Petri-Netze

S. Gerber

WS 1999/2000

Inhaltsverzeichnis

Literatur (Auswahl):

Baumgarten, B.: *Petri-Netze*, BI-Mannheim, 1990
Peterson, J. L.: *Petri Net Theory and the Modelling of Systems*, Prentice Hall, London, 1981
Reisig, M.: *Petri-Nets*, Springer, Berlin, 1985
Vogler, W.: *Modular Construction and Partial Order Semantics of Petri Nets*, Springer, 1992
Starke, P.H.: *Petri-Netze*, Deutscher Verlag der Wissenschaften, Berlin, 1980

0. Einführung

Petri-Netze wurden als Modelle zur Beschreibung nebenläufiger kommunizierender Prozesse 1962 von C. A. Petri vorgeschlagen. Damit sind erfaßbar dynamische Systeme mit fester Grundstruktur (Betriebssysteme, Produktionsprozesse u.a.). Automaten beschreiben nicht Kausalgefüge von Prozessen, die aus mehreren unabhängigen, nebenläufigen, parallelen Teilprozessen bestehen. Es wird auf einen Gesamtzustand projeziert. Petri-Netz-Theorie erlaubt einheitliche Systembeschreibungen auf verschiedenen Detaillierungsstufen. Dazu werden verschiedene Klassen von Netzen verwendet. (Kausalnetze, Bedingungs-/Ereignis-Systeme, Stellen-/Transitions-Systeme, höhere Petri-Netze). Abstrakte Modelle dynamischer Prozesse auf unterschiedlichen Abstraktionsebenen und unterschiedlicher Detailliertheit mit einheitlicher Sprache. Modell des Informationsflusses in informationsverarbeitenden Systemen, Informationsfluß zwischen parallelen (nebenläufigen) Teilen des Systems muß organisiert und gesteuert werden. Große Systeme sind abhängig vom Abstraktions- und Komplexitätsgrad, nebenläufige Abläufe sind schwieriger zu beherrschen als sequentielle Abläufe, es finden Zustandsänderungen unabhängig voneinander statt. Anwendung von Petri-Netzen zur Modellierung von Systemen, in denen Ereignisse nebenläufig eintreten u.U. mit Beschränkungen (Schaltsysteme, Informationssysteme, Produktionssysteme, Rechtssysteme u.a.).

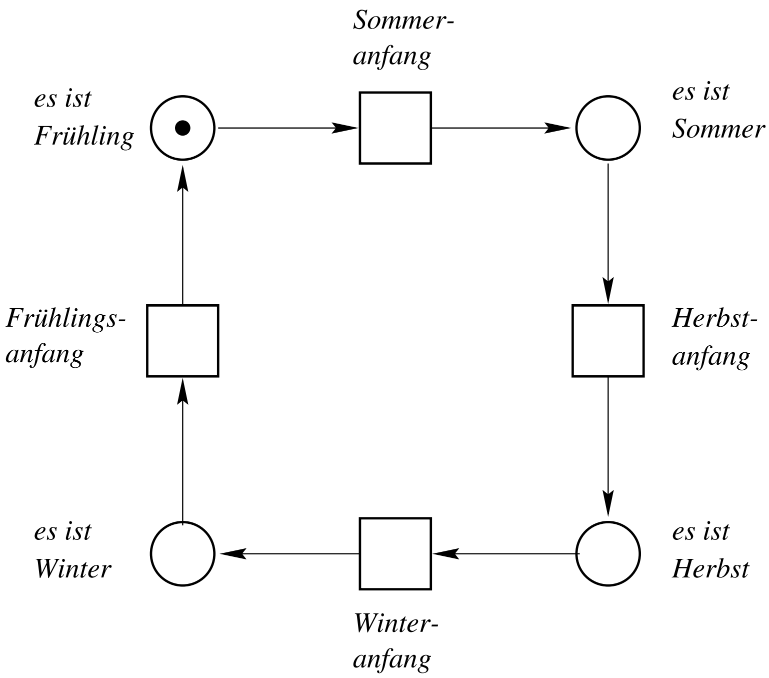
Eigenschaften von Petri-Netz-Modellen:

- Beschreibung von Aufbau, Arbeitsweise und Eigenschaften nichtsequentieller Systeme
- Darstellung kausaler Abhängigkeiten, wobei unabhängige Ereignisse nicht auf Zeitachse projeziert werden
- Einführung einer Nebenläufigkeitsrelation
- Darstellung auf verschiedenen Abstraktionsebenen mit einheitlicher Beschreibungssprache
- Nachweis von Systemeigenschaften und Korrektheitsbeweise.

Arten von Petri-Netzen:

(a) Systeme aus Bedingungen und Ereignissen.

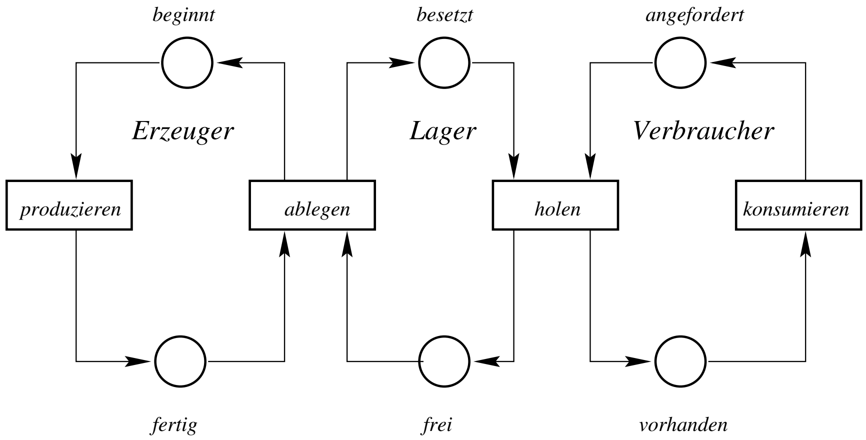
Marken kennzeichnen erfüllte Bedingungen. Beziehungen zwischen Bedingungen (Frühling, Sommer, Herbst und Winter) und Ereignissen (Frühlings-, Sommer-, Herbst-, Winteranfang). Durch den Eintritt des Ereignisses e wird Bedingung b erfüllt. b Nachbedingung von e ($e \to b$). Durch den Eintritt von e wird b unerfüllt. b Vorbedingung von e ($b \to e$). Ein Ereignis kann eintreten, wenn alle seine Vorbedingungen erfüllt sind. Tritt das Ereignis dann ein, werden alle Nachbedingungen erfüllt und alle Vorbedingungen unerfüllt.



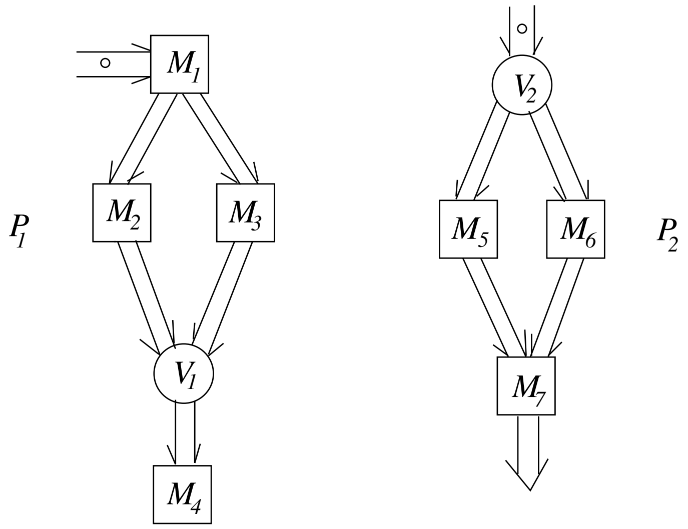
Die vier Jahreszeiten und ihr Wechsel

(b) Systeme mit Stellen und Transitionen.

Oft sind nicht nur Einzelereignisse oder Bedingungen von Interesse, sondern auch Eigenschaften von Objekten (z.B. Anzahl von produzierten Gütern). Wir unterscheiden dann Stellen und Transitionen. Stellen können mehrere Marken tragen (Anzahl von unterscheidbaren Dingen). Wenn eine Transition schaltet, dann wird jeder Vorstelle eine Marke entnommen und jeder Nachstelle eine Marke hinzugefügt wird. (Erzeuger-Verbraucher-System).



(c) Auch individuelle Eigenschaften von Dingen sind von Interesse. Stellen bezeichnen passive Teile. Transitionen bezeichnen aktive Teile. Pfeile charakterisieren Kopplung und Fluß von Objekten, die durch Marken dargestellt werden (Produktionssysteme).



(d) Pfeile können ungerichtet sein. Es werden nur Zusammenhänge beschrieben, ohne daß ein Fluß stattfindet (Handlungsgefüge einer Schachpartie).

Anwendungsbeispiele von Petri-Netzen:

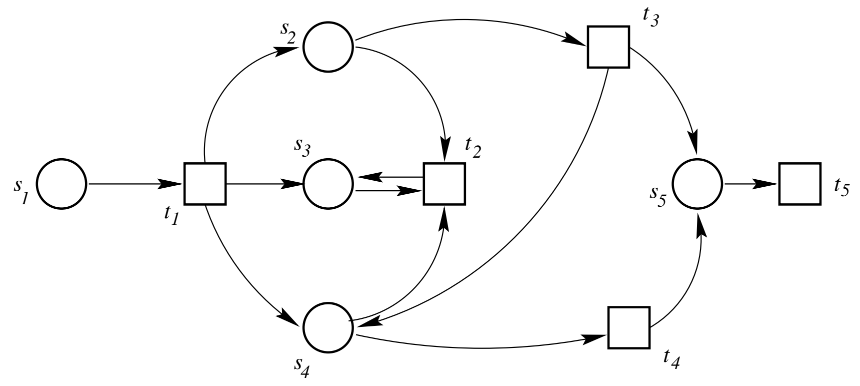
- Schaltlogik, Zugriffsrechte von Prozessen in Betriebssystemen, Nichtsequentielle Programme (Konstruktion von Reihenfolgeentscheidungen mit notwendigen oder gewünschten Anfangsbedingungen).
- Rechenalgorithmen: Anweisungen=Ereignisse, Programmzustände=Bedingungen, für nebenläufige Anweisungen muß keine Reihenfolge spezifiziert werden.
- Systemanalyse (Softwaretechnologie): Netzdarstellung nichtformaler Systeme (Aufgabenstellung) mit Strukturmerkmalen werden zu programmierbaren Systemteilen verfeinert.
- Prädikat/Ereignis-Netze (Systemeigenschaften können berechnet und Korrektheitsbeweise geführt werden).

Verhalten von Petri-Netzen:

Der Petri-Netz Begriff wird für die zugrundeliegende Struktur verwendet, die ausschließlich statische Aspekte beschreibt. Das Verhalten des Petri-Netzes beschreibt die dynamischen Aspekte. Ein *Petri-System* besteht aus einem *Petri-Netz*, einer *Markierung* und einer *Transitionsregel* (Schaltregel).
Elementare Netze sind einfachste netzbasierten Modelle für verteilte Systeme. Als Grundmuster werden zwei Sorten von Elementen und Beziehungen zwischen diesen benutzt, (passive) S-Elemente (Stellen,Plätze) und (aktive) T-Elemente (Transitionen, Ereignisse). Pfeile (Flußrelation) verbinden Elemente unterschiedlicher Sorte (bipartiter Graph).

Netz-Definition:

$N = (S, T, F)$ heißt <u>*Netz*</u>, falls S, T disjunkte (endliche) Mengen (Stellen S als Kreise, Transitionen T als Kästen oder Striche) und $F \subseteq (S \times T) \cup (T \times S)$ Flußrelation von N.
S_N, T_N, F_N bezeichnen die Bestandteile von N.
${}^{\cdot}x = \{y \ / \ y \ F_N \ x\}$ Vorbereich von x in N, $x^{\cdot} = \{y \ / \ x \ F_N \ y\}$ Nachbereich von x in N.
Für $X \subseteq S \cup T$ sei ${}^{\cdot}X = \bigcup\limits_{x \in X} {}^{\cdot}x$, $X^{\cdot} = \bigcup\limits_{x \in X} x^{\cdot}$ mit $x \in {}^{\cdot} y \Leftrightarrow y \in x^{\cdot}$ für alle $x, y \in S \cup T$
.



$S=\{s_1,\ldots,s_5\}$

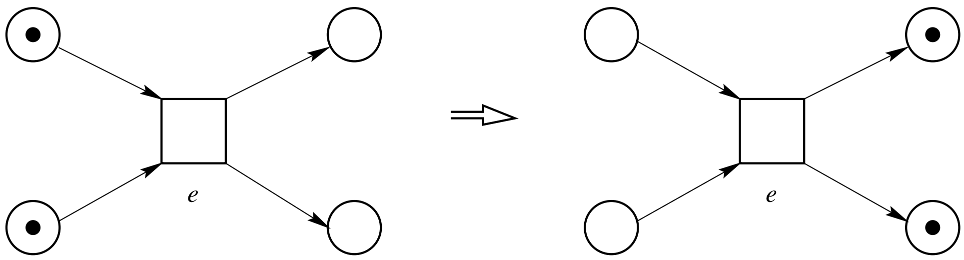
$T=\{t_1,\ldots,t_5\}$

$f=\{(s_1,t_1),(t_1,s_2),(t_1,s_3),(t_1,s_4),(s_3,t_2),(t_2,s_3),(s_2,t_2),(s_2,t_3),(s_4,t_2),(s_4,t_4),(t_3,s_4),(t_3,s_5),(t_4,s_5),(s_5,t_5)\}$

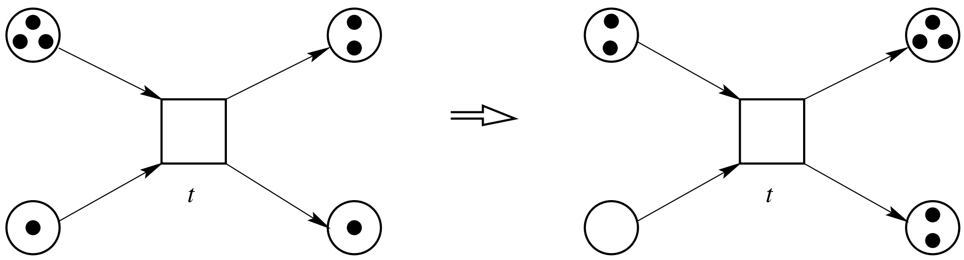
Graphische Darstellung eines Netzes

Andere Schreibweisen:

$F : (S \times T) \cup (T \times S) \rightarrow \{0, 1\}$ als Funktion mit $F(s,\, t) = 1 \leftrightarrow s$ Vorbedingung von t bzw. $Ft = \{s/F(s,t) \succ 0\} =^{\cdot} t$ Menge der Vorbedingungen von t.
$F(t,\, s) = 1 \leftrightarrow t$ schafft s bzw. $tF = \{s\ /\ F(t,s) \succ 0\} = t^{\cdot}$ Menge der Nachbedingungen von t oder pre, $post : T \rightarrow 2^P$ mit $F(s,t) = 1 \leftrightarrow s \in pre(t)$ bzw. $F(t,s) = 1 \leftrightarrow s \in post(t)$ oder $Post \subseteq T \times S$, $Pre \subseteq S \times T$ mit $F(s,t) = 1 \leftrightarrow (s,t) \in Pre$ bzw. $F(t,s) = 1 \leftrightarrow (t,s) \in Post$.
Durch das Netz N werden zunächst nur die statischen Verhältnisse zwischen Stellen (Bedingungen) und Transitionen (Ereignisse) beschrieben, d.h. welche Ereignisse hängen von welchen Bedingungen ab und welche Bedingungen werden durch welche Ereignisse beeinflußt. Wann die Ereignisse als aktive Elemente eintreten, ist damit noch nicht festgelegt. Dies geschieht durch die Transitionsregel:



Das Stattfinden eines Ereignisses e



Das Schalten einer Transition t

Allgemeine Transitionsregel:

Das Ereignis t kann höchstens dann eintreten,

(a) wenn alle Vorbedingungen s mit $F(s,t) = 1$ erfüllt sind,

(b) bei Eintritt des Ereignises t werden alle Vorbedingungen von t beendet (unerfüllt) und alle Nachbedingungen von t erfüllt.

Unsicherheit: Wenn Nachbedingungen bereits erfüllt sind, wird Wirkung des Ereignisses verfälscht.

Sichere Transitionsregel:

(c) Das Ereignis t kann nur dann eintreten, wenn keine Nachbedingung s mit $F(t, s) = 1$ erfüllt ist.

Netze bei denen die Forderung für die sichere Transitionsregel immer erfüllt ist, werden selbst als sicher bezeichnet. Wann das Ereignis (Transition) tatsächlich eintritt (schaltet) hängt von dem konkreten Prozeßablauf ab, der modelliert werden soll.

Markierung:
Die konkrete Situation, in der sich ein Petri-Netz befindet, wird durch die Markierung $m : S \to Z$ beschrieben, bei der jeder Stelle eine natürliche Zahl oder allgemein Objekt aus Z zugeordnet wird. Bei endlicher Stellenmenge S kann m durch einen Vektor über Z beschrieben werden.

Je nachdem welche Prozeßabläufe modelliert werden sollen, unterscheidet man verschiedene Klassen von Netzen:

- **Bedingungs- / Ereignis-Netze** (s-Elemente markiert (erfüllt) oder unmarkiert (nicht erfüllt))
- **Stellen- / Transitions-Netze** (s-Elemente können mehrere nicht unterscheidbare Marken tragen, Markierung mit natürlichen Zahlen)
- **Individual-Netze** (s-Elemente können mehrere unterscheidbare Objekte tragen, Markierung mit Individuen)

1. Grundbegriffe

1.1. Bedingungen und Ereignisse

Sei $N = (B, E, F)$ mit B, E nichtleere disjunkte Mengen und $F \subseteq (B \times E) \cup (E \times B)$. Dann heißt

1. $c \subseteq B$ *Fall* (case),
2. $e \in E$ *aktiviert* unter c (c-aktiviert) gdw. ${}^{\cdot}e \subseteq c$ (pre(e)$\subseteq$ c) und $e^{\cdot} \subseteq B \backslash c$ $(c \cap post(e) = \emptyset)$
3. c' *Folgefall* von c unter e $(c \rightarrow_e c')$ gdw. $c' = (c \backslash {}^{\cdot}e) \cup e^{\cdot}$. ($c'$ entsteht durch Eintritt von e im Fall c.)

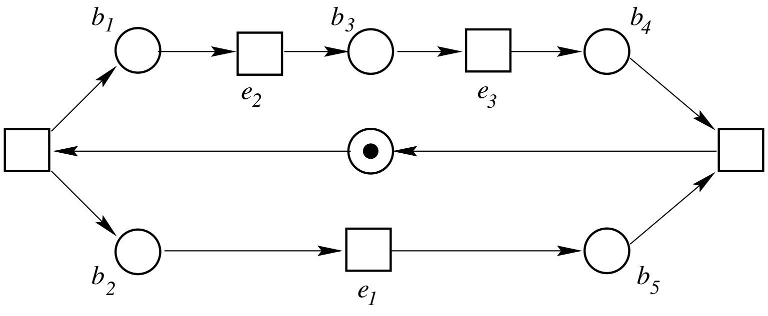
e kann im Fall c eintreten, wenn alle Vorbedingungen von e erfüllt, d.h. zu c gehören, und alle Nachbedingungen von e nicht erfüllt sind, d.h. nicht zu c gehören. Wenn e eintritt, werden alle Vorbedingungen von e unerfüllt und alle Nachbedingungen von e erfüllt. Kontaktsituation im Fall c: ${}^{\cdot}e \subseteq c$ und $e^{\cdot} \cap c \neq \emptyset$. e kann nicht eintreten, weil Nachbedingungen von e erfüllt sind. Eintritt von Ereignissen führt Fall in einen neuen Fall über.

Schritte:
Eine Menge von Ereignissen $G \subseteq E$ kann *eintreten* (hat *Konzession*), wenn alle Ereignisse von G aktiviert und paarweise disjunkten Vor- und Nachbereich haben.

1. G *unabhängig* (*nebenläufig*), falls für alle $e_1 \neq e_2 \in G \rightarrow {}^{\cdot} e_1 \cap {}^{\cdot} e_2 = \emptyset = e_1^{\cdot} \cap e_2^{\cdot}$
2. Für $c, c' \subseteq B$, $G \subseteq E$ unabhängig heißt G *Schritt* von c nach c', falls alle $e \in G$ c-aktiviert sind und $c' = (c \backslash {}^{\cdot}G) \cup G^{\cdot}$.

   G überführt c nach $c'(c \rightarrow_G c')$.

Schritte in Netzen



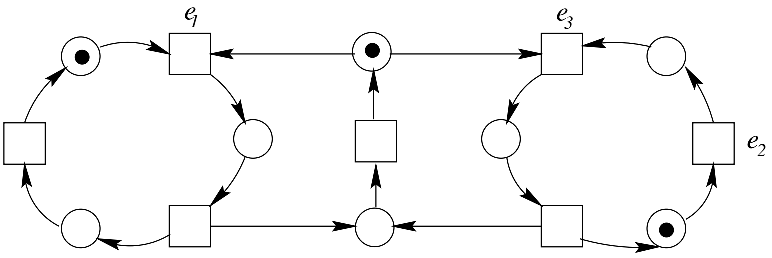
$\{e_1,e_2\}$ ist ein Schritt von $\{b_1,b_2\}$ nach $\{b_3,b_5\}$.

$\{e_1,e_3\}$ ist ein Schritt von $\{b_2,b_3\}$ nach $\{b_4,b_5\}$.

Bemerkung 1:
$G \subseteq E$ unabhängig und $c, c' \subseteq B : c \rightarrow_G c'$ gdw. $c \backslash c' = {}^{\cdot}G$ und $c' \backslash c = G^{\cdot}$.

Bemerkung 2:
$c, c' \subseteq B$, G endlicher Schritt $c \rightarrow_G c'$ mit $G = \{e_1, ..., e_n\}$ (Anordnung), dann existieren Fälle $c_0, ..., c_n$ mit $c = c_0 \rightarrow_{e_1} c_1 \rightarrow_{e_2} ...c_n = c'$, d.h. $c_{i-1} \rightarrow_{e_i} c_i$ für $1 \leq i \leq n$.

Konfliktsituation:
e_1 vor e_2 dann e_1, e_3 *konfliktfrei*, e_2 vor e_1 dann e_1, e_3 im Konflikt. Wenn Reihenfolge e_1, e_2 nicht festgelegt, dann *konfuse* Situation. e_1, e_3 im Konflikt, da gemeinsame Vor- und Nachbedingungen.



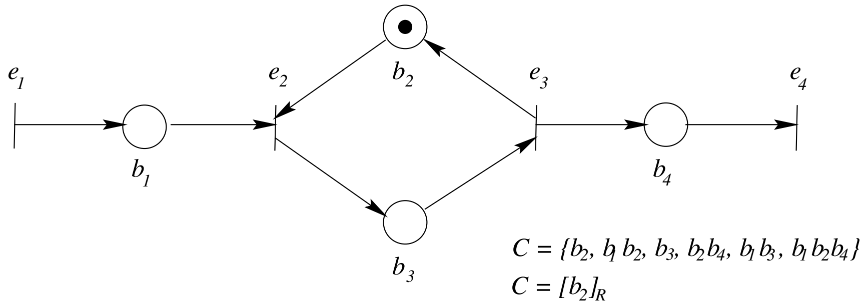
Erreichbarkeit:

c' <u>*direkt erreichbar*</u> von $c(c \to c')$, falls ein Schritt G existiert mit $c \to_G c'$.
c' <u>*erreichbar*</u> von $c(c \to *c')$, falls Schritte G_i exisitieren mit $c = c_0 \to_{G_1} c_1 \to_{G_2} \cdots \to_{G_n} c_n = c'$ (Äquivalenzklassen bzgl. $\to^*$) .
Für $R = (\to \cup \to^{-1})^*$ heißt $(c, c') \in R$ erreichbar und $[c]_R$ durch c repräsentierte Äquivalenzklasse.

Sei $N = (B, E, F)$ ein Netz aus Bedingungen B und Ereignissen E und $G \subseteq E$ ein Schritt mit $c \in C$, $c \to_G c' \Rightarrow c' \in C$ und $c' \in C$, $c \to_G c' \Rightarrow c \in C$.
N enthalte keine isolierten Bedingungen b bzw. Ereignisse e mit ${}^{\cdot}b \cup b^{\cdot} = \emptyset$ bzw. ${}^{\cdot}e \cup e^{\cdot} = \emptyset$.
F heißt <u>*schlingenfrei*</u>, wenn für kein Bedingungs-Ereignis-Paar $(b, e) \in F$ und $(e, b) \in F$ gilt.
N heißt <u>*schlicht*</u>, wenn für alle $b_1, b_2 \in B$ und $e_1, e_2 \in E$ gilt ${}^{\cdot}b_1 = {}^{\cdot}b_2 \wedge b_1^{\cdot} = b_2^{\cdot} \Rightarrow b_1 = b_2$ und ${}^{\cdot}e_1 = {}^{\cdot}e_2 \wedge e_1^{\cdot} = e_2^{\cdot} \Rightarrow e_1 = e_2$, d.h. es existieren keine ununterscheidbaren Bedingungen und keine ununterscheidbaren Ereignisse.

B/E-System: $\sum = (B, E, F, C)$ heißt <u>*B/E-System*</u>, falls

1. (B, E, F) schlichtes Netz ohne isolierte Elemente, B$\cup E \neq \emptyset$
2. $C \subseteq 2^B$ ist Äquivalenzklasse von R ($C = [c]_R$ wird durch ein Element $c \in C$ vollständig bestimmt.)
3. für alle $e \in E$ existiert $c \in C$,wo e c-aktiviert ist.



Eigenschaften:

(a) $B \neq \emptyset$, $E \neq \emptyset$, $F \neq \emptyset$, $|C| \succ 1$

(b) $c \in C$, $c' \subseteq B$, $G \subseteq E : c \to_G c'$ oder $c' \to_G c \Rightarrow c' \in C$

(c) für alle $b \in B$ existiert $c, c' \in C$ mit $b \in c$ und $b \notin c'$

(d) $\sum$ rein, d.h. F schlingenfrei

(e) $c_1 \to c_2$ gdw. exist. $e \in E$ mit $c_1 \to_e c_2$.

Eigenschaften:

(a) $\sum$ heißt <u>*zyklisch*</u> gdw. $(c, c') \in \rightarrow^*$für alle $c, c' \in C$ (alle Fälle sind reproduzierbar.)

(b) $\sum$ heißt <u>*lebendig*</u> gdw. für alle $c \in C, e \in E$ existiert $c' \in C$ mit $(c, c') \in \rightarrow^*$ und e ist c'-aktiviert. (Ereignis kann immer wieder eintreten.)

(c) $\sum \sim \sum'$ <u>*äquivalent*</u> gdw. existieren Bijektionen φ_c, φ_e mit $\varphi_c : C \rightarrow C'$, $\varphi_e : E \rightarrow E'$, für alle $G \subseteq E$ Schritt und $c, c' \in C$ gilt: $c \rightarrow_G c'$ gdw. $\varphi_c(c) \rightarrow_{\varphi_e(G)} \varphi_c(c')$.

Satz: $\sum$ System, $c_i \in C$ Fälle, $G_i \subseteq E$ Schritte

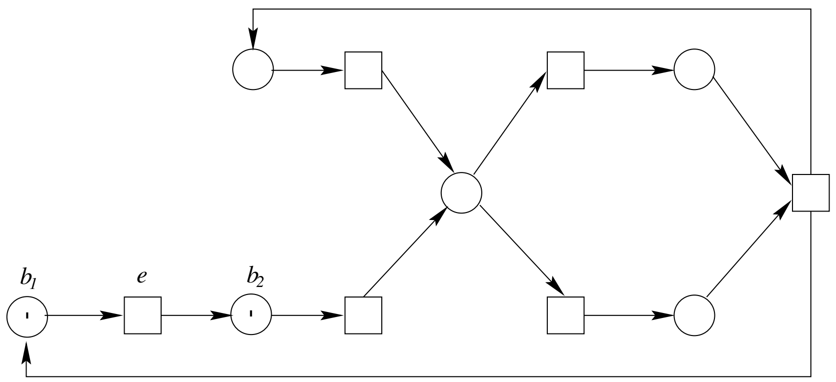
(a) $c_1 \rightarrow_{G_1} c_2 \rightarrow_{G_2} c_3 \Rightarrow G_1 \cap G_2 = \emptyset$,

(b) $G_1 \cap G_2 = \emptyset$, $c_1 \rightarrow_{G_1 \cup G_2} c_3 \Rightarrow$ existiert $c_2 = (c_1 \setminus {}^{\cdot}G_1) \cup G_1^{\cdot}$ mit $c_1 \rightarrow_{G_1} c_2 \rightarrow_{G_2} c_3$.

Eigenschaften:

1. $\sum$ zyklisch, $c \in C \Rightarrow C = [c]_{\rightarrow^*} = \{c'/(c, c') \in \rightarrow^*\}$

2. $\sum$ zyklisch $\Rightarrow \sum$ lebendig
Bemerkung: Aus $\sum$ lebendig folgt nicht $\sum$ zyklisch (Im Beispiel tritt $c = \{b_1, b_2\}$ nicht wieder ein. e kann immer wieder aktiviert werden.)



lebendiges, azyklisches BE-System

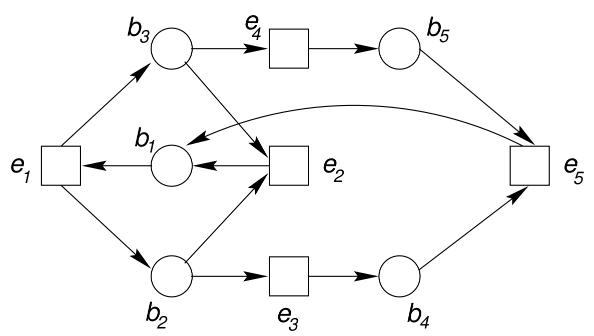
3. $\sum \sim \sum' \Rightarrow \sum$ zyklisch (lebendig) gdw. $\sum'$ zyklisch (lebendig).

4. Für $|c| = 1$ (einelementige Fälle) ist $\sum \sim \sum'$ gdw. $\sum$ isomorph $\sum'$ (Für einelementige Fälle sind φ_c, φ_e Bijektionen, gleiche Anzahl von Bedingungen.)

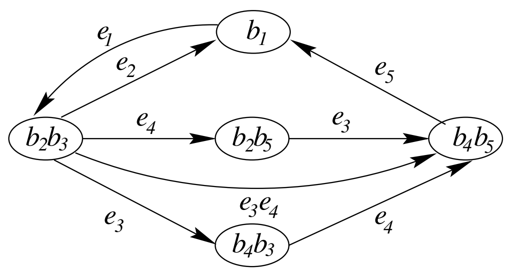
Definition: Fallgraph

$\sum = (B, E, F, C)$ System, $\Gamma \subseteq 2^E$ Schrittmenge von $\sum$.
$\Phi_{\sum} = (C, K)$ heißt *Fallgraph* von $\sum$, falls
$K = \{(c, G, c')|\ c \rightarrow_G c'$ mit $c, c' \in C$ und $G \in \Gamma\}$.

BE-System



$C = \{b_1, b_2b_3, b_2b_5, b_3b_4, b_4b_5\}$



Fallgraph

Satz:

1. System $\sum$ zyklisch gdw. Fallgraph $\Phi_{\sum}$ stark zusammenhängend.

2. System $\sum$ lebendig gdw. für alle $c_0 \in C, e_n \in E$ existieren c_i, e_i $(1 \leq i < n), c_n$ mit $c_0 \rightarrow_{e_1} c_1 \rightarrow_{e_2} ... \rightarrow_{e_n} c_n$ (e_i ist c_{i-1} - aktiviert).

3. $\sum_1 \sim \sum_2$ gdw. $\Phi_{\sum_1}$ isomorph $\Phi_{\sum_2}$.

1.2 Prozesse und Netze

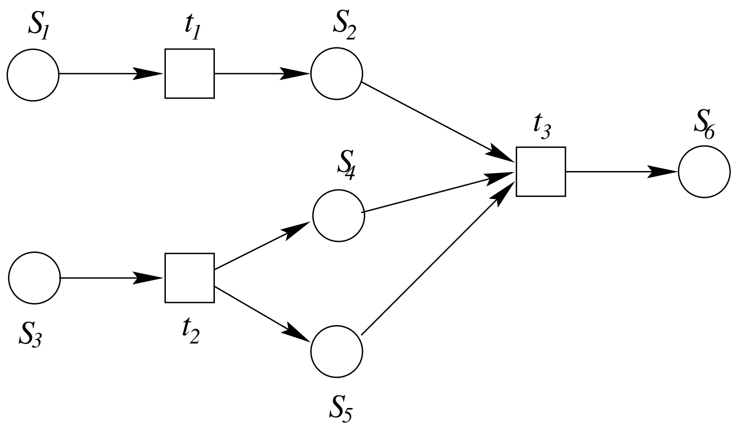
Prozesse können in Systemen ablaufen. Prozeßprotokolle beschreiben Ereignisfolgen, die in Prozessen eintreten können. Prozeß als Gesamtheit möglicher Protokolle. Durch Fallgraphen ungenügend beschrieben, weil Nebenläufigkeit nicht explizit darstellbar.
Wichtig: Darstellung der kausalen Abhängigkeiten von Ereignissen. Ziel, dies selbst als Netz darzustellen. Dabei Wiederholung von Ereignissen und Bedingungen möglich. Da Protokolle eindeutig sind, müssen Bedingungen unverzweigt sein, sonst verschiedene Möglichkeiten. Dazu werden Halbordnungen verwendet. Halbgeordnete Netze (Kausalnetze) sind kreisfreie Netze mit unverzweigten Bedingungen.

Definition: Kausalnetz
$K = (S, T, F)$ heißt *Kausalnetz*, falls

1. $(a, b) \in F^+ \Rightarrow (b, a) \notin F^+$ für alle $a, b \in S \cup T$
   (F^+ antisymmetrische Ordnungsrelation)
2. $|{}^{\cdot}s| \leq 1$ und $|s^{\cdot}| \leq 1$ für alle $s \in S$ (S unverzweigt).

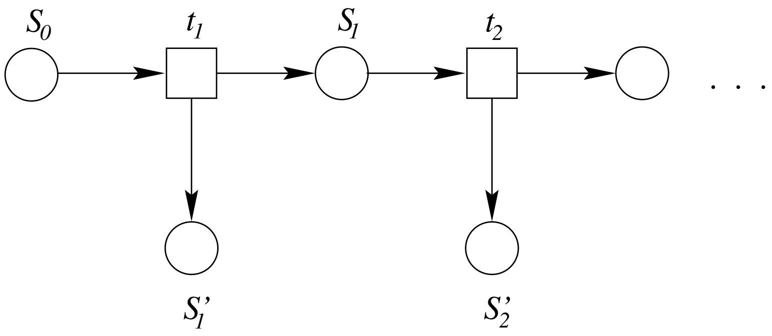
Bemerkung: F^+ ist Ordnungsrelation $\prec$ auf $S \cup T$ ($a \prec b :=: (a, b) \in F^+$).
Vergleichbarkeit $\diamondsuit$: $a \diamondsuit\, b \Leftrightarrow a \prec b$ oder $a \succ b$ oder $a = b$
Nebenläufigkeit $\natural$: $a \natural b \Leftrightarrow a \not\diamondsuit\, b$ oder $a = b$
Restklassenbildung: $(S \cup T)|_{\diamondsuit} = L$ *Linien*, $(S \cup T)|_{\natural} = D$ *Schnitte*, $(S \cup T)|_{\natural} \cap S = I$ *Scheiben*

Beispiel: Linien $\{s_3,t_2,s_4,t_3,s_6\}$, $\{s_3,t_2,s_5,t_3,s_6\}$, $\{s_1,t_1,s_2,t_3,s_6\}$, Schnitte $\{t_1,s_4,s_5\}$, $\{s_1,t_2\}$,$\{t_1,s_3\}$,$\{t_2,s_2\}$,$\{t_1,t_2\}$, Scheiben $\{s_1,s_3\}$, $\{s_2,s_4,s_5\}$, $\{s_1,s_4,s_5\}$, $\{s_2,s_3\}$, $\{s_6\}$



Ein Kausalnetz heißt *beschränkt*, wenn ein n existiert mit $| \; l \; | \leq n$ für alle Linien $l \in (S \cup T) \;|_{\diamondsuit}$.
Ein Kausalnetz heißt *dicht*, wenn $l \cap d \neq \emptyset$ für alle Linien $l \in (S \cup T) \;|_{\diamondsuit}$ und alle Schnitte $d \in (S \cup T) \;|_{\natural}$.

Bemerkung: Jedes beschränkte, nichtleere Kausalnetz ist dicht.



Kontaktfreies System

$\sum = (B, E, F, C)$ heißt <u>*kontaktfrei*</u>, falls für alle $e \in E$ und $c \in C$ gilt:
${}^\cdot e \subseteq c \Rightarrow e^\cdot \subseteq B \backslash c$ und $e^\cdot \subseteq c \Rightarrow {}^\cdot e \subseteq B \backslash c$.

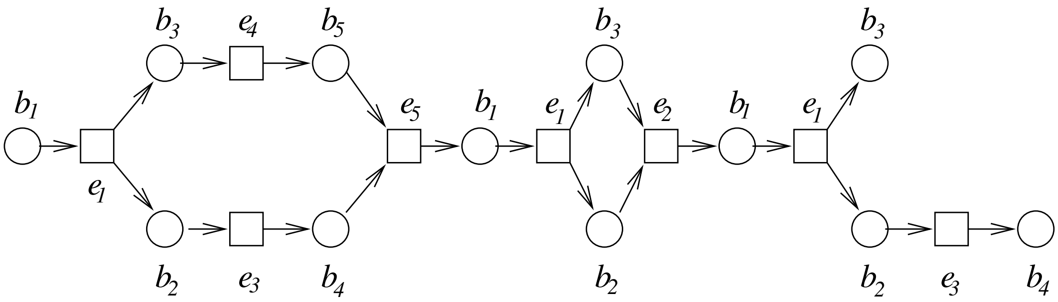
Prozeß

Sei K beschränktes Kausalnetz und $\sum$ kontaktfreies System.
$\pi : K \to \sum$ heißt <u>*Prozeß*</u> auf $\sum$, falls für alle Scheiben $i \in S \cup T|_\natural$ und $t \in T$ gilt:

(a) $\pi|_i$ injektiv und $\pi(i) \in C$ und

(b) $\pi({}^\cdot t) = {}^\cdot \pi(t)$ und $\pi(t^\cdot) = \pi(t)^\cdot$.

Beispiel: Prozeß zum BE-System Seite 13



Bemerkung: Die Beschränktheit von K garantiert, daß Elemente in Linien Kausalketten entsprechen. Ein Schnitt ist eine Momentaufnahme des Prozesses durch äußeren Beobachter. Dichtheit von K garantiert, daß jede Kausalkette in jeder Momentaufnahme (Schnitt) vertreten ist.

Eigenschaften von Prozessen: $\pi : K \to \sum$

(a) $\pi\ (S) \subseteq B$, $\pi\ (T) \subseteq E$ (π sortenrein)

(b) $(x, y) \in F \Rightarrow (\pi(x), \pi(y))$ (π bzgl. F homomorph, flußtreu)

(c) $\pi(x) = \pi(y) \Rightarrow x \diamondsuit y$ vergleichbar (Ereignisse und Bedingungen zu sich selbst nicht nebenläufig)

(d) $t \in T \Rightarrow {}^{\cdot}t \neq \emptyset$ und $t^{\cdot} \neq \emptyset$ (Ereignisse haben Ursache und Wirkung)

(e) π injektiv auf $(S \cup T)|_{\natural}$

(f) $t \natural t'$ für alle $t, t' \in T_0 \subseteq T \Rightarrow$ existieren $c, c' \in C$ mit $c \rightarrow_{\pi(T_0)} c'$ (Schritt).

Prozeßisomorphie:

$\pi_1 \simeq \pi_2$ gdw. existiert φ mit $K_1 \simeq_{\varphi} K_2$ und $\pi_1(x) = \pi_2(\varphi(x))$ für alle $x \in K_1$
(Bildung von Äquivalenzklassen als Prozesse).

Satz:
Sei $\sum_i$ kontaktfrei und Π_i Prozeßmenge von $\sum_i$ dann gilt: $\Pi_1 = \Pi_2$ gdw. $\sum_1 = \sum_2$.
(Kontaktfreie Systeme sind durch ihre Prozeßmengen eindeutig bestimmt.)

Bemerkung: Wenn $\Pi : K \rightarrow \sum$ Prozeß , dann Max K, Min $K \subseteq I$ (Scheibe).

Komposition (Verkettung) von Prozessen

Seien $\Pi_i : K_i \rightarrow \sum$ beschränkte Prozesse und K_i disjunkt mit $\Pi_1(Max\ K_1) = \Pi_2(Min\ K_2)$ (d.h. Π_1 endet mit Fall, mit dem Π_2 beginnt), dann heißt
$\Pi : K_1 \cup K_2 \rightarrow \sum$ mit $\Pi(k) = \Pi_i(k)$ falls $k \in K_i$ <u>*Komposition*</u> von Π_1 mit Π_2
($\Pi = \Pi_1 \circ \Pi_2$).

Lemma:
Wenn $\Pi = \Pi_1 \circ \Pi_2$ dann existiert eine Scheibe i (Trennscheibe von Π_1 und Π_2) mit $\Pi|_{i^-} = \Pi_1$ und $\Pi|_{i^+} = \Pi_2$.
(wo $i^+ = \{k \in K |\ \{k\} > i\}$ und $i^- = \{k \in K |\ \{k\} \leq i\}$)

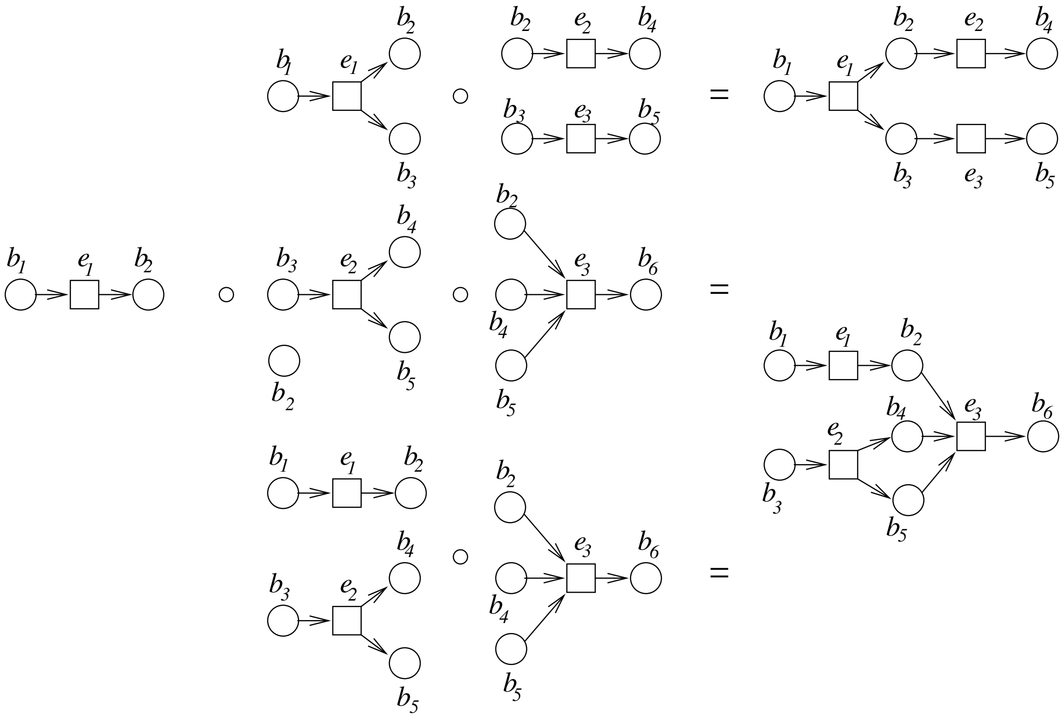
Eigenschaften:

(a) $\Pi_1 \circ (\Pi_2 \circ \Pi_3) = (\Pi_1 \circ \Pi_2) \circ \Pi_3$ (Assoziativität)

(b) $\Pi : K \to \sum$, i Scheibe, dann sind $\Pi^- = \Pi|_{i^-}$, $\Pi^+ = \Pi|_{i^+}$ Prozesse und es gilt $\Pi = \Pi^- \circ \Pi^+$

(c) $S =$ Max $K \cup$ Min K (elementarer Prozeß) gdw. $\Pi($ Min $K) \to_{\Pi(T)} \Pi($Max $K)$ Schritt in $\sum$ gdw. $t_1 \natural\, t_2$ für alle $t_i \in T$.

Beispiel: Verkettung von Prozessen



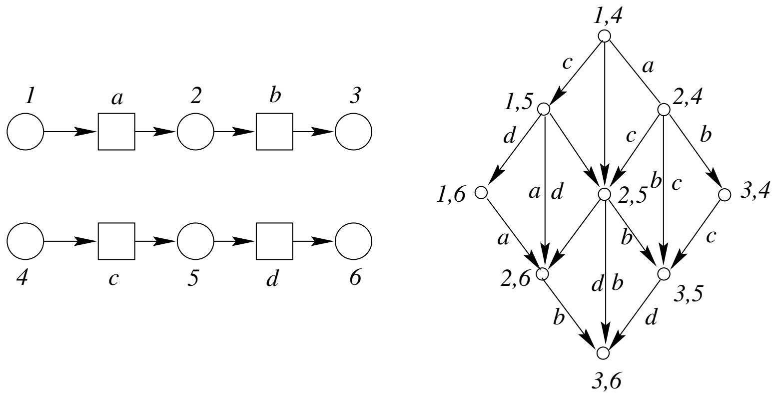
Satz:
Wenn Π beschränkt, dann existieren n Elementarprozesse Π_i mit $\Pi = \Pi_1 \circ \Pi_2 \circ \ldots \circ \Pi_n$.

Zusammenhang zu Fallgraphen:

Wege in Fallgraphen:
$\Pi : K \to \sum$ elementar gdw. existiert genau eine Kante (c, G, c') im Fallgraph Φ von $\sum$ mit $\Pi($Min $K) = c$, $\Pi($Max $K) = c'$, $\Pi(T) = G$, d.h. $(\Pi($Min $K) \to_{\Pi(T)} \Pi($Max $K))$, elementare Prozesse Π entsprechen Kanten k_Π im Fallgraphen. (Π Prozeß von k_Π, k_Π Kante von Π).

Erweiterung auf Wege:
$w = k_1 \ldots k_n$ Weg in Fallgraph Φ von $\sum$, $\Pi_w = \Pi_{k_1} \circ \Pi_{k_2} \circ \ldots \circ \Pi_{k_n}$ Prozeß von w , w Weg von Π_w .

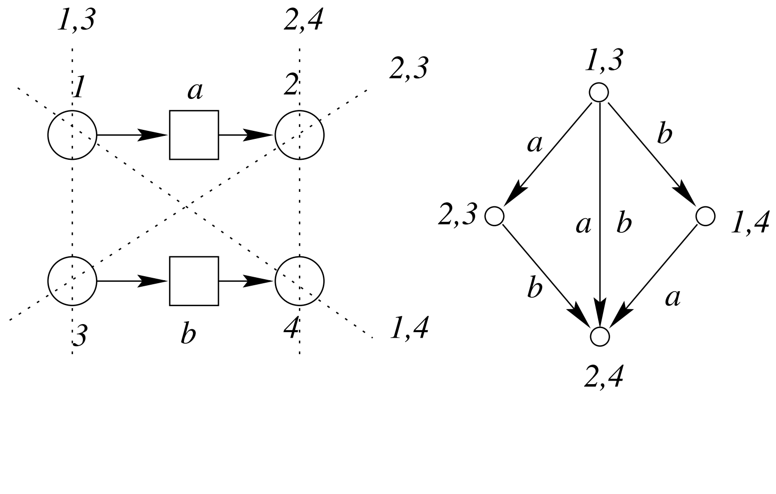
Bemerkung: Zu jedem Weg im Fallgraph gibt es genau einen Prozeß. Es kann aber mehrere Wege eines Prozesses geben.



Zerlegung (Verschmelzung) von Wegen:
$u = (c_1, G_1, c_3)$, $v = (c_3, G_2, c_2)$, $w = (c_1, G_1 \cup G_2, c_2)$ Kanten im Fallgraphen,
uv heißt *Zerlegung* von w , w heißt *Verschmelzung* von uv,
v heißt *Umordnung* von u, falls existieren Wege $w_1, ..., w_4$ mit $u = w_1 w_2 w_3$, $v = w_1 w_4 w_3$,
$(u_1 ... u_n)$ heißt *Umordnungsfolge* falls u_{i+1} Umordnung von u_i .

Beispiele:

[(1,3) a (2,3)][(2,3) b (2,4)] Zerlegung von [(1,3) (a,b) (2,4)]
[(1,3) (a,b) (2,4)] Verschmelzung von [(1,3) a (2,3)][(2,3) b (2,4)]
[(1,3) b (1,4)][(1,4) a (2,4)] Umordnung von [(1,3) (a,b) (2,4)]
[(1,3) a (2,3)][(2,3) b (2,4)], [(1,3) (a,b) (2,4)], [(1,3) b (1,4)][(1,4) a (2,4)] Umordnungsfolge



Bemerkung: Sei $\sum$ kontaktfrei und G_1 , G_2 disjunkt, nichtleer, dann gilt:

(a) Wenn $k = (c_1, G_1 \cup G_2, c_2)$ Kante in $\Phi_{\sum}$, dann existiert ein $c_3 \in C$ und eine Zerlegung von k mit $(c_1, G_1, c_3)(c_3, G_2, c_2)$

(b) Seien $k_1 = (c_1, G_1, c_3)$, $k_2 = (c_3, G_2, c_2)$ Kanten in $\Phi_{\sum}$ von $\sum$, $\pi_{k_1} \circ \pi_{k_2} : K \to \sum$ Prozeß, dann gilt für alle $t, t' \in T : t \sharp t'$ gdw. $(c_1, G_1 \cup G_2, c_2)$ Kante in $\Phi_{\sum}$ von $\sum$ mit $c_1 = \pi(Min\ k)$, $c_2 = \pi(Max\ k)$ und $G_1 \cup G_2 = \pi(T)$.

Satz:
u, v Wege in einem beschränkten Prozeß gdw. Umordnungsfolge von u nach v existiert.

Weitere Zielstellungen sind Untersuchung des dynamischen Verhaltens von B/E-Systemen, Synchronisation zwischen Ereignissen, Ereignisabhängigkeiten, Verallgemeinerung des Netzbegriffes.

2. Netztheorie

2.1 Stellen-Transitions-Netze

Definition: $N = (S, T, F, K, W, m_0)$ heißt *Stellen-Transitions-Netz*, wenn folgendes gilt:

a) (S, T, F) ist ein Petri-Netz mit der Stellenmenge S, der Transitionsmenge T und der Flußrelation F,

b) $K : S \to N \cup \{\omega\}$ (Kapazität von Stellen),

c) $W : F \to N^+$ (Gewicht von Kanten) und

d) $m_0 : S \to N \cup \{\omega\}$ (Anfangsmarkierung) mit $m_0(s) \leq K(s)$ für alle $s \in S$.

Eine Abbildung $m : S \to N \cup \{\omega\}$ heißt *Markierung* von N, falls $m(s) \leq K(s)$ für alle $s \in S$.

Vektorschreibweise, falls S endlich: $S = \{s_1, s_2, ..., s_n\}$, $m = (m(s_1), ..., m(s_n))$, $K = (K(s_1), ..., K(s_n))$ und Erweiterung von Ordnungsrelationen und arithmetischen Operationen auf Vektoren wie üblich.

Bezeichnungen: Für $s \in {}^{\cdot}t$ bezeichnet $W(s, t)$ das Gewicht von $(s, t) \in F$.
Für $s \in t^{\cdot}$ bezeichnet $W(t, s)$ das Gewicht von $(t, s) \in F$.
$t \in T$ heißt *m-aktiviert* (hat Konzession bei m), falls $m(s) \geq W(s, t)$ für alle $s \in {}^{\cdot}t$ und $m(s) \leq K(s) - W(t, s)$ für alle $s \in t^{\cdot}$.

Definition: *Folgemarkierung*
$f : M \times T \to M$, wo M die Markierungsmenge von N mit

$$f(m,t)(s) = \begin{cases} m(s) - W(s,t) & \\ m(s) + W(t,s) & \\ m(s) - W(s,t) + W(t,s) & \\ m(s) & \end{cases} , \text{ falls } \begin{array}{l} s \in {}^{\cdot}t \backslash t^{\cdot} \\ s \in t^{\cdot} \backslash {}^{\cdot}t \text{ und } t \text{ m-aktiviert} \\ s \in {}^{\cdot}t \cap t^{\cdot} \\ \textit{sonst} \end{array}$$

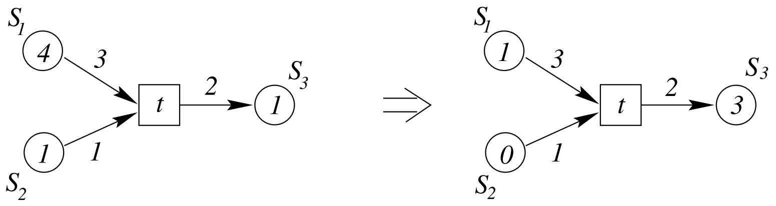
Die Markierung $m' = f(m, t)$ heißt Folgemarkierung von m. (In Zeichen: $m \to_t m'$)
Wenn $[m]$ die kleinste Markierungsmenge bezeichnet, die m und alle Folgemarkierungen für beliebige t enthält, dann gilt

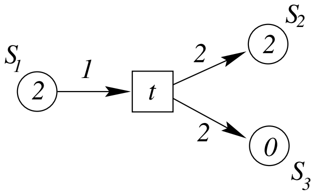
a) $m \in [m]$ und

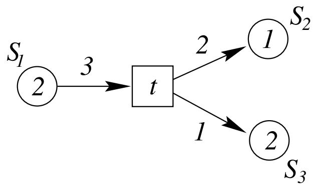
b) wenn $m' \in [m]$ und $m' \to_t m''$ dann ist auch $m'' \in [m]$.

Bemerkung: B/E-Systeme sind spezielle Stellen-Transitions-Netze (S/T-Netze) mit Kapazität und Gewicht = 1.
Bei B/E-Systemen Fallklasse C, bei S/T-Netzen Anfangsmarkierung m_0 .

Beispiel:







1. $m = (4, 1, 1), f(m, t) = m' = (1, 0, 3), W(s_1, t) = 3, W(s_2, t) = 1, W(t, s_3) \ = 2$, t m-aktiviert.
2. $m = (2, 2, 0), m(s_2) \succ K(s_2) - W(t, s_2), t$ nicht m-aktiviert (Kontaktsituation, Kapazität von s_2 unzureichend).
3. $m = (3, 2, 1), m(s_1) \prec W(s_1, t), t$ nicht m-aktiviert (unzureichende Markierung von s_1).

Vektorschreibweise:

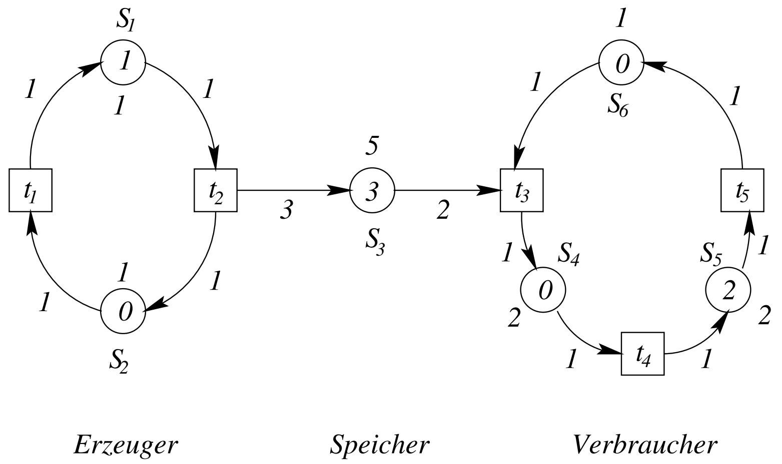
$t^+(s) = W(t, s)$ falls $(t, s) \in \ F$ sonst 0
$\underline{t^-(s) = W(s, t)}$ falls $(s, t) \ \in \ F$ sonst 0
$\Delta t(s) = t^+(s) - t^-(s)$
$f(m, t) = m + \Delta t$ falls $m \geq t^-$ und $m \leq k - t^+$.

Matrixschreibweise:

$$\underline{N}(s, t) = \begin{cases} t^+(s) & \\ -t^-(s) & \\ \Delta \mathrm{t(s)} & \\ 0 & \end{cases} , \text{ falls } \begin{array}{l} \mathrm{s} \in t^{\cdot} \backslash {}^{\cdot}t \\ \mathrm{s} \in {}^{\cdot}\, t \backslash t^{\cdot} \\ \mathrm{s} \in {}^{\cdot}\, t \cap t^{\cdot} \\ \textit{sonst} \end{array}$$

Bemerkung:
$\underline{N}$ legt N eindeutig fest, falls N rein (schlingenfrei).
$\underline{N}$, m_0 beschreibt das Verhalten von N eindeutig, falls N rein und kontaktfrei.
Wenn N rein, dann t m-aktiviert gdw. $0 \leq m + \Delta t \leq K$ und
N kontaktfrei gdw. $0 \leq m + \Delta t \Rightarrow m + \Delta t \leq K$ für alle $m \in [m_0]$.

Beispiel: Erzeuger-Verbraucher Problem



m_0 = (1, 0, 3, 0, 2, 0),
t_1^+ = (1, 0, 0, 0, 0, 0), t_2^+ = (0, 1, 3, 0, 0, 0), t_3^+ = (0, 0, 0, 1, 0, 0),
t_4^+ = (0, 0, 0, 0, 1, 0), t_5^+ = (0, 0, 0, 0, 0, 1),
t_1^- = (0, 1, 0, 0, 0, 0), t_2^- = (1, 0, 0, 0, 0, 0), t_3^- = (0, 0, 2, 0, 0, 1),
t_4^- = (0, 0, 0, 1, 0, 0), t_5^- = (0, 0, 0, 0, 1, 0),
Δt_1 = (1, -1, 0, 0, 0, 0), Δt_2 = (-1, 1, 3, 0, 0, 0), Δt_3 = (0, 0, -2, 1, 0, -1),
Δt_4 = (0, 0, 0, -1, 1, 0), Δt_5 = (0, 0, 0, 0, -1, 1),
$\Delta t = t^+ - t^- = \underline{N}$.

Erweiterung der Folgemarkierung $F(m, t)$ auf *Transitionsfolgen:*
$f : M \times W(T) \to M$
$G \subseteq T$ Schritt bei m (m-Schritt), falls alle $t \in G$ m-aktiviert und paarweise unabhängig sind, d.h. wenn $t_1 \neq t_2 \in G$ dann t m-aktiviert und $({}^{\cdot}t_1 \cup t_1^{\cdot}) \cap ({}^{\cdot}t_2 \cup t_2^{\cdot}) = \emptyset$.

Bemerkung:

a) $\{t\}$ Schritt bei m gdw. t m-aktiviert.

b) $G' \subseteq G$ Schritt bei m, dann ist auch G' Schritt bei m.

Schreibweise: $G^- = \sum\limits_{t \in G} t^-$, $G^+ = \sum\limits_{t \in G} t^+, \Delta G = G^+ - G^-$

Für Transitionsfolgen (Wörter) $t_1 ... t_n = q \in W(T)$ und Schrittfolgen $Q \in W(G)$ führen wir ein:

$f(m, G) = m + \Delta G$ falls $G^- \leq m \leq K - \Delta G$;
nicht definiert sonst

$f(m, e) \ = m$
$f(m, tq) = f(m + \Delta t, q)$ falls $t^- \leq m \leq K - t^+$ und $f(m + \Delta t, q)$ definiert;
nicht definiert sonst

$f(m, GQ) = f(m + \Delta G, Q)$ falls $G^- \leq m \leq K - \Delta G$ und $f(m + \Delta G, Q)$ definiert
nicht definiert sonst

Bemerkung:

a) $f(m, q) = m + \Delta q$ gdw. $t_i^- \leq m + \sum\limits_{1 \leq j \leq i} \Delta t_j \leq K$ mit $\Delta q = \sum\limits_{1 \leq i \leq n} \Delta t_i$

b) Wenn G endlicher Schritt bei m und $q \in W(G)$ eine Transitionsfolge aus G, wo jede Transition genau einmal vorkommt, dann ist

   $f(m, q) = m + \Delta q = m + \Delta G = f(m, G)$.

   (Transitionen können in beliebiger Folge schalten. Jede Folge erzeugt gleiche Endmarkierung.)

c) Wenn N rein (schlingenfrei), $G \subseteq T$ und $K = \omega$ (unendliche Kapazität) dann gilt:

   $G^- \leq m$ (alle $t \in G$ haben gemeinsam bei m Konzession) gdw. jede Ordnung von G (Wort) hat Konzession (ist feuerbar) bei m (m-aktiviert).
   (Bei kapazitätsbeschränkten Netzen gilt dies im allgemeinen nicht.)

Definition:

a) *<u>Erreichbarkeitsklasse</u>* R_N von N

   $R_N(m_0) = \{f(m_0, q) \mid f(m_0, q) \text{ definiert}, q \in W(T)\}$.
   $m \in R(m_0)$ erreichbar von m_0.

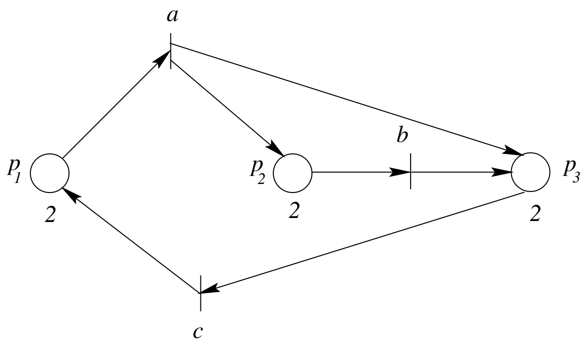
b) *<u>Petri-Netz-Sprache</u>* L_N

   $L_N(m_0) = \{q \mid f(m_0, q) \text{ definiert}, q \in W(T)\}$ von m_0 erzeugte Sprache.

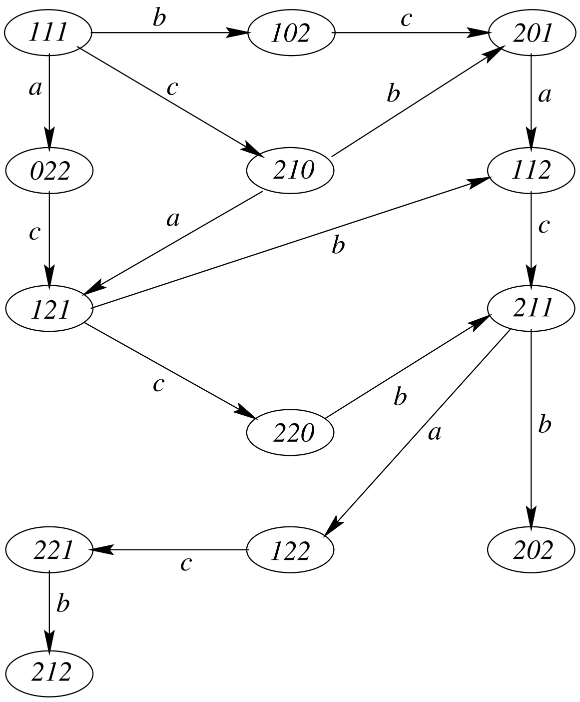
c) Netz N heißt *beschränkt* bei m_0, falls jeder Platz $s \in S$ bei m_0 beschränkt ist, d.h. falls eine natürliche Zahl k existiert mit $m(s) \leq k$ für alle erreichbaren Markierungen $m \in R_N(m_0)$.

d) Netz N heißt K-*sicher* (sicher, wenn $K(s) = 1$ für alle $s \in S$), falls $L_N(m_0) = L_{N_\omega}(m_0)$ mit $N_\omega = (S, T, F, \omega, W, m_0)$.

e) *Erreichbarkeitsgraph* (analog Fallgraph) $\Xi_N = (M_N, P)$ mit M_N Markierungsmenge von N und $P \subseteq M_N \times T \times M_N$ mit $(m, t, f(m, t)) \in P$, falls $f(m, t)$ definiert.



Netz



Erreichbarkeitsgraph

Bemerkung:

a) Ξ_N ist endlich gdw. N beschränkt (in Kapazität) ist.

b) Ξ_N ist Automatengraph von $\Theta_N = (T, R_N(m_0), f, m_0)$ (partieller Automat).

Satz: Für Netze N_1, N_2, die sich nur in der Kapazität unterscheiden, gilt:
$L_{N_1}(m_0) \subseteq L_{N_2}(m_0)$ gdw. $R_{N_1}(m_0) \subseteq R_{N_2}(m_0)$

Folgerung:

a) Für Netze N_1, N_2, die sich nur in der Kapazität unterscheiden, gilt:
$L_{N_1}(m_0) = L_{N_2}(m_0)$ gdw. $R_{N_1}(m_0) = R_{N_2}(m_0)$

b) N ist K-sicher genau dann, wenn $R_N(m_0) = R_{N_\omega}(m_0)$

c) $R_N(m_0) \subseteq R_{N_\omega}(m_0)$ (Kapazitätsvergrößerung schränkt die Erreichbarkeitsklassen nicht ein.)

d) Jedes K-sichere Netz N ist beschränkt, d.h. wenn N K-sicher ist, dann ist $m \leq K$ für alle $m \in R_N(m_0)$.

Definition: $N = (S, T, F, K, W, m_0)$, m Markierung

a) t *lebendig* bei m (m-lebendig), wenn von jeder Markierung m' aus, die von m erreichbar ist, eine Markierung m'' erreicht werden kann, wo t Konzession hat (m''-aktiviert), d.h. für alle $m' \in R_N(m)$ existiert m'' mit
$m'' \in R_N(m') \wedge t^- \leq m'' \wedge m'' + \Delta t \leq K$.

b) m *lebendig*, wenn alle $t \in T$ lebendig bei m.

c) N *lebendig*, wenn m_0 lebendig.

d) t *tot* bei m, wenn für alle $m' \in R_N(m)$ gilt $t^- \succ m'$ (t nicht m'-aktiviert).

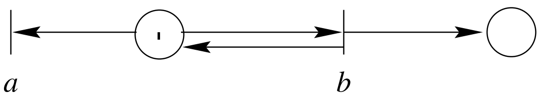
e) m *tot*, wenn jede Transition t tot bei m.

f) N *tot*, wenn die Anfangsmarkierung m_0 tot ist.

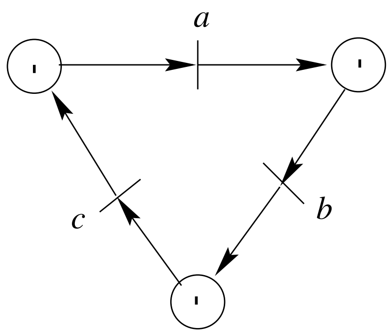
Bemerkung: Wenn m lebendig in N, dann ist $L_N(m)$ unendlich. (Umkehrung gilt allgemein nicht.)

Beispiele:

a) $L_N(m_0) = \{b^i a^j | i \geq 0,\ 0 \leq j \leq 1\}$, $R_N(m_0) = \{(j,\ i)\ | i \geq 0,\ 0 \leq j \leq 1\}$, aber a, b nicht lebendig. Es existiert keine lebendige Markierung.



b) Netz lebendig, a,b,c bei m_0 lebendig.

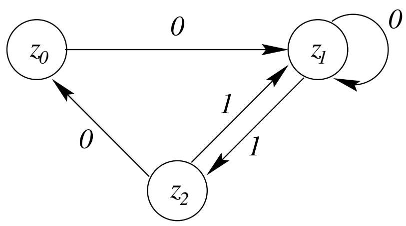


2.2 Automaten und Petri-Netze

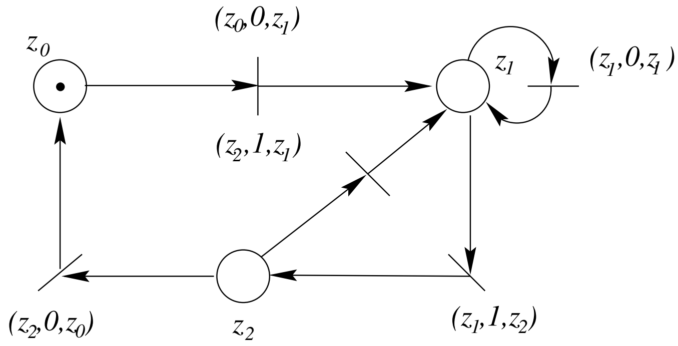
Sei $A = (X, Z, f, z_0)$ ein endlicher Automat ohne Ausgabe. Der Automatengraph kann als Petri-Netz N_A mit Transitionen in den Kanten $t = (t_1, t_2, t_3)$ aufgefaßt werden.
$N_A = (Z, T, F, 1, \omega, m_0)$ mit $T = \{(z, x, f(x, z))|z \in Z, x \in X\}$,
F $= \{$(z, t) $|t_1 = z\} \cup \{(t, z)|t_3 = z\}, m_0(z_0) = 1$ sonst 0 (genau eine Marke, Zustandsmaschine).

Beispiel:

Automat



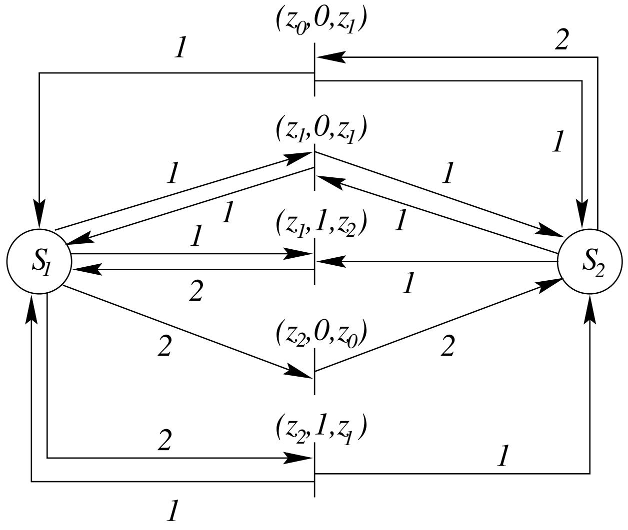
Zustandsmaschine



Bemerkung:
$|t^\cdot| = |^\cdot t| = 1, m_z(z) = 1$ gdw. z aktueller Zustand, $m_z(z') = 1 \Leftrightarrow z = z'$
Erreichbarkeitsgraph $\Xi_{N_A} \cong G_A$ Automatengraph
Erreichbarkeitsklasse $R_{N_A}(m_0) \subseteq \{m_z \mid z \in Z\}$
Netz mit zwei Plätzen möglich (Zustände entsprechen Markierung der beiden Plätze)
$Z = \{z_0, ..., z_n\}$, $m_{z_0} = (0, n)$, $m_{z_i} = (i, n - i)$, $N_A = (\{s_1, s_2\}, T, F, K, W, m_{z_0})$ mit
$T = \{(z, x, f(x, z))|z \in Z, x \in X\}$,
$F = \{(s_i, t)|t_1 = z, m_z(s_i) \succ 0\} \cup \{(t, s_i)|t_3 = z, m_z(s_i) \succ 0\}$
$K = n$, $W(s_i, t) = m_z(s_i)$ für $t_1 = z$, $W(t, s_i) = m_z(s_i)$ für $t_3 = z$
N_A ist K-sicher (konstante Anzahl n von Marken)
$m_z \leq m_{z'} \Rightarrow m_z = m_{z'}$ (keine verschiedenen Markierungen sind vergleichbar)
deshalb gilt: t m_z-aktiviert gdw. $t_1 = z$ (t hat Konzession bei m_z) und $\Xi_{N_A} \cong G_A$.

Beispiel:

Netz



Probleme:

a) Kann man jedem endlichen Netz N einen endlichen Automaten A_N zuordnen (N beschränkt)?

b) Welche Automaten (eventuell mit Speicher) können durch endliche Netze dargestellt werden?

Definition: Beschränktheit von Petri-Netzen

1. m *<u>überdeckbar</u>* in N, wenn existiert $m' \in R_N(m_0)$ mit $m' \geq m$

   (durch erreichbare Markierung überdeckt).

2. $s \in S$ *<u>beschränkt</u>* in N, wenn ein $n \in Nz$ existiert, so daß für alle $m' \in R_N(m_0)$ gilt $m'(s) \leq n$.

   (alle erreichbaren Markierungen sind durch n beschränkt).

3. $Q \subseteq S$ *<u>beschränkt</u>* in N, wenn für alle $s \in Q : s$ beschränkt in N.

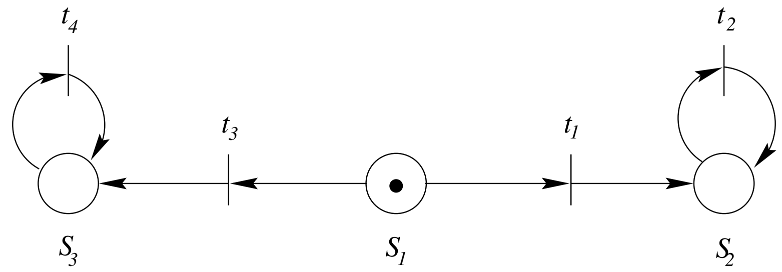
4. $s \in S$ *<u>sicher unbeschränkt</u>* in N, wenn für jede nat. Zahl n und alle $m \in R_N(m_0)$

   existiert $m' \in R_N(m)$ mit $m'(s) \succ n$.

5. $Q \subseteq S$ *<u>simultan unbeschränkt</u>* in N, wenn für jede nat.Zahl n existiert ein $m \in R_N(m_0)$ mit $m(Q) \succ n$. (d.h. für alle $s \in Q$ gilt $m(s) > n$).

6. t m-*<u>tot</u>* , wenn für alle $m' \in R_N$ (m) gilt: t ist nicht m' -aktiviert.

Beispiel:



$m = (0,2,0)$ erreichbar, überdeckbar durch (0, 4, 0)
$m = (0,0,2)$ nicht erreichbar, überdeckbar durch (0, 0, 3)
$m = (2,0,0)$ nicht erreichbar, nicht überdeckbar
s_1 beschränkt, s_2, s_3 unbeschränkt, nicht sicher unbeschränkt, $\{s_2, s_3\}$ nicht simultan unbeschränkt.

Definition: *Überdeckbarkeitsgraph*
$N = (S, T, F, K, W, m_0)$ Netz
$G_N = (P, B, b)$ Überdeckbarkeitsgraph von N mit $P \subseteq L_N(m_0)$ Transitionswörter,
$B \subseteq P \times P$ Bögen (Kanten),
$b : P \to (N_z^\omega)^S$ (verallgemeinerte) Markierung und
$B \to T \cup \{e\}$ Transitionswörter der Länge ≤ 1 .

Konstruktion von G_N:
Anfangsschritt: $P^0 = \{e\}$, $B^0 = \emptyset$, $b^0(e) = m_0$
i-ter Schritt: $k \in P^i$ mit $(k, k') \in B^i \Rightarrow k = k'$ (keine anderen Bögen, Rücksteller)
(i+1)-ter Schritt:

a) für alle $t \in T : t$ nicht $b(k)$ - aktiviert (k totes Blatt)

b) existiert k' echtes Anfangsstück von k mit $b(k) = b(k')$(k Schleifenblatt) dann
$B^{i+1} = B^i \cup \{(k, k')\}$, $b^{i+1}(k, k') = e$

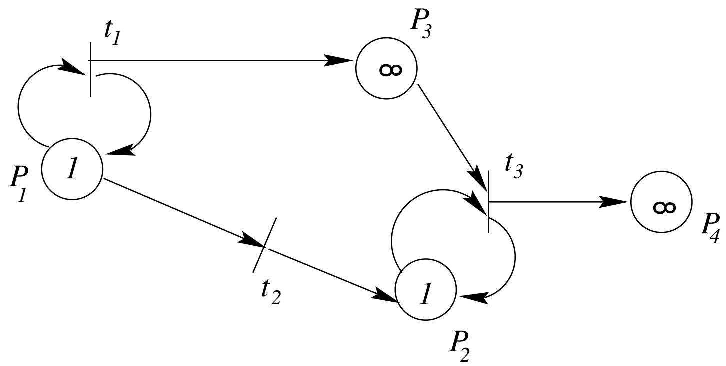
c) sonst $P^{i+1} = P^i \cup \{kt | t^- \leq m, m + \Delta t \leq K, b^{i+1}(k) = m\}$
$B^{i+1} = B^i \cup \{(k, kt) | t^- \leq m, m + \Delta t \leq K, b^{i+1}(k) = m\}$
$b^{i+1}(k, kt) = t$ (Bewertungskante),
$b^{i+1}(kt) = b^{i+1}(k) + \Delta t$ falls $P_{kt}^{i+1} = \emptyset$ (Bewertungsknoten)
mit $P_{kt}^{i+1} = \{k' | k' \in P^{i+1}$, k' echtes Anfangsstück von $kt, b^{i+1}(k') \leq b^{i+1}(k) + \Delta t$
und für alle $s : b^{i+1}(k')(s) \prec (b^{i+1}(k) + \Delta t)(s)$ gilt $K(s) = \omega\}$.
Falls $P_{kt}^{i+1} \neq \emptyset$ dann
$b^{i+1}(kt)(s) = \omega$ falls $k' \in P_{kt}^{i+1}$ mit $b^{i+1}(k')(s) \prec (b^{i+1}(k) + \Delta t)(s)$;
$(b^{i+1}(k) + \Delta t)(s)$ sonst

$B^{i+1} = B^i \cup \{kt, k'\}$ mit $k' \in P^{i+1}_{kt}$ (Rücksteller)

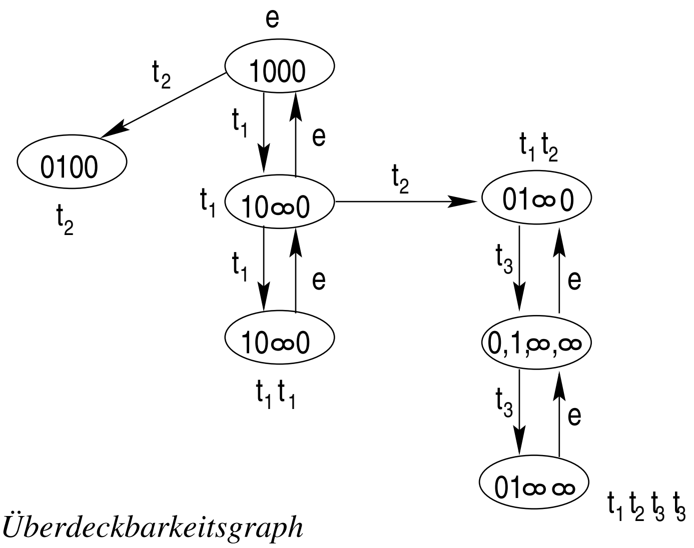
$b^{i+1}(kt, k') = e$

Abschluß: $P^i = P^{i+1}$

$G_N = (P^\omega, B^\omega, b^\omega)$
Beispiel: ohne e-Kanten ein Baum



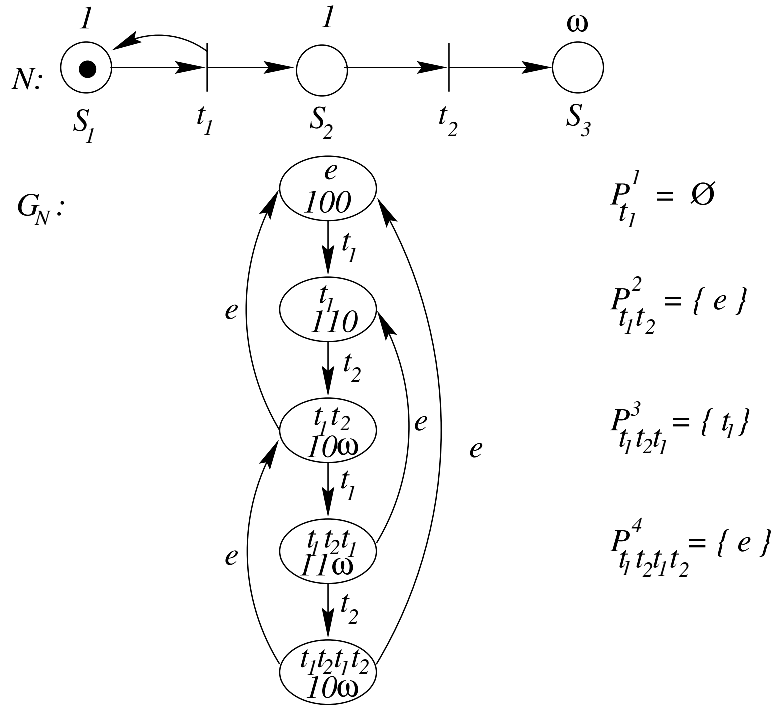
Netz



Überdeckbarkeitsgraph

Satz: Wenn N endlich, dann ist auch G_N endlich.

Beispiel:



Schleifenblatt

Eigenschaften:

a) $s \in S$ unbeschränkt in N gdw. es existiert ein $k \in P$ in G_N mit $b(k)(s) = \omega$

b) m überdeckbar in N gdw. es existiert ein $k \in P$ in G_N mit $b(k) \geq m$

c) $Q \subseteq S$ simultan unbeschränkt in N gdw.
es existiert ein $k \in P$ in G_N mit $b(k)(Q) = \omega$

d) t m-tot in N gdw. (m, t, m') kein Bogen (Kante) in G_N.

Satz: Wenn N endlich, dann ist K-Sicherheit von N entscheidbar, d.h. es existiert ein Algorithmus mit dem festgestellt werden kann, ob NK-sicher ist.

Folgerung: Für endliche N ist die Beschränktheit entscheidbar.

2.3 Lebendigkeit, Sicherheit, Deadlocks

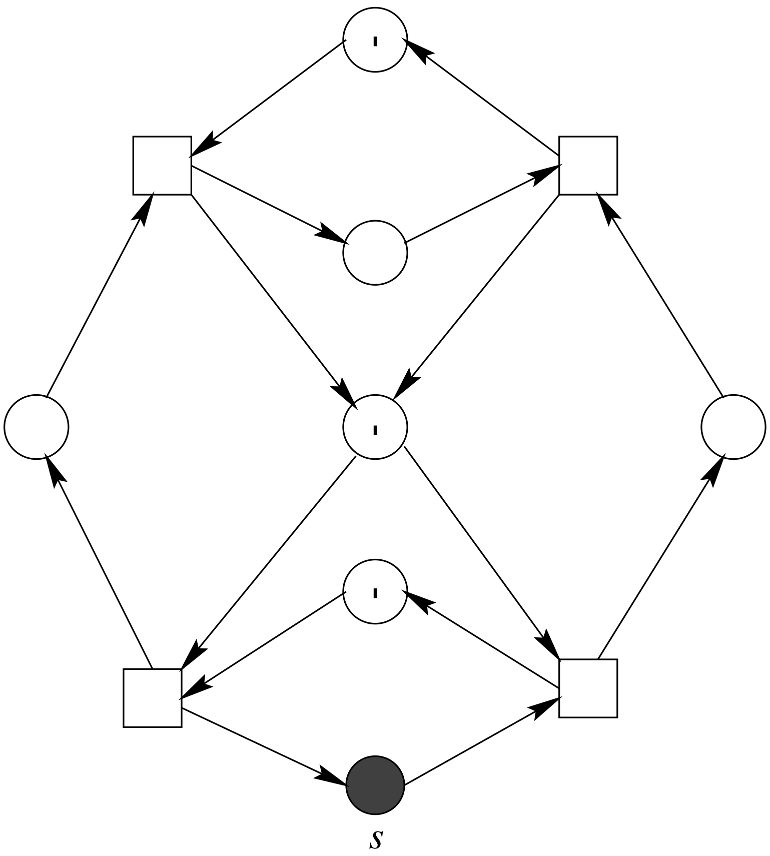
Stellen/Transitions-Netze werden zur Verhaltensbeschreibung von Prozessen benutzt, bei denen die Anzahl und Verteilung veränderlicher Objekte wesentlich ist (Daten im Speicher, Waren in Lagern, Werkstücke in Produktionssystemen). Dabei ist wichtig, daß sich Anzahl und Verteilung der Objekte in gewissen Grenzen hält. Störungen können zum Stillstand (Blockierung) des Systems führen (Stau o. Mangelbildung)

Blockierung: Transitionen (aktive Elemente) können auf Grund von leeren (unvollständigen) Eingangsstellen oder vollen (unzureichenden) Ausgangsstellen nicht mehr schalten (Netz ist nicht lebendig). Es wurden verschiedene Lebendigkeitsbegriffe betrachtet. Bisher so verwendet, daß bei jeder erreichbaren Markierung jede Transition aktivierbar bleibt.

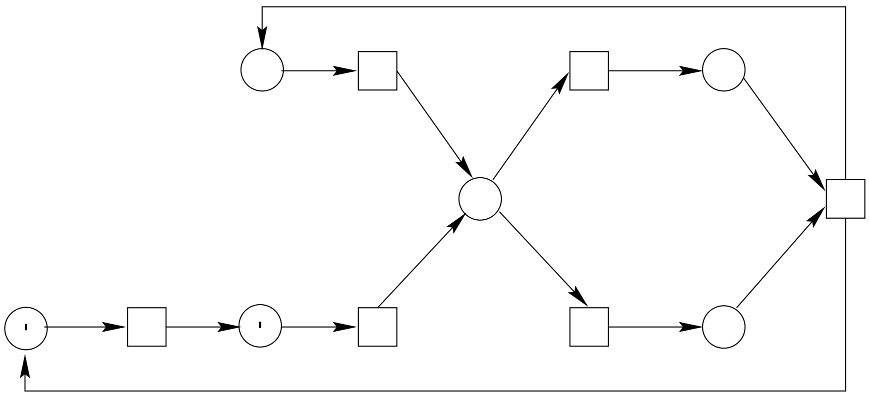
Vermutung: Eine Vergrößerung der Anfangsmarkierung kann zum Verlust der Lebendigkeit führen.

Beispiele:

a) Markierung bei s führt zu nicht lebendigem Netz.



b) Nicht jede Markierung muß reproduzierbar sein.



Für beschränkte Petri-Netze ist Lebendigkeit mit Hilfe des Erreichbarkeitsgraphen zu entscheiden.

Satz: Ein beschränktes Netz N ist lebendig gdw. in jeder stark zusammenhängenden Komponente des Erreichbarkeitsgraphen Ξ_N von N zu jeder Transition $t \in T$ eine mit t markierte Kante (Bogen) existiert, d.h. für beschränkte Netze ist das Lebendigkeitsproblem mit dem Erreichbarkeitsproblem (algorithmisch) äquivalent.

Bemerkung: Falls ein Verfahren existiert, mit dem entschieden werden kann, ob $N = (P, T, F, m)$ lebendig ist, dann existiert auch ein Verfahren, welches entscheidet, ob $m \in R_N(m_0)$ ist und umgekehrt.

Satz: $N = (P, T, F, m_0)$ Netz
$t \in T$ tot bei m gdw. in $G_N = (P, B, b)$ existiert <u>keine</u> Kante (Bogen) (k, k') aus B mit $b\ (k, k') = t$.

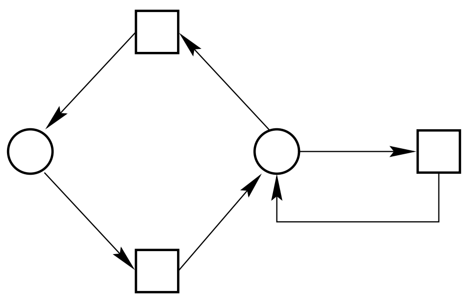
Bemerkung: Für endliche Netze ist die Eigenschaft tot entscheidbar.

Definition: Sicherheit eines Netzes $N = (P, T, F, m_0)$
$p \in P$ <u>*sicher*</u> bei m, wenn für alle $m' \in R_N(m)$ gilt $m'(p) \leq 1$.
m *sicher*, wenn alle $p \in P$ sicher bei m.
N *sicher*, wenn m_0 sicher.

Spezielle Petri-Netze:

1. $N = (P, T, F)$ heißt <u>*Zustandsmaschine*</u> gdw. für alle $t \in T : |{}^{\cdot}t| = |t^{\cdot}| = 1$, (d.h., jede Transition hat genau einen Vor- und genau einen Nachplatz(stelle). (Zusammenhang mit Automaten beachten.)

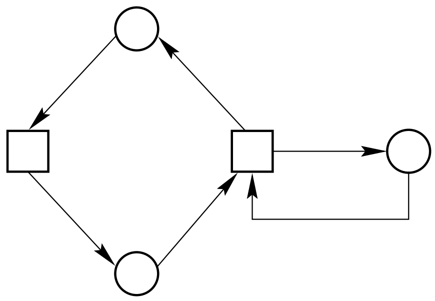
   Beispiel:

   Zustandsmaschine

2. $N = (P, T, F)$ heißt <u>*Synchronisationsgraph*</u> gdw. für alle $p \in P : |{}^{\cdot}p| = |p^{\cdot}| = 1$, (d.h., jeder Platz (Stelle) hat genau eine Vor- und genau eine Nachtransition. (Dualität zur Zustandsmaschine.)

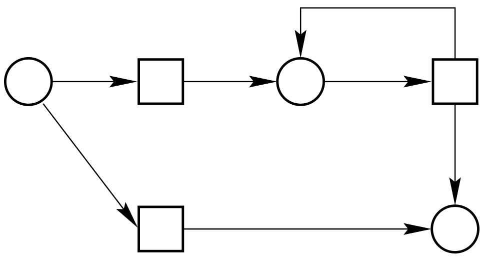
   Beispiel:

   Synchronisationsgraph

3. $N = (P, T, F)$ heißt <u>*free-choice-Netz*</u> gdw. für alle $p \in P$ und $t \in T$ mit $(p, t) \in F$ gilt: $|p^{\cdot}| = 1$ oder $|{}^{\cdot}t| = 1$, (d.h., jeder Platz hat nur eine Nachtransition oder alle Nachtransitionen besitzen nur einen gemeinsamen Vorplatz).

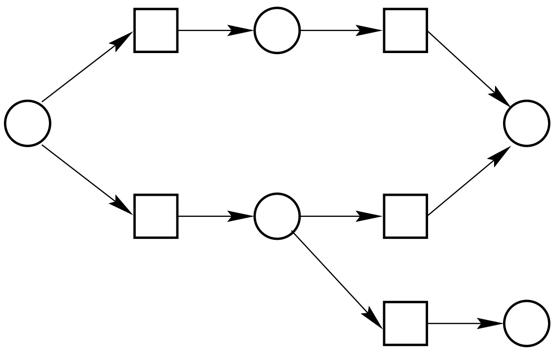
   Beispiel:

   Free-choice Netz

4. $N = (P, T, F)$ heißt <u>*einfaches Netz*</u> gdw. für alle Transitionen $t \in T$ gilt: $|{}^{\cdot}t \cap \bigcup\limits_{t \neq t'} {}^{\cdot}t'| \leq 1$, (d.h., jede Transition hat mit allen anderen Transitionen höchstens einen Vorplatz gemeinsam).

   Beispiel:



Einfaches Netz

Eigenschaften:

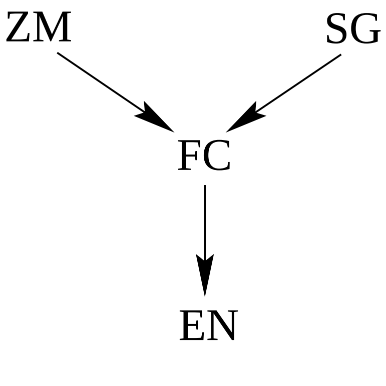
a) Zustandsmaschinen und Synchronisationsgraphen sind *free-choice-Netze.* (Umkehrung gilt nicht.)

b) *Free-choice-Netze* sind einfache Netze (Umkehrung gilt nicht).

c) Zustandsmaschinen besitzen sichere Markierungen.
Wenn N stark zusammenhängend, dann ist jede Markierung mit genau einer Marke lebendig und sicher.

d) Synchronisationsgraphen sind bei jeder Markierung konfliktfrei. (Es existieren keine geteilten Eingabeplätze.) (t_1 und t_2 bei m in Konflikt gdw. t_1 verliert Konzession , wenn t_2 Konzession hat und umgekehrt. N konfliktfrei, wenn bei keiner erreichbaren Markierung zwei Transitionen in Konflikt stehen.)

e) Synchronisationsgraphen N sind lebendig bei m gdw. jeder Kreis in N bei m markiert ist.



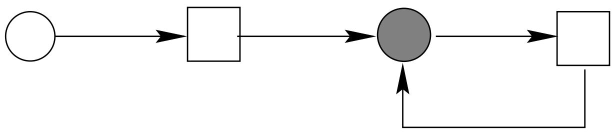
Definition: $N = (P, T, F),\ Q \subseteq P,\ Q \neq \emptyset,\ m : P \to Nz$

a) Q <u>*sauber*</u> bei m, wenn $\sum\limits_{p \in Q} m(p) = 0$, (d.h., kein Platz aus Q ist markiert), sonst heißt Q markiert.

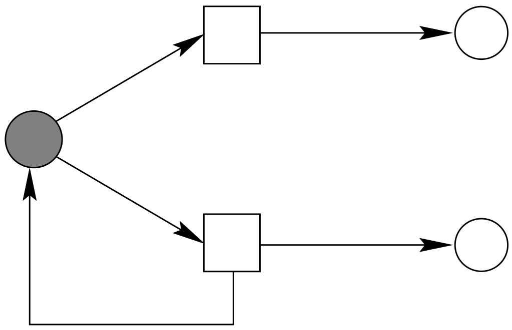
b) Q <u>*Falle*</u>, wenn $QF \subseteq FQ$ (bzw. $Q^{\cdot} \subseteq {}^{\cdot}Q$) , (d.h., Q markiert kann nicht gesäubert werden).

c) Q <u>*Deadlock*</u>, wenn $FQ \subseteq QF$ (bzw. ${}^{\cdot}Q \subseteq Q^{\cdot}$), (d.h., Q sauber kann nicht markiert werden.)

Beispiele



Falle



Deadlock

Eigenschaften:

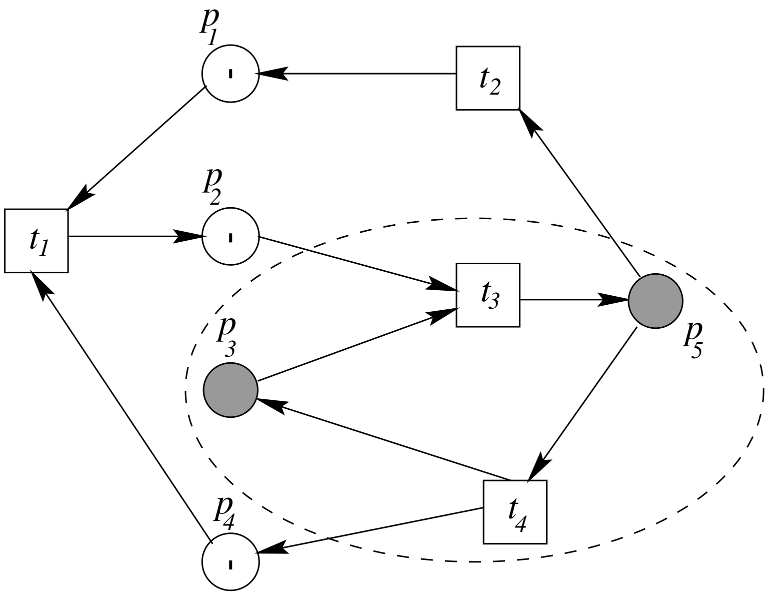
a) Jede bei m markierte Falle Q bleibt bei jeder von m erreichbaren Markierung m' markiert (d.h., $m(Q) \succ 0$ und $m' \in R_N(m)$, dann $m'(Q) \succ 0$.)

b) Jeder bei m saubere Deadlock Q bleibt bei jeder von m erreichbaren Markierung m' sauber (d.h., $m(Q) = 0$ und $m' \in R_N(m)$, dann $m'(Q) = 0$.)

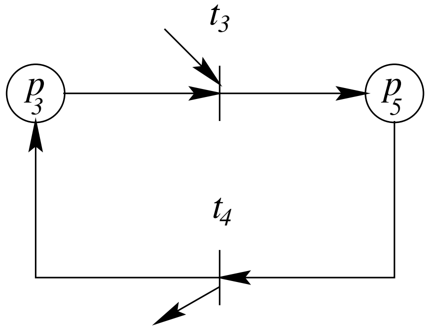
Satz (Lebendigkeit von *free-choice-Netzen*):
Eine Markierung m ist lebendig in einem free-choice-Netz $N = (P, T, F)$ gdw. zu jedem Deadlock $Q \subseteq P$ existiert eine bei m markierte Falle $Q' \subseteq Q$ (d.h., m lebendig in N gdw jeder Deadlock enthält eine bei m markierte Falle).

(Da jede Transition höchstens einen Platz mit einer anderen Transition teilt, könnten sonst Transitionen im Deadlock niemals aktiviert werden, da keine Marke an die Vorplätze dieser Transitionen gelangen kann.)

Beispiel:

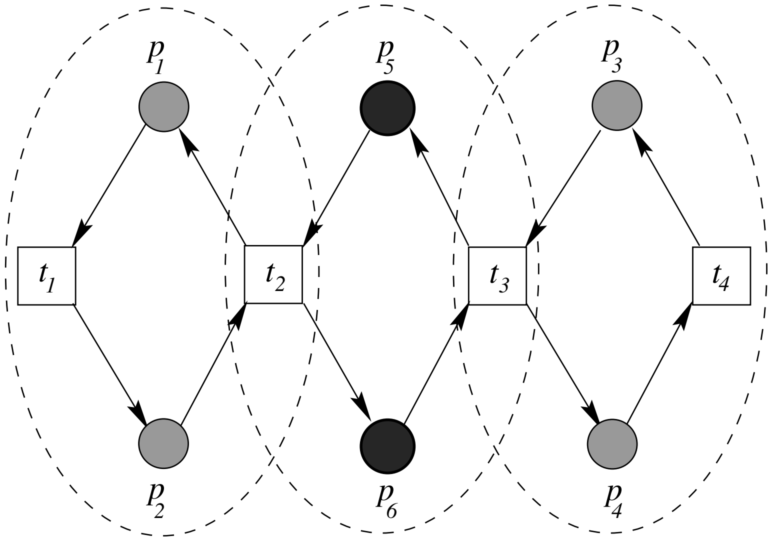


$m = (1,1,0,1,0)$, sauberer Deadlock $Q = \{p_3, p_5\}, QF = \{t_2, t_3, t_4\}, FQ = \{t_3, t_4\}$, $FQ \subseteq QF$.
N nicht lebendig bei m, da t_3, t_4 nicht lebendig.



Bemerkung: Jedes *free-choice-Netz*, welches einen sauberen Deadlock enthält, ist nicht lebendig.

Beispiel:



Bemerkung:
Für bei m lebendige *free-choice-Netze* müssen *minimale Deadlocks* (Deadlocks, die selbst keine Deadlocks als Teilmengen besitzen) bei m markierte Fallen enthalten.

Minimale Deadlocks: $\{p_1, p_2\}, \{p_3, p_4\}, \{p_5, p_6\}$. Da diese Deadlocks auch Fallen sind, muß mindestens ein Knoten jeder dieser Mengen markiert sein. In lebendigen Markierungen sind mindestens drei Plätze markiert.

Lebendige Markierungen:
(1,0,1,0,1,0) , (1,0,1,0,0,1) , (1,0,1,0,1,1) , (1,0,0,1,1,0) , (1,0,1,1,0,1) , (0,1,1,0,1,0).

Überdeckung von Netzen durch Zustandsmaschinen:

Zusammenhang mit Lebendigkeit und Sicherheit, Zerlegung in Automaten, isolierte Transitionen können ausgenommen werden. Lebendigkeit und Sicherheit sind gemeinsam von Interesse.

Definition: $N = (P, T, F, m),\ Q \neq \emptyset,\ Q \subseteq P$

a) Das Netz $N_Q = (Q, T_Q, F_Q, m_Q)$ heißt das von Q *erzeugtes Teilnetz* von N, wenn $T_Q = FQ \cup QF, \quad F_Q = F|_{(Q \times T_Q) \cup (T_Q \times Q)}$ und $m_Q = m|_Q$.

b) Seien $N_{Q_i} (i \in I)$ die von Q_i erzeugten Teilnetze von N und N sei frei von isolierten Transitionen.
$\{N_{Q_i}\}_{i \in I}$ heißt *Überdeckung* von N gdw. $\bigcup\limits_{i \in I} Q_i = P$.

c) Ein Deadlock Q heißt *minimal*, wenn keine echte Teilmenge von Q Deadlock ist.

Satz (Hack):
Free-choice-Netze $N = (P, T, F, m)$ ohne isolierte Transitionen sind lebendig und sicher gdw.:

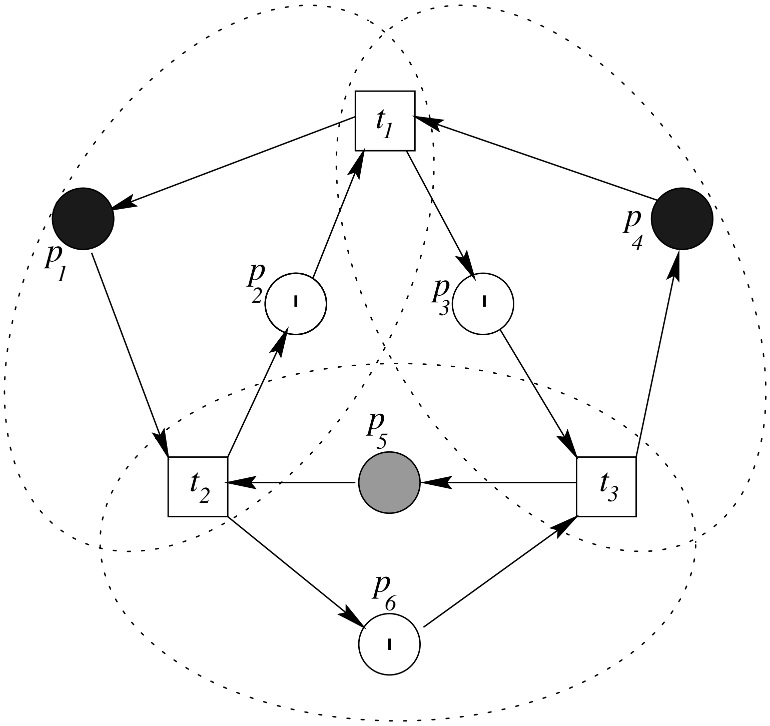
a) existiert eine Überdeckung von N mit stark-zusammenhängenden 1-markierten (nur eine Marke) Zustandsmaschinen und

b) jeder minimale Deadlock ist eine stark-zusammenhängende markierte (mindestens eine Marke) Zustandsmaschine.

Beispiel:

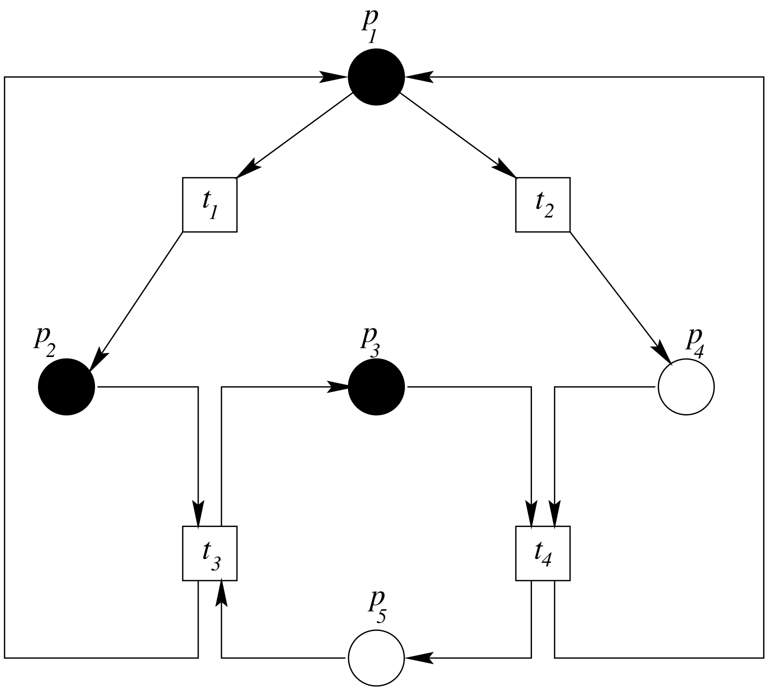
1. $Q_1 = (p_1, p_2),\ Q_2 = (p_3, p_4),\ Q_3 = (p_5, p_6),\ Q_4 = (p_2, p_3, p_5),\ Q_5 = (p_1, p_4, p_6)$

   sind minimale Deadlocks, die erzeugten Teilnetze sind stark-zusammenhängende Zustandsmaschinen. $Q_1 \cup Q_2 \cup Q_3$ ist Überdeckung des Gesamtnetzes, welches lebendig und sicher, da die Zustandsmaschinen je eine Marke enthalten und alle Q_i markiert sind.



Free-choice Netz lebendig und sicher

2. $Q = (p_1, p_2, p_3)$ ist minimaler Deadlock, $FQ = (t_1, t_3, t_4) \subseteq (t_1, t_2, t_3, t_4) = QF$. Das von Q erzeugte Teilnetz ist keine Zustandsmaschine, das Netz besitzt keine lebendige und sichere Markierung.



Free-choice Netz ohne lebendige und sichere Markierung

Verallgemeinerte Petri-Netze:

Es werden mehrere Kanten zwischen Transitionen und Plätzen zugelassen (Aufgabe der Schlichtheit) bzw. können Kanten entsprechend ihrer Vielfachheit mit nat.Zahlen bewertet werden. Flußrelation wird als Abbildung in die Menge Nz beschrieben.

Definition: $N = (P, T, F)$ heißt <u>*verallgemeinertes Petri-Netz*</u>, wenn

a) P, T disjunkte, endliche Mengen, $P \cup T \neq \emptyset, P \cap T = \emptyset$ und

b) $F : (P \cup T) \times (T \cup P) \to Nz$. (partielle Funktion)

Bezeichnung: $F(p, t)$ bzw. $F(t, p)$ Anzahl der Kanten von p nach t bzw. von t nach p (keine Kante = 0). Markierung $m : P \to Nz$, $m(p)$ Anzahl der Marken auf Platz p.

Vektorschreibweise:
$t^- = (F(p_1, t), ..., F(p_n, t)) \in Nz^n$ bzw. $t^+ = (F(t, p_1), ..., F(t, p_n)) \in Nz^n$ (Zahl der Marken, die den Plätzen entnommen bzw. zugefügt werden.) für $P = \{p_1, ..., p_n\}, \Delta t = t^+ - t^-$.

Folgemarkierung:
$f_N(m, t) = m + \Delta t$ falls $m \geq t^-$ und $\Sigma_{p \in P} m(p) * t^+(p) = 0$ (sichere Transitionsregel)
nicht definiert sonst

Beispiel:

verallgemeinertes Petri-Netz

$m = (5, 0, 1), t^- = (3, 0, 0), t^+ = (0, 1, 3), \Delta t = (-3, 1, 3), f(m, t) = (2, 1, 4)$ definiert, da $t^- \leq (5, 0, 1)$, Folgemarkierung nicht definiert, da $t^- \succ (2, 1, 4)$.

Bemerkung:

1. Jedes einfache Netz ist auch verallgemeinertes Netz.

2. Jedem verallgemeinerten Netz kann ein einfaches Netz zugeordnet werden, welches dieselbe Menge feuerbarer Transitionsfolgen
<u>*Transitionssprache*</u> $L_N(m) = \{q | q \in W(T_N) \wedge f_N(m, q)$ definiert$\}$ besitzt.

Definition: *Simulation*

Seien $N^i = (P^i, T^i, F^i, m_0^i)$ verallgemeinerte Petri-Netze.
N^1 <u>*simuliert*</u> N^2 falls existieren Abbildungen $g: \ Nz^{P^1} \to Nz^{P^2}$ und
$h: W(T^1) \to W(T^2)$ mit

a) $g(f_{N^1}(m,q)) = f_{N^2}(g(m), h(q))$ und $g(m_0^1) = m_0^2$ für alle $m \ \in Nz^{P^1}, q \in W(T^1)$,

b) $h(q) \in L_{N^2}(g(m))$ für alle $m \in Nz^{P^1}$ und $q \in L_{N^1}(m)$ und

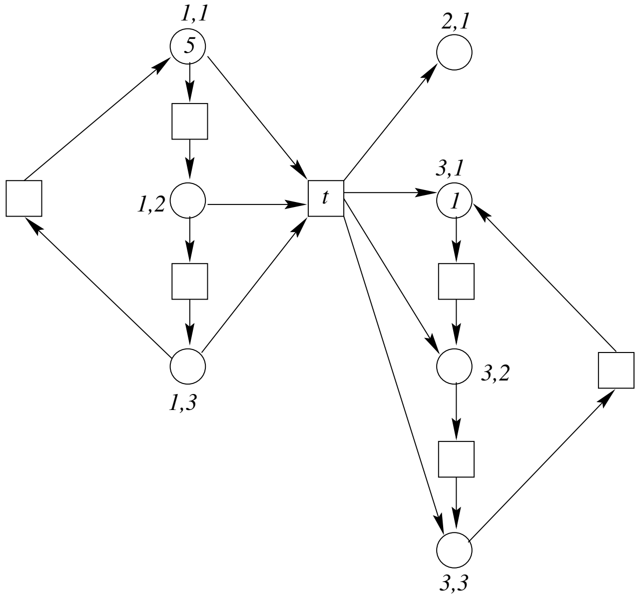
c) es existiert $q \in L_{N^1}(m), m \in Nz^{P^1}$ mit $h(q) = q'$ und $g(m) = m'$ für alle $m' \in Nz^{P^2}, q' \in L_{N^2}(m')$.

(Simuliert bedeutet, daß zu jedem Wort der Sprache $L_{N^1}(m_0^1)$ ein Wort der Sprache $L_{N^2}(m_0^2)$ und umgekehrt existiert, und die entsprechenden Transitionswörter zu denselben Markierungen führen.)

Satz: Zu jedem verallgemeinerten Petri-Netz N^2 existiert ein einfaches Petri-Netz N^1, welches N^2 simuliert (verschärfend kann N^1 als schlingenfrei gefordert werden).

(Beweis durch Konstruktion des einfachen Netzes, indem die Anzahl der Plätze des erweiterten Netzes entsprechend der Vielfachheit der Kanten vervielfacht wird, wobei Hilfstransitionen zwischen den hinzugenommenen Plätzen angeordnet werden (siehe Abbildung). Lebendigkeit, Sicherheit und Erreichbarkeit bleibt dabei erhalten bzw. überträgt sich.)

Beispiel:



einfaches Petri-Netz

Definition: Sei $N = \ (P, T, F, m_0)$ verallgemeinertes Netz, m Markierung von N, p $\in P$ Platz und $Q \subseteq P$ Platzmenge.

a) m *erreichbar* in N, wenn $m \in R_N(m_0)$.

b) m *überdeckbar* in N, wenn eine in N erreichbare Markierung m' mit $m' \geq m$ existiert.

c) p *beschränkt* in N, wenn es eine nat. Zahl k gibt, so daß für alle $m \in R_N(m_0)$ gilt $m(p) \leq k$.

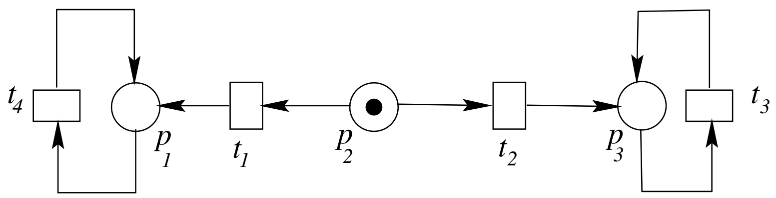
d) Q *beschränkt* in N, wenn alle $p \in Q$ beschränkt in N sind.

e) N *beschränkt*, wenn P beschränkt in N ist.

f) p *sicher unbeschränkt* in N, wenn für jede nat. Zahl k und alle $m \in R_N(m_0)$ ein $m' \in R_N(m)$ existiert mit $m'(p) \succ k$.

g) Q *simultan unbeschränkt* in N, wenn kein $p \in Q$ beschränkt in N ist.

Beispiel:



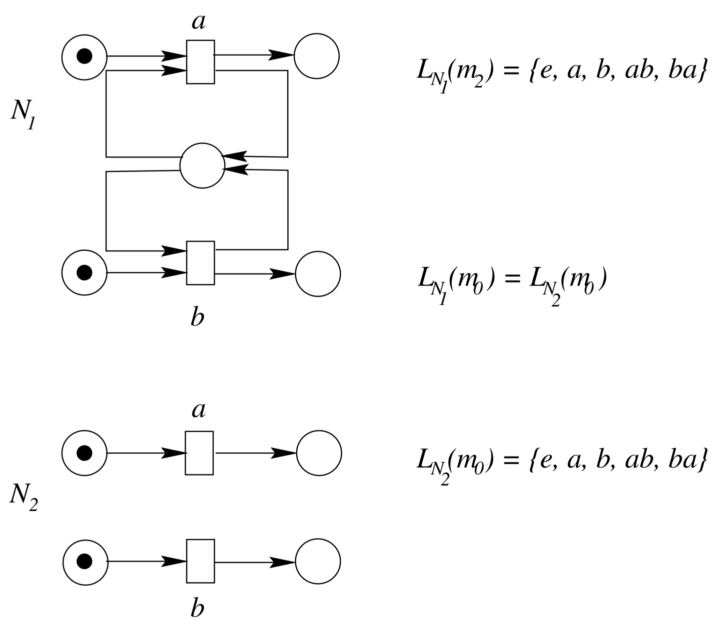
$m_0 = (0, 1, 0)$,
$m_1 = (0, 0, 2)$ erreichbar,
$m_2 = (2, 0, 0)$ nicht erreichbar, aber überdeckbar durch $m = (3, 0, 0)$,
$\mathrm{m}_3 = (0, 2, 0)$ nicht erreichbar und nicht überdeckbar.

2.4 Petri-Netz-Sprachen

Asynchrone Prozeßmodellierung durch Petri-Netze ohne zentrale Zeitskala für Synchronisation des Eintretens von Ereignissen (Aktivierung von Transaktionen). Aktivierbare Transitionen, die nicht in Konflikt stehen, können in beliebiger Reihenfolge bzw. nebenläufig (gleichzeitig) eintreten. Diese möglichen Reihenfolgen (Protokolle, Transitionswörter) des Eintretens von Transitionen in einem Petri-Netz $N = (P, T, F, m_0)$ bei der Anfangsmarkierung m_0 beschreibt die Petri-Netz-Sprache $L_N(m_0)$ als die Menge aller Transitionswörter $q \in W(T)$, die bei m_0 beginnend eintreten (gefeuert werden) können.

Definition: *Netz-Sprache*:
$L_N(m_0) = \{q | q \in W(T) \wedge (m_0, q) \in Df_N\}$, ($Df_N$ Definitionsbereich von f_N).
Wenn nur diejenigen Transitionsfolgen interessieren, die zu bestimmten Endmarkierungen m bzw. Menge M von Endmarkierungen führen, ergeben sich die Sprachen:
$L_N(m_0, m) = \{q | q \in W(T) \wedge (m_0, q) \in Df_N \wedge f_N(m_0, q) = m\}$ (m Markierung) und
$L_N(m_0, M) = \{q | q \in W(T) \wedge (m_0, q) \in Df_N \wedge f_N(m_0, q) \in M\}$ (M Markierungsmenge).
Die Sprachen $L_N(m_0), L_N(m_0, m), L_N(m_0, M)$ beschreiben nur die möglichen Sequentialisierungen des Eintretens von Transaktionen (Ereignissen), aber nicht die Nebenläufigkeitsstruktur der durch das Netz beschriebenen Prozesse.

Beispiel:



Wie schon beim Übergang von verallgemeinerten zu einfachen Netzen benutzt, werden auch bei praktischen Anwendungen vielfach Hilfstransaktionen eingeführt, die im eigentlich beschriebenen Prozeß keine eigenständige Bedeutung besitzen und deren Eintreten nicht protokolliert bzw. nicht beachtet zu werden braucht. Solche Hilfstransitionen können aus den Protokollen gestrichen bzw. verschiedene Transitionen identifiziert werden. Wir nutzen dazu die Begriffe *Homomorphismus* und *Umbenennung*.

Definition: Eine Wortfunktion $h : W(X) \to W(Y)$ mit X, Y – endliche nichtleere Mengen heißt *Homomorphismus* von X und Y, wenn für alle Wörter $p, q \in W(X)$ gilt: $h(pq) = h(p)h(q)$.
$HOM(X, Y)$ bezeichnet die Menge aller solcher Homomorphismen.

Bemerkung: Für jedes $h \in HOM(X, Y)$ gilt: $h(e) = e$ und $h(x_1, x_2, \cdots, x_n) = h(x_1)h(x_2)\cdots h(x_n)$. Ein Homomorphismus heißt *e-frei*, wenn $h(x) \neq e$ für alle $x \in X$. Die Menge der e-freien Homomorphismen soll durch $eHOM(X, Y)$ bezeichnet werden.

Definition: Ein Homomorphismus $h \in HOM(X, Y)$ wird *Umbenennung* genannt, wenn für alle $x \in X$ gilt: $h(x) \in Y \cup \{e\}$.
Die entsprechenden Gesamtheiten werden durch $UMB(X, Y)$ bzw. $eUMB(X, Y)$ notiert.
In Erweiterung auf Wortmengen (Sprachen) $L \in W(X)$ führen wir ein:
$h(L) = \ \{h(q)|q \in L\}$.
Unter einer *Sprachfamilie* Λ verstehen wir ein System von Wortmengen über endlichen Grundalphabeten und bezeichnen durch REG bzw. Λ_3 die Familie aller *regulären* Sprachen, durch CF bzw. Λ_2 die Familie der *kontextfreien* Sprachen, durch CS bzw. Λ_1 die Familie der *kontextabhängigen* Sprachen, durch RA bzw. Λ_0 die Familie der *rekursiv-aufzählbaren* Sprachen und durch FIN bzw. Λ_{fin} die Familie der *endlichen* Sprachen, wofür gilt:
FIN $\subset$ REG $\subset$ CF $\subset$ CS $\subset$ RA (Chomsky-Hierarchie).
Es seien $N = (P, T, F, m_0)$ endliche Petri-Netze mit unbeschränkter Kapazität und FN $= \{L_N(m_0)|N$ endliches Petri-Netz$\}$ die Familie aller (umbenennungs-) *freien Petri-Netz-Sprachen*, sowie
FTN $= \{L_N(m_0, M)|N$ endliches Petri-Netz und M endliche Markierungsmenge $\}$ die Familie aller *freien terminalen Petri-Netz-Sprachen*.
Durch Umbenennungen bzw. e-freie Umbenennungen enstehen aus freien terminalen Petri-Netz-Sprachen die Sprachfamilien UTN $=\{h(L)|L \in$ FTN $\wedge h \in UMB\}$ sowie eUTN $=\{h(L)|L \in$ FTN $\wedge h \in eUMB\}$, hierfür gelten die Beziehungen
FTN $\subseteq$ eUTN $\subseteq$ UTN $\subseteq$ RA und REG $\subset$ eUTN $\subset$ CS $\subset$ RA.

Operationen für Netz-Sprachen:

1. **Definition:** *Mischung* (*Shuffle-Produkt*):

   $u||v$ für Wörter $u, v \in W(X)$

   $u||e = e||u = \{u\}$

   $xu||yv = \{x\} \cdot (u||yv) \cup \{y\} \cdot (xu||v)$

   Die Mischung $u||v$ der Wörter u und v ist die Menge aller Wörter, die aus dem Wort u entstehen, wenn alle Buchstaben von v in der durch v gegebenenen Reihenfolge von links nach rechts vor, zwischen oder hinter die Buchstaben von u geschrieben werden, d.h., $u||v = \{u_1v_1u_2v_2 \, ... \, u_kv_l | u_i, v_j \in W(X), u = u_1 \, ... \, u_k, v = v_1 \, ... \, v_l\}$.

   Die Mischung beschreibt die Parallelverarbeitung der Wörter u und v in zwei unabhängigen Einheiten. FTN wird demnach gegen Mischung allgemein nicht abgeschlossen sein. Treten aber in den FTN-Sprachen L_1 und L_2 keine gemeinsamen Buchstaben auf (die entsprechenden Netze können vollständig parallel arbeiten), dann wird auch ihre Mischung in FTN sein.

   Analog für Sprachen L_i über X sei: $L_1 \, ||L_2 = \bigcup\limits_{u_i \in L_i} u_1||u_2$.

   Beispiele : a$||bc = \{abc, bac, bca\}$, $\quad a||aba^2b = \{a^2ba^2b, aba^3b, aba^2ba\}$.

2. **Definition:** *Produkt* und *Iteration*:

   L, L_i seien Sprachen über X.

   Produkt: $L_1 \circ L_2 = \{uv | u \in L_1 \wedge v \in L_2\}$,
   Produktiteration: $L^* = \bigcup\limits_{i \geq 0} L'_i$ mit $L'_0 = e$ und $L'_i = L \circ L'_{i-1}$ für $i \geq 1$,
   Mischiteration : $L^{||} = \bigcup\limits_{i \geq 0} L_i$ mit $L_0 = e$ und $L_i = L||L_{i-1}$ für $i \geq 1$.

   Satz:

   a) Beide Iterationsoperationen sind Hüllenoperationen.

   b) $L||\{e\} = L$

   c) $L_1||L_2 = L_2||L_1$

   d) $L_1||(L_2||L_3) = (L_1||L_2)||L_3$

   e) $L_1||(L_2 \cup L_3) = L_1||L_2 \cup L_1||L_3$

   f) $L_1 \circ L_2 \subseteq L_1||L_2$

g) FIN, REG, CS und RA sind gegen Mischen abgeschlossen.

h) CF und FTN sind nicht abgeschlossen gegen Mischen.

i) eUTN und UTN sind abgeschlossen gegen Mischen.

j) Für alle $L_1, L_2 \in CF$ gilt: $L_1 || L_2 \in$ CF $\Leftrightarrow L_1 \in$ REG $\vee L_2 \in$ REG .

Satz: Für $L \subseteq W(X)$ sei $X_L = \bigcap\{X' | L \subseteq W(X')\}$ das kleinste Alphabet (*Minimalalphabet*), über dem L Sprache über X' ist.
Dann gilt für disjunkte Minimalalphabete :
$L_1, L_2 \in$ FTN $\wedge (X_{L_1} \cap X_{L_2} = \emptyset) \Rightarrow L_1 || L_2 \in$ FTN.

3. **Definition:** <u>*Restriktion*</u> (Verallgemeinerter Durchschnitt): Für Sprachen L_i über X.
$L_1 \sqcap L_2 = \{L_1 || (X_{L_2} \backslash X_{L_1})^*\} \cap \{L_2 || (X_{L_1} \backslash X_{L_2})^*\}$.

Bemerkung:
Falls $X_{L_1} = X_{L_2}$ gilt $L_1 \sqcap L_2 = L_1 \cap L_2$ (Bei gleichen Minimalalphabeten Übereinstimmung mit normalem Durchschnitt).

Definition: Wenn $h \in HOM(X, Y)$ Homomorphismus und $L \subseteq W(Y)$, so heißt die Sprache $h^{-1}(L) = \{p | h(p) \in L\}$ durch Anwendung des *inversen Homomorphismus* h^{-1} aus L entstanden.

Darüber gilt der
Satz:
Für $L_i \subseteq W(Y)$ und $h_i \in HOM((X_{L_1} \cup X_{L_2}), X_{L_i}), i = 1, 2$ sei
$h_i(x) = x \quad$ falls $x \in X_{L_i}, \quad$ sonst $h_i(x) = e$.
Dann gilt
$h_i^{-1}(L_i) = L_i || (X_{L_{3-i}} \backslash X_{L_i})^* \quad$ für $i = 1, 2 \quad$ und $\quad L_1 \sqcap L_2 = h_1^{-1}(L_1) \cap h_2^{-1}(L_2)$.

Eigenschaften:

a) REG, CS und RA sind gegen Restriktion abgeschlossen.

b) Wegen obiger Bemerkung ist CF aber gegen Restriktion nicht abgeschlossen.

c) FTN, eUTN und UTN sind abgeschlossen gegen Restriktion und gegen inversen Homomorphismen.

4. **Definition:** *Linksderivat* und *Rechtsquotient*:

$D_pL = \{q|pq \in L\}$ heißt <u>*Linksderivat*</u> der Sprache $L \subseteq W(Y)$ für das Wort p.

$L_1/L_2 = \{p|\exists q(q \in L_2 \wedge pq \in L_1)\}$ heißt <u>*Rechtsquotient*</u> von L_1 bezüglich L_2.

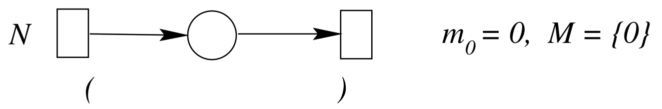
Bemerkung: Die vorgenannten Sprachfamilien sind gegen Linksderivat- und Rechtsquotientbildung mit endlichen Sprachen abgeschlossen.

Die Tabelle gibt die Abgeschlossenheitseigenschaften der behandelten Sprachfamilien gegenüber den angegebenen Operationen insgesamt wieder.

Operation	*FTN*	*eUTN*	*UTN*
Vereinigung	*nein*	*ja*	*ja*
Durchschnitt	*ja*	*ja*	*ja*
Komplement	*nein*	*nein*	*nein*
Durchschnitt mit regulären Sprachen	*nein*	*ja*	*ja*
Produkt	*nein*	*ja*	*ja*
Hülle	*nein*	*?*	*?*
Linksderivat	*ja*	*ja*	*ja*
Rechtsquotient mit endlichen Sprachen	*ja*	*ja*	*ja*
Homomorphismen	*nein*		*ja*
e-freie Homomorphismen	*nein*	*ja*	*ja*
inverse Homomorphismen	*ja*	*ja*	*ja*
Mischung	*nein*	*ja*	*ja*
Restriktion	*ja*	*ja*	*ja*

Abgeschlossenheitseigenschaften

Definition: Unter einer *Dyck-Sprache* D_1 versteht man die Menge aller richtig gebildeten Klammergebirge aus einer Sorte von Klammern (,).
$D_1 = \{(), (()), (()(())), ...\} = D_{(,)} = L_N \ (m_0, M)$ Petri-Netz-Sprache des Netzes N mit $m_0 = 0$ und $M = \{0\}$.



Darüber gilt der **Satz**:

FTN ist die kleinste Familie von Sprachen, die die *Dyck-Sprachen* D_1 enthält und abgeschlossen ist gegen Linksderivat und Rechtsquotient mit endlichen Sprachen, inversen Homomorphismen und Restriktion (bzw. Durchschnitt).

3. Anwendungen

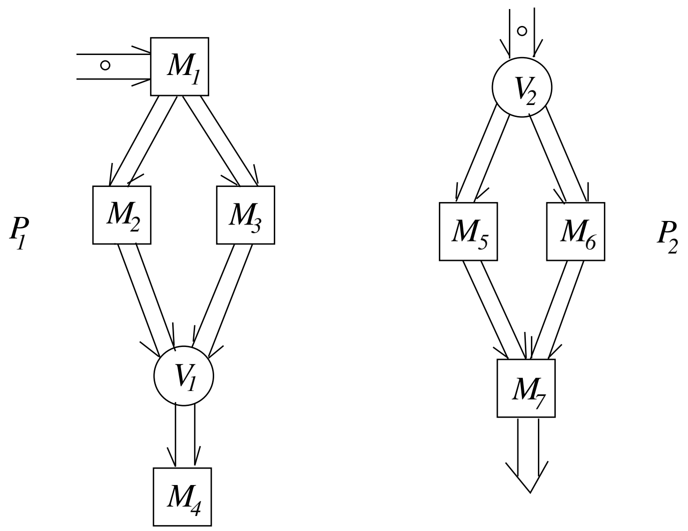
3.1. Produktionssysteme

Modellierung des Objektstromes in Produktionssystemen, d.h. Systemen aus Maschinen, Verteilern und Verbindungskanälen.

Maschinen können arbeiten, wenn alle zu verarbeitenden Eingabeobjekte vorhanden sind. Die von Maschinen produzierten Ausgabeobjekte werden über den Ausgabekanal weitergeleitet.

Verteiler steuern die Objektströme, indem sie jeweils ein Objekt von einem ihrer Eingabekanäle aufnehmen und unverändert an Ausgabekanäle übertragen.

Das Gesamtsystem arbeitet asynchron. Im Beispiel

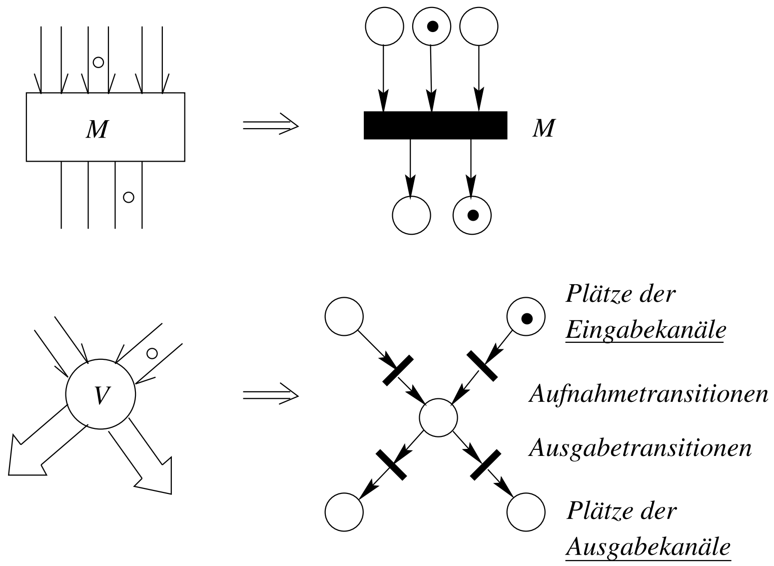


sind die Maschinen durch Quadrate, Verteiler als Kreise und Kanäle als Pfeile dargestellt. Im System P_1 erzeugt die Maschine M_1 die Eingangsobjekte für die Maschinen (gleichen Typs) M_2 und M_3. Die Ausgänge dieser Maschinen sind über den Verteiler V_1 mit der Maschine M_4 verbunden. Bei V_1 kann es zur Überlastung (Akkumulation) kommen, wenn die Verarbeitungsgeschwindigkeit der Maschine M_4 nicht aussreicht. In diesem Fall (Stau) muß mit einem Stillstand von M_2 oder M_3 gerechnet werden.
Bei dem System P_2 verteilt der Verteiler V_2 die Eingangsobjekte auf die Maschinen M_5 und M_6. M_7 kann nur dann arbeiten, wenn M_5 und M_6 entsprechende Ausgangsobjekte produziert haben. Falls der Verteiler eine dieser Maschinen bevorzugt, kann es zum Stillstand der anderen und damit auch von M_7 kommen.

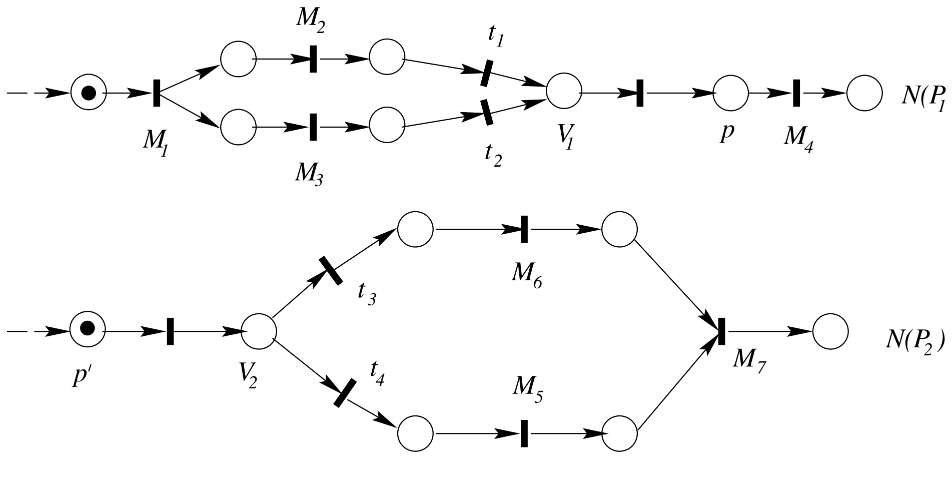
Ziel ist es, durch geeignete Strukturen (Vernetzung) derartige Störungen zu vermeiden bzw. diese zu erkennen. Dazu stellen wir solche Produktionssysteme durch spezielle Petri-Netze vom Typ free-choice dar. Auftreten von Akkumulation bedeutet Unsicherheit des Petri-Netzes. Stillstand entspricht der Nichtlebendigkeit des Netzes.

Objekten (bzw. <u>*Kanälen*</u>, bzw. <u>*Maschinen*</u>, bzw. <u>*Verteiler*</u>) werden Marken (bzw. Plätze, bzw. Transitionen, bzw. Teilnetze) zugeordnet.

Elemente von Produktionssystemen



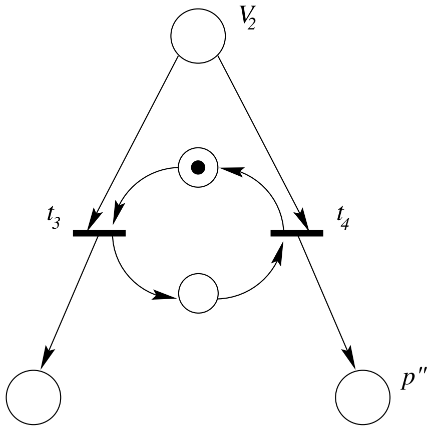
Die vorher angegebenen Systeme sind dann wie folgt dargestellt.



Netz

Im Netz $N(P_1)$ des Produktionssystems P_1 sind die Plätze V_1 (innerer Platz des Verteilers) und p unsicher, da bei der angenommenen Anfangsmarkierung Folgemarkierungen mit 2 Marken bei den Plätzen V_1 bzw. p entstehen, womit ihre Kapazität überschritten wird. Ein Platz, der einem Kanal des Produktionssystems entspricht, ist genau dann unsicher, wenn bei der der Anfangsbelegung entsprechenden Objektbelegung im System auf dem zugeordneten Kanal Akkumulation auftritt.

Im Netz $N(P_2)$ des Systems P_2 stehen die Transitionen t_3 und t_4 in Konflikt, sobald die Eingangstransition des Verteilers V_2 gefeuert hat. Wird dieser Konflikt immer zugunsten von t_4 gelöst, kommt es zum Stillstand, so daß M_6 und M_7 niemals Konzession erhalten. Dieser Stillstand kann durch Einführung eines *Regulationskreises* zwischen den Ausgangstransitionen des Verteilers vermieden werden.



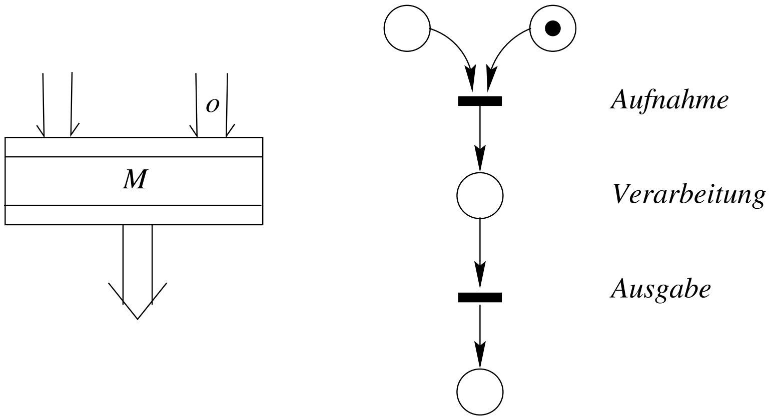
Es besteht offensichtlich ein Zusammenhang zwischen dem Stillstand eines Teilsystems in einem Produktionssystem und dem Tod der entsprechenden Transition im zugehörigen Netz. Das Petri-Netz kann auf Akkumulation (Sicherheit) oder Stillstand (Lebendigkeit) hin untersucht werden.

Bemerkung:

Jedes Produktionssystem läßt sich durch ein Petri-Netz $N = (P, T, F)$ mit free-choice Eigenschaft (FC-Netz) , d.h. wenn $(p, t) \in F$, dann ist $pF = \{t\}$ oder $Ft = \{p\}$. (Wenn p einem Kanal, t einer Maschine oder t einer Eingangstransition eines Verteilers entspricht, dann ist t die einzige Nachtransition von p, da Kanäle sich nicht verzweigen. Wenn p innerer Platz eines Verteilers, dann ist t Ausgangstransition dieses Verteilers, womit p einziger Ausgangsplatz dieser Transition ist.)

Bei FC-Netzen besteht zur Lösung des Konflikts freie Auswahl unter den beteiligten Transitionen, da diese keine weiteren Vorplätze besitzen. Für FC-Netze ist Lebendigkeit und Sicherheit entscheidbar, damit die o.g. Störungen (Stillstand und Akkumulation) kontrollierbar.Bei allgemeinen Netzen ist dies nicht gegeben.

Maschinen können durch Teilnetze repräsentiert sein,

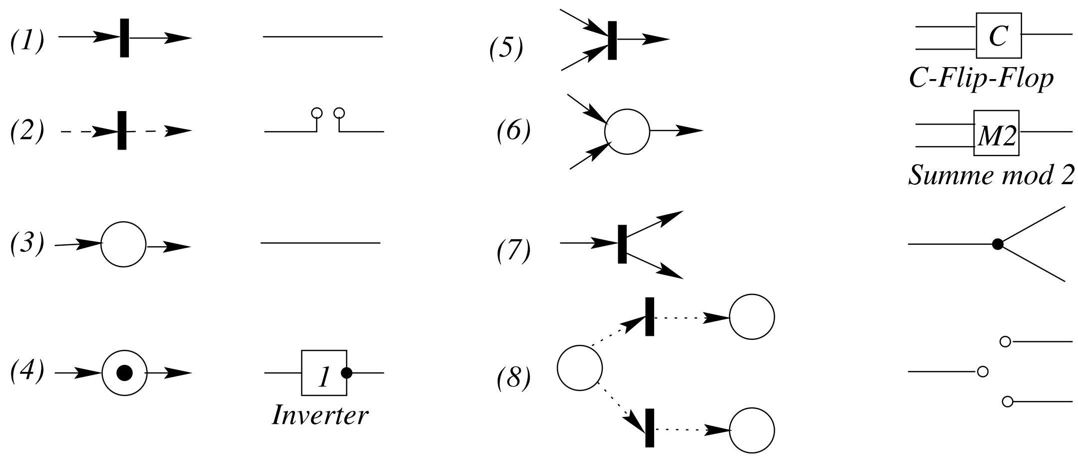


womit dann zeitliches Verhalten der aktiven Elemente des Systems beschreibbar wird. Den Plätzen können Verzögerungszeiten zugeordnet werden. Die Aussage der o.g. Bemerkung bleibt bestehen.

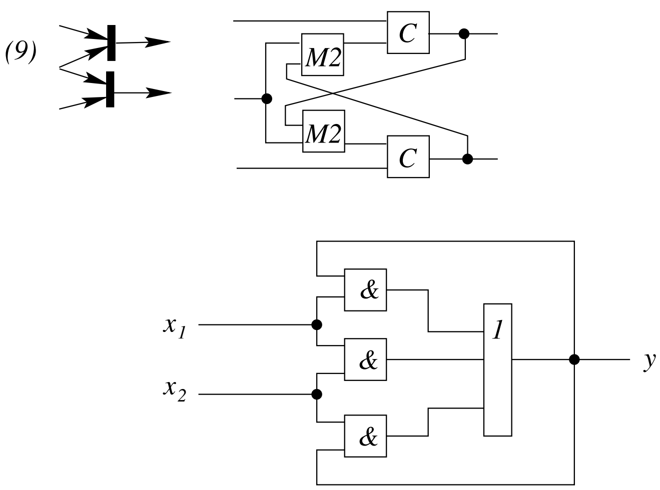
3.2. Schaltwerke

Schaltwerke sind Strukturbeschreibungen von Automaten (digitale sequentielle Systeme) unter Verwendung vorgegebener Schaltelemente (logische bzw. speichernde). Petri-Netze können zur Modellierung socher Schaltwerke, deren Analyse und Synthese eingesetzt werden. Dabei ist zu berücksichtigen, daß das Schaltwerk mit seiner Umgebung, z.B. als Teil einer größeren Steuerung, durch Ein- und Ausgaben in Wechselwirkung tritt.

Wir stellen die Schaltelemente durch spezielle Netze dar.



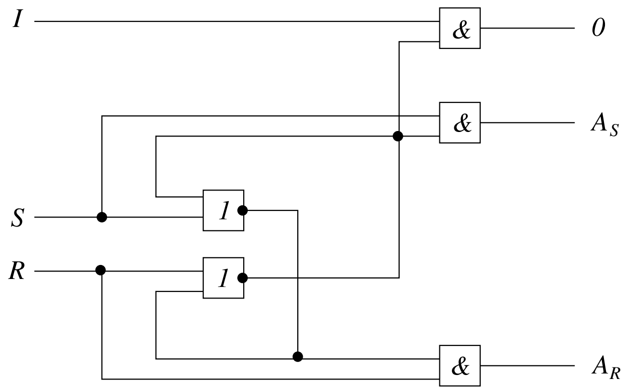
und setzen das gesamte Schaltwerk aus diesen Teilnetzen zusammen, wobei zu beachten ist, daß jede Transition höchstens einen Eingangsplatz mit einer anderen Transitionen teilen darf (einfaches Netz).



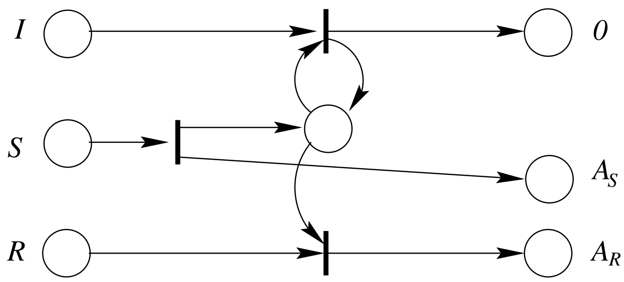
Schaltwerke

Die darstellenden markierten Petri-Netze sollen sicher und Konflikte sollen immer von außen (durch binäre Entscheidungen) lösbar sein. Einer Marke im Netz entspricht eine Flanke (Zustandsänderung) im Schaltwerk. Man unterscheidet dann zwischen passiven (logische Verknüpfung ohne Speicher) und aktiven (logische Verknüpfung mit Speicher) Schaltungen.

p-Modul als Schaltwerk



Das C-Flip-Flop nimmt z.B. am Ausgang den Wert 1 an, sobald beide Eingänge den Wert 1 erhalten (siehe Abbildung). Das Gesamtschaltwerk kann aus einzelnen Modulen aufgebaut werden, die selbst wieder durch Petri-Netze beschrieben sind. Das Netz aus der Abbildung beschreibt z.B. einen *Modul*,

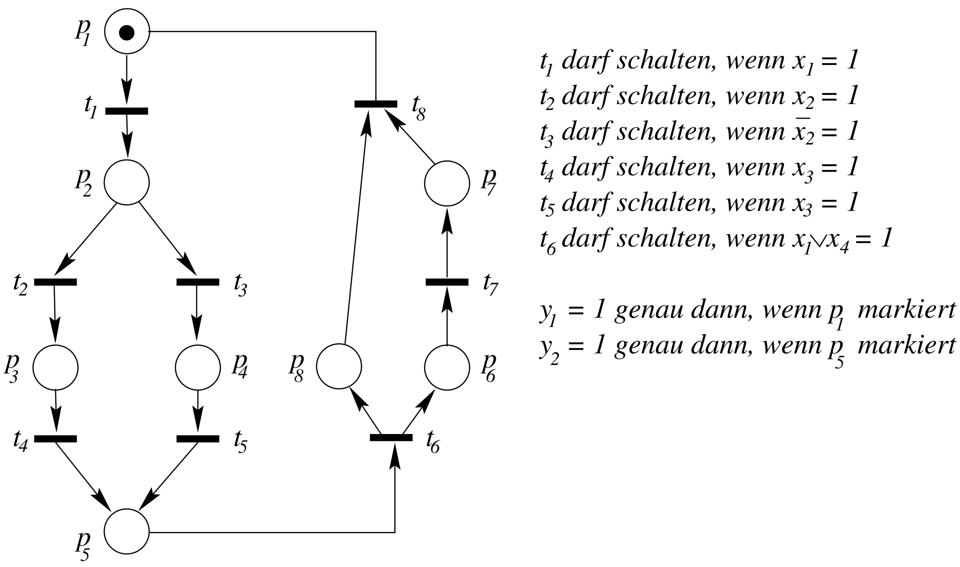


p-Modul als Netz

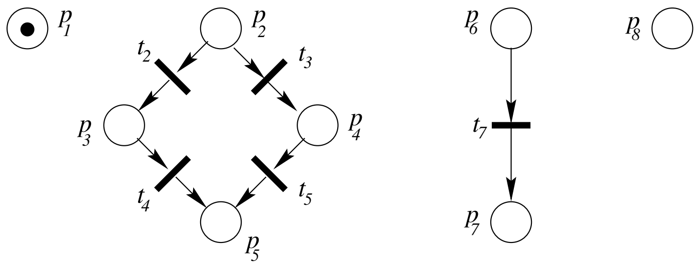
wo eine Marke bei S das *Setzen* (Ausgang A_S wird markiert) und bei R das *Rücksetzen* (Ausgang A_R wird markiert) bewirkt. Marken bei I werden genau dann durchgeleitet zu O, wenn der Modul gesetzt ist. I und R bzw. S und R dürfen nicht gleichzeitig markiert sein.(Marken werden hier als Zustandsänderung von 0 auf 1 interpretiert.)

Bei vielen Aufgaben mit Schaltwerken führt eine Zerlegung der Gesamtschaltung in Module zur Beherrschung der Komplexität. Dies entspricht einer Zerlegung des zugeordneten Petri-Netzes in Teilnetze. Dabei wird von einem lebendigen und sicheren Netz ausgegangen. Eingaben im Schaltwerk werden als Zusatzbedingungen für das Schalten von Transitionen im Netz dargestellt. Die Ausgaben ergeben sich aus der Markierung.

Beispiel:

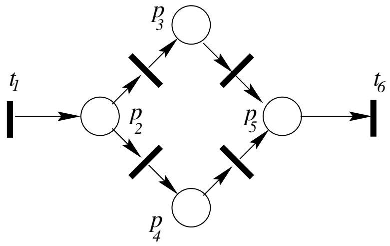


Die Funktion des Gesamtnetzes kann durch eine Automatentafel beschrieben werden. Bei großen Netzen ist dies aber sehr aufwendig. Besser ist dann eine Zerlegung des Gesamtnetzes in Zustandsmaschinen, die höchstens eine Marke tragen. Diesen Zustandsmaschinen entsprechen Automaten, dessen Zustände durch die Plätze des Teilnetzes gegeben sind. Die Teilnetze werden als Makroplätze bezeichnet und können zur Konstruktion von größeren Netzen als Plätze verwendet werden.



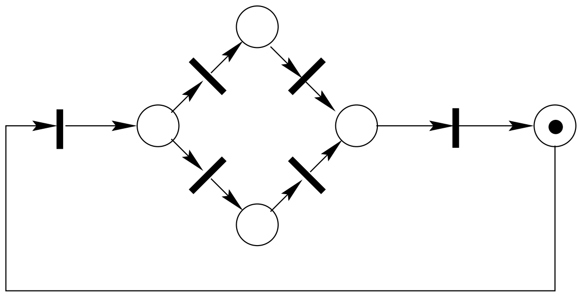
Makroplätze

Zur Analyse von Schaltwerken werden ebenfalls Petri-Netze eingeführt, wobei das Verhalten von *Gattern* (Verknüpfungselementen) und *Speicherelementen* (z.B. Flip-Flops) durch Teilnetze modelliert wird.



ohne Zustandsmaschine

Den Ein- und Ausgängen des Schaltwerkes entsprechen Ein- uns Ausgangsplätze (Endplätze) des Netzes.



mit Ruheplatz

Die Netzmarkierung beschreibt die Zustände der Schaltelemente und die vorhandenen Signalflanken. Eine tote Markierung entspricht einem stabilen (nur von außen veränderlichen) Zustand des Schaltwerkes. Die möglichen Prozeßverläufe des Petri-Netzes korrespondieren mit denen des Schaltwerkes. Hazards (Wettläufe) können dort auftreten, wo der entsprechende Platz im Netz nicht sicher ist. Ein Teil des Schaltwerkes kann instabil werden, wenn eine Markierung erreichbar ist, bei der der betreffende Teil des Netzes lebendig ist. Spezielle damit zusammenhängende Eigenschaften des Schaltwerkes können über den Erreichbarkeitsgraphen geklärt werden. Für unterschiedliche Typen von Schaltwerken und unterschiedliche Analyse- und Synthesemethoden werden unterschiedliche Petri-Netz-Modelle verwendet.

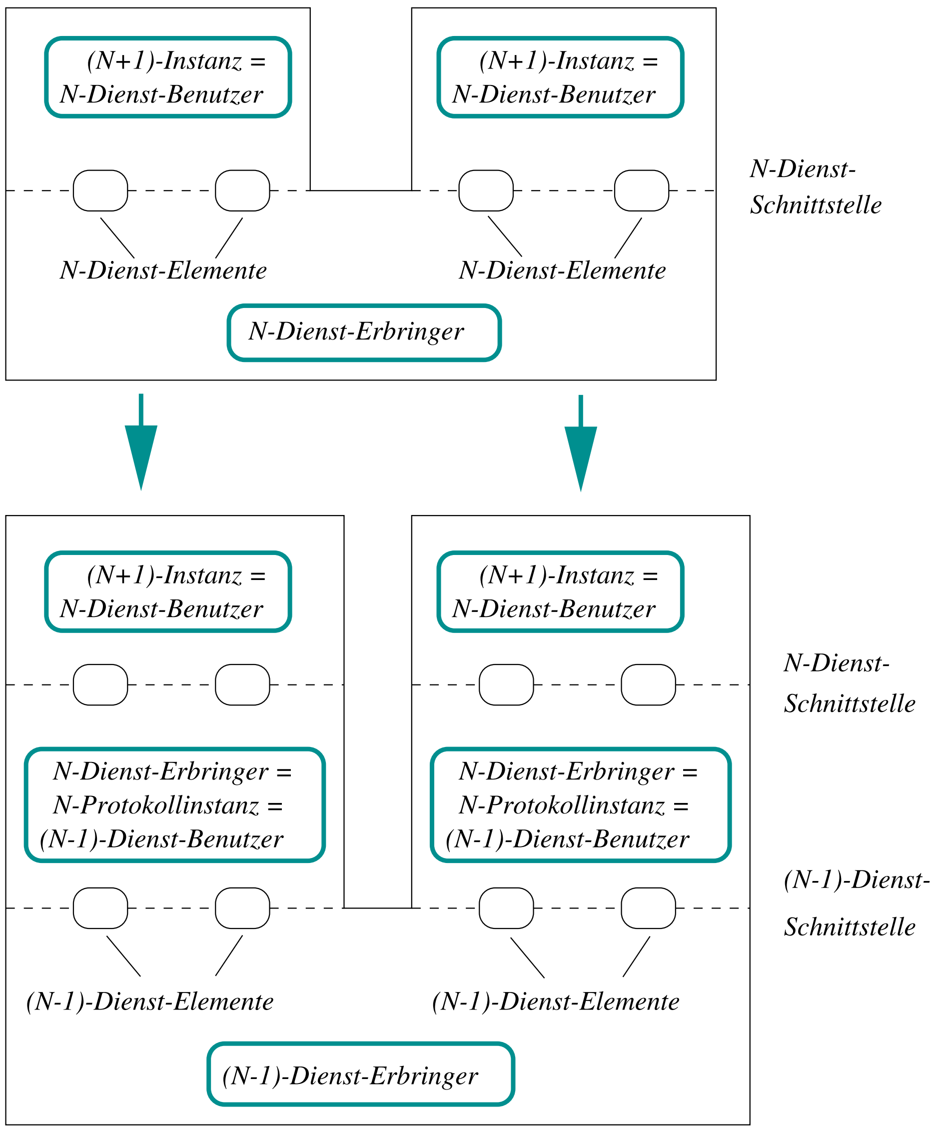
3.3 Kommunikationsnetze

Kommunikation dient der Übertragung von Information (Daten, Signale, Befehle u.a.) zwischen verschiedenen Partnern (Agenten), die nebenläufig und anteilig arbeiten. Die Kommunikation selbst unterliegt Protokollen, die eine Übereinkunft zwischen den Partnern über den Informationsaustausch festlegen. Sie sind Verhaltensvorschriften für Teile des Gesamtsystems zur Erreichung von gemeinsamen Zielen, die zu erreichen bzw. zu vermeiden sind. Unter einem Protokoll kann die Gesamtheit der Algorithmen verstanden werden, die jeder Systemteil (Agent) ausführen muß, um das Übertragungsziel zu erreichen oder Nachteile zu vermeiden. Zu beachten ist dabei, daß die Agenten räumlich getrennt sind, auf getrennten Datenbeständen arbeiten, unterschiedliche Datenformate benutzen, nebenläufig arbeiten und unterschiedliches Verhalten (Plattformen) besitzen. Insbesondere der Verteiltheitaspekt ist für das Kommunikationssystem charakteristisch. Dabei treten Störfaktoren (Nachrichtenverlust, Verfälschung, Ausfälle, Verklemmungen, Unverträglichkeiten) auf, die durch geeignete Protkolle zu berücksichtigen und wenn möglich, zu vermeiden bzw. auszugleichen sind.

Aufgabenbereiche von Kommunikationsprotokollen:

- Datenübertragung (gesicherter Daten- und Signalaustausch)
- Wechselseitiger Ausschluß (Vermeidung unerwünschter Zustandskombinationen bzw. Aktionsfolgen)
- Fehlertoleranz (Resistenz gegen Ausfälle oder Fehlfunktionen)

Zur Beschreibung von Kommunikationsprotokollen bedient man sich dem Paradigma der virtuellen Maschinen auf hierarchisch gestuften Sprachebenen, als verteilte Systeme. Die Ausführung von Protokollen bzw. Teilprotokollen wird von Diensten übernommen, die durch das System spezifiziert werden. Die Dienstbeschreibung spezifiziert deren Verfügbarkeit, Benutzung und Wirkung auf unterschiedlichen Ebenen (Stufen). Aufgabe eines Protokolls ist es, einen <u>*Dienst*</u> unter Benutzung eines Dienstes einer niedrigeren Ebene zu erbringen.



Hierarchisches Dienst-/Protokollmodell

Dadurch wird ein Dienst in eine höhere (abstraktere) Ebene gehoben. Eine solche hierarchische Kommunikationsarchitektur wird in einzelne Module gegliedert, die hierarchisch aufeinander aufgebaut werden. Durch die internationale Normungsorganisation (ISO) wurde das folgende *OSI-Referenzmodell* (Open Systems Interconnection) eingeführt, dem die meisten hierarchischen Kommunikationsprotokolle folgen.

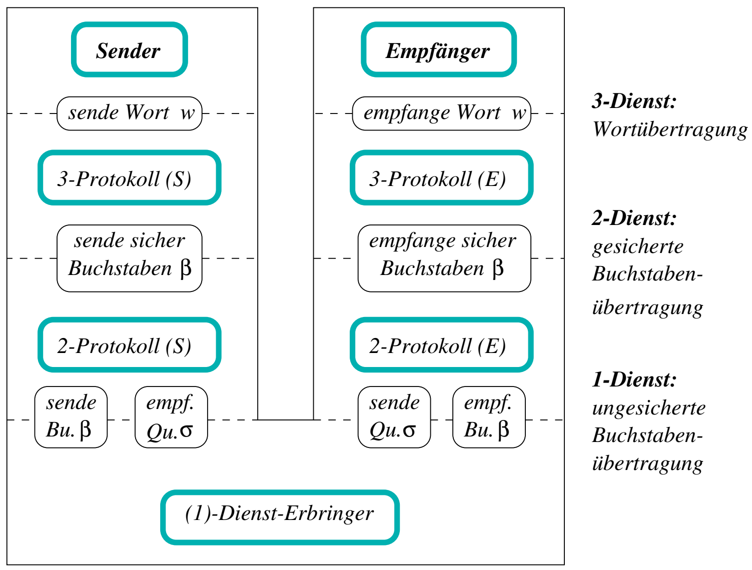
Anwendung	*Anwendung*
Darstellung	*Darstellung*
Kommunikationssteuerung	*Kommunikationssteuerung*
Transport	*Transport*
Vermittlung	*Vermittlung*
Sicherung	*Sicherung*
Bitübertragung	*Bitübertragung*
Übertragungsmedium	

Schema des OSI-Referenzmodells

Zur vereinfachten Beschreibung einer derartigen Protokollhierarchie durch Petri-Netze gehen wir von folgender Problemstellung aus:

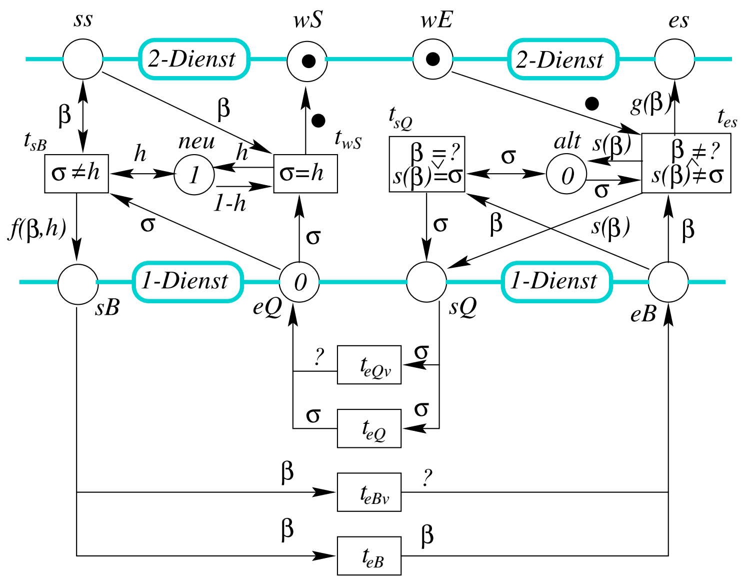
Zwischen zwei Agenten, einem *Sender* und einem *Empfänger*, soll ein Wort $w = w_1 w_2 ... w_k$ über einem Alphabet $\{$A, B ,... , Z$\}$ übertragen werden. Senderseitig soll der Dienst "Sende Wort" und empfängerseitig der Dienst "Empfange Wort" realisiert werden. Zur Verfügung stehen aber nur weniger leistungsfähigere Dienstelemente, wie "Sende Buchstaben" , "Empfange Quittung" senderseitig und "Empfange Buchstaben", "Sende Quittung" empfängerseitig. Mit diesen niedrigeren Diensten können Buchstaben aus einem erweiterten Alphabet $\{$A, B,... , Z, a, b,... , z, $\odot, \oplus\}$ eventuell gestört und ungeordnet vom Sender zum Empfänger übertragen werden. Entsprechendes gilt für die Quittungen $\{0{,}1\}$ vom Empfänger zum Sender. Dieser Dienst soll aber eine Fehlererkennung und die Einhaltung der Ordnung sicherstellen.

Die Agenten lösen das Problem mehrstufig. Auf den gegebenen niederen Dienst (1-Dienst) setzt ein 2-Protokoll auf, welches die Dienstelemente "Sende sicher Buchstabe" und "Empfange sicher Buchstabe" als 2-Dienst von Sender zu Empfänger zur Verfügung stellt (Fehlerkorrektur). Außerdem soll das 2-Protokoll die Einhaltung der Reihenfolge (Ordnung im Wort) garantieren. Auf diesen 2-Dienst setzt nun ein 3-Protokoll auf, was den gewünschten Dienst als 3-Dienst zur Verfügung stellt.



Protokollhierarchie zur Wortübertragung

Diese Protokollhierarchie "Sichere Zeichenübertragung" kann als Petri-Netz (Dienstelemente als Stellen, der Übersichtlichkeit wegen) wie folgt beschrieben werden:



Dabei bedeutet sB - "sende Buchstaben", eQ - "empfange Quittung", sQ - "sende Quittung", eB - "empfange Buchstaben".

Das Netz beschreibt, wie das asymmetrische 2-Protokoll mit Hilfe des 1-Dienstes den 2-Dienst (Zeichen sicher übertragen) erbringt. An den so realisierten 2-Dienst kann durch Ankopplung eines 3-Protokolls an die Dienstelementstellen ss - "sende sicher Buchstaben", wS - "weiter" (Sender-gesteuert), wE - "weiter" (Empfänger-gesteuert), es - "empfange sicher Buchstaben" der gewünschte Dienst (3-Dienst) ausgeführt werden.

Das 2-Protokoll benutzt die Kanäle so, daß immer höchstens ein "Nutz"-Buchstabe vom Sender zum Empfänger bzw. eine Quittung in umgekehrter Richtung transportiert wird. Dies garantieren die Dienstschnittstellen (warten auf "weiter"-Marke) und das Quittierungsschema (aufheben des Buchstabens beim Sender bis passende Quittung erfolgt). Der zu übertragende Buchstabe wird vom Sender solange aufgehoben (auf ss) bzw. mehrfach gesendet, bis die der Empfänger den Empfang durch Quittung bestätigt. Dabei wird nach folgendem Sende- und Quittierungsschema verfahren:

Die Satzfunktion s ordnet den Großbuchstaben 1 und den Kleinbuchstaben 0 zu mit $s(X) = 1$, wenn $X \in \{A, B, ..., Z, \oplus\}$ sonst 0.

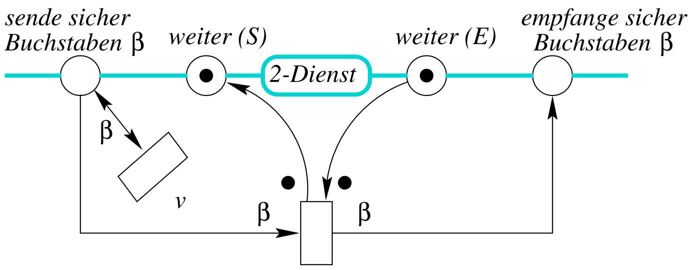
Das 2-Protokoll verwandelt mit der Funktion

$$f(X, 1) = X \text{ und } f(X, 0) = x \text{ Kleinbuchstabe von } X$$

senderseitig jeden zweiten Buchstaben in einen Kleinbuchstaben. Gegebenenfalls wird der Buchstabe mehrfach übertragen.

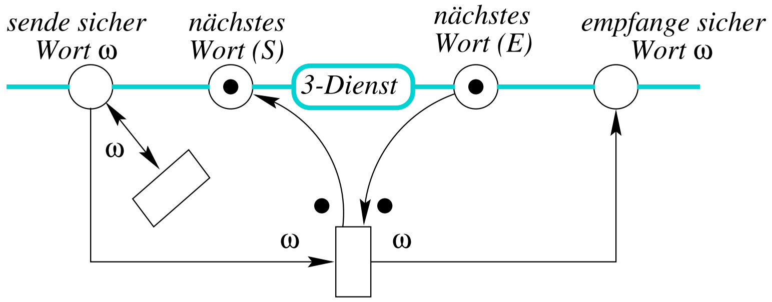
Jedes empfangene Zeichen (echt oder verfälscht) quittiert der Empfänger mit $s(X) = \sigma$ für den zuletzt unverfälscht empfangenen Buchstaben X (anfangs 0). Erreicht diese Quittung den Sender unverfälscht und paßt sie zu dem gesendeten Buchstaben, wird der Wert von s von 1 nach 0 oder umgekehrt gewechselt. Ist die Quittung verfälscht (?) oder unpassend ($s(X) \neq \sigma$), sendet der Sender diesen Buchstaben nochmals. Jedesmal, wenn eine passende Quittung empfangen wird, signalisiert das 2-Protokoll senderseitig durch eine Marke in wS die Bereitschaft für den nächsten Buchstaben. Empfangsseitig wird nur dann der letzte unverfälscht und nicht wiederholt empfangene Buchstabe weitergereicht, wenn wE eine Marke trägt. Gegebenenfalls wird der weitergereichte Buchstaben durch eine Funktion g wieder in einen Großbuchstaben verwandelt.

Um die Korrektheit des 2-Protokolls zu überprüfen, kann die Äquivalenz des vom Protokoll realisierten 2-Dienstes mit der 2-Dienst-Spezifikation untersucht werden. Meist werden dabei bestimmte Dienst-Benutzungsordnungen für Protokolle vorausgesetzt, d.h. auf Netze gegebener Form eingeschränkt.



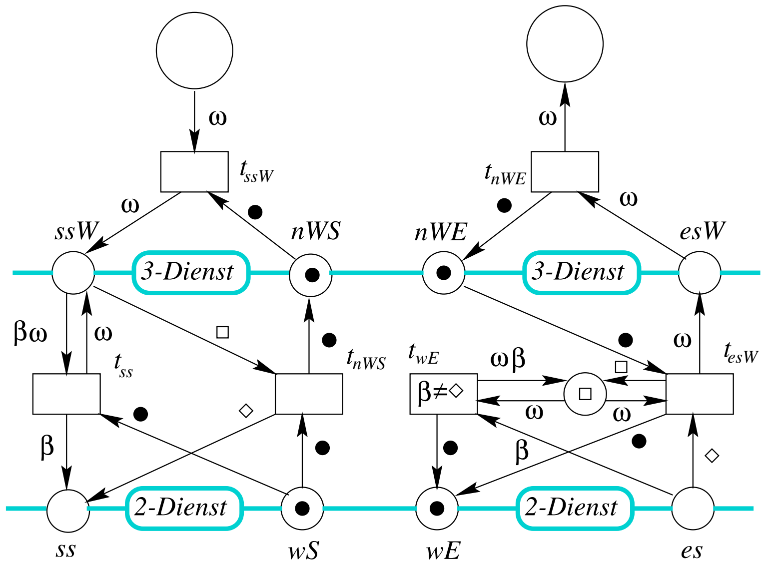
2-Dienst

Auf den realisierten 2-Dienst baut nun das 3-Protokoll auf, welches den schließlich gewünschten 3-Dienst gewährleisten soll. Wir verwenden dazu die 3-Dienstelementstellen ssW - sende sicher Wort, nWS - nächstes Wort senderseitig, nWE - nächstes Wort empfängerseitig, esW - empfange sicher Wort. 3-Dienstspezifikation und das den 3-Dienst erbringende 3-Protokoll sind wie folgt zu beschreiben:



3-Dienst

Ein auf ssW liegendes Wort wird zeichenweise gesendet. Dabei wird ein Zeichen erst dann gesendet, wenn das vorhergehende auf "weiter" senderseitig bestätigt ist. Anschließend wird das Endesymbol ⊕ gesendet. Empfangsseitig wird das Wort wieder zusammengesetzt und weitergereicht, nachdem das Endesymbol ⊕ empfangen und erkannt wurde.



Falls die Korrektheit der vorhergenden Stufen nachgewiesen ist, braucht jetzt nur noch die Kopplung von 2-*Dienstspezifikation* und 3-*Protokoll* untersucht zu werden. Damit kann man bei Korrektheitsbeweisen stufenweise vorgehen und muß nicht das Gesamtnetz untersuchen, welches sehr unübersichtlich und komlex sein kann. Auch dabei wird natürlich eine einheitliche Benutzerordnung der Dienste durch die Protokolle auf den einzelnen Ebenen vorausgesetzt.

INDEX