Petri Netze

Petri Netze

Wolfgang Garn
Für den Bereich:
Regelungsmathematik, Hybridrechner- und Simulationstechnik
Technical University of Vienna

Abstrakt

Austria

The fundamental theory of petri nets will be introduced. It will be presented theoretical as well as with examples. Terms like finite capacity, siphon, trap, etc. will be explained. There will be an introduction for the reachability and the state equation. With the help of an example the reduction method will be explained. Timed PN will be shown.

Die Grundlagen von Petri Netzen werden formal und anschaulich eingeführt. Die Begriffe Finite Capacity, Siphon, Trap, etc. werden erklärt. Es werden die Erreichbarkeit, die Zustandsgleichung vorgestellt. Die Faltung wird anhand eines Beispieles erklärt. Zeitbehaftete PN werden gezeigt.

Einleitung

Motiviert sind die PN durch ihre mannigfache Anwendbarkeit.
PN werden in der Wirtschaft und Industrie häufig verwendet, da sie eine einfache Beschreibung für komplexe Systeme liefern und z.T. analytische als auch bei Simulationen Ergebnisse liefern. PN sind beispielsweise die Schnittstelle zwischen Designern und Managern.
Weit verbreitet sind die PN in der Fertigungs-, Verfahrens-, Energie- und Verkehrstechnik. Sie eignen sich jedoch auch sehr gut zur Modellierung von Rechnersystemen und Netzwerken.
Diese „Arbeit“ versucht die Theoretische Grundlage von Petri Netzen anschaulich und einfach zu vermitteln.

1. Einfache Beispiele

Wir beginnen mit dem klassischen Erzeuger/Verbrauchersystem.

1.1 Erzeuger/Verbrauchersystem

Wir erzeugen „ETWAS“ und senden es ab. Falls „ETWAS“ da ist entnehmen wir es und verbrauchen „ETWAS“. Man kann „ETWAS“ nur absenden, falls es bereits erzeugt wurde. Ebenso ist es mit dem Verbrauchen. Man kann „ETWAS“ nur dann verbrauchen, falls man es bereits entnommen hat.
Die Abbildung 1 gibt genau wieder, daß „ETWAS“ bereits erzeugt wurde und das „ETWAS“ entnommen wurde.

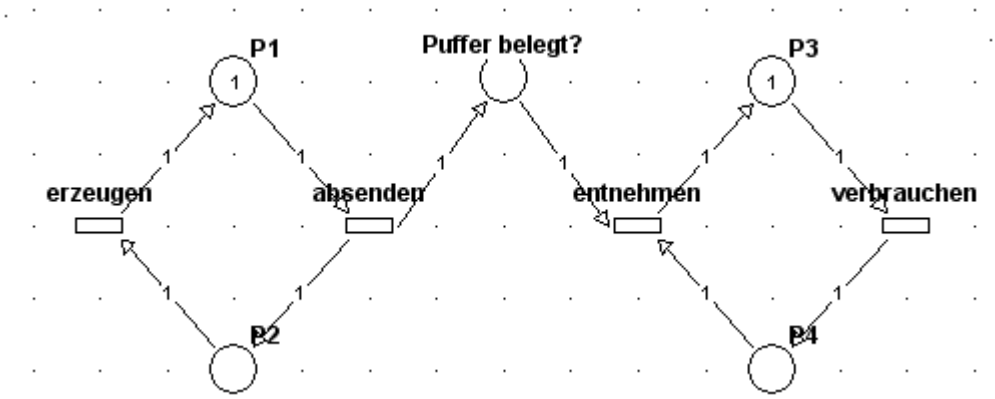


Abbildung 1: Erzeuger/Verbraucher-System

Beim Absenden geben wir „ETWAS“ in den Puffer für das Ereignis-Entnehmen und benachrichtigen die Instanz Erzeuger davon. Ebenso benachrichtigen wir die Instanz Entnehmen davon, daß wir „ETWAS“ verbraucht haben.
Dies ergibt die Abbildung 2.

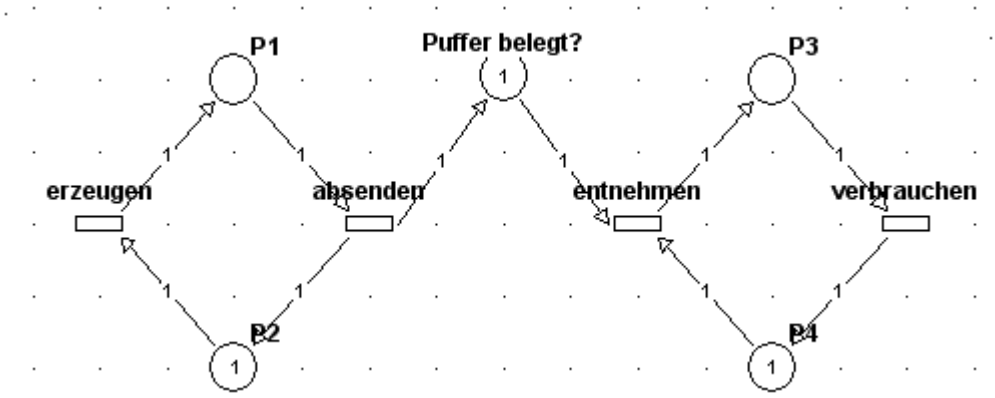


Abbildung 2: Erzeuger/Verbraucher-System nach den „Feuern“

1.2 Übergangsregel

Diese Ereignisse, wie erzeugen, absenden, etc., treten immer nur dann ein, falls alle erforderlichen Bedingungen erfüllt sind(siehe Abbildung 3).

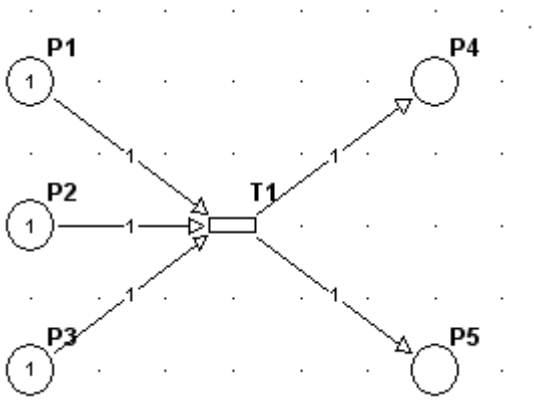


Abbildung 3: Übergangsregel (vor den Feuern)

Die Ereignisse ihrerseits bewirken eine Zustandsänderung.

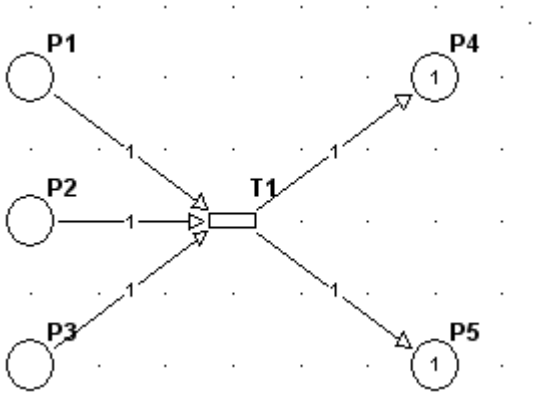


Abbildung 4: Übergangsregel (nach den Feuern)

Der Übergang der Zustände (P1, P2, P3) in die Zustände (P4, P5) kann nur Erfolgen falls die Instanz T1 es „erlaubt". T1 „erlaubt"(feuert) es nur falls alle zu ihm führende Bedingungen erfüllt sind. Sobald die Instanz feuert sind alle wegführenden Bedingungen erfüllt.
Sehen wir uns dazu das vorige Beispiel mit einer kleinen Erweiterung an.

1.3 Erzeuger/Verbraucher mit sequentiellem Puffer

Wir möchten bei diesem Beispiel in der Lage sein sofort zwei „ETWAS" zu erzeugen.
Welche Ereignisse können in der untenstehenden Abbildung überhaupt eintreten?
Nach der vorhin erwähnten Übergangsregel werden nur jene Ereignisse, bei welchen alle hinführenden Bedingungen erfüllt sind, aktiviert. Wir sehen uns also die Kreise mit einer Eins an und folgen dem Pfeil zum relevantem Ereignis. Beim Ereignis angekommen prüfen wir ob die Übergangsregel durchführen können.
Im untenstehenden Beispiel können wir also die Übergangsregel auf die Ereignis: absenden und verbrauchen anwenden.

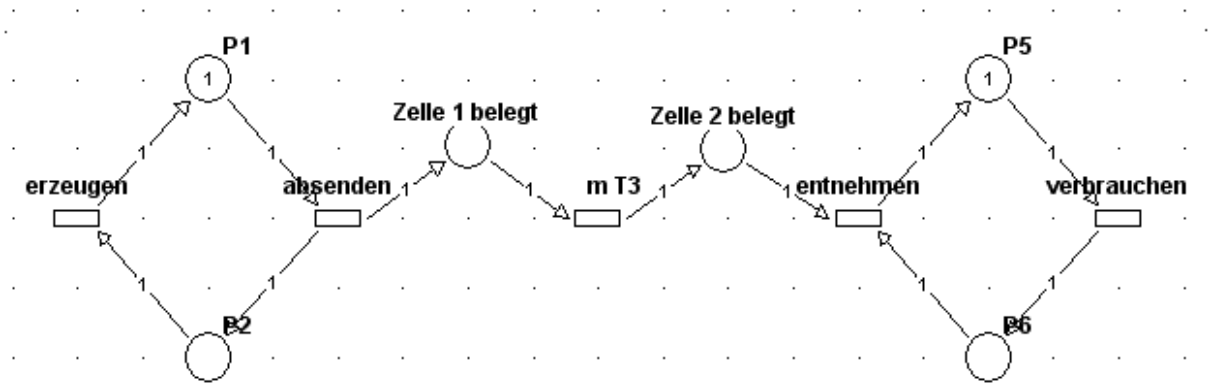


Abbildung 5: Erzeuger/Verbraucher mit sequentiellem Puffer

Damit treten die Zustände „P2" und „Zelle 1 belegt" ein, während die Bedingung P1 reduziert wird. Gleichzeitig kann die Instanz Verbrauchen zuschlagen. (P6 erfüllt, P5 unerfüllt).

Nun betrachten wir die gefüllten Zustände: Durch P2 kommen wir auf das Ereignis Erzeugen und sehen das es eintreten kann. Mit dem Zustand „Zelle 1 belegt“ kommen wir auf „m T3“ auch dieses kann eintreten. Bei der Bedingung P6 kommen wir zwar zum Ereignis Entnehmen, sehen jedoch das „ETWAS“ nicht da ist und somit nichts entnommen werden kann.
Man sieht bereits regelrecht die Bäume für all diese möglichen Übergänge wachsen.

2. Grundlegende Definitionen und Begriffe

2.1 Begriffe Beispielhaft

Die nächste Abbildung zeigt uns mögliche Begriffe für die „Rechtecke“ und „Kreise“. Die wirklich Verwendeten hängen natürlich von der jeweiligen Anwendung(Fachgebiet) ab.

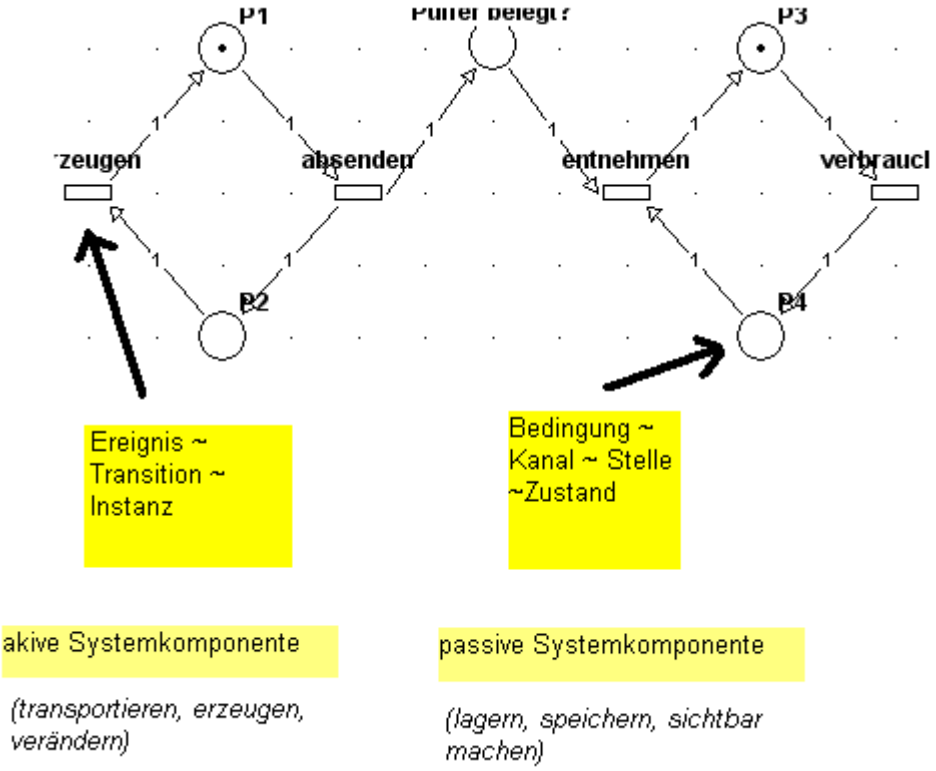


Abbildung 6: Begriffe

Der am meist verbreitet Begriff für die aktive Systemkomponente ist die Transition. Für die passive Systemkomponente ist der Begriff Place international üblich.

2.2 Formale Definitionen:

Damit wir uns auf einer allgemein klaren und verständlichen Ebene bewegen benötigen wir formale Definitionen. Da die Mathematik das Fundament jeder Wissenschaft ist, verwenden wir die Mengenlehre, den Funktionsbegriff und die Matrizenrechnung um unsere ersten Werkzeuge zu erzeugen.

2.2.1 Petri Netz

Ein *Petri Netz* ist ein Fünfer Tupel $P_N = (P, T, A, W, M_0)$ wobei die Komponenten folgende Bedeutung haben: *P* steht für *Places(Bedingung, Kanal, Stelle, Zustand)*; *T* steht für *Transitions(Ereignis, Instanz); P* und *T* sind endliche Mengen von Elementen, *A* steht für *Arcs(Pfeile)* und ist eine endliche Teilmenge von Elementen(siehe unten), *W* steht für *weight function(Gewichtsfunktion)* und ist eine Abbildung von *A* in **N**, M_0 steht für *initial marking(Anfangszustand)* und ist eine Abbildung von *P* in **N**.

$$P := \{p_1,p_2,...,p_n\}$$
$$T := \{t_1,t_2,...t_q\}$$

$$A \subseteq (PxT) \cup (TxP)$$
$$W : A \to \{1,2,..., m\}$$
$$M_0 : P \to \{0,1,2,..., k\}$$

Ein PN ist *normal* (*gewöhnlich, ordinary*), falls $W:A \to 1$ (also alle „Gewichte“ sind Eins).

Beispiel:
Die anfangs erwähnten Beispiele waren alle normale PN.
■

Beispiel 2.2.1:
Wir wollen uns die formale Definition des PN anhand des nächsten Beispieles verdeutlichen.

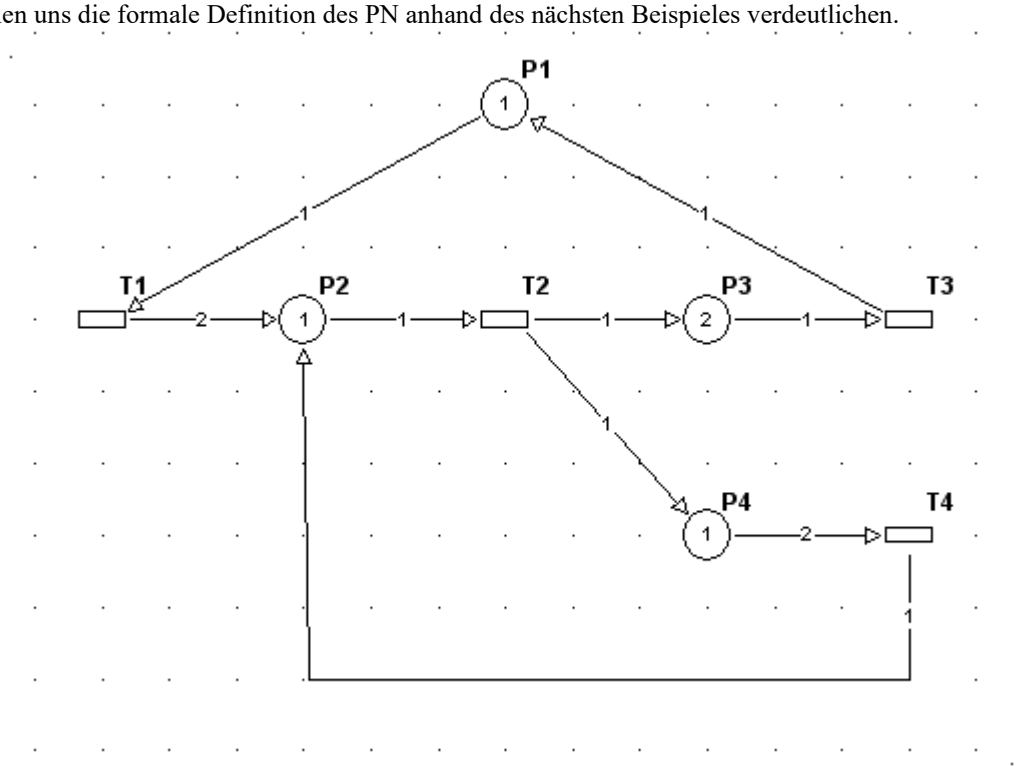


Abbildung 7: Allgemeines PN

In diesem Beispiel sind die Places P die Menge der Elemente *{P1, P2, P3, P4}*. Die Transitions T ist die Menge der Element *{T1, T2, T3, T4}*. Die Arcs A ist die Menge *{ (p1,t1), (p2,t2), (p3,t3), (p4,t4) } ∪ { (t1,p2), (t2,p3), (t2,p4), (t3,p1), (t4,p2) }.*
Die Gewichtsfunktion W ordnet jedem Element der Arcs A eine endliche natürliche Zahl zu.
W: A → {1, 2}, {(p1,t1) → 1, (p2,t2) → 1, (p3,t3) → 1, (p4,t4) → 2, (t1,p2) → 2, (t2,p3) → 1, (t2,p4) → 1, (t3,p1) → 1, (t4,p2) → 1 }.
Also beispielsweise ist *W(t1, p2) = 2.*
Die initial marking ist M_0*: P → {1, 2}.{P1 → 1, P2 → 1, P3 → 2, P4 → 1}.*
■

Damit haben wir das PN formal vollständig festgelegt und haben alles statischen Daten erfaßt.

2.2.2 Dynamik

Die Abbildung M (*Marking, Markierung*) definiert das *Feuern* einer Instanz t (das *Eintreten* eines Ereignisses) .

$$M(p) = \begin{cases} M_0(p) - W(p,t) \; if \; p \in {}^\circ t \\ M_0(p) + W(t,p) \; if \; p \in t^\circ \\ M_0(p) \; otherwise \end{cases}$$

Eine *Feuersequenz* (Feuerfolge, Folge von Ereignissen) σ ist eine Folge von „möglichen“ Transitionen.

Eine Transition t ist *möglich(aktiv)*, falls die resultierenden Markierungen positiv sind. Ansonsten ist die Transition *unmöglich(passiv)*.

Das Feuern ist also das gleiche wie die Durchführung einer mögliche Transition.

Eingangsbedingungen(input places) $^\circ t$ ist die Menge von Places *{p}*, so daß $(p,t) \in A$.
Ausgangsbedingungen(output places) t° ist die Menge von Places *{p}*, so daß $(t,p) \in A$.

Eingangsereignis(input transitions) $^\circ p$ ist die Menge von Transitions *{t}*, so daß $(t,p) \in A$.
Ausgangsereignis(output transitions) p° ist die Menge von Transitions *{t}*, so daß $(p,t) \in A$.

Beispiele:

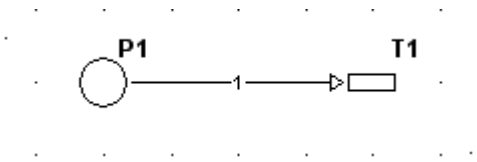


Abbildung 8: Eingangsbedingung $P1 \in {}^\circ t$ und Ausgangsereignis $T1 \in p^\circ$

Die Bezeichnung Eingangsbedingung alleine ergibt eigentlich wenig Sinn, gemeint ist das *P1* eine Eingangsbedingung **für die** Transition *T1* ist.
Analog ist *T1* ein Ausgangsereignis **von** der Bedingung *P1*.

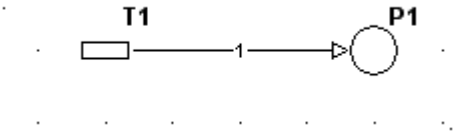


Abbildung 9: Ausgangsbedingung $P1 \in t^\circ$ und Eingangsereignis $T1 \in {}^\circ p$

P1 ist also die Ausgangsbedingung **von der** Transition *T1* und *T1* ist das Eingangsereignis **für die** Bedingung *p1*.

Mit diesen Beispielen sollte es klar sein, warum $^\circ t$ etwa für die Eingangsbedingungen steht.
Das „ETWAS“ $^\circ$ geht in den Eingang des Ereignisses **t** . Natürlich handelt es sich meistens um Mengen und nicht bloß um einzelne Komponenten.
■

Fortsetzung von Beispiel 2.2.1: Zuerst Feuern dann markieren.
Wir feuern etwa *T1*, dies ist möglich, da die Eingangsbedingung p1 für die Transition *T1* gleich $M_0(p1) = 1 > 0$ ist. Diese Feuerung bewirkt Zustandsänderungen in *p1* und *p2*. Wir erhalten somit neue Markierungen *{M(p1), M(p2)}*.
$P1 \in {}^\circ T1$ deshalb *ist* $M(p1) = M_0(p1) - W(p1,t1) = 1 - 1 = 0$ und
$P2 \in T1^\circ$ deshalb ist $M(p2) = M_0(p2) + W(t1,p1) = 1 + 2 = 3$

Feurersequenz
Nun möchten wir irgendeine Feuersequenz σ illustrieren. Wir könnten im Moment irgend eine aktive Transition nehmen, also *T2* nicht, da $M(p4) = -1 < 0$ per Definition unmöglich ist. Deshalb feuern wir einfach mal *T1* und erhalten aus der anfänglichen Markierung

$$M_0: P \rightarrow \{1, 2\}.\{P1 \rightarrow 1, P2 \rightarrow 1, P3 \rightarrow 2, P4 \rightarrow 1\}$$

die Markierung

$$M_1: P \rightarrow \{0, 1, 2, 3\}.\{P1 \rightarrow 0, P2 \rightarrow 3, P3 \rightarrow 2, P4 \rightarrow 1\}$$

Und somit die Feuersequenz $\sigma = <t_1>$
Nun können wir *T1* und *T4* nicht mehr feuern, da wir keine „Ladung“ mehr haben. Statt dessen feuern wir etwa *T3*, damit ist

$$M_2: P \rightarrow \{1,3\}.\{P1 \rightarrow 1, P2 \rightarrow 3, P3 \rightarrow 1, P4 \rightarrow 1\}$$

und $\sigma = <t_1, t_3>$.
Die nochmalige Feuerung von t_3 ergibt:

$$M_3: P \rightarrow \{1,3\}.\{P1 \rightarrow 2, P2 \rightarrow 3, P3 \rightarrow 0, P4 \rightarrow 1\}$$

$$\sigma = <t_1, t_3, t_3>.$$

Und so weiter und so fort $\sigma = <t_1, t_3, t_3, ... >$.
■

2.2.3 Source- und Sinktransitionen

Eine *Source-Transition (Quell-Ereignis, Anfangsereignis)* ist ein Ereignis, welches keine Eingangsereignisse hat.
Eine *Sink-Transition (Senken-Ereignis(Senke))* ist ein Ereignis, welches keine Ausgangsereignisse hat.

Beispiel:
Hier stellt *T1* die Source-Transition dar, welcher immer gefeuert werden kann. Also eine unerschöpfliche Quelle darstellt. *T4* ist eine Sink-Transition und „saugt" die Stelle *P2* ab.

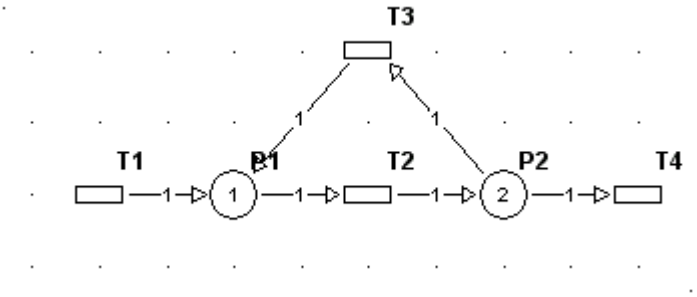


Abbildung 10: Source-Transition *T1* und Sink-Transition *T4*

■

Die Source-Transition kann immer gefeuert werden. Man kann sie sich vorstellen wie eine Transition der ein „unendliche" passive Systemkomponete vorgeschaltet ist.

2.2.4 Finite Capacity

Finite Capacity Place, falls ein Marking *M* kleiner gleich einer natürlichen Zahl sein muß.

Beispiel:
Sobald *T1* feuert würde sich der Zustand von *P1* um zwei erhöhen. *P1* läßt dies jedoch nicht zu. In der Ausgangssituation kann *T1* nicht feuern, da $M(p1) = 4 + 2 = 6 > 5$ ist. Die maximale „Speicherkapazität" von 4 wurde überschritten. *T1* muß warten bis *T2* zweimal gefeuert hat.

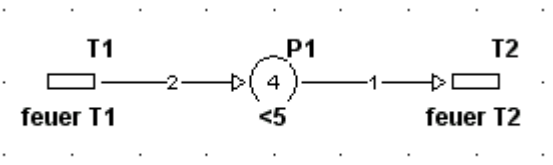


Abbildung 11: Finite Capacity

■

Satz: Jedes Finite Capacity PN kann in ein PN transformiert werden.

Beispiel:
P2 verhindert das mehr als vier „Einheiten" in *P1* eindringen. Dazu muß *P1* natürlich wissen wer die Eingangs- und Ausgangstransition von *P1* ist und was diese von *T1* wollen bzw. geben.
T1 kann nicht feuern, da sonst *M(p2)* unter Null sinkt, somit haben wir *M(p1)* auf maximal vier „Einheiten" beschränkt.

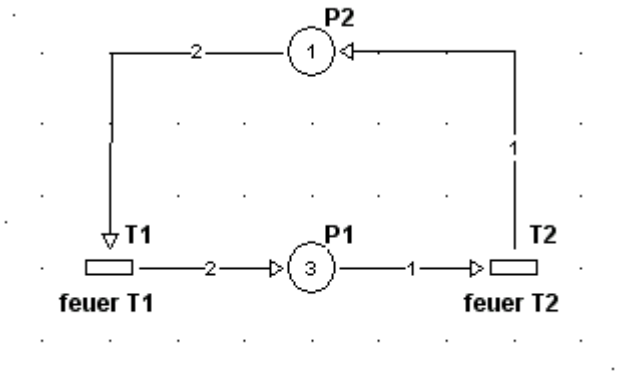


Abbildung 12: Aufgelöste Finite Capacity

■

Im Prinzip müßten wir einige Änderungen bzw. Ergänzungen in den Definitionen vornehmen, falls wir die Definition Finite Capacity verwenden. Jedoch aufgrund des Transformation Satzes ist dies nicht notwendig.

2.2.5 Selfloop

Selfloop (p,t), falls *p* eine Ein- und Ausgangsbedingung von *t* ist ($p \in {}^\circ t$ und $p \in t^\circ$).

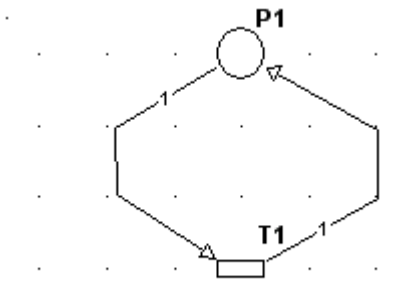


Abbildung 13: Selfloop

2.2.6 Siphon und Trap

Eine Menge *S* von Places ist ein *Siphon*, falls jede Transition die eine Ausgangsbedingung $p_a \in S$ hat mindestens eine Eingangsbedingung $p_e \in S$ hat (also aus $(t, p_a) \Rightarrow \exists(p_e,t)$).

Beispiel*:*
Wir behaupten die Menge $S := \{p1, p2, p3, p4\}$ ist ein Siphon.
Die Transition *T1* hat eine Ausgangsbedingung *p1*, deshalb muß sie mindesten eine Eingangsbedingung aus *S* haben. Sie hat sogar zwei, nämlich *p2* und *p3*. Die Transition *T2* hat auch eine Ausgangsbedingung *p4* welche Element von S ist sowie die Eingangsbedingung *p3*. Somit ist *T2* auch in Ordnung.
T3 hat keine Ausgangsbedingung und braucht deshalb keine Eingangsbedingung von *S* haben. *T3* hat Glück und bekommt trotzdem „ETWAS“ von *S*.

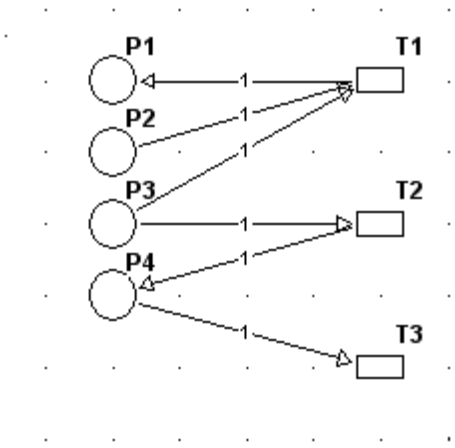


Abbildung 14: Siphon („Man kommt immer heraus.“)

Wie man leicht sieht sind $S_1 := \{p1, p2, p3\}$ und $S_2 := \{p1, p2\}$ ebenfalls Siphone. Jedoch sind $S_3 := \{p1\}$ und $S_1 := \{p1, p4\}$ keine Siphone, weil *T1* keine Eingangsbedingung hat.
■

Bei einem Siphon kommen also mehr Bedingungen heraus als hinein gehen.

Eine Menge *T* von Places ist eine *Trap*(Falle), falls jede Transition die eine Eingangsbedingung $p_e \in T$ hat mindestens eine Ausgangsbedingung $p_a \in T$ hat(also aus $(p_e,t) \Rightarrow \exists (p_a,t)$)

Bei einer Falle gehen mehr hinein als heraus.

Beispiel:
Wir behaupten die Menge $T := \{p1, p2, p3, p4\}$ ist eine Falle. *T1* hat eine Eingangsbedingung *p1* von *T* und muß deshalb mindestens einmal wieder zurück in die Falle. *T2* „flieht“ ebenfalls von der Falle mittels *P2* wird jedoch durch *P3* und *P4* wieder „gefangen genommen“. *T3* hat überhaupt keine Chance.

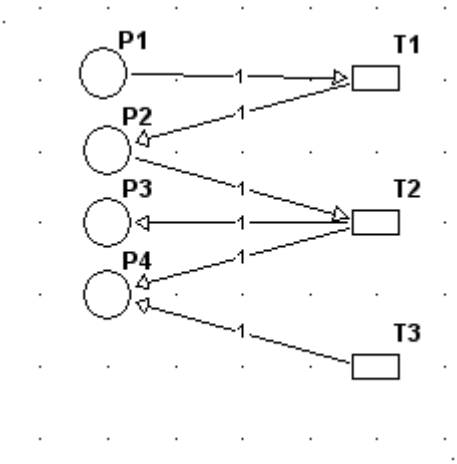


Abbildung 15: Trap („Hinein kommt man leicht.“)

■

3. Erreichbarkeit

3.1 Der Baum

Beispiel:
Wir haben die Anfangsmarkierung $M_0 = (2, 1, 0)$.

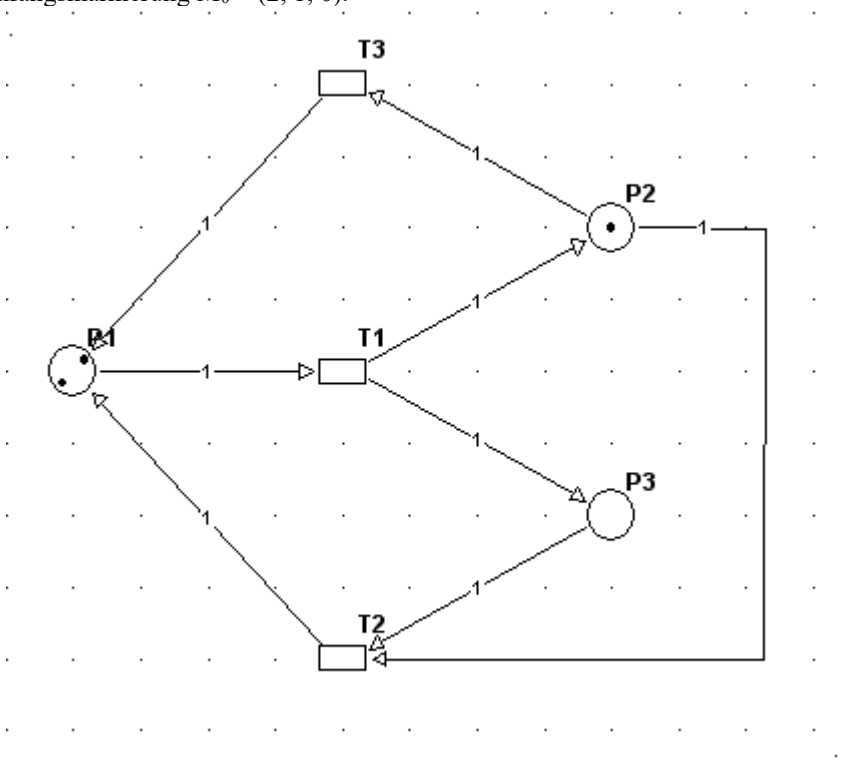
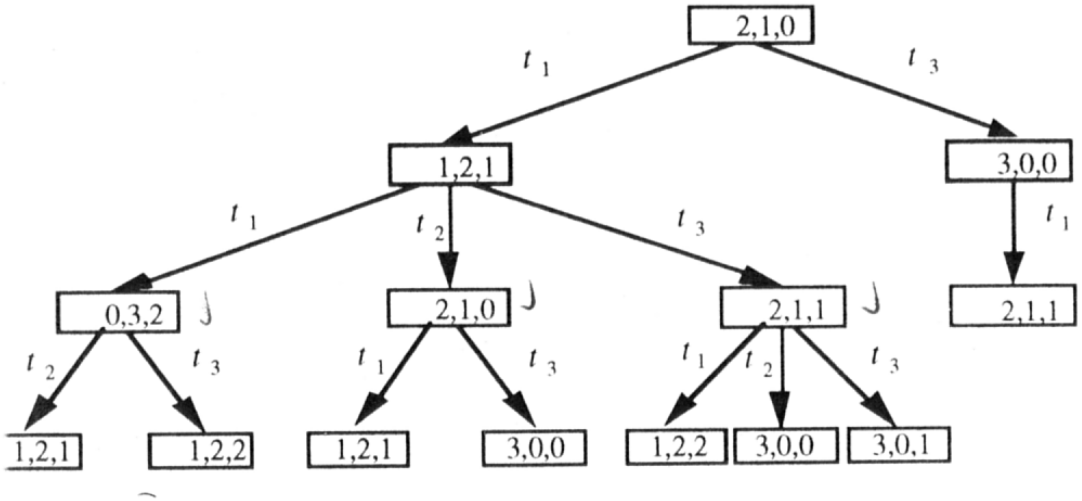


Abbildung 16: Normales PN („für den Baum“)

Die Transition *T1* und *T3* können feuern. Wir erhalten die erste Ebene unseres Baumes $\{M_1^1, M_2^1\}$.
Die nächste Tabelle gibt die ersten drei Ebenen des entstehenden Baumes wieder.

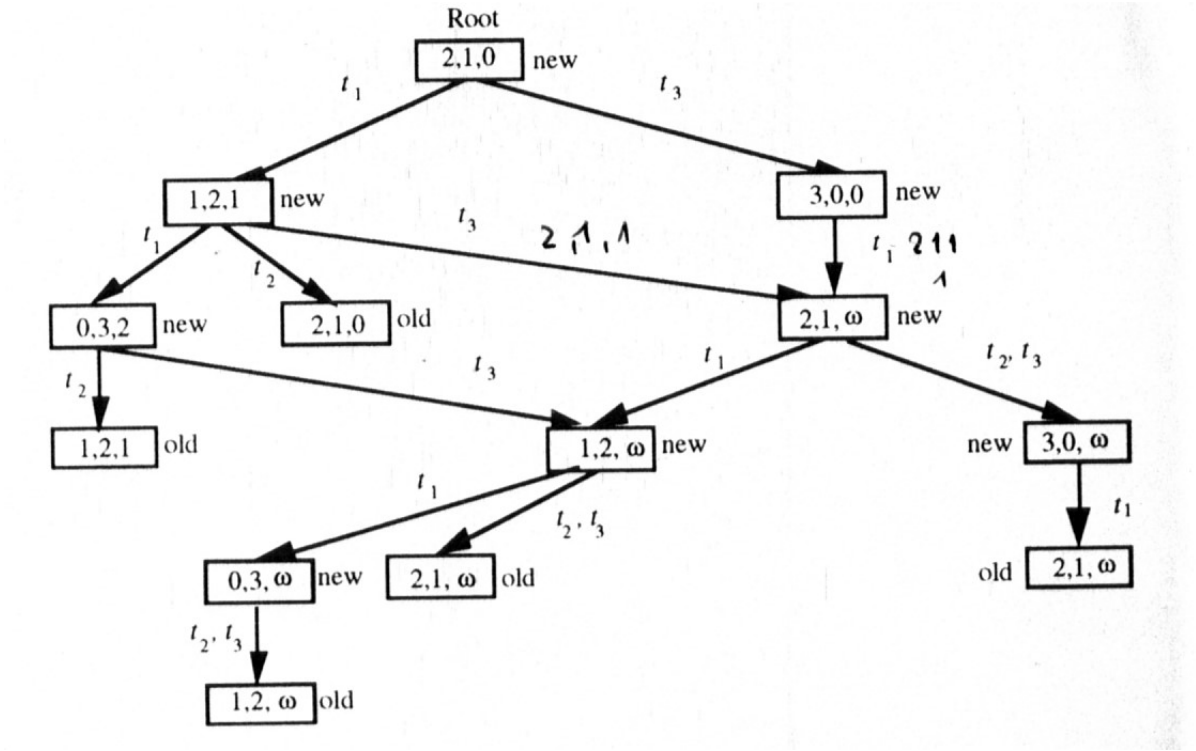
Considered marking	Fired transition	Obtained marking
$M_2^1 = [0,3,2]$	t_2	$M_3^1 = [1,2,1] = M_1^1$
$M_2^1 = [0,3,2]$	t_3	$M_3^2 = [1,2,2]$
$M_2^2 = [2,1,0]$	t_1	$M_3^3 = [1,2,1] = M_1^1$
$M_2^2 = [2,1,0]$	t_3	$M_3^4 = [3,0,0] = M_1^2$
$M_2^3 = [2,1,1]$	t_1	$M_3^5 = [1,2,2] = M_3^2$
$M_2^3 = [2,1,1]$	t_2	$M_3^6 = [3,0,0] = M_3^4$
$M_2^3 = [2,1,1]$	t_3	$M_3^7 = [3,0,1]$

Tabelle: Drei Ebenen des Erreichbarkeit-Baumes.
Die Knoten sind die Markierungen, einee Kante ist die möglichen Transitiondie eine Markierung in die nächste überführt.



Grafik: Die ersten drei Ebenen des Baumes

Es ist nun zweckmäßig gleiche Knoten zu erkennen. Hierzu gibt es leicht Entwickelbare Algorithmen. Mit dem Algorithmus von Jean-Marie Proth, Xiaolan Xie [1] erhält man folgenden „Coverbility Tree“.



Grafik: Coverbility Tree.

Diesen Baum kann man wieder umformen und so zu einem Graph kommen. Damit sind natürlich alle Wege in die Graphentheorie geöffnet.
■

3.2 Beschränktheit

Eine Markierung eines azyklisches PN (also ein PN ohne Zyklen(Kreise)) ist erreichbar genau dann und nur dann falls $M^t = M_0^t + U.Y^t$ mindestens eine Lösung hat.

4. Zustandsgleichung und Inzidenzmatrix

4.1 Inzidenzmatrix

$$u_{i,j} = \begin{cases} W(t_j, p_i)\ if\ t_j \in {}^{\circ}\mathrm{p_i} \\ -W(p_i, t_j)\ if\ t_j \in \mathrm{p_i}^{\circ} \\ 0\ \text{otherwise} \end{cases}$$

Beispiel: Konstruktion der Inzidenzmatrix

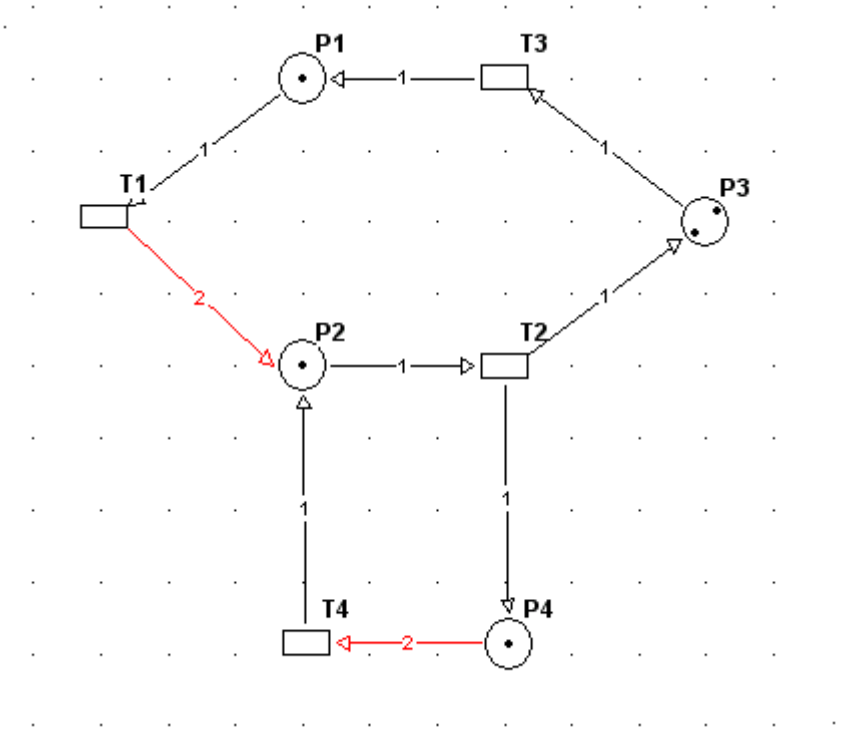


Abbildung 17: Das Petri Netz aus dem die Inzidenzmatrix gebaut wird.

Wir sind P1 und fragen T1 ob er uns „ETWAS“ gibt. Nein! Er nimmt uns ein „ETWAS“ weg (u_{11} = -1). Als nächstes fragen wir T2. T2 kennt uns nicht einmal (u_{12} = 0). Verzweifelt fragen wir T3 und bekommen endlich „ETWAS“ (u_{13} = 1). T4 ist wieder unbekannt (u_{14} = 0).
Nun sind wir P2 und fragen uns wieder durch von T1 bis T4. Letztendlich sollten wir zur P/T-Tabelle kommen.

	T1	T2	T3	T4
P1	-1	0	1	0
P2	2	-1	0	1
P3	0	1	-1	0
P4	0	1	0	-2

Dies ist unsere Inzidenzmatrix:

$$U = \begin{pmatrix} -1 & 0 & 1 & 0 \\ 2 & -1 & 0 & 1 \\ 0 & 1 & -1 & 0 \\ 0 & 1 & 0 & -2 \end{pmatrix}$$

■

Wir sehen sofort das wir die Selfloops nicht identifizieren können, da sie wie die nicht existierenden Pfeile Null sind. Die Inzidenzmatrix gibt die Pfeile mit ihrer Gewichtung wieder. Aber ebenso gibt sie uns die Menge der Places und Transitionen.
Wir haben also die Places, die Transitionen, die Arcs und die Gewichtsfunktion von unserem PN Fünfer Tupel, daß einzige was noch abgeht, um das statische PN vollständig zu beschreiben ist M_0.

4.2 Zustandsgleichung

Counting vector (Feuerungszählvektor) V_σ ist die Folge von Feuerungszählern $<v_i>$. Der *Feuerungszähler* v_i ist dabei die Anzahl des Auftretens der Transition in der Feuersequenz.

$$M^t = M_0^t + U.V_\sigma^t$$

Beispiel:
Wir betrachten die Abbildung 17 und versuchen die Zustandsgleichung auszuwerten.
Wir sehen sofort M_0 = (1, 1, 2, 1) , die Inzidenzmatrix haben wir im vorigen Beispiel ermittelt. Der Feuerungszählvektor ist der Einzig abgehende Mitspieler.
Wir definieren uns eine Feuerfolge und zählen dann die Transitionen.
Die Feuerfolge σ sei < t1, t2, t2, t4, t3, t3 >.
Wir fragen uns: Wie oft tritt t1 auf? Einmal; t2: zweimal; t3: zweimal; t4: einmal.
Damit haben wir den Feuerungszähler: (1, 2, 2, 1).
Wenn wir die Feuerfolge σ durchführen sollten wir folgende neue Markierung erhalten.

$$M^t = \begin{pmatrix}1\\1\\2\\1\end{pmatrix} + \begin{pmatrix}-1 & 0 & 1 & 0\\ 2 & -1 & 0 & 1\\ 0 & 1 & -1 & 0\\ 0 & 1 & 0 & -2\end{pmatrix}\begin{pmatrix}1\\2\\2\\1\end{pmatrix} = \begin{pmatrix}2\\2\\2\\1\end{pmatrix}$$

Wir haben also eine einfache analytische Methode um die Zustandsänderungen zu berechnen.
■

4.3 p-Invariante

$Z := (z_1, z_2, \ldots, z_n)^t$ ist eine *p-Invariante* falls
$Z.U = 0$
Und alle $z_i > 0$ sind.

4.4 t-Invariante

$W := (w_1, w_2, \ldots, w_n)$ ist eine *t-Invariante* falls
$U.W^t = 0$
Und alle $w_i > 0$ sind.

5. Vereinfachungen

5.1 Faltung (Betriebsmittel und ihre Nutzer)

Beispiel: Betriebsmittel und ihre Nutzer (W.Reisig)
Es gibt zwei Betriebsmittel {m, n}, die von drei Benutzern {u, v, w}verwendet werden. In der Ausgangssituation sind alle „Arbeits-“ Zustände aktiv. Nach Feuern des Eingangsereignisses vom Zustand „arbeitet“, sind die Wartebedingungen erfüllt. Falls das Betriebsmittel zur Verfügung steht kann es vom Benutzer verwendet werden. Dies wird durch die Place „hat_es“ angezeigt. Mit Eintreten des nächst möglichen Ereignisses wird das Betriebsmittel frei gegeben und dem nächsten Benutzer übergeben.

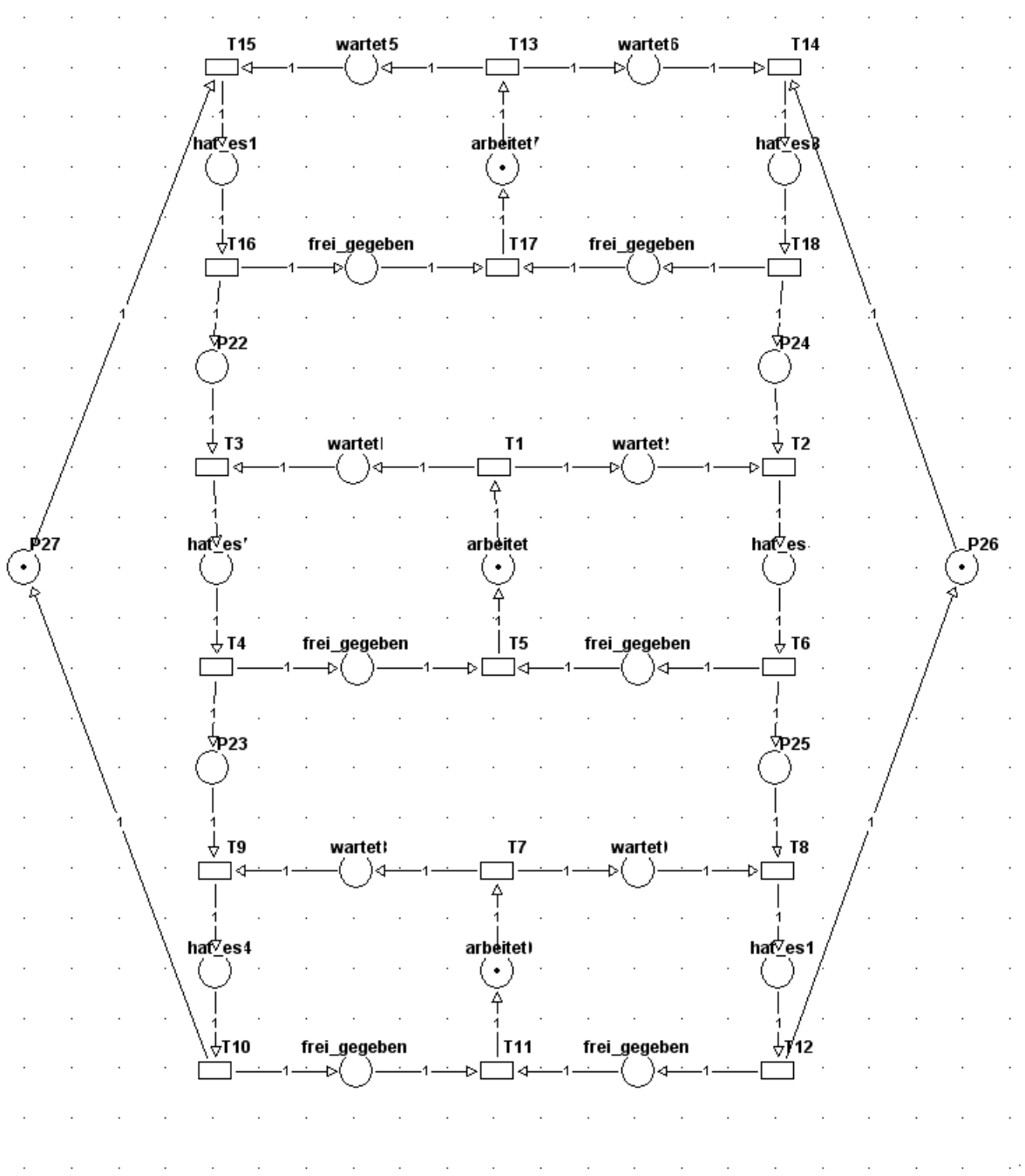


Abbildung 18: Betriebsmittel und ihre Nutzer

Wir sehen vor uns ein komplex anmutendes Gebilde, welches schön symmetrisch erscheint.
Unser Ziel ist ein kleines, einfach aussehendes graphisches PN. Wir falten deshalb dieses PN in der Mitte.

Dies bedeutet die Zusammenfassung der Betriebsmittel.
m + n

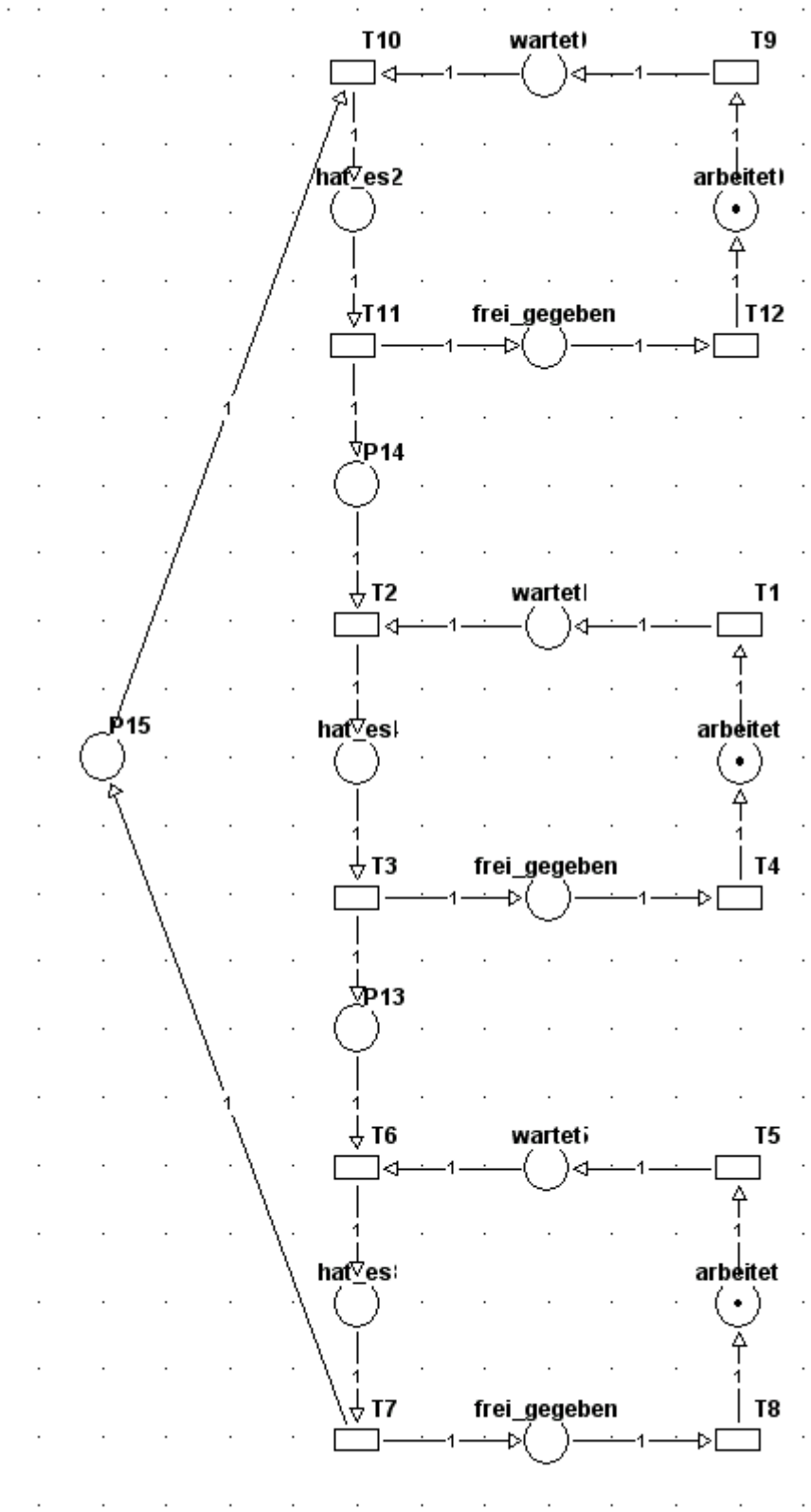


Abbildung 19: Einfache Faltung (ohne Änderung der Notation)

Ein halb so komplex aussehendes PN ist entstanden. Wiederum untersuchen wir das PN auf Äquivalente Strukturen. Diesmal legen wir die Benutzer übereinander.
u + v + w

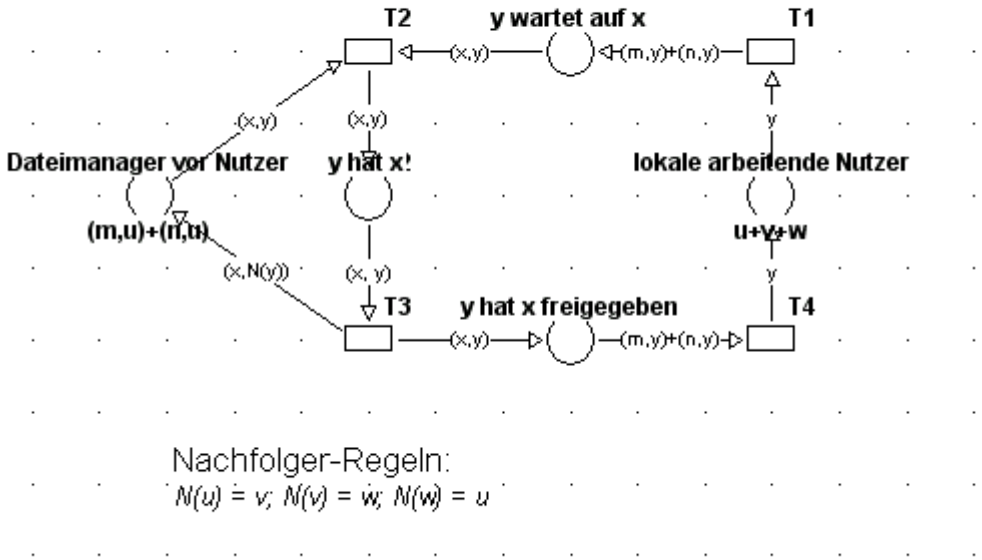


Abbildung 20: Zweifache Faltung mit Notation

Die Betriebsmittel m und n fassen wir unter der Variable x zusammen. Die Benutzer u, v und w resümieren wir unter der Variable y. „y hat x“ bedeutet, daß der Benutzer y das Betriebsmittel x hat. Sobald der Benutzer y das Betriebsmittel x frei gegeben hat kann der nächste Benutzer das Betriebsmittel nehmen. Wer ist der nächste Benutzer? Diese Frage wird durch die Nachfolgerregel festgelegt.
Wir wollen uns an die Notation gewöhnen.
Betrachten wir den Benutzer u in der untenstehenden Warteposition. Wie sieht diese Situation nach der Zweifachen Faltung aus?

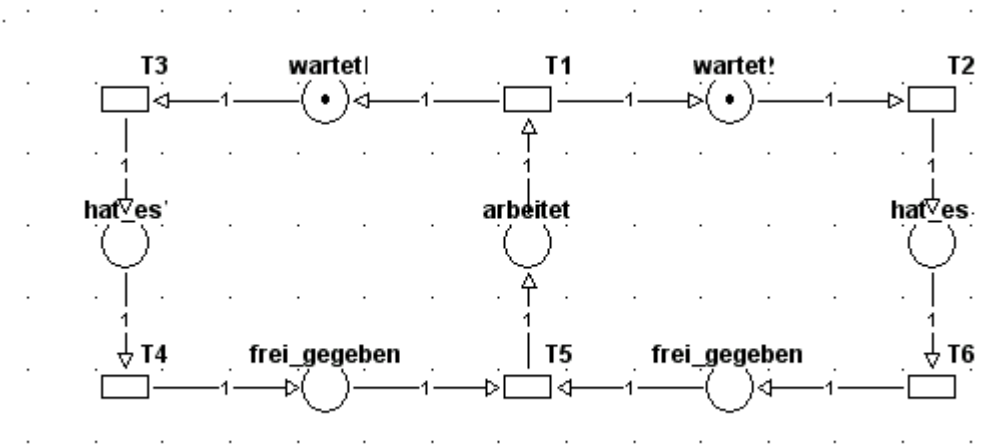


Abbildung 21: Ein Benutzer und die Betriebsmittel

Da u nicht arbeitet sondern wartet, muß er aus der Verfügbarkeit arbeitender Benutzer genommen werden. Er wartet sowohl auf das Betriebsmittel m als auch auf das Betriebsmittel n. Deshalb müssen wir den Zustand „y wartet auf x“ mit „(m,u)+(n,u)“ markieren.
Die Abbildung 22 gibt die Markierung aus Abbildung 21 für den Benutzer u wieder.

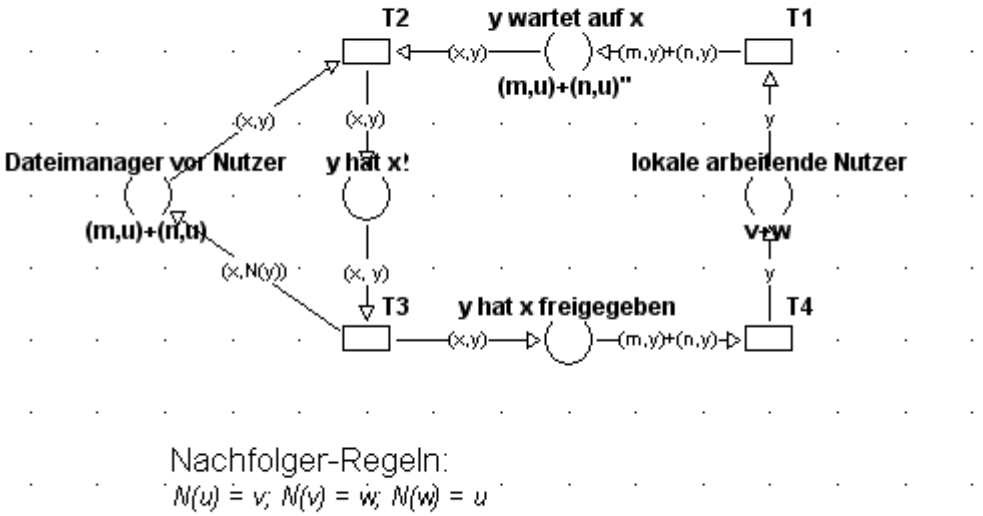


Abbildung 22: Benutzer u in Warteposition für Betriebsmittel m und n.

■

Die Faltung vereinfacht die graphisch komplexen PN.

6. Zeitbehaftete PN

6.1 Prinzip

Man spricht von einem *Timed PN (zeitbehafteten PN, zeiterweiterte PN)*, falls eine Transition ein Zeit t_1 braucht um die Eingangsbedingungen zu reduzieren und/oder eine Zeit t_2 um die Ausgangsbedingungen zu setzen.

Ein *Place Timed PN* (*Zustandszeitbehaftetes PN*), falls nach Feuerung einer Transition die „Angekommen" Tokens (Einheiten) mindestens eine Zeit t_1 im Place bleiben müssen.

Beispiel 6.1: Zeitbehaftetes deterministisches PN
Zur Zeit t_0 befindet sich das Timed PN im Zustand der durch Abbildung 23 wiedergegeben ist. Exakt zur Zeit t_0 wird die Timed Transition gefeuert.

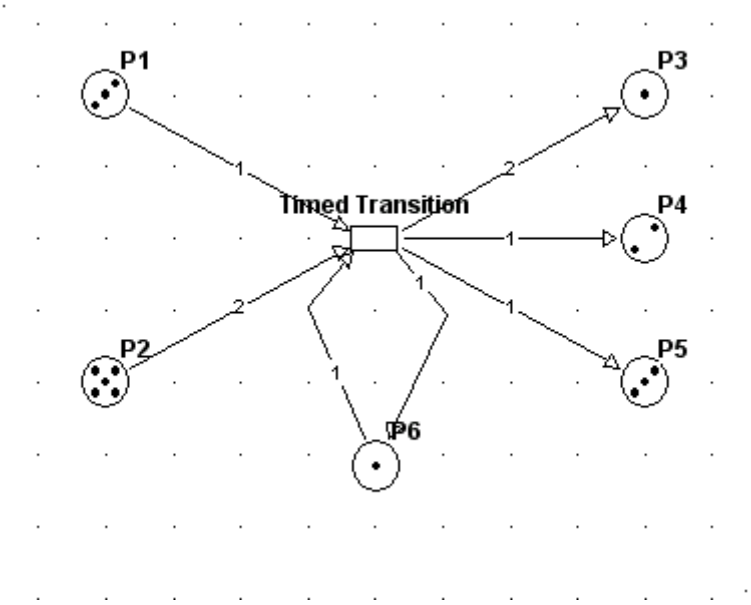


Abbildung 23: Zeitbehaftete Transition zur Zeit t_0

Durch die Feuerung ist der Zustand in Abbildung 24 eingetreten. Wobei wir die Nützlichkeit der Bedingung P6 erkennen. Diese verhindert eine Feuerung im Zeitintervall $(t, t + \Delta t)$.

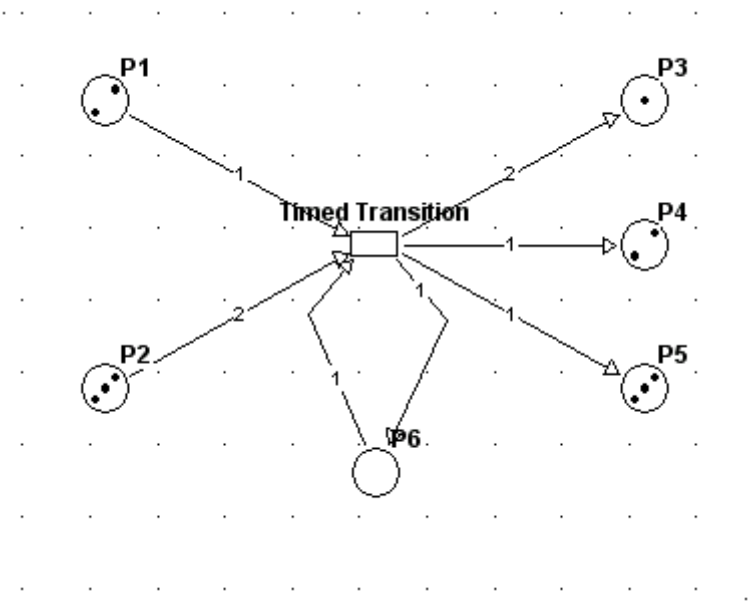


Abbildung 24: Zeitbehaftete Transition zwischen $t_0 + \Delta t$. Offenes Intervall $(t, t + \Delta t)$

Die nächste Abbildung gibt den exakten Zeitpunkt $t_0 + \Delta t$ wieder.

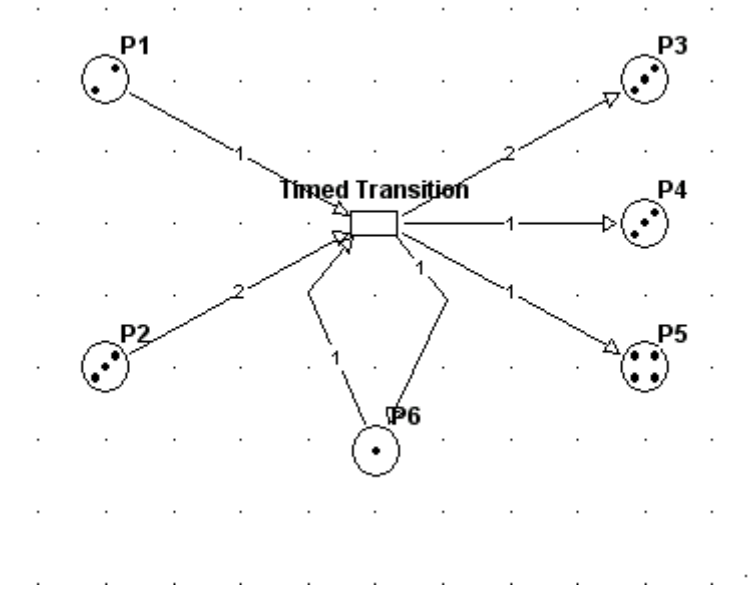


Abbildung 25: Zeitbehaftete Transition zum Zeitpunkt $t + \Delta t$.

■

6.2 Deterministisch und Stochastisch

Deterministische Transition, falls die Zeit konstant ist.

Das Beispiel 6.1 ist ein deterministisches PN.

Stochastische Transition, falls die Zeit stochastisch (also von einer Zufallsvariable abhängig) ist.
Man nennt ein *PN stochastisch*, falls es mindestens eine stochastische Transition gibt.

Beispiel:
Im Beispiel 6.1 nehmen wir an das die „Wartezeit“ Δt eine gleichverteilte Wahrscheinlichkeitsfunktion hat.
Wir erzeugen Δt mit einem Zufallszahlengenerator zwischen Null und Eins.
Zum Zeitpunkt $t_0 = 0$ liefert der Zufallszahlengnerator die zufällige Zeit $\Delta t = 0.41$.
Zur Zeit $t_1 = t_0 + \Delta t$ bekommen wir $\Delta t = 0.16$. und erhalten zum Zeitpunkt $t_2 = 0.57$ die Abbildung 26.

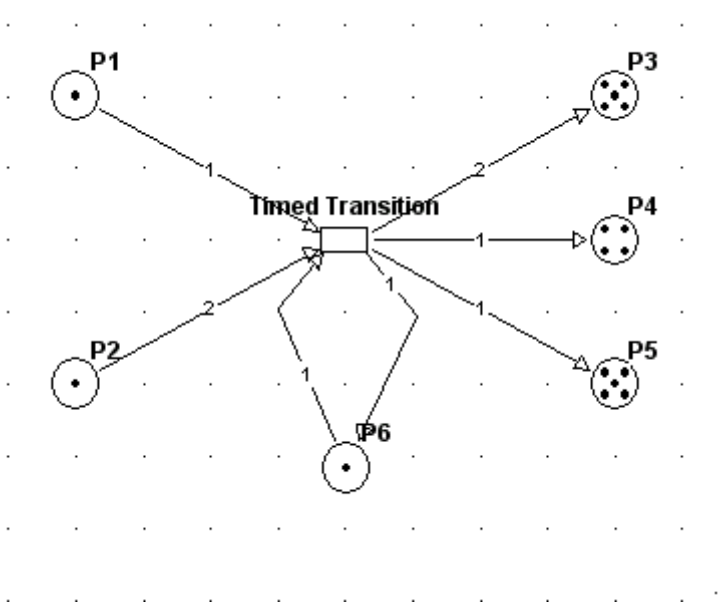


Abbildung 26: Stochastisches PN zum Zeitpunkt t_2

■

PN Software

	Features [and]					PN Type [and]							Platform [or]					Distribution [or]			Information links [and]					
Since 1.4.99	Graph. editor	Token game	Perform. analysis	Struct. analysis	Transfor mation	Time	Stoch astic	Color	Queue	Priority	Proba bility	Hier archy	Win	DOS	Unix	Mac	Other	Free ware	Free educ.	Free trial	Down load	Mail	Home	Daimi	Unimi	Crim
Agnes	■														■				■		■	■	■			
CABERNET	■	■		■	■										■			■			■	■	■		■	■
CAPSNET	■													■				■			■	■			■	
CodeSign	■	■				■							■		■	■			■			■	■	■		
CPN/AMI	■			■	■										■	■			■		■	■	■	■	■	■
DESIGN/CPN	■	■	■	■		■		■				■			■	■		■				■	■	■	■	■
DNAnet	■	■	■	■		■	■						■		■			■				■	■	■	■	■
DNS	■					■		■				■			■			■				■	■		■	
DSPNExpress	■	■	■	■		■	■								■			■				■	■	■	■	■
GDToolkit	■												■		■				■			■	■	■		
GreatSPN	■	■	■	■	■	■	■	■							■				■			■	■	■	■	
HiQPN-Tool	■	■	■	■		■	■	■	■						■			■				■	■	■		
Looping	■	■											■						■			■	■	■		
MOBY	■	■	■	■		■		■				■		■	■	■	OS/2		■		■	■	■		■	■
NETMAN	■					■	■					■	■					■			■	■			■	■
ONE	■			■	■										■			■				■			■	■
PAPETRI	■	■		■	■	■		■							■			■			■	■			■	■
PDS	■	■												■					■		■	■	■			
PED	■					■				■		■			■				■			■	■	■		
PEP	■	■		■	■										■				■			■	■	■	■	
PESIM	■	■	■	■	■	■	■						■					■			■	■			■	■
PETRI Maker	■	■	■	■									■					■			■	■	■	■	■	
PetriSim	■		■			■								■				■			■	■	■	■		
PNS	■	■													■		Java	■			■	■	■		■	■
PNTalk	■	■	■			■							■		■			■			■	■	■	■		
QPN-Tool	■	■	■	■		■	■	■	■						■			■				■	■		■	■
SANDS	■														■			■				■	■	■	■	■
SEA	■	■				■									■			■			■	■	■	■		
STROBOSCOPE	■		■			■	■	■	■	■	■		■						■		■	■	■	■		
THORN/DE	■		■			■									■				■			■	■	■		
TimeNET	■	■	■	■		■	■								■				■			■	■	■	■	■
UltraSAN	■		■			■	■								■				■			■	■		■	■
VIPtool	■	■											■		■				■		■	■	■	■		

Tabelle: Petri Netz Software

Internet Links

[I1]	Carl Adam Petri	seine Homepage	Bibliography, CV	http://www.informatik.uni-hamburg.de/TGI/mitarbeiter/profs/petri_eng.html
[I2]	W. Garbe	Petri Net tool survey	Software	http://ourworld.compuserve.com/homepages/wolf_garbe/petrisoft.html
[I3]	CPN group	World of PN	International Confernces, Applications, Theory	http://www.daimi.aau.dk/PetriNets/
[I4]	CPN group	PN Standard	ISO / IEC standards	http://www.daimi.au.dk/PetriNets/standard/
[I5]	Universität Kiel	„Links“	Einige nützliche Links	http://www.informatik.uni-kiel.de/~petri00/links.html
[I6]	„Educational“	PN Simulator	In Java	http://www.ee.uwa.edu.au/~braunl/pns/java/
[I7]	Visual Sim Net	Stochastic&Queuing PN	Software und vieles mehr	http://ourworld.compuserve.com/homepages/wolf_garbe/
[I8]	H. Boers	PN	Einführung (von Student)	http://www.voerde.globvill.de/~heinrich.boers/computer/petrinetze/index.html
[I9]	CPN group	Coloured PN	Einführung	http://www.daimi.aau.dk/CPnets/intro/index.html

Literaturverweise

[1]	Jean-Marie Proth, Xiaolan Xie	Petri Nets	Einfach, viele Beispiele
[2]	TU Braunschweig	Effizientes Engineering komplexer Automatisierungssysteme	Verschiedenste Berichte, empfehlenswert ist Reisigs Beschreibung
[3]	Christian Kelling	Simulationsverfahren für zeiterweiterte Petri-Netze	Interessante Anwendungen (z.B.: Tokenring)
[4]	Heinrich	Systemplanung I	Oberflächlich, einfach
[5]	Abel Dirk	Petri-Netze für Ingenieure	Standardreferenz

Index