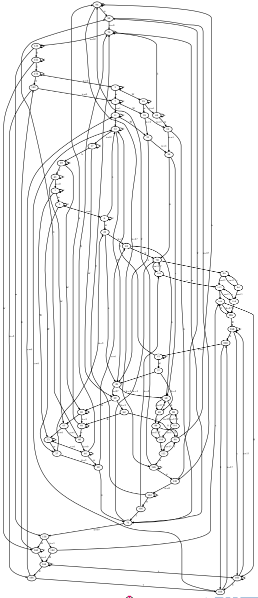
```
while(true) {
  /* non critical */
  x:=1;
  s:=1;
  while(u=1 ∧ s=1) { }
  /* critical section */
  x:=0;
}
```

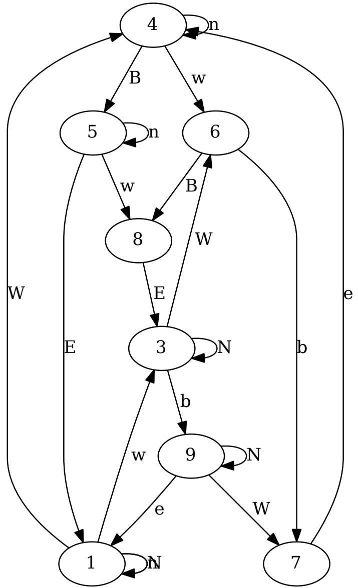


Was passiert, wenn P_0 und P_1 laufen?

Betrachte $(P_0 \sqcup P_1) \circ B_u \circ B_x \circ B_s$.







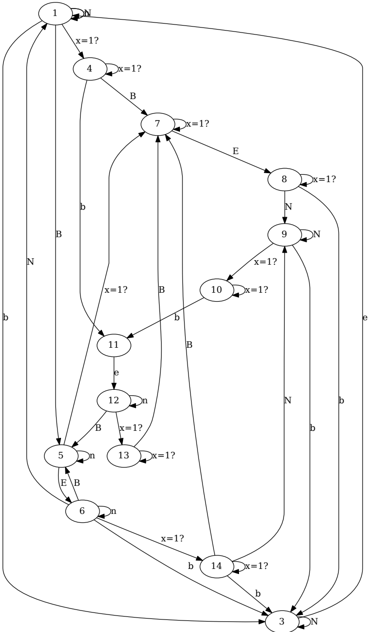
h ist wieder ein Homomorphismus, der alle Symbole außer W, B, E, N, w, b, e und n löscht.

$L' = h(L(M))$ mit
$M = (P_0 \sqcup P_1) \circ B_u \circ B_x \circ B_s$.

$$L' \cap \Sigma^* b(\Sigma \setminus \{e\})^* B\Sigma^* = \emptyset$$

$$L' \cap \Sigma^* B(\Sigma \setminus \{E\})^* b\Sigma^* = \emptyset$$

Also kann sich wieder nur ein Prozeß im kritischen Bereich befinden.



h löschte jetzt alle Symbole außer B, E, N, b, e, n und $x = 1?$.

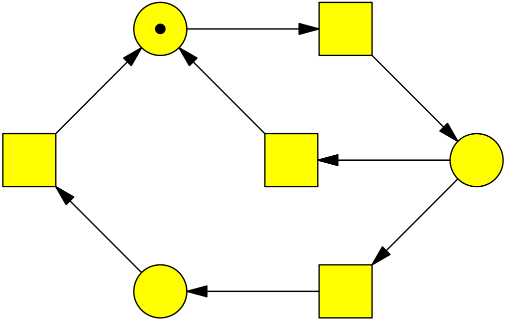
Wir stellen die Frage:

Kann in einem Lauf beliebig of $x = 1?$ gefolgt von B vorkommen, ohne daß b vorkommt?

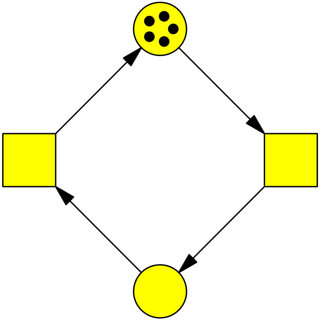
Gibt es einen Kreis, der $x = 1?$ und B enthält, aber nicht b?

P_0 kann nicht verhungern.

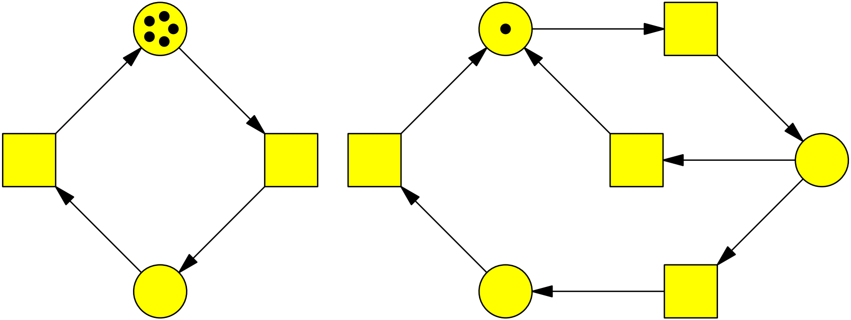
Petrinetze



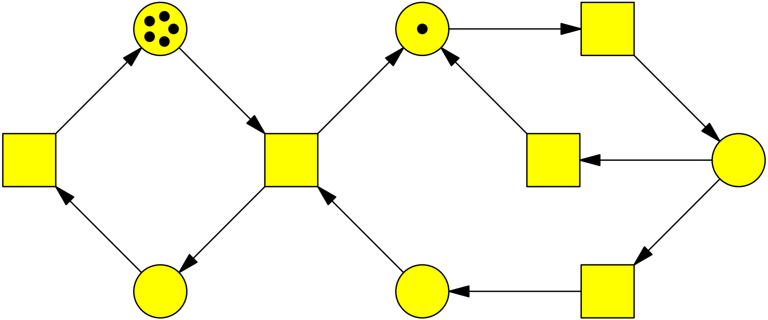
Petrinetze



Petrinetze



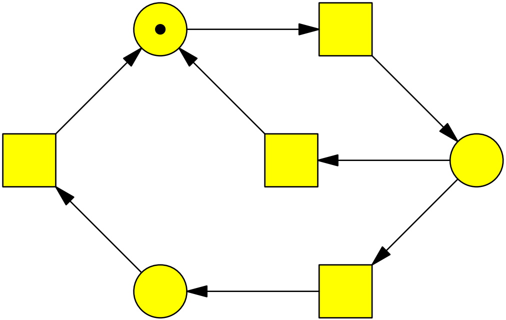
Petrinetze



Definition

Ein *Petrinetz* ist ein gerichteter, bipartiter Graph $N = (P, T, F)$ mit:

1. P, der Menge der Stellen,
2. T, der Menge der Transitionen,
3. $F \subseteq P \times T \cup T \times P$.



Markierungen

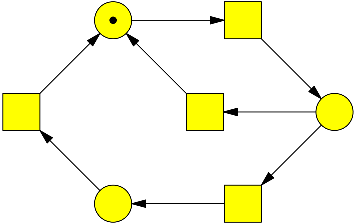
Definition

Es sei $N = (P, T, F)$ ein Petrinetz.

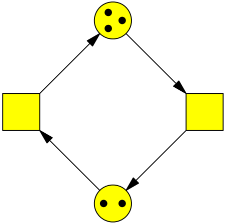
Eine *Markierung* ist eine Funktion $m\colon P \to \mathbf{N_0}$.

Sie ordnet jeder Stelle eine natürliche Zahl zu.

Falls wir die Stellen durch $p_1, \dots, p_n$ ordnen, können wir eine Markierung kurz als Vektor $(m_1, \dots, m_n)$ schreiben.



$(1, 0, 0)$



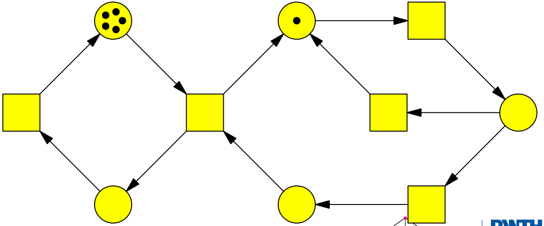
$(3, 2)$

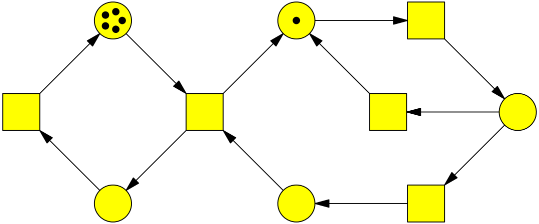
Definition

Es sei $N = (P, T, F)$ ein Petrinetz, $p \in P$, $t \in T$.

1. ${}^\bullet t = \{ p' \in P \mid (p', t) \in F \}$ (*Vorbereich* von t)
2. $t^\bullet = \{ p' \in P \mid (t, p') \in F \}$ (*Nachbereich* von t)
3. ${}^\bullet p = \{ t' \in T \mid (t', p) \in F \}$ (*Vorbereich* von p)
4. $p^\bullet = \{ t' \in T \mid (p, t') \in F \}$ (*Nachbereich* von p)

Erweiterung: $R \subseteq P$, dann $R^\bullet = \bigcup_{r \in R} r^\bullet$.





- t bezüglich m aktiviert, falls $m(p) > 0$ für alle $p \in {}^\bullet t$.
- t_1 und t_2 *in Konflikt* bezüglich m, falls beide aktiviert, aber nur eine schalten kann.
- t_1 und t_2 *nebenläufig*, falls ${}^\bullet t_1 \cap {}^\bullet t_2 = \emptyset$.
- $m > (0, \ldots, 0)$ ist eine *Verklemmung*, falls keine Transition schalten kann.

Die Schaltrelation

Definition

Es seien $N = (P, T, F)$ ein Petrinetz, $t \in T$ und m, m' Markierungen.

Es gilt $m \xrightarrow{t} m'$ gdw.

1. $m(p) > 0$ für alle $p \in {}^\bullet t$
2. $m'(p) = \begin{cases} m(p) - 1 & \text{falls } p \in {}^\bullet t \setminus t^\bullet, \\ m(p) + 1 & \text{falls } p \in t^\bullet \setminus {}^\bullet t, \\ m(p) & \text{sonst} \end{cases}$

Frage: Ist die erste Bedingung redundant?

m' ist von m *erreichbar*, falls

- $m = m'$ oder
- $m \xrightarrow{t} m''$ für ein $t \in T$ und m' ist von m'' erreichbar.

Petrinetze und synchronisierte Produkte

Offensichtlich: Ein Petrinetz kann einen NFA simulieren.

Gegeben seien NFAs M_1, $M_2, \ldots, M_k$.

Dann ist $M = M_1 \circ \cdots \circ M_k$ wieder ein NFA.

Können wir ein Petrinetz für M konstruieren?

Können wir etwas besseres machen?

Petrinetz, dessen Größe die Summe der Größen von M_i ist!

Analyse von Petrinetzen

Erreichbarkeitsbaum:

Gegeben ein Petrinetz und eine Markierung m.

Konstruiere einen Baum (Idee):

1. Die Wurzel besteht aus m.
2. Die Kinder eines Knotens sind die möglichen Folgemarkierungen.
3. (Kinder eines doppelt vorkommenden Knotens weglassen.)

Erreichbarkeit von Markierungen kann so oft leicht nachgewiesen werden.

Beschränktheit kann ebenfalls so nachgewiesen werden.

Es sei $N = (P, T, F)$ ein Petrinetz mit $P = (p_1, \ldots, p_n)$ und $T = (t_1, \ldots, t_m)$.
Definiere die $m \times n$-Matrizen D^-, D^+ und D:

$$D^-_{i,j} = \begin{cases} -1 & \text{falls } p_j \in {}^\bullet t_i \\ 0 & \text{sonst} \end{cases}$$

und

$$D^+_{i,j} = \begin{cases} 1 & \text{falls } p_j \in t_i^\bullet \\ 0 & \text{sonst} \end{cases}$$

$$D = D^- + D^+$$

Theorem

Es sei $N = (P, T, F)$ ein Petrinetz und $m, m' \in \mathbf{N}^n$ Markierungen.

Falls m' von m erreichbar ist, dann gibt es ein $x \in \mathbf{N}^m$ mit

$$m' = m + xD.$$

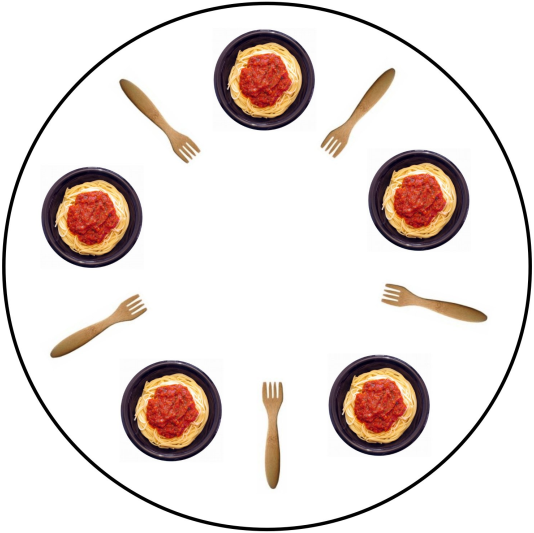
Beweis.

$m' = m + (0, \ldots, 0, 1, 0, \ldots, 0)D$ falls *eine* Transition *einmal* schaltet.

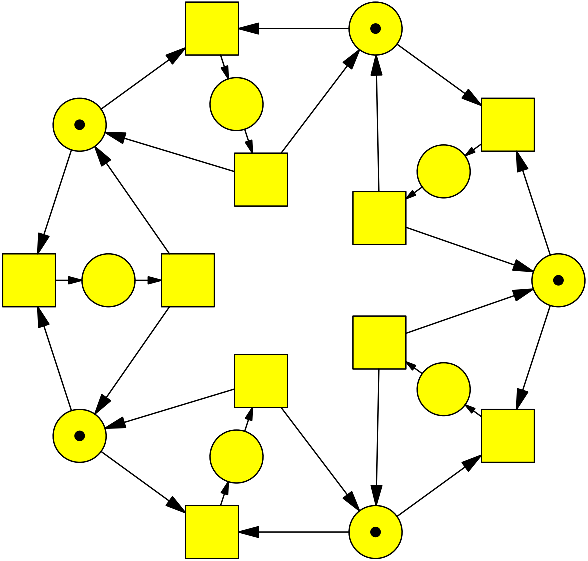
x ergibt sich als Summe solcher Vektoren einer Schaltfolge. □

Auf diese Weise kann oft gezeigt werden, daß eine Markierung nicht erreichbar ist.

Beispiel



Die essenden und denkenden Philosophen.



Denkende und essende Philosophen.