

# Petrinetze

Vorversion eines Skripts zur Vorlesung  
Petrinetze  
im SS 2002

an der Universität Paderborn

Ekkart Kindler

Version: 0.0.8 vom 18. Juli 2002



# Vorwort

Dies ist eine Vorversion eines Skripts zur Vorlesung „Petrietze“, die ich im Sommersemester 2002 an der Universität Paderborn halte. Dieses Dokument ist zunächst als Ergänzung der handschriftlichen Notizen im WWW gedacht, die teilweise nur schwer lesbar sind. In der aktuellen Fassung liegt der Schwerpunkt auf den formalen Definitionen und den Sätzen. Die Motivation und weitere Erläuterungen werde ich nach und nach ergänzen.

Eines Tages wird vielleicht aus diesem Text ein richtiges Skript oder eine Monographie werden. Bis dahin muß man sich für die Information zwischen den Zeilen entweder in die Vorlesung begeben (was ohnehin empfehlenswert ist) oder die einschlägige Literatur bemühen.

Für Korrekturen und Verbesserungsvorschläge bin ich natürlich dankbar.

Paderborn, im Mai 2002,  
Ekkart Kindler



# Inhaltsverzeichnis

<b>Vorwort</b>	<b>iii</b>
<b>A Vorlesung</b>	<b>1</b>
<b>1 Informelle Einführung</b>	<b>3</b>
1 Der Buchstabe . . . . .	3
2 Der Geist . . . . .	5
3 Analyse . . . . .	8
4 Varianten . . . . .	9
5 Zusammenfassung . . . . .	9
<b>2 Stellen/Transitions-Systeme</b>	<b>11</b>
1 Grundlagen . . . . .	11
1.1 Standardbegriffe und -notationen . . . . .	11
1.2 Petrinetzbegriffe . . . . .	12
2 Stellen/Transitions-Systeme . . . . .	14
3 Der Überdeckungsbaum . . . . .	18
4 Lineare Invarianten . . . . .	32
4.1 Grundbegriffe der linearen Algebra . . . . .	32
4.2 Inzidenzmatrix und Zustandsgleichung . . . . .	34
4.3 Stellen-Invarianten . . . . .	36
4.4 Transitions-Invarianten . . . . .	41
5 Deadlocks und Traps . . . . .	46
6 Varianten von Petrinetzen . . . . .	51
6.1 Kapazitäten . . . . .	51
6.2 Elementare Netzsysteme und Bedingungs/Ereignis- Systeme . . . . .	53

6.3	Inhibitorkanten . . . . .	53
6.4	Zeitbehaftete Petrinetze . . . . .	55
6.5	Nebenläufigkeit und Parallelität . . . . .	56
6.6	Netze mit strukturierten Marken . . . . .	61
<b>3</b>	<b>Geschäftsprozeßmodellierung</b>	<b>67</b>
1	Überblick und Begriffsbildung . . . . .	67
2	Workflow-Netze . . . . .	71
3	Workflow-Konstrukte . . . . .	75
<b>4</b>	<b>Verifikation verteilter Algorithmen</b>	<b>81</b>
1	S/T-Systemen und ihre Prozesse . . . . .	81
2	Systemnetze und ihre Abläufe . . . . .	83
3	Spezifikation . . . . .	85
	3.1 Zustandsformeln . . . . .	86
	3.2 Temporale Formeln . . . . .	87
4	Verifikation . . . . .	88
5	Verteilte Algorithmen: Einige Beispiele . . . . .	94
<b>C</b>	<b>Literatur und Index</b>	<b>97</b>
	<b>Literaturverzeichnis</b>	<b>99</b>
	<b>Index</b>	<b>100</b>

Teil A  
Vorlesung



# Kapitel 1

## Informelle Einführung

Petrinetze sind eine Technik zur Modellierung, zur Analyse und zur Steuerung von Vorgängen. Sie eignen sich insbesondere für Vorgänge, an denen verschiedene Agenten beteiligt sind, die miteinander kooperieren und interagieren. Bei diesen Vorgängen kann es sich sowohl um Abläufe technischer Systeme, wie z.B. verteilt Informationssysteme, als auch Abläufe nicht-technischer Systeme, wie z.B. Geschäftsabläufe oder die Betreuung von Patienten, handeln. Wegen ihrer Allgemeinheit, werden Petrinetze heute in so verschiedenen Anwendungsgebieten wie dem Chip-Entwurf, der Software-Entwicklung<sup>1</sup>, der Telekommunikation, der Geschäftsprozessmodellierung und des Workflowmanagements eingesetzt.

Eine der Stärken<sup>2</sup> ist ihre intuitiv verständliche Semantik und ihre graphisch anschauliche Notation. Um diese Stärke zu betonen, beginnen wir mit einer informellen Einführung der Petrinetze.

### 1 Der Buchstabe

Wir stellen diese Einführung unter das folgende Motto:

Denn der *Buchstabe* tötet, der *Geist* aber macht lebendig.

2. Korinther 3, 6

---

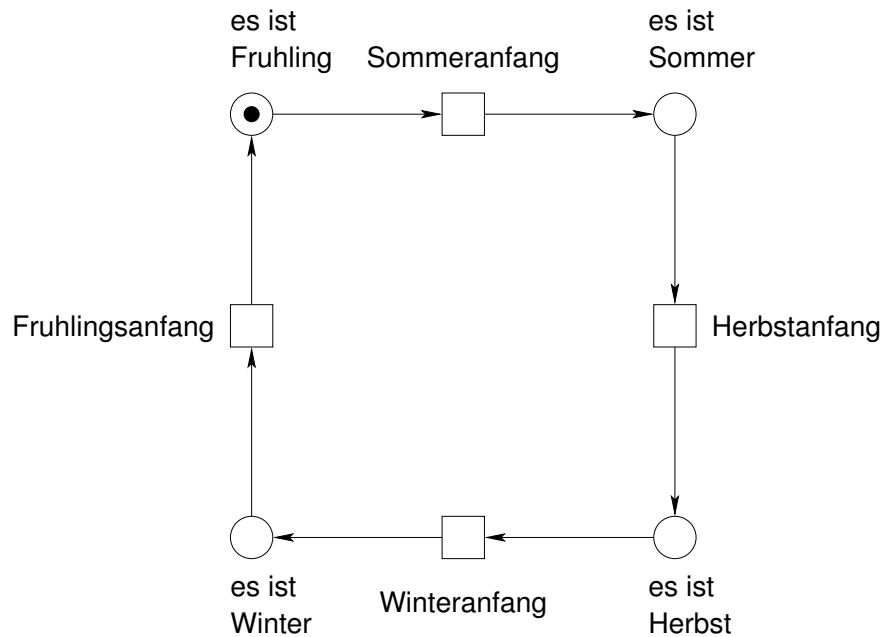
<sup>1</sup> Die Aktivitätsdiagramme von UML kann man auch als eine Variante von Petrinetzen ansehen.

<sup>2</sup>Vielleicht ist das zugleich auch ihre größte Schwäche.

Der Buchstabe steht für die formale mathematische Definition der Petrinetze. Ohne vermittelt zu bekommen, wie man diesen Formalismus zweckmäßig einsetzt, bleibt diese Definition jedoch tot. Erst das Verständnis für ihren zweckmäßigen Einsatz erweckt Petrinetze zum Leben. Ziel der Vorlesung ist es, den Geist der Petrinetze zu vermitteln. Allerdings wird es uns nicht immer erspart bleiben, uns auch mit dem Buchstaben der Petrinetze auseinanderzusetzen. Denn der Geist läßt sich leider nur über den Buchstaben „transportieren“.

Als erstes betrachten wir ein der klassisches Beispiel: *Die vier Jahreszeiten*.

### Beispiel 1.1 (Die vier Jahreszeiten)

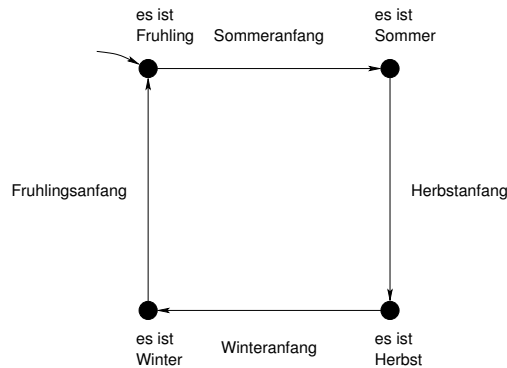


Diese Petrinetz modelliert den Lauf der Jahreszeiten. Dabei repräsentieren die *Stellen*, die als Kreise dargestellt sind, die mögliche Zustände: es ist Frühling, Sommer, Herbst oder Winter.

Die *Transitionen*, die durch Quadrate dargestellt sind, repräsentieren die Übergänge in die jeweils nächste Jahreszeit. Welche Zustandsänderung eine Transition bewirkt wird durch die *Kanten* zwischen den Stellen und den Transitionen dargestellt. Einen Zustandsübergang nennen wir dann auch das *Schalten* einer Transition.

Ein Zustand wird durch *Marken* (schwarze Kreise) auf den Stellen dargestellt. Ein Zustand eines Petrinetzes heißt deshalb auch *Markierung*. Im Beispiel beginnt der Lauf der vier Jahreszeiten mit dem Frühling. Ausgehend von dieser *Anfangsmarkierung* werden alle vier Jahreszeiten durchlaufen.

Obwohl Beispiel 1.1 das klassische Einführungsbeispiel für Petrinetze ist, ist es ein ziemlich schlechtes Beispiel. Denn der Witz von Petrinetzen kommt darin noch gar nicht zum Ausdruck. Dem Geiste nach ist es gar kein Petrinetz, sondern nur ein endlicher Automat. Es ist nur eine etwas andere Darstellung des folgenden endlichen Automaten:

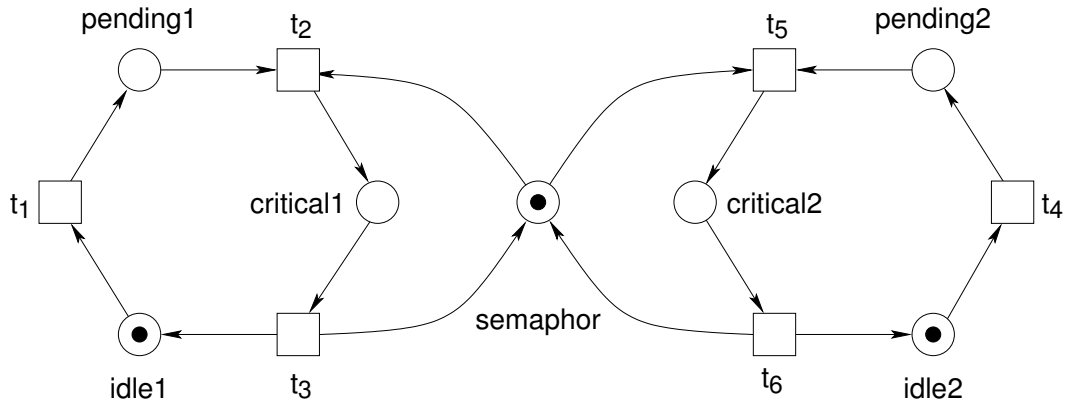


## 2 Der Geist

Wir betrachten deshalb ein Beispiel das dem Geist der Petrinetze besser entspricht:

### Beispiel 1.2 (Die wechselseitige Ausschluß)

Diese Petrinetz modelliert einen Semaphore, der den wechselseitigen Ausschluß(engl. mutual exclusion oft kurz mutex genannt) zweier Agenten 1 und 2 in ihren kritischen Bereichen `critical1` und `critical2` gewährleistet.



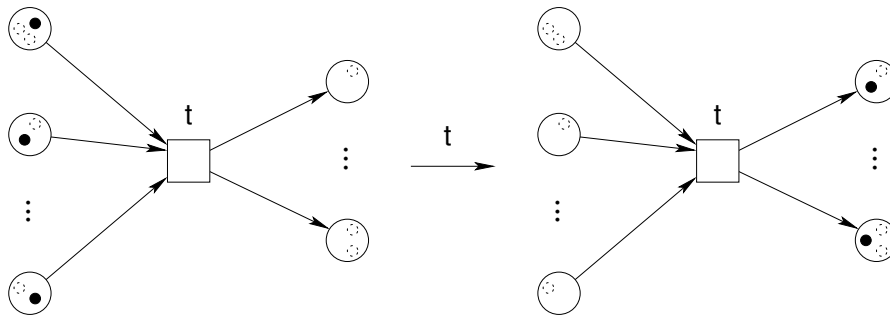
Die beiden Agenten durchlaufen dabei zyklisch die Zustände *idle*, *pending*, *critical*. Die zusätzliche Stelle **semaphor** gewährleistet, den wechselseitigen Ausschluß. Bei Eintritt in den kritischen Bereich muß eine Marke auf der Stelle **semaphor** vorhanden sein und der Agent nimmt diese Marke gewissermaßen mit in den kritischen Bereich. Erst beim Verlassen des kritischen Bereiches wird diese Marke zurückgegeben. Auf diese Weise ist gewährleistet, daß sich niemals beide Agenten zugleich im kritischen Bereich aufhalten können; denn es gibt nur einen Semaphor.

Dieses Beispiel zeigt, daß eine Transition mehrere eingehende Kanten und mehrere ausgehende Kanten besitzen kann. Für das Schalten einer Transition müssen alle Stellen mit einer Kante zu der Transition mindestens eine Marke enthalten; wir sagen dann, daß die Transition *aktiviert* ist. Beim Schalten wird dann jeweils eine Marke von diesen Stellen entfernt und eine Marke zu jeder Stelle hinzugefügt, zu denen eine Kante von der Transition führt. Wir nennen die Stellen mit einer Kante zu einer Transition den *Vorbereich* der Transition und die Stellen, zu denen eine Kante der Transition führt, den *Nachbereich* der Transition. Ob eine Transition in einer Markierung schalten kann, ist also durch ihren Vorbereich bestimmt; die Veränderung der Markierung beim Schalten einer Transition wird durch ihren Vor- und Nachbereich bestimmt.

Nachfolgend ist das Schaltverhalten einer Transition nochmals graphisch dargestellt; dabei deuten die gestrichelten Kreise Marken an, die vorhanden sein können, aber nicht vorhanden sein müssen:

Vorbereich

Nachbereich

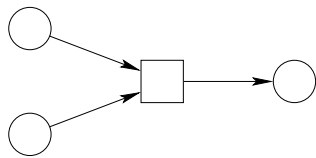


vorher (Transition ist aktiviert)

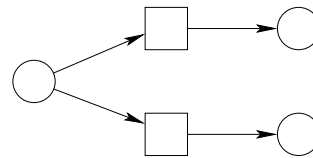
nachher (Transition hat geschaltet)

Dem Geiste nach ist also eine Transition bestimmt durch ihren Vor- und Nachbereich. Petrinetze sind nun insbesondere keine endlichen Automaten mehr. Wir können mit Ihnen verschiedene Situationen in Vorgängen modellieren:

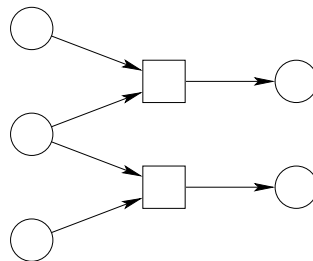
**Synchronisation,**



**Alternative,**



**und deren Kombination**



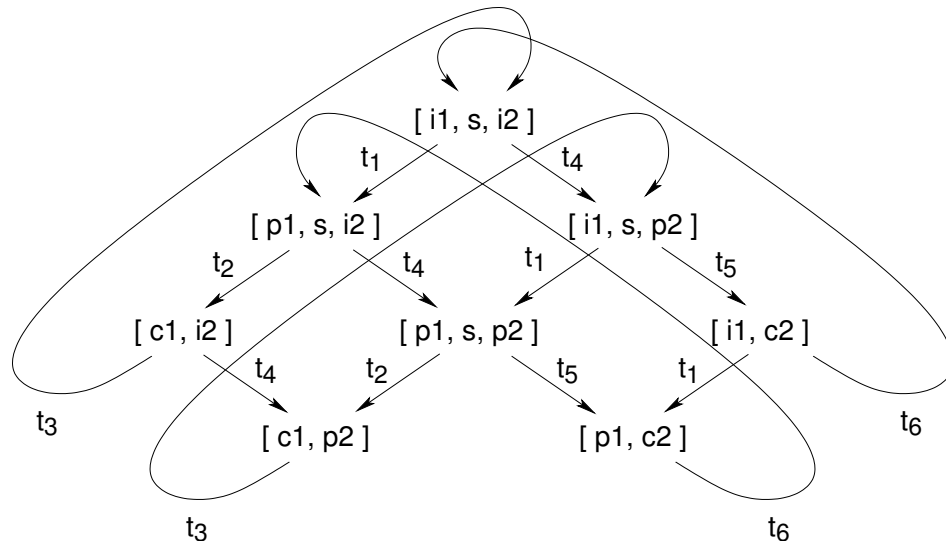
Durch die geeignete Kombination von *Synchronisationen* und *Alternativen* (Konflikten) können wir Systeme bzw. Vorgänge modellieren.

### 3 Analyse

Eine der Stärken von Petrinetzen sind die Techniken zu ihrer Analyse. Die Techniken basieren zum großen Teil auf linearer Algebra. Hier lassen wir jedoch den Geist der Petrinetze sprechen – zu den mathematischen Grundlagen kommen wir später.

Wir kommen dazu nochmals auf das Beispiel 1.2 zurück. Dort haben wir behauptet, daß in diesem Modell der wechselseitige Ausschluß gewährleistet ist, d. h. daß niemals zugleich eine Marke auf `critical1` und auf `critical2` vorkommen kann. Im Beispiel ist das ziemlich offensichtlich. Aber sobald die Modelle etwas komplizierter werden, ist das nicht mehr so offensichtlich, und man kann man sich bei solchen Aussagen leicht vertun. Die Frage ist nun, wie man ohne „Vertun“ beweisen kann, daß in einem Petrinetzmodell eine gewünschte Eigenschaft gilt.

Eine Möglichkeit besteht darin, ausgehend von der Anfangsmarkierung systematisch alle durch Schalten von Transitionen erreichbaren Markierungen zu konstruieren. Nachfolgend ist das Ergebnis für Beispiel 1.2 dargestellt. Wir nennen dies den *Erreichbarkeitsgraphen* des Petrinetzes: Dabei stellen wir eine Markierung durch Aufzählung der markierten Stellen zwischen eckigen Klammern dar (mit abgekürzten Stellennamen).



... Hier fehlt noch

- Stellen-Invarianten

- Beispiel für high-level Netze
- Hinweis auf Varianten
- Zusammenfassung

## 4 Varianten

...

## 5 Zusammenfassung

...



# Kapitel 2

## Stellen/Transitions-Systeme

In diesem Kapitel betrachten wir *Stellen/Transitions-Systeme* (*S/T-Systeme*). Sie sind eine der verbreitetsten und typischsten Varianten von Petrinetzen. Auch wenn sie in vielen Fällen zu unpraktikabel großen Netzmodellen führen, können wir an ihnen die wichtigsten Aspekte von Petrinetzen und ihren Analysetechniken verstehen. Viele dieser Techniken lassen sich relativ einfach auf andere Petrinetzvarianten übertragen.

### 1 Grundlagen

Bevor wir mit der Definition von S/T-Systemen beginnen können, legen wir uns in diesem Abschnitt das mathematische Handwerkszeug zurecht. In Abschnitt 1.1 führen wir die von uns benutzten Notationen für die Standardbegriffe aus der Mathematik ein. In Abschnitt 1.2 führen wir dann einige Basisbegriffe der Petrinetztheorie ein, die in allen Varianten von Petrinetzen benutzt wird.

#### 1.1 Standardbegriffe und -notationen

... Wird später ergänzt.

## 1.2 Petrinetzbegriffe

Wie wir in Kapitel 1 gesehen haben, besteht ein Netz aus *Stellen*, *Transitionen* und *Kanten*. Dem Buchstaben nach ist ein Petrinetz<sup>1</sup> also wie folgt definiert:

### Definition 2.1 (Netz)

Ein Netz  $N = (S, T, F)$  besteht aus zwei disjunkten Mengen  $S$  und  $T$  und einer Relation  $F \subseteq (S \times T) \cup (T \times S)$ .

Ein Element  $s \in S$  heißt *Stelle*, ein Element  $t \in T$  heißt *Transition*, und ein  $f \in F$  heißt *Kante* des Netzes  $N$ . Die Relation  $F$  wird auch *Flußrelation* von  $N$  genannt. Die Stellen und Transitionen des Petrinetzes nennen wir auch die *Elemente* des Netzes.

Wie wir haben bereits gesehen, daß man ein Petrinetz als Graph darstellen kann, wobei die Stellen als Kreise oder Ellipsen, die Transition als Quadrate oder Rechtecke (manchmal sogar nur als senkrechte Striche) und die Kanten als Pfeile zwischen den entsprechenden Elementen repräsentiert werden.

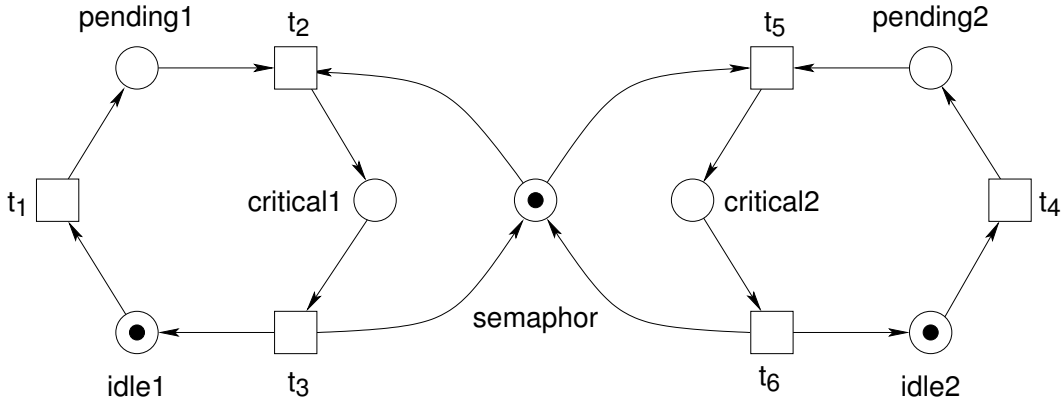
*Die graphische Darstellung ist nicht Bestandteil der mathematischen Definition. Jedoch gehört sie dem Geiste nach zur Definition – und auch in verschiedenen Normierungsbestrebungen wird die graphische Darstellung festgelegt. Wer in einem Petrinetz die Stellen durch Quadrate und die Transitionen durch Kreise darstellt, ist ein „Verbrecher“ (vgl. Übung 1, Aufgabe 1).*

Das folgende Beispiel zeigt die mathematische und die graphische Repräsentation des Netzes aus Beispiel 1.2:

### Beispiel 2.1 (Die wechselseitige Ausschluß)

---

<sup>1</sup> Im folgenden werden wir nur noch von Netzen reden.



$$\begin{aligned}
 N &= (S, T, F) \text{ mit} \\
 S &= \{\text{idle1, pending1, critical1, semaphore, idle2, pending2, critical2}\} \\
 T &= \{t_1, t_2, t_3, t_4, t_5, t_6, \} \\
 F &= \{(\text{idle1, } t_1), (t_1, \text{pending1}), \\
 &\quad (\text{pending1, } t_2), (\text{semaphor, } t_2), (t_2, \text{critical1}), \\
 &\quad (\text{critical1, } t_3), (t_3, \text{idle1}), (t_3, \text{semaphor}), \\
 &\quad (\text{idle2, } t_4), (t_4, \text{pending2}), \\
 &\quad (\text{pending2, } t_5), (\text{semaphor, } t_5), (t_5, \text{critical2}), \\
 &\quad (\text{critical2, } t_6), (t_6, \text{idle2}), (t_6, \text{semaphor})\}
 \end{aligned}$$

In Kapitel 1 haben wir bereits gesehen, daß der Vor- und der Nachbereich einer Transition eine wesentliche Rolle spielt. Diese Begriffe definieren wir nun – allgemein für beliebige Netzelemente und für Mengen von Netzelementen.

### Definition 2.2 (Vor- und Nachbereich)

Sei  $N = (S, T, F)$  ein Netz und  $x \in S \cup T$  ein Element von  $N$ . Die Menge  $\bullet x = \{y \in S \cup T \mid y F x\}$  heißt der *Vorbereich* von  $x$ . Die Menge  $x^\bullet = \{y \in S \cup T \mid x F y\}$  heißt der *Nachbereich* von  $x$ .

Für eine Menge  $X \subseteq S \cup T$  von Elementen von  $N$  definieren wir den *Vorbereich*  $\bullet X = \bigcup_{x \in X} \bullet x$  und den *Nachbereich*  $X^\bullet = \bigcup_{x \in X} x^\bullet$ .

In der informellen Diskussion haben wir gesehen, daß das Wesen einer Transition durch ihren Vorbereich und ihren Nachbereich bestimmt ist. Deshalb ist es oft nicht sinnvoll, Netze zu betrachten, in denen zwei Transitionen denselben Vorbereich und denselben Nachbereich haben. Denn dem Wesen nach sind beide Transitionen gleich. Ähnliches kann man sich leicht für Stellen

mit demselben Vorbereich und demselben Nachbereich überlegen. Deshalb betrachtet man oft nur *schlichte* Netze, in denen dies ausgeschlossen ist. Ebenso werden oft Netze mit isolierten Elementen oder mit Schlingen von der Betrachtung ausgeschlossen (wir werden aber sehen, daß Schlingen in vielen Fällen sehr sinnvoll sind).

**Definition 2.3 (Schlichtes Netz, isoliertes Element, Schlinge)**

Sei  $N = (S, T, F)$  ein Netz.

1. Ein Netz heißt *schlicht*, wenn für alle  $x, y \in S \cup T$  mit  $\bullet x = \bullet y$  und  $x^\bullet = y^\bullet$  gilt  $x = y$ .
2. Ein Element  $x \in S \cup T$  heißt *isoliert* in  $N$ , wenn  $\bullet x = x^\bullet = \emptyset$ .
3. Ein ungeordnetes Paar  $\{x, y\} \subseteq S \cup T$  von Elementen von  $N$  heißt *Schlinge*, wenn sowohl  $xFy$  als auch  $yFx$  gilt.

... Graphische Darstellung der Schlingen.

... Etwas ausführlicher Diskussion, der Bedeutung der Schlichkeit

## 2 Stellen/Transitions-Systeme

Nun werden wir S/T-Systeme definieren. Dazu definieren wir zunächst *Markierungen*, dann den Begriff des *S/T-Systems* selbst und zuletzt ihre Schaltregel.

Eine Markierung sagt für jede Stelle eines Petrinetzes, wieviele Marken gerade auf dieser Stelle vorkommen:

**Definition 2.4 (Markierung)**

Sei  $N = (S, T, F)$  ein Netz. Eine Abbildung  $m : S \rightarrow \mathbb{N}$  heißt *Markierung* von  $N$ .

Für zwei Markierungen  $m_1$  und  $m_2$  von  $N$  definieren wir die *Addition* und die *Inklusion* wie folgt:

**Addition**  $m = m_1 + m_2$  ist wieder eine Markierung mit  $m(s) = m_1(s) + m_2(s)$  für alle  $s \in S$ .

**Inklusion** Es gilt  $m_1 \leq m_2$ , wenn für alle  $s \in S$  gilt  $m_1(s) \leq m_2(s)$ . Es gilt  $m_1 < m_2$ , wenn  $m_1 \leq m_2$  und  $m_1 \neq m_2$  gilt.

Für Markierungen gibt es verschiedene Notationen:

**Graphisch:** Im Netz kann die Markierung durch eine entsprechende Zahl von Marken in der jeweiligen Stelle dargestellt werden. Bei einer großen Markenzahl wird dies durch eine Zahl dargestellt.

**Multimenge:** Die Markierung wird ähnlich wie eine Menge zwischen eckigen Klammern notiert. Allerdings zählt das mehrfache Auftreten desselben Elementes: deshalb nennt man das eine Multimenge. Beispielsweise bedeutet  $[a, c, a, d]$ , daß zwei Marken auf der Stelle  $a$ , und je eine auf der Stelle  $c$  und  $d$  vorkommen.

**Formale Summe:** Jede Stelle wird explizit mit der Anzahl ihrer Marken multipliziert (wobei unmarkierte Stellen weggelassen werden können). Zum Beispiel  $2a + c + d$ .

**Tupel/Vektor:** Wenn auf den Stellen eine Ordnung festgelegt ist, können wir eine Markierung auch als Vektor oder Tupel notieren. Wenn beispielsweise die Ordnung alphabetisch ist, können wir die obige Markierung durch  $(2, 0, 1, 1)$  notieren.

Damit können wir nun endlich definieren, was ein S/T-System ist. Zum Netz kommt eine Markierung hinzu und Kantengewichte, die besagen, wieviele Marken beim Schalten durch diese Kante fließen.

### Definition 2.5 (Stellen/Transitions-System)

Ein *Stellen/Transitions-System* (*S/T-System*)  $\Sigma = (N, W, m)$  besteht aus einem Netz  $N = (S, T, F)$ , einer Abbildung  $W : F \rightarrow \mathbb{N} \setminus \{0\}$  und einer Markierung  $m$  von  $N$ .

$W$  heißt das *Kantengewicht* und  $m$  die *Anfangsmarkierung* von  $N$ .

Das Kantengewicht wird graphisch durch Zahlen an den entsprechenden Kanten dargestellt, wobei das Gewicht 1 meist weggelassen wird. Wir nennen ein S/T-System *gewöhnlich*, wenn alle Kantengewichte 1 sind.

**Bemerkungen:**

1. Es gibt noch einige Erweiterungen von S/T-Systemen. Beispielsweise kann man in einigen Varianten die *Kapazität* einer Stelle angeben, die eine obere Schranke für die Anzahl der Marken angibt, die auf der Stelle vorkommen dürfen. Wir werden am Ende dieses Kapitels einige Varianten von S/T-Systemen diskutieren.
2. Wenn wir nichts anderes sagen, gehen wir im folgenden davon aus, daß das einem S/T-System zugrundeliegende Netz endlich ist, d. h. daß  $S$  und  $T$  endlich sind. Wenn wir S/T-Systeme mit einem unendlichen Netz betrachten wollen, sagen wir dies explizit dazu.

Um die Schaltregel für S/T-Systeme zu definieren, definieren wir für jede Transition  $t$  zwei Markierungen  ${}^{-}t$  und  $t^{+}$ . Dabei repräsentiert  ${}^{-}t$ , die Marken, die beim Schalten von  $t$  konsumiert werden, und  $t^{+}$  repräsentiert die Marken, die beim Schalten von  $t$  produziert werden.

**Definition 2.6**

Sei  $\Sigma = (N, W, m)$  ein S/T-System mit Netz  $N = (S, T, F)$ . Für jede Transition  $t \in T$  definieren wir die beiden Markierungen  ${}^{-}t : S \rightarrow \mathbb{N}$  und  $t^{+} : S \rightarrow \mathbb{N}$  wie folgt:

$${}^{-}t(s) = \begin{cases} W((s, t)) & \text{falls } (s, t) \in F \\ 0 & \text{sonst} \end{cases}$$

$$t^{+}(s) = \begin{cases} W((t, s)) & \text{falls } (t, s) \in F \\ 0 & \text{sonst} \end{cases}$$

Die Abbildung  $\underline{t} : S \rightarrow \mathbb{Z}$  definieren wir durch  $\underline{t} = t^{+} - {}^{-}t$ .

*Streng genommen hätten wir die Subtraktion auf Markierungen auch noch definieren müssen. Sie ist aber wie die Addition punktweise definiert. Allerdings ist  $\underline{t}$  im allgemeinen keine Markierung von  $N$ , denn  $\underline{t}(s)$  kann negativ sein.*

Damit sind wir nun in der Lage, die Schaltregel für S/T-Systeme zu definieren. Um dem Geist der Petrinetze gerecht zu werden, benutzen wir die Markierungen  ${}^{-}t$  und  $t^{+}$  und verzichten auf die Verwendung der Subtraktion.

**Definition 2.7 (Aktiviertheit und Schaltregel)**

Seien  $\Sigma = (N, W, m)$  ein S/T-System mit Netz  $N = (S, T, F)$ ,  $m_1$  eine Markierung von  $N$  und  $t \in T$  eine Transition von  $N$ .

1. Die Transition  $t$  heißt in Markierung  $m_1$  *aktiviert*, wenn es eine Markierung  $m'$  von  $N$  mit  $m_1 = m' + {}^-t$  gibt. Wir schreiben dann  $m_1 \xrightarrow{t}$

$m_1 \xrightarrow{t}$  gilt genau dann, wenn  $m_1 \geq {}^-t$  gilt.

2. Eine Markierung  $m_2$  heißt *Folgemarkierung* von  $m_1$  unter Schalten von  $t$ , wenn es eine Markierung  $m'$  von  $N$  mit  $m_1 = m' + {}^-t$  und  $m_2 = m' + t^+$  gibt. Wir schreiben dann  $m_1 \xrightarrow{t} m_2$ .

Es gilt  $m_2 = (m_1 - {}^-t) + t^+ = m_1 + t$ .

Die durch die Schaltregel definierte 3-stellige Relation  $m_1 \xrightarrow{t} m_2$  nennen wir auch *Übergangsrelation*

Aufbauend auf der Übergangsrelation führen wir noch einige weitere Begriffe und Notationen als Abkürzungen ein.

**Definition 2.8 (Notationen)**

Wir schreiben

$m_1 \xrightarrow{t_1 \dots t_n} m_2$ , wenn es Markierungen  $m_2, m_3, \dots, m_n$ , gibt so daß für alle  $i \in \{1, \dots, n\}$  gilt  $m_i \xrightarrow{t_i} m_{i+1}$ ,

$m_1 \longrightarrow m_2$ , wenn es eine Transition  $t \in T$  mit  $m_1 \xrightarrow{t} m_2$ , gibt,

$m_1 \xrightarrow{*} m_2$ , , wenn es ein *Schaltwort*  $w \in T^*$  mit  $m_1 \xrightarrow{w} m_2$  gibt und

$[m\rangle$  für die Menge der von Markierung  $m$  aus erreichbaren Markierungen:  
 $[m\rangle = \{m' \mid m \xrightarrow{*} m'\}$ . Für ein S/T-System  $\Sigma$  mit Anfangsmarkierung  $m$  nennen wir  $[m\rangle$  die Menge der in  $\Sigma$  erreichbaren Markierungen.

Die Definition der Schaltregel hat einige einfache und doch erstaunliche weitreichende Konsequenzen – wie wir im Verlauf der Vorlesung noch sehen werden.

**Folgerung 2.9**

Seien  $\Sigma = (N, W, m)$  ein S/T-System mit Netz  $N = (S, T, F)$ ,  $t_1, \dots, t_n \in T$  Transitionen,  $m', m_1, m_2, m'_2$  Markierungen von  $\Sigma$  und  $\pi : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$  eine Permutation (d.h.  $\pi$  ist bijektiv).

1. Aus  $m_1 \xrightarrow{t_1 \dots t_n} m_2$  und  $m_1 \xrightarrow{t_1 \dots t_n} m'_2$  folgt  $m_2 = m'_2$ .
2. Aus  $m_1 \xrightarrow{t_1 \dots t_n} m_2$  und  $m_1 \xrightarrow{t_{\pi(1)} \dots t_{\pi(n)}} m'_2$  folgt  $m_2 = m'_2$ .
3. Aus  $m_1 \xrightarrow{t_1 \dots t_n} m_2$  folgt  $m_1 + m' \xrightarrow{t_1 \dots t_n} m_2 + m'$ .

**Beweis:** Das wesentliche Argument ist  $m_2 = m_1 + \underline{t}$ . Ein ausführlicher Beweis wird in Übung 3, Aufgabe 3 angegeben.  $\square$

Das Gesamtverhalten eines S/T-Systems, das sich aus der Schaltregel ergibt, wird im *Erreichbarkeitsgraphen* des S/T-Systems zusammengefaßt, den wir nun formal definieren.

**Definition 2.10 (Erreichbarkeitsgraph)**

indexErreichbarkeitsgraph

Sei  $\Sigma = (N, W, m)$  ein S/T-System mit Netz  $N = (S, T, F)$ . Der *Erreichbarkeitsgraph*  $\Gamma = ([m], m, R)$  besteht aus der Menge der erreichbaren Markierungen, der Anfangsmarkierung und der dreistelligen Relation  $R \subseteq [m] \times T \times [m]$  mit  $R = \{(m_1, t, m_2) \mid m_1 \xrightarrow{t} m_2\}$ .

*R ist die Einschränkung der Übergangsrelation des S/T-Systems auf die Menge der erreichbaren Markierungen.*

In Kapitel 1 haben wir in Abschnitt 3 bereits ein Beispiel eines Erreichbarkeitsgraphen kennengelernt. Deshalb verzichten wir hier auf ein weiteres Beispiel.

### 3 Der Überdeckungsbaum

In der Übung 2 haben wir gesehen, daß es selbst unter den vernünftigen S/T-Systeme (gewöhnlich ohne leere Vorbereiche) solche gibt, die unendlich viele erreichbare Markierungen haben. Abbildung 2.1 zeigt ein weiteres Beispiel, das sogenannte Erzeuger-Verbraucher-System. Zwischen dem Erzeuger (links) und dem Verbraucher (rechts) gibt es einen unbeschränkten Puffer.

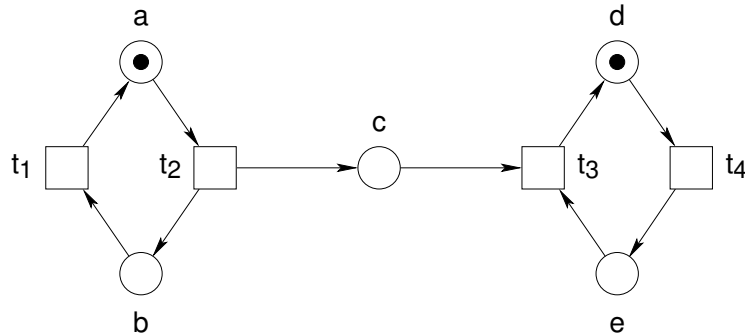


Abbildung 2.1: Das Erzeuger-Verbraucher-System

Dies ist also ein Beispiel, in dem die Unendlichkeit bewußt in Kauf genommen wird.

Zunächst geben wir eine äquivalente Charakterisierung der Endlichkeit des Erreichbarkeitsgraphen an:

**Definition 2.11 (Beschränktheit)**

Ein S/T-System  $\Sigma$  heißt *beschränkt*, wenn es eine (nicht notwendigerweise erreichbare) Markierung  $m'$  von  $\Sigma$  gibt, so daß für jede erreichbare Markierung  $m''$  von  $\Sigma$  gilt  $m'' \leq m'$ .

*Die erreichbaren Markierungen von  $\Sigma$  sind also alle durch die Markierung  $m'$  beschränkt.*

**Folgerung 2.12 (Beschränktheit und Endlichkeit)**

Ein S/T-System<sup>2</sup> ist genau dann beschränkt, wenn es endlich viele erreichbare Markierungen besitzt.

**Beweis:** Für die eine Richtung des Beweises gibt es ein einfaches kombinatorisches Argument. Für die andere Richtung konstruiert man explizit die Markierung  $m'$  explizit. Einen ausführlichen Beweis werden wir in Übung 3, Aufgabe 4 sehen.  $\square$

Für viele Analysetechniken will man den Erreichbarkeitsgraphen des S/T-Systems konstruieren. Allerdings wird die Konstruktion nicht terminieren, wenn der Erreichbarkeitsgraph unendlich ist. Wir müssen uns also Gedanken

<sup>2</sup> Zur Erinnerung: Wenn wir nichts anderes sagen, ist das zugrundeliegende Netz des S/T-Systems endlich. Sie können sich ja einmal überlegen, ob die Aussage auch für S/T-Systeme mit einem unendlichen Netz auch gilt.

machen, wie man feststellen kann, ob der Erreichbarkeitsgraph endlich ist. Dabei ist zunächst die Frage, ob diese überhaupt entscheidbar ist.

*Um ein Gefühl dafür zu einwickeln, ob eine solche Frage im allgemeinen schwer zu entscheiden ist, betrachten wir zunächst zwei etwas vertrautere Maschinenmodell: den linear beschränkte Automaten und die Turingmaschine (die den Typ-1- bzw. Typ-0-Sprachen der Chomsky-Hierarchie entsprechen).*

*Hier ist die Frage, ob für eine bestimmte Eingabe die Menge der erreichbaren Konfigurationen endlich ist oder nicht. Bei linear beschränkten Automaten ist die Menge der erreichbaren Konfigurationen per Definition endlich (da ein linear beschränkter Automat nur ein festgelegtes Stück des Bandes beschreiben darf). Dagegen ist die Menge der erreichbaren Konfigurationen bei (beliebigen) Turingmaschinen manchmal endlich und manchmal unendlich. Noch schlimmer: Für Turingmaschinen ist nicht entscheidbar, ob sie unendlich viele erreichbare Konfigurationen besitzen (denn sonst wäre das Halteproblem entscheidbar).*

*Petrinetze sind zwar ein vollkommen andersartiges Automatenmodell: Sie werden zwar manchmal auch zum Berechnen von Funktionen „mißbraucht“, sie sind aber eigentlich zum Modellieren von (Vorgängen oder von) reaktiven Systemen gedacht. Technisch gesehen ist es aber dieselbe Fragestellung, der wir hier nachgehen.*

Bezüglich dieser Frage liegen S/T-Systeme zwischen den linear beschränkten Automaten und den Turingmaschinen: Es gibt sowohl S/T-Systeme mit endlich vielen erreichbaren Markierungen als auch S/T-Systeme mit unendlich vielen erreichbare Markierungen. Aber für S/T-Systeme ist entscheidbar, ob die Menge der der erreichbarekn Markierungen endlich oder unendlich ist. In diesem Abschnitt stellen wir ein Verfahren vor, mit dem wir diese Frage entscheiden können. Dazu konstruieren wir den sogenannten *Überdeckungsbaum*. Der Überdeckungsbaum liefert darüber hinaus auch noch eine ganze Reihe weiterer Informationen über die erreichbaren Markierungen liefert.

*Deshalb lohnt es sich den Überdeckungsbaum auch noch weiter zu konstruieren, wenn wir schon wissen, daß das S/T-System unendlich viele erreichbare Markierungen hat.*

Um die Idee des Überdeckungsbaumes zu illustrieren, betrachten wir zunächst ein einfaches Beispiel.

### **Beispiel 2.2 (Überdeckungsbaum)**

Wir betrachten einen Ausschnitt des Erzeuger-Verbraucher-Systems, der in Abb. 2.2 dargestellt ist. Für dieses S/T-System konstruieren wir nun aus-

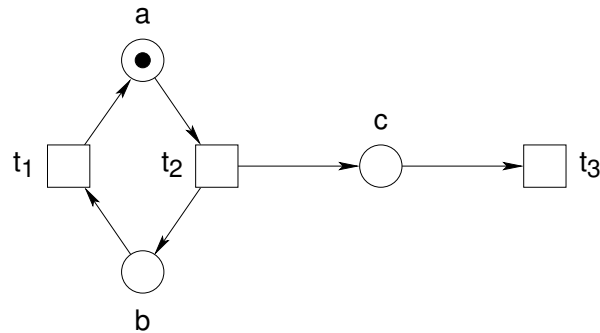
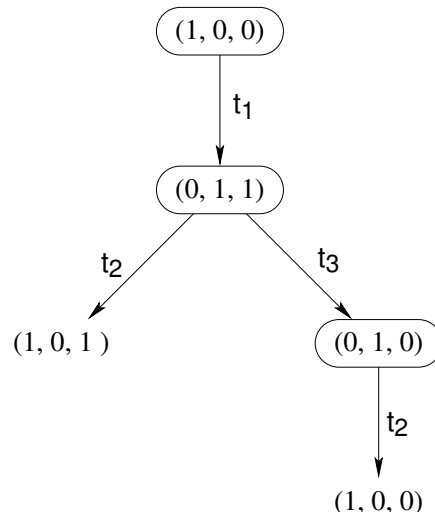


Abbildung 2.2: Ausschnitt des Erzeuger-Verbraucher-Systems

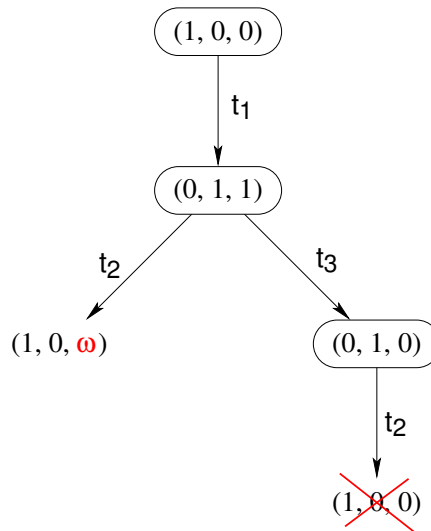
gehend von der Anfangsmarkierung<sup>3</sup>  $(1, 0, 0)$  einen Baum der erreichbaren Markierungen. Nach vier Schritten entsteht dabei der folgende Baum:



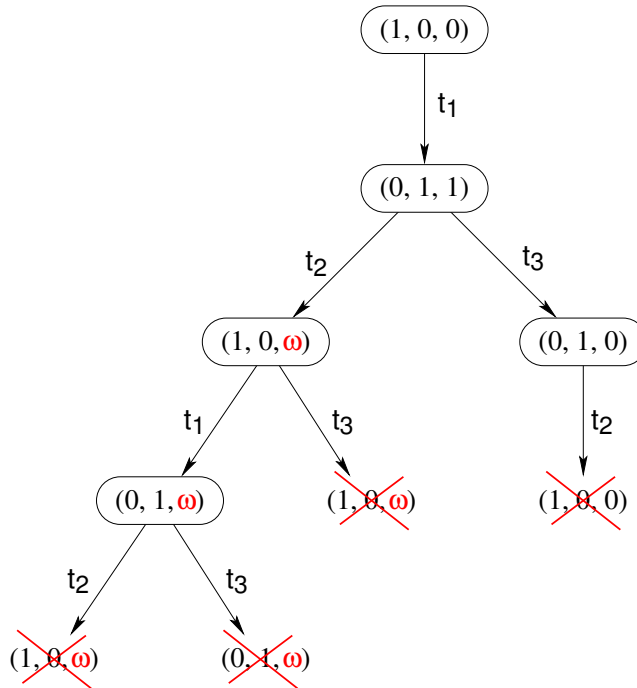
Die „eingekringelten“ Markierungen sind bereits fertig bearbeitet; bei den nicht „eingekringelten“ Markierung muß die Konstruktion noch fortgesetzt werden. Wir könnten nun diese Konstruktion beliebig weit fortsetzen, allerdings würde dieser Baum unendlich werden. Das verhindern wir, indem wir die Markierung rechts  $(1, 0, 0)$  streichen, da sie auf dem Pfad (ganz am Anfang schon mal vorkam). Denn, wenn wir dort weitere Nachfolger konstruieren, entstehen keine neue Markierungen mehr.

<sup>3</sup> Wir benutzen hier die Tupel-Schreibweise für Markierungen mit der Ordnung  $a, b, c$ .

Etwas anders verhält es sich bei der Markierung  $(1, 0, 1)$ . Diese kam bisher nirgends vor. Allerdings ist die Markierung echt größer als eine Markierung, die auf dem Pfad bereits vorkam:  $(1, 0, 0)$ . Wir wissen also, daß wir die Schaltfolge  $t_1 t_2$  beliebig häufig wiederholen können; dabei gelangen beliebig viele Marken auf die Stelle  $c$  (vgl. Folgerung 2.9.3). Wir ersetzen deshalb die Markierung  $(1, 0, 1)$  durch die *verallgemeinerte Markierung*  $(1, 0, \omega)$ . Dabei drückt  $\omega$  aus, daß auf dieser Stelle beliebig viele Marken liegen werden können. Damit erhalten wir den folgenden Baum, in dem der Knoten  $(1, 0, \omega)$  noch weiter bearbeitet werden muß:



Wir können nun die weiteren erreichbaren Markierungen konstruieren, wobei mit  $\omega$  markierte Stellen immer mit  $\omega$  markiert bleiben. Dann erhalten wir den folgenden Baum (wobei nun alle unfertigen Knoten gestrichen worden sind, weil auf dem Pfad dorthin diese Markierung bereits vorkommt):



Die wesentliche Beobachtung, die der Konstruktion des Überdeckungsbaumes zugrunde liegt, ist die folgende: Wenn es zwei erreichbare Markierungen  $m_1 \xrightarrow{t_1 \dots t_n} m_2$  mit  $m_1 < m_2$  gibt, sind alle Stellen  $s \in S$  mit  $m_2(s) > m_1(s)$  unbeschränkt. Denn wir können die Schaltfolge  $t_1 \dots t_n$  beliebig oft durchlaufen. Bei jedem Durchlauf wird die Markenzahl auf diesen Stellen erhöht. Wir können also beliebig viele Marken auf diesen Stellen produzieren, was wir im Überdeckungsbaum mit durch ein  $\omega$  ausdrücken. Es gilt sogar die umgekehrte Richtung dieser Aussage:

**Satz 2.13 (Äquivalente Charakterisierung der Unbeschränktheit)**

Ein S/T-System ist genau dann unbeschränkt, wenn zwei erreichbare Markierungen  $m_1$  und  $m_2$  des S/T-Systems mit  $m_1 \xrightarrow{*} m_2$  und  $m_1 < m_2$  existieren.

**Beweis:** Übung 3, Aufgabe 4. □

Diesen Zusammenhang machen wir uns bei der Konstruktion des Überdeckungsbaumes zunutze. Zunächst verallgemeinern wir dazu den Begriff der Markierung, indem wir auch Einträge  $\omega$  zulassen. Für ein S/T-System mit Stellenmenge  $S$  nennen wir  $m : S \rightarrow \mathbb{N} \cup \{\omega\}$  eine *verallgemeinerte Markierung* des S/T-Systems. Für  $\omega$  gelten die folgenden Gesetze:

$$\omega + n = n + \omega = \omega = \omega + \omega \text{ für alle } n \in \mathbb{N}$$

$$\omega - n = \omega \text{ für alle } n \in \mathbb{N}$$

$$\omega > n \text{ für alle } n \in \mathbb{N}$$

Die Addition und die Inklusion auf verallgemeinerten Markierungen können wir – wie bei Markierungen – punktweise (stellenweise) definieren. Da die Definition der Schaltregel für S/T-Systeme nur die Addition auf Markierungen voraussetzt, ist damit auch die Schaltregel auf verallgemeinerten Markierungen definiert. Dabei bleibt eine mit  $\omega$  markierte Stelle beim Schalten einer Transition mit  $\omega$  markiert.

Damit definieren wir nun den Überdeckungsbaum, bzw. einen Algorithmus für die Konstruktion des Überdeckungsbaumes.

**Definition 2.14 (Überdeckungsbaum)**

Sei  $\Sigma = (N, W, m_0)$  ein S/T-System mit Netz  $N = (S, T, F)$ . Der folgende Algorithmus erzeugt einen Überdeckungsbaum von  $\Sigma$ .

**Datenstruktur** Die Knoten des zu konstruierenden Baumes sind mit verallgemeinerten Markierungen  $m$  von  $\Sigma$  beschriftet.

Während der Konstruktion des Baumes unterscheiden wir zwei Sorten von Knoten:

 *fertige Knoten*, die sicher zum konstruierten Baum gehören.

 *unfertige Knoten*, die noch nicht sicher zum Baum gehören.

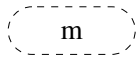
An diesen Knoten wird die Konstruktion des Baumes fortgesetzt.

Am Ende der Konstruktion hat der Baum nur noch fertige Knoten.

**Initialisierung** Die Konstruktion des Baumes beginnt mit einem einzigen Knoten, einem unfertigen Knoten, der mit der Anfangsmarkierung des

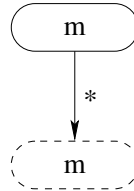
S/T-Systems beschriftet ist: 

**Algorithmus** Solange in dem bereits konstruierten Baum noch ein unfertiger

Knoten existiert, wählen wir einen unfertigen Knoten  aus: Mit diesem Knoten verfahren wir gemäß der folgenden Fallunterscheidung:

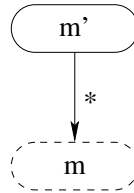
- 1. Fall:** Auf dem Pfad von von der Wurzel zu dem gewählten Knoten

 gibt es einen fertigen Knoten mit derselben Beschriftung:

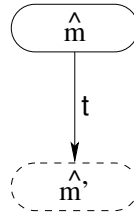


In diesem Fall löschen wir den gewählten Knoten  (nebst der eingehenden Kante).

- 2. Fall:** Auf dem Pfad von der Wurzel zu dem gewählten Knoten gibt es einen fertigen Knoten, der mit einer verallgemeinerten Markierung  $m'$  beschriftet ist, für die  $m' \leq m$  gilt und  $m'(s) < m(s) \neq \omega$  für mindestens eine Stelle  $s \in \text{gilt}$ :

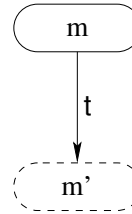


In diesem Falle ersetzen wir den gewählten Knoten durch einen fertigen Knoten mit der Beschriftung  $\hat{m}$ , wobei  $\hat{m}$  wie folgt definiert ist: für alle Stellen  $s$  mit  $m'(s) = m(s)$  gilt  $\hat{m}(s) = m(s)$ ; sonst gilt  $\hat{m}(s) = \omega$ . Außerdem erzeugen wir für jede Transition  $t \in T$  und jede verallgemeinerte Markierung  $\hat{m}'$  mit  $\hat{m} \xrightarrow{t} \hat{m}'$  einen unfertigen Nachfolgeknoten, der mit  $\hat{m}'$  beschriftet ist:



- 3. Fall:** Anderenfalls ersetzen wir den gewählten Knoten durch einen fertigen Knoten (mit derselben Beschriftung) und erzeugen für jede Transition  $t \in T$  und jede verallgemeinerte Markierung  $m'$  mit

$m \xrightarrow{t} m'$  einen unfertigen Nachfolgeknoten, der mit  $m'$  beschriftet ist:



Bevor wir uns einige schöne Eigenschaften des Überdeckungsbaumes überlegen, sollten wir uns tunlichst überlegen, ob den der Algorithmus für jedes S/T-System terminiert. Denn nur dann liefert uns der Algorithmus einen Überdeckungsbaum als Ergebnis.

**Satz 2.15 (Existenz eines Überdeckungsbaumes)**

Der Algorithmus zur Konstruktion eines Überdeckungsbaumes terminiert für jedes S/T-System; insbesondere existiert für jedes S/T-System ein Überdeckungsbaum.

**Beweis:** Der Beweis ist dem Beweis von Satz 2.13 (Übung 3, Aufgabe 4) sehr ähnlich. Die wesentliche Idee ist die folgende: Wenn der Algorithmus nicht terminiert, muß es einen unendlichen Pfad geben. Auf diesem Pfad müssen immer wieder verallgemeinerte Markierungen  $m$  vorkommen, die echt größer sind als eine Markierung, die auf dem Pfad bereits vorkam. Dann entsteht bei der Konstruktion des Überdeckungsbaumes an dieser Stelle mindestens ein weiteres  $\omega$ . Die Anzahl der  $\omega$ -Einträge der verallgemeinerten Markierungen auf diesem Pfad nimmt auf diesem Pfad immer wieder echt zu. Da die Anzahl der  $\omega$ -Einträge durch die Anzahl der Stellen beschränkt ist, kann der Pfad nicht unendlich sein.

Ein ausführlicher Beweis wird in der Übung (Blatt 4, Aufgabe 3) besprochen.

**Bemerkung:** *Tatsächlich könnte man Satz 2.13 als Folgerung des Satzes 2.15 auffassen. Aus didaktischen Gründen haben wir hier jedoch darauf verzichtet.*

□

*Der Algorithmus zur Konstruktion eines Überdeckungsbaumes ist nicht deterministisch. Dann ist natürlich die Frage, ob es für jedes S/T-System einen eindeutigen Überdeckungsbaum gibt. Eine Antwort auf diese Frage werden*

*wir uns in der Übung (Blatt 4, Aufgabe 3) überlegen. Solange wir nicht wissen, ob der Überdeckungsbaum eines S/T-Systems eindeutig ist, können wir also nicht von DEM Überdeckungsbaum eines S/T-Systems sprechen, sondern nur von EINEM Überdeckungsbaum des S/T-Systems.*

Damit haben wir nun ein Verfahren, um zu entscheiden, ob ein S/T-System beschränkt ist oder nicht.

**Folgerung 2.16 (Entscheidbarkeit der Beschränktheit)**

Für jedes S/T-System ist entscheidbar, ob es beschränkt ist.

**Beweis:** Wir konstruieren zunächst einen Überdeckungsbaum des S/T-Systems. Wenn der Überdeckungsbaum keine Markierungen  $m$  mit einem  $\omega$ -Eintrag (d.h. mit  $m(s) = \omega$  für ein  $s \in S$ ) besitzt, ist der Erreichbarkeitsgraph offensichtlich endlich, denn bei der Konstruktion des Überdeckungsbaumes haben wir alle erreichbaren Markierungen erzeugt (es wurden nur Knoten gestichen, deren Markierung zum zweiten Mal vorkommt). Nach Folgerung 2.12 ist das Netz dann beschränkt.

Wenn im Überdeckungsbaum eine verallgemeinerte Markierung mit einem  $\omega$ -Eintrag vorkommt, dann gibt es im S/T-System zwei erreichbare Markierungen  $m_1 \xrightarrow{*} m_2$  mit  $m_1 < m_2$ , denn nur so kann ein Eintrag  $\omega$  bei der Konstruktion entstehen. Nach Satz 2.13 ist das S/T-System dann unbeschränkt.  $\square$

... Hier kommt vielleicht noch irgendwann mal ein Beispiel für einen Überdeckungsbaum ...

Wenn wir nur an der Frage interessiert sind, ob ein S/T-System beschränkt ist oder nicht, ist es natürlich nicht nötig, den Überdeckungsbaum fertig zu konstruieren. Sobald wir einen Knoten mit einem  $\omega$ -Eintrag erzeugt haben, wissen wir, daß das S/T-System unbeschränkt ist. Die Frage ist nun, ob es sich lohnt, den Überdeckungsbaum fertig zu konstruieren. Für den Augenblick<sup>4</sup> beantworten wir diese Frage mit „j“, denn wir können aus dem Überdeckungsbaum weitere Aussagen über das S/T-System ableiten. Für das Beispiel 2.2 können wir aus dem Überdeckungsbaum ersehen, daß es keine erreichbare Markierung des S/T-Systems gibt, in der sowohl die Stelle **a** als auch die Stelle **b** markiert sind.

*In diesem Beispiel ist das auch mit einer S-Invariante beweisbar. Aber das ist nicht immer möglich.*

---

<sup>4</sup> Bei der Konstruktion des Überdeckungsbaumes stößt man leider schnell an praktische Grenzen.

Im folgenden untersuchen wir deshalb den Zusammenhang zwischen dem Erreichbarkeitsgraphen und dem Überdeckungsbaum eines S/T-Systems. Die erste Eigenschaft besagt, daß die verallgemeinerten Markierungen des Überdeckungsbaumes die erreichbaren Markierungen *überdecken*. Dies hat dem Überdeckbarkeitsbaum auch seinem Namen gegeben.

**Satz 2.17 (Überdeckung)**

Sei  $\Sigma$  ein S/T-System und  $\Gamma$  ein Überdeckungsbaum von  $\Sigma$ .

Für jede erreichbare Markierung  $m$  von  $\Sigma$  gibt es eine verallgemeinerte Markierung  $m'$  des Überdeckungsbaumes  $\Gamma$  mit  $m \leq m'$ .

**Beweis:** Die wesentliche Idee dabei ist, den Pfad von der Anfangsmarkierung des S/T-Systems zu  $m$  im Erreichbarkeitsgraphen im Überdeckungsbaum zu simulieren. Dies wird durch eine Induktion formalisiert (vgl. Übung 4, Aufgabe 3).  $\square$

In der Praxis wird dieser Satz häufig in der umgekehrten Richtung angewendet: Wenn es für eine Markierung  $m$  keine Markierung  $m'$  des Überdeckungsbaumes mit  $m \leq m'$  gibt, dann ist  $m$  im S/T-System nicht erreichbar. Das ist das Argument, warum im Beispiel 2.2 die Markierung  $m = (1, 1, 0) = [a, b]$  nicht erreichbar ist, denn im Überdeckungsbaum gibt es keine verallgemeinerte Markierung  $m'$ , die  $m$  *überdeckt*.

Allerdings liefert uns Satz 2.17 keine exakte Charakterisierung der Menge der erreichbaren Markierungen des S/T-Systems. Wenn für eine Markierung  $m$  eine verallgemeinerte Markierung  $m'$  des Überdeckungsbaumes mit  $m \leq m'$  gilt, dann muß  $m$  nicht unbedingt erreichbar sein! Der Überdeckungsbaum liefert also keine notwendige und hinreichende Bedingung für die Erreichbarkeit einer Markierungen des Systems. Als nächstes überlegen wir uns deshalb, wie scharf der Überdeckungsbaum die Menge der erreichbaren Markierungen eingrenzt. Dazu überlegen wir uns, daß im Überdeckungsbaum wirklich nur solche  $\omega$ -Einträge vorkommen, die auch aufgrund der Unbeschränktheit nötig sind. Der folgende Satz besagt informell, daß sich jede verallgemeinerte Markierung  $m'$  des Überdeckungsbaumes als „Limes“ einer unendlichen Folge von erreichbaren Markierungen  $m_1, m_2, m_3, \dots$  darstellen läßt, wobei  $\omega$  der Limes einer aufsteigenden Folge von Zahlen ist:  $m' = \lim_{i \rightarrow \infty} m_i$ . Allerdings sparen wir uns bei der Formulierung des Satzes die Definition des Limes.

**Satz 2.18 (Simultane Unbeschränktheit)**

Sei  $\Sigma$  ein S/T-System und  $\Gamma$  ein Überdeckungsbaum von  $\Sigma$  und  $m'$  eine verallgemeinerte Markierung von  $\Gamma$ . Dann gibt es eine Folge von erreichbaren

Markierungen  $m_1, m_2, \dots$  von  $\Sigma$  mit  $m_i(s) = m'(s)$  für jede Stelle  $s \in S$  mit  $m'(s) \neq \omega$  und  $m_i(s) \geq i$  für jede Stelle  $s \in S$  mit  $m'(s) = \omega$ .

**Beweis:** Auch hier gilt wieder: Der Beweis wird in der Übung ausformuliert. Die wesentliche Idee dabei ist, für ein gegebenes  $i \in \mathbb{N}$  einen Pfad zu Markierung  $m_i$  im Erreichbarkeitsgraphen von  $\Sigma$  zu konstruieren, indem wir uns von der verallgemeinerten Markierung  $m'$  ausgehend rückwärts durch den Überdeckungsbaum bewegen und die Schleifen  $m_1 \xrightarrow{w} m_2$  mit  $m_1 < m_2$ , die zur Konstruktion eines  $\omega$  geführt haben, hinreichend häufig durchlaufen).  $\square$

*Wir sagen, daß alle Stelle die in  $m'$  einen  $\omega$ -Eintrag besitzen simultan unbeschränkt sind, weil wir für jedes  $i \in \mathbb{N}$  „simultan“ in einer einzelnen Markierung in jeder dieser Stellen mehr als  $i$  Marken erzeugen können.*

**Achtung:** In Satz 2.18 wird nicht gefordert, daß für die Folge  $m_1, m_2, \dots$  auch gilt  $m_1 \xrightarrow{*} m_2 \xrightarrow{*} \dots$ . Wir werden uns in Übung 4, Aufgabe 4 überlegen, ob der Satz auch unter diesen verschärften Bedingungen gilt.

Der Überdeckungsbaum liefert auch Aussagen über einige andere Eigenschaften von S/T-Systemen. Dazu definieren wir zunächst einen Katalog von Eigenschaften von S/T-Systemen.

### Definition 2.19 (Katalog von Eigenschaften)

Sei  $\Sigma = (N, W, m_0)$  ein S/T-System mit Netz  $N = (S, T, F)$ .

1. Für eine natürliche Zahl  $n \in \mathbb{N}$  heißt eine Stelle  $s \in S$  *n-sicher* in  $\Sigma$ , wenn für jede erreichbare Markierung  $m$  von  $\Sigma$  gilt  $m(s) \leq n$
2. Eine Stelle  $s \in S$  heißt *beschränkt* in  $\Sigma$ , wenn die Stelle  $s$  für ein  $n \in \mathbb{N}$  *n-sicher* ist.

*Oft wird eine beschränkte Stelle auch sicher genannt. Da dies aber leicht zu Verwechslungen mit 1-sicheren Stellen (ein häufiger Spezialfall der n-Sicherheit) führt, benutzen wir die Bezeichnung „beschränkt“.*

3. Für eine natürliche Zahl  $n \in \mathbb{N}$  heißt das S/T-System  $\Sigma$  *n-sicher*, wenn jede Stelle  $s \in S$  von  $\Sigma$  *n-sicher* ist.

*Die Beschränktheit eines S/T-Systems gehört eigentlich an dieser Stelle in den Katalog der Eigenschaften. Da wir aber schon eine Definition dafür haben (Def. 2.11) verzichten wir hier darauf.*

4. Das S/T-System ist *strukturell beschränkt*, wenn es für jede beliebige Anfangsmarkierung  $m$  beschränkt ist, d.h., wenn jedes S/T-System  $\Sigma' = (N, W, m)$  beschränkt ist.
5. Eine Teilmenge  $S' \subseteq S$  von Stellen des S/T-Systems  $\Sigma$  heißt *simultan unbeschränkt*, wenn es für jede natürliche Zahl  $n \in \mathbb{N}$  eine erreichbare Markierung  $m$  von  $\Sigma$  mit  $m(s) \geq n$  für alle  $s \in S'$  gibt.
6. Eine Transition  $t \in T$  heißt *tot* in einer Markierung  $m$ , wenn es keine Markierung  $m' \in [m]$  gibt, in der  $t$  aktiviert ist.
7. Eine Transition  $t \in T$  heißt *tot* in  $\Sigma$ , wenn sie in der Anfangsmarkierung  $m_0$  von  $\Sigma$  tot ist.
8. Das S/T-System heißt *verklemmungsfrei*, wenn in keiner erreichbaren Markierung alle Transitionen tot sind.

*Die Verklemmungsfreiheit kann man auch wie folgt formulieren: In jeder erreichbaren Markierung ist mind. eine Transition aktiviert.*

9. Eine Transition heißt *lebendig* in  $\Sigma$ , wenn sie in keiner erreichbaren Markierung von  $\Sigma$  tot ist.

*Eine Transition  $t$  ist also genau dann lebendig, wenn für jeder erreichbaren Markierung  $m$  des S/T-Systems eine Markierung  $m'$  mit  $m \xrightarrow{*} m' \xrightarrow{t}$  gibt.*

10. Das S/T-System heißt *lebendig*, wenn jede Transition des S/T-Systems lebendig ist.
11. Das S/T-System heißt *strukturell lebendig*, wenn es eine Anfangsmarkierung  $m$  gibt, für die das S/T-System lebendig ist, d.h. wenn es ein S/T-System  $\Sigma' = (N, W, m)$  gibt, das lebendig ist.

### Bemerkungen

1. Ein S/T-System ist genau dann beschränkt (gemäß Def. 2.11), wenn jede Stelle des S/T-Systems beschränkt ist.
2. Für jedes  $m \geq n$  ist jedes  $n$ -sichere S/T-System auch  $m$ -sicher; und jede  $n$ -sichere Stelle ist  $m$ -sicher.

3. **Achtung:** Tot ist (leider) nicht ganz das Gegenteil von lebendig.
4. **Achtung:** Die strukturelle Lebendigkeit ist über die Existenz einer lebendigen Anfangsmarkierung definiert; die strukturelle Beschränktheit ist über die Beschränktheit für alle Anfangsmarkierungen definiert.

Es gilt: Jedes strukturell beschränkte S/T-System ist beschränkt. Jedes lebendige S/T-System ist strukturell lebendig. Die Umkehrungen dieser Aussagen gelten im allgemeinen nicht.

Als nächstes überlegen wir uns, welche Eigenschaften wir mit Hilfe des Überdeckungsbaumes entscheiden können. Leider sind das nicht sehr viele. Aber für einige Eigenschaften können wir wenigstens notwendige Bedingungen formulieren. Beispielsweise liefert Satz 2.17 eine notwendige Bedingung dafür, daß eine Markierung  $m$  erreichbar ist: es muß eine verallgemeinerte Markierung  $m'$  des Überdeckungsbaumes geben, mit  $m \leq m'$ . Der Überdeckungsbaum liefert eine notwendige und hinreichende Bedingung für die folgenden Fragen:

1. Ist eine bestimmte Stelle  $s$  beschränkt?
2. Ist das S/T-Systems beschränkt?
3. Ist eine bestimmten Stellenmenge  $S'$  simultan unbeschränkt?
4. Ist eine bestimmte Transition  $t$  tot?

Der Überdeckungsbaum liefert eine notwendige (und im allgemeinen nicht hinreichende) Bedingung für die folgenden Eigenschaften:

1. Ist das System verklemmungsfrei?
2. Ist eine bestimmte Transition  $t$  lebendig?
3. Ist das S/T-Systems lebendig?
4. Ist eine bestimmte Markierung erreichbar?

Wir können dem Überdeckungsbaum also durchaus noch einige Eigenschaften des S/T-Systems ansehen. Leider ist der Überdeckungsbaum in der Praxis jedoch weniger nützlich als es im Augenblick den Anschein hat. Der Grund dafür ist, daß der Überdeckungsbaum eines S/T-Systems extrem groß werden kann: Die Größe des Überdeckungsbaumes wächst stärker als die primitiv rekursiven Funktionen in der Größe des S/T-Systems.

*Das Wachstum ist stärker als alles, was man sich üblicherweise intuitiv vorstellen kann!*

*Dies zeigt, daß wir uns bei Eigenschaften von S/T-Systemen hart an der Grenze des Entscheidbaren befinden. Wir werden später noch sehen, daß bereits für einfach aussehende Erweiterungen der S/T-Systeme die Beschränktheit nicht mehr entscheidbar ist.*

Deshalb beschäftigen wir uns im folgenden mit anderen Techniken, die uns helfen Eigenschaften eines S/T-Systems zu analysieren. In den meisten Fällen liefern auch diese Verfahren nur notwendige oder hinreichende Kriterien – aber die Kriterien sind effizienter überprüfbar.

## 4 Lineare Invarianten

Beim Beweis der Folgerung 2.9 haben wir bereits gesehen, daß für zwei Markierungen  $m_1$  und  $m_2$  eines S/T-Systems mit  $m_1 \xrightarrow{t_1 \dots t_n} m_2$  gilt<sup>5</sup>  $m_2 = m_1 + \underline{t}_1 + \dots + \underline{t}_n$ . Die neue Markierung ergibt sich also durch Addition einer Linearkombination der *Schaltvektoren*  $\underline{t}$  auf die Ausgangsmarkierung. Diese einfache Beobachtung hat einige interessante und weitreichenden Konsequenzen und liefert uns Analysetechniken für Eigenschaften von Petrinetzen.

### 4.1 Grundbegriffe der linearen Algebra

Für eine möglichst elegante und einheitliche Darstellung dieser Analysetechniken begeben wir uns auf das Gebiet der linearen Algebra. Hier fassen wir die für uns wesentlichen Begriffe kurz zusammen, wobei wir die Notationen für unsere Zwecke anpassen.

**Vektoren** Für eine Menge  $A$  können wir eine Abbildung  $\underline{v} : A \rightarrow \mathbb{Z}$  als einen *Vektor* auffassen. Wir nennen ihn einen Vektor über  $A$  oder kurz *A-Vektor*. Auf  $A$ -Vektoren  $\underline{v}_1$  und  $\underline{v}_2$  ist die Addition punktweise definiert, d. h.  $\underline{v} = \underline{v}_1 + \underline{v}_2$  ist definiert durch  $\underline{v}(a) = \underline{v}_1(a) + \underline{v}_2(a)$  für  $a \in A$ . Das Produkt eines *Skalars*  $z \in \mathbb{Z}$  mit einem  $A$ -Vektor  $\underline{v}_1(a)$  ist definiert als  $A$ -Vektor  $\underline{v} = z \cdot \underline{v}_1$  mit  $\underline{v}(a) = z \cdot \underline{v}_1(a)$  für  $a \in A$ .

Mit  $\underline{0}$  bezeichnen wir den Vektor mit  $\underline{0}(a) = 0$  für alle  $a \in A$ . Wir nennen ihn den *Nullvektor*.

---

<sup>5</sup> Weil diese Beziehung so fundamental ist, bekommt sie später einen eigenen Namen. Wir nennen sie die *Zustandsgleichung* oder die *Markierungsgleichung*.

Markierungen eines  $S/T$ -Systems  $\Sigma = (N, W, m)$  mit Netz  $N = (S, T, F)$  können wir als spezielle  $S$ -Vektoren auffassen, für die alle Einträge nicht negativ sind. Insbesondere sind für eine Transition  $t$  die Markierungen  $-t$  und  $t^+$   $S$ -Vektoren; und auch der Schaltvektor  $\underline{t}$  ein  $S$ -Vektor.

Für zwei  $A$ -Vektoren  $\underline{v}_1$  und  $\underline{v}_2$  definieren wir das *Skalarprodukt*  $\underline{v}_1 \cdot \underline{v}_2 = \sum_{a \in A} \underline{v}_1(a) \cdot \underline{v}_2(a)$ .

**Achtung:** Das Produkt mit einem Skalar ist nicht zu verwechseln mit dem Skalarprodukt. Im ersten Fall wird ein Skalar mit einem Vektor multipliziert, und das Ergebnis ist ein Vektor. Im zweiten Fall werden zwei Vektoren miteinander multipliziert, und das Ergebnis ist ein Skalar.

Beim Skalarprodukt wird  $\underline{v}_1$  oft als Zeilenvektor und  $\underline{v}_2$  als Spaltenvektor aufgefaßt. Wir geben die „Orientierung“ der Vektoren meist nicht an, da sie sich bei uns immer aus dem Kontext ergibt.

Manchmal wollen wir über die Einträge eines Vektors reden, die von 0 verschieden sind. Wir nennen diese die *Trägermenge* (engl. support) des Vektors:  $\text{supp}(\underline{v}) = \{a \in A \mid \underline{v}(a) \neq 0\}$

Für ein Wort  $w \in A^*$  bezeichnet  $\#w$  einen  $A$ -Vektor, der für jedes  $a \in A$  die Anzahl der Vorkommen des Zeichens  $a$  in  $w$  angibt. Dieser Vektor heißt der *Parikh-Vektor* von  $w$ . Den Wert  $\#w$  können wir induktiv über die Länge von  $w$  definieren:

- $(\#\varepsilon)(a) = 0$
- $(\#(a \circ w))(a) = 1 + \#w(a)$
- $(\#(b \circ w))(a) = \#w(a)$  für  $b \neq a$

**Matrizen** Für zwei Mengen  $A$  und  $B$  können wir eine Abbildung  $M : A \times B \rightarrow \mathbb{Z}$  als eine  $A \times B$ -Matrix auffassen. Für eine  $A \times B$ -Matrix  $M$  und einen  $B$ -Vektor  $\underline{v}$  definieren wir das Produkt  $\underline{v}' = M \cdot \underline{v}$  als  $A$ -Vektor mit  $\underline{v}'(a) = \sum_{b \in B} M(a, b) \cdot \underline{v}(b)$ .

Für eine  $A \times B$ -Matrix  $M$  heißt die  $B \times A$ -Matrix  $M^T$  mit  $M^T(b, a) = M(a, b)$  die zu  $M$  *transponierte Matrix*.

Eine  $A \times B$ -Matrix  $M$  wird oft durch eine Liste von Spaltenvektoren  $\underline{t}(b_1), \underline{t}(b_2), \dots, \underline{t}(b_n)$  für die Spalten  $B = \{b_1, b_2, \dots, b_n\}$  notiert:

$$M = [\underline{t}(b_1) \ \underline{t}(b_2) \ \dots \ \underline{t}(b_n)]$$

Für ein  $b_i \in B$  kann man mit Hilfe des Parikh-Vektors  $\#b_i$  leicht den Spaltenvektor  $\underline{t}(b_i)$  aus der Matrix  $M$  extrahieren: Es gilt

$$M \cdot \#b_i = \underline{t}(b_i)$$

Für eine  $A \times B$ -Matrix  $M$ , einen  $A$ -Vektor  $\underline{v}$  und einen  $B$ -Vektor  $\underline{x}$  mit  $M \cdot \underline{x} = \underline{v}$  nennen wir  $\underline{x}$  auch die Lösung des linearen Gleichungssystems  $M \cdot \underline{x} = \underline{v}$ .

**Achtung:** Im strengen Sinne der linearen Algebra bilden unsere Vektoren *keinen* Vektorraum, da wir nur Vektoren über den ganzen Zahlen betrachten, die keinen Körper bilden. Mit Gleichungssystemen und Ungleichungssystemen über den ganzen und später sogar über den natürlichen Zahlen, befinden wir uns auf dem Gebiet der *ganzzahligen linearen Programmierung*, worüber man eine eigenständige Vorlesung halten könnte. Allerdings werden wir das hier nicht vertiefen; häufig werden wir einfach so tun, als ob wir Gleichungssysteme über den rationalen Zahlen betrachten. Deshalb reichen im folgenden elementare Grundkenntnisse der linearen Algebra aus.

## 4.2 Inzidenzmatrix und Zustandsgleichung

Die *Inzidenzmatrix* eines S/T-Systems  $\Sigma$  ist eine Liste von Spaltenvektoren  $(\underline{t}_1 \underline{t}_2 \dots \underline{t}_n)$ , wobei die  $\underline{t}_i$  die Schaltvektoren der Transitionen des S/T-Systems sind.

### Definition 2.20 (Inzidenzmatrix eines S/T-Systems)

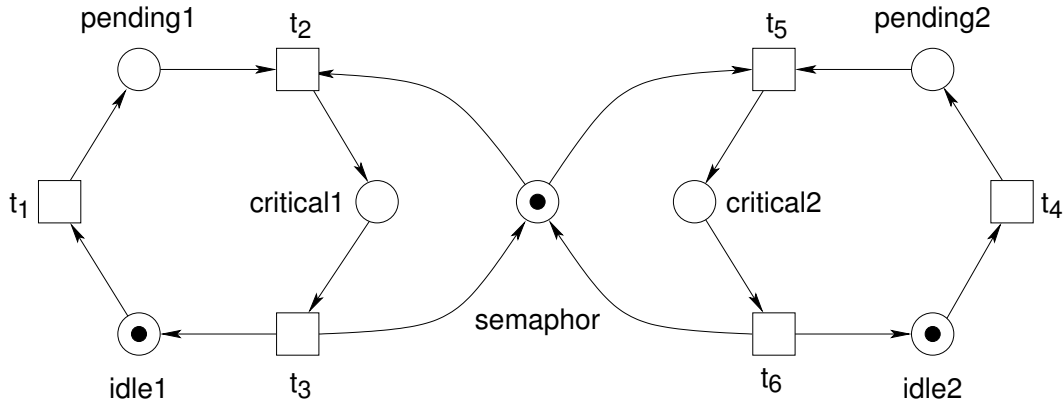
Sei  $\Sigma = (N, W, m)$  ein S/T-System mit Netz  $N = (S, T, F)$ .

Wir definieren die *Inzidenzmatrix* von  $\Sigma$  als  $S \times T$ -Matrix  $C_\Sigma$  mit  $C_\Sigma(s, t) = \underline{t}(s)$  für alle  $s \in S$  und  $t \in T$ .

Wenn  $\Sigma$  aus dem Kontext klar ist, lassen wir den Index  $\Sigma$  bei der Bezeichnung der Inzidenzmatrix  $C_\Sigma$  weg und schreiben  $C$ .

### Beispiel 2.3 (Inzidenzmatrix des Mutex-Beispiels)

Wir betrachten als Beispiel die Inzidenzmatrix unseres „Mutex-Beispiels“. Hier ist nochmal das System:



Die Inzidenzmatrix sieht dann wie folgt aus:

$C$	$t_1$	$t_2$	$t_3$	$t_6$	$t_5$	$t_4$
$i1$	-1		1			
$p1$	1	-1				
$c1$		1	-1			
$s$		-1	1	1	-1	
$c2$				-1	1	
$p2$					-1	1
$i2$				1		-1

Dabei haben wir die Stellen und Transitionen so angeordnet, daß sich die Achsensymmetrie des S/T-Systems in der Punktsymmetrie der Inzidenzmatrix widerspiegelt. Die Reihenfolge der Stellen und Transitionen in der graphischen Darstellung ist frei wählbar, sie muß aber angegeben werden.

*In Def. 2.20 haben wir für ein S/T-System die Inzidenzmatrix  $C$  definiert. Frage: Können wir umgekehrt aus der Inzidenzmatrix eindeutig auf das zugehörige S/T-System schließen?*

*Die Antwort ist nein! Das Problem ist, daß Schlingen in der Inzidenzmatrix nicht ausreichend berücksichtigt werden. Nur dann, wenn wir wissen, daß das S/T-System schlingenfrei war, können wir es eindeutig aus der Inzidenzmatrix rekonstruieren.*

Wie bereits eingangs erwähnt gilt mit  $m_1 \xrightarrow{t_1 \dots t_n} m_2$  auch  $m_2 = m_1 + \underline{t}_1 + \dots + \underline{t}_n$ . Mit der Beobachtung über den Parikh-Vektor gilt  $C \cdot \#t = \underline{t}$ . Wir können die Gleichung also wie folgt umformen  $m_2 = m_1 + (C \cdot \#t_1) + \dots + (C \cdot \#t_n) = m_1 + C \cdot (\#t_1 + \dots + \#t_n) = m_1 + C \cdot \#(t_1 \dots t_n)$ .

Damit erhalten wir die *Zustandsgleichung*<sup>6</sup>:

**Satz 2.21 (Zustandsgleichung)**

Sei  $\Sigma = (N, W, m)$  ein S/T-System mit Netz  $N = (S, T, F)$  und Inzidenzmatrix  $C$ . Für zwei Markierungen  $m_1$  und  $m_2$  von  $\Sigma$  und ein Schaltwort  $w$  mit  $m_1 \xrightarrow{w} m_2$  gilt  $m_2 = m_1 + C \cdot \#w$ .

**Beweis:** Ausformulierung der Vorüberlegung. □

**Bemerkung:** Die Umkehrung von Satz 2.21 gilt nicht. Dazu kann man sich leicht ein Gegenbeispiel überlegen (vgl. Übung 3, Aufgabe 3f).

Aber für jedes beliebige Schaltwort  $w \in T^*$  kann man sich leicht überlegen, daß eine Markierung  $m$  mit  $m \xrightarrow{w} m + C \cdot \#w$  existiert. Man muß nur genügend viele Marken „spendieren“ damit alle Transitionen in  $w$  aktiviert sind (für  $w = t_1 \dots t_n$  können wir beispielsweise  $m = {}^-t_1 + \dots + {}^-t_n$  wählen).

### 4.3 Stellen-Invarianten

Bereits in der informellen Einführung haben wir *Stellen-Invarianten* kennengelernt. Jetzt werden wir sie formal definieren. Die Gewichtung der Stellen können wir durch einen  $S$ -Vektor  $\underline{i}$  formalisieren. Die mit  $\underline{i}$  *gewichtete Markierung*  $m$  können wir als Skalarprodukt  $m \cdot \underline{i}$  ausdrücken. Daß das Schalten einer Transition  $t$  die *gewichtete Markensumme* nicht verändert, bedeutet dann  ${}^-t \cdot \underline{i} = t^+ \cdot \underline{i}$ , oder  $\underline{t} \cdot \underline{i} = 0$ .

Dies können wir durch  $C^T \cdot \underline{v} = \underline{0}$  für alle Transitionen gleichzeitig formulieren, was uns die folgende Definition liefert:

**Definition 2.22 (Stellen-Invariante)**

Sei  $\Sigma$  ein S/T-System mit Inzidenzmatrix  $C$ . Ein  $S$ -Vektor  $\underline{i}$  heißt *Stellen-Invariante* (S-Invariante) von  $\Sigma$ , wenn  $C^T \cdot \underline{i} = \underline{0}$  gilt.

*Eine S-Invariante  $\underline{i}$  ist also eine ganzzahlige Lösung des linearen Gleichungssystems  $C^T \cdot \underline{i} = \underline{0}$ .*

**Beispiel 2.4 (S-Invarianten des Mutex-Systems)**

Unser Mutex-Beispiel hat beispielsweise die drei folgenden S-Invarianten. Wir repräsentieren sie als Spaltenvektoren neben der Inzidenzmatrix, weil man auf

---

<sup>6</sup> Die Zustandsgleichung wird manchmal auch *Markierungsgleichung* genannt.

diese Weise einfach überprüfen kann, ob es sich wirklich um S-Invarianten handelt.

$C$	$t_1$	$t_2$	$t_3$	$t_6$	$t_5$	$t_4$	$i_1$	$i_2$	$i_3$
$i1$	-1		1				1		
$p1$	1	-1	1				1		
$c1$		1	-1				1	1	
$s$		-1	1	1	-1			1	
$c2$				-1	1			1	1
$p2$					-1	1			1
$i2$				1		-1			1

*Zum Berechnen der Invarianten ist es zweckmäßiger die transponierte Matrix  $C^T$  zu benutzen – einfach deshalb, weil wir es gewöhnt sind, Gleichungssysteme in dieser Form zu lösen.*

Ähnlich wie Markierungen können wir S-Invarianten auch als formale Summen darstellen:

- $\underline{i_1} = i1 + p1 + c1$
- $\underline{i_2} = c1 + s + c2$
- $\underline{i_3} = i2 + p2 + c2$

Diese Darstellung ist insbesondere bei der Anwendung in Beweisen zweckmäßig.

**Bemerkung:** Um S-Invarianten zu finden, können wir den Gauß-Algorithmus anwenden. Der liefert und allerdings eine Basis für die rationalen Lösungen des Gleichungssystems. Das ist für allerdings kein Problem; wenn wir eine rationale Lösung haben können wir sie durch geeignete Skalierung zu einer ganzzahligen Lösung machen. Schwerer wird es, wenn wir darüber hinaus noch positive Lösungen suchen – dafür verweisen wir auf die Techniken der linearen Programmierung. Für uns reicht es zu wissen, daß es Algorithmen gibt, die solche Lösungen finden.

Die Definition der S-Invarianten haben wir bereits über ihre wesentliche Eigenschaft motiviert: Beim Schalten ändert sich die gewichtete Markensumme nicht. Tatsächlich ist dies eine „genau-dann-wenn“-Beziehung.

**Satz 2.23**

Sei  $\Sigma$  ein S/T-System mit Inzidenzmatrix  $C$ . Ein  $S$ -Vektor  $\underline{i}$  ist genau dann eine  $S$ -Invariante von  $\Sigma$ , wenn für alle Markierungen  $m_1$  und  $m_2$  von  $\Sigma$  mit  $m_1 \rightarrow m_2$  gilt  $m_1 \cdot \underline{i} = m_2 \cdot \underline{i}$ .

**Beweis:** Wir beweisen die beiden Richtungen der ‘‘genau-dann-wenn‘‘-Beziehung einzeln:

„ $\Rightarrow$ “ Sei  $C^T \cdot \underline{i} = \underline{0}$  und gelte  $m_1 \rightarrow m_2$ . Es ist zu zeigen:  $m_1 \cdot \underline{i} = m_2 \cdot \underline{i}$ .

Gemäß Def. von  $m_1 \rightarrow m_2$  gibt es eine Transition  $\mathbf{t} \in T$  mit  $m_1 \xrightarrow{\mathbf{t}} m_2$ . Also gilt  $m_2 = m_1 + \underline{\mathbf{t}}$ . Wegen  $C^T \cdot \underline{i} = \underline{0}$  gilt insbesondere  $\underline{\mathbf{t}}^T \cdot \underline{i} = 0$ , wobei wir konventionsgemäß  $\underline{\mathbf{t}} \cdot \underline{i} = 0$  schreiben.

Also gilt  $m_2 \cdot \underline{i} = (m_1 + \underline{\mathbf{t}}) \cdot \underline{i} = (m_1 \cdot \underline{i}) + (\underline{\mathbf{t}} \cdot \underline{i}) = (m_1 \cdot \underline{i}) + 0 = m_1 \cdot \underline{i}$ .

*Diese Richtung des Beweises folgt (mit elementaren Kenntnissen aus der linearen Algebra) unmittelbar aus der Zustandsgleichung:*

*Für  $m_1 \xrightarrow{\mathbf{w}} m_2$  gilt gemäß Zustandsgleichung  $m_2 = m_1 + C \cdot \#\mathbf{w}$ . Also gilt  $m_2 \cdot \underline{i} = \underline{i} \cdot m_2 = \underline{i} \cdot (m_1 + C \cdot \#\mathbf{w}) = \underline{i} \cdot m_1 + \underline{i} \cdot C \cdot \#\mathbf{w} = \underline{i} \cdot m_1 + \underline{0} \cdot \#\mathbf{w} = \underline{i} \cdot m_1 = m_1 \cdot \underline{i}$ .*

*Allerdings müssten wir in diesem Falle mit der ‘‘Orientierung‘‘ der Vektoren etwas sorgfältiger umgehen, da wir Vektoren sowohl von links als auch von rechts mit der Matrix multiplizieren. Da es sich aber nur um eine Randnotiz handelt, nehmen wir es hier damit nicht so genau.*

„ $\Leftarrow$ “ Gelte nun  $m_1 \cdot \underline{i} = m_2 \cdot \underline{i}$  für jedes Paar von Markierungen  $m_1 \rightarrow m_2$ . Es ist zu zeigen, daß dann gilt  $C^T \cdot \underline{i} = \underline{0}$ ; dafür genügt es zu zeigen, daß für jede Zeile  $\underline{\mathbf{t}}$  der Matrix  $C^T$ , d. h. für jede Transition  $\mathbf{t}$  des S/T-Systems, gilt  $\underline{\mathbf{t}} \cdot \underline{i} = 0$ .

Sei also  $\mathbf{t} \in T$  eine beliebige Transition von  $\Sigma$ . Offensichtlich gilt für die beiden Markierungen  $m_1 = \mathbf{t}^-$  und  $m_2 = \mathbf{t}^+$  auch  $\mathbf{t}^- \rightarrow \mathbf{t}^+$ . Also gilt gemäß Voraussetzung  $\mathbf{t}^- \cdot \underline{i} = \mathbf{t}^+ \cdot \underline{i}$ . Durch elementare Umformung erhalten wir  $0 = (\mathbf{t}^+ \cdot \underline{i}) - (\mathbf{t}^- \cdot \underline{i}) = (\mathbf{t}^+ - \mathbf{t}^-) \cdot \underline{i} = \underline{\mathbf{t}} \cdot \underline{i}$ .

□

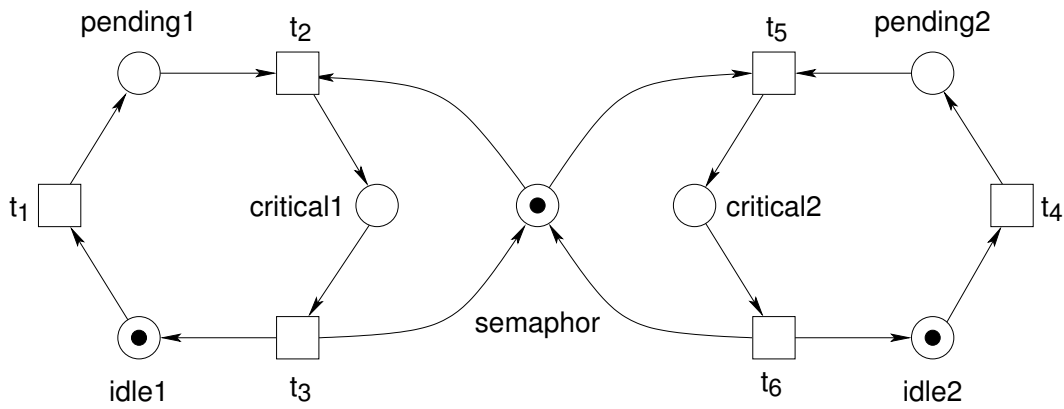
**Folgerung 2.24**

Sei  $\Sigma$  ein S/T-System mit  $S$ -Invariante  $\underline{i}$ , und sei  $m$  eine Markierung von  $\Sigma$ . Dann gilt für jede Markierung  $m' \in [m]$ :  $m' \cdot \underline{i} = m \cdot \underline{i}$ .

Folgerung 2.24 liefert eine notwendige Bedingung für die Erreichbarkeit einer Markierung. Dies können wir im Umkehrschluß ausnutzen, um zu beweisen, daß bestimmte Markierungen nicht erreichbar sind – wie bereits in der Einleitung gesehen. Hier präzisieren wir nun diese Argumentation.

### Beispiel 2.5 (S-Invariante: Anwendung)

Wir betrachten wieder das Mutex-Beispiel:



Wie im Beispiel 2.4 gesehen hat dieses Beispiel die S-Invariante  $i_2 = c1 + s + c2$ .

Für die Anfangsmarkierung  $m_0$  gilt  $m_0 \cdot i_2 = 1$ . Nun muß für alle erreichbaren Markierungen  $m$  des Systems gelten  $m \cdot i_2 = 1$ . Dies notieren wir durch  $\square c1 + s + c2 = 1$ , wobei der vorangestellte Kasten  $\square$  besagt, daß diese Gleichung in allen erreichbaren Markierungen gelten muß (sprich „immer gilt“ oder engl. „always“).

Da die Bezeichner  $c1$ ,  $s$  und  $c2$  eine Anzahl von Marken repräsentieren, gilt  $c1 \geq 0$ ,  $s \geq 0$  und  $c2 \geq 0$ . Zusammen mit der obigen Aussage folgt also  $\square c1 + c2 \leq 1$ ; damit können also die Stellen  $c1$  und  $c2$  in keiner erreichbaren Markierung beide eine Marke enthalten. Das System erfüllt also den wechselseitigen Ausschluß.

Eine S-Invariante kann auch negative Einträge besitzen. Auch diese S-Invarianten haben sinnvolle Interpretationen. Dazu diskutieren wir ein weiteres Beispiel.

### Beispiel 2.6 (S-Invariante: Anwendung)

... Übungsblatt 5, Aufgabe 5

Mit  $S$ -Invarianten können wir bestimmte Eigenschaften der erreichbaren Markierungen eines S/T-Systems nachweisen. Wir können auch eine hinreichende Bedingung für die Beschränktheit bzw. für die strukturelle Beschränktheit eines S/T-Systems mit Hilfe von  $S$ -Invarianten formulieren. Dafür benötigen wir positive  $S$ -Invarianten.

**Definition 2.25 (Positive Invarianten)**

Sei  $\Sigma$  ein S/T-System mit Stellenmenge  $S$ .

1. Eine  $S$ -Invariante  $\underline{i}$  von  $\Sigma$  heißt *positiv*, wenn gilt  $\underline{i} > \underline{0}$ .
2.  $\Sigma$  heißt von *positiven  $S$ -Invarianten* überdeckt, wenn es eine positive  $S$ -Invarianten  $\underline{i}$  mit Trägermenge  $S$  gibt (d. h.  $\text{supp}(\underline{i}) = S$ ).

*Manchmal nennt man diese  $S$ -Invariante auch strikt positiv.*

*Die Bezeichnung von „positiven  $S$ -Invarianten“ überdeckt kommt daher, daß die Eigenschaft äquivalent wie folgt formuliert werden kann: Für jede Stelle  $s \in S$  gibt es eine  $S$ -Invariante  $\underline{i}_s \geq \#s$ . Per Definition ist  $\underline{i}_s$  positiv und überdeckt die Stelle  $s$ . Die  $S$ -Invariante  $\underline{i} = \underline{i}_{s_1} + \dots + \underline{i}_{s_n}$  ist dann eine Überdeckung in unserem Sinne.*

**Satz 2.26 (Positive  $S$ -Invarianten und Beschränktheit)**

Sei  $\Sigma$  ein S/T-System mit Stellenmenge  $S$ .

1. Sei  $\underline{i}$  eine positive  $S$ -Invariante. Dann ist jede Stelle  $s \in \text{supp}(\underline{i})$  beschränkt.
2. Wenn  $\Sigma$  mit positiven  $S$ -Invarianten überdeckt ist, dann ist  $\Sigma$  strukturell beschränkt.

**Beweis:** Sei  $\underline{i}$  eine positive  $S$ -Invariante. Sei  $m_0$  eine beliebige Anfangsmarkierung und sei  $n = m_0 \cdot \underline{i}$ . Dann gilt für alle erreichbaren Markierungen  $m$  gemäß Folgerung 2.24

$n = m_0 \cdot \underline{i} = m \cdot \underline{i} = \sum_{s \in S} m(s) \cdot \underline{i}(s)$ . Da  $\underline{i}$  positiv ist, gilt für jeden Summanden  $m(s) \cdot \underline{i}(s) \geq 0$ . Also gilt  $n \geq m(s) \cdot \underline{i}(s)$  für alle  $s \in S$ .

Sei nun  $s \in \text{supp}(\underline{i})$ . Dann gilt  $\underline{i}(s) > 0$  und damit  $\frac{n}{\underline{i}(s)} \geq m(s)$ . Die Stelle  $s$  ist also beschränkt.

Falls  $\Sigma$  mit positiven Invarianten überdeckt ist, ist also jede Stelle beschränkt. Das S/T-System ist also beschränkt. Da wir keine Annahme über die Anfangsmarkierung  $m_0$  getroffen haben, ist  $\Sigma$  strukturell beschränkt.  $\square$

Im Gegensatz zum Überdeckungsbaum liefert Satz nur eine hinreichende Bedingung für die Beschränktheit (und die strukturelle Beschränktheit) eines S/T-Systems. Es gibt S/T-Systeme, die strukturell beschränkt sind, die aber nicht von positiven S-Invarianten überdeckt sind. Hier ist ein einfaches Beispiel (dieses S/T-System besitzt keine nicht-triviale S-Invariante):



In vielen praktischen Fällen reicht jedoch Satz 2.26 für den Nachweis der Beschränktheit eines S/T-Systems aus.

Es gibt sogar eine präzise Charakterisierung der strukturellen Unbeschränktheit durch ein lineares Ungleichungssystem, die wir hier der Vollständigkeit halber erwähnen (aber nicht im einzelnen diskutieren).

### Satz 2.27 (Charakterisierung der strukturellen Beschränktheit)

Ein S/T-System  $\Sigma$  mit Stellenmenge  $S$  und Inzidenzmatrix  $C$  ist genau dann strukturell beschränkt, wenn das lineare Ungleichungssystem  $C^T \cdot \underline{x} \leq \underline{0}$  eine Lösung mit  $\underline{x} \geq \underline{0}$  und  $\text{supp}(\underline{x}) = S$  hat.

**Beweis:** Wir verzichten hier auf einen Beweis.

**Hinweis für Interessierte:** Die eine Richtung des Beweises ist analog zum Beweis von Satz 2.26, denn aus der Ungleichung  $C^T \cdot \underline{x}$  folgt für alle  $m_1 \xrightarrow{*} m_2$  unmittelbar  $m_1 \cdot \underline{x} \geq m_2 \cdot \underline{x}$ .

Die andere Richtung folgt aus Satz 2.13, der Zustandsgleichung (Satz 2.21) und Farkas Lemma (einem sog. Alternativsatz aus dem Gebiet der linearen Programmierung, der den Zusammenhang zwischen der Lösbarkeit und Unlösbarkeit verschiedener Formen von linearen Gleichungssystemen herstellt).

□

## 4.4 Transitions-Invarianten

Im vorangegangenen Abschnitt haben wir Lösungen des linearen Gleichungssystems  $C^T \cdot \underline{i} = \underline{0}$  betrachtet. Nun betrachten wir Lösungen des linearen Gleichungssystems  $C \cdot \underline{j} = \underline{0}$ . Wir nennen sie *Transitions-Invarianten*<sup>7</sup> (*T-Invariante*).

<sup>7</sup> Achtung: Die Bezeichnung Transitions-Invariante ist sehr irreführend, da Transitions-Invarianten im Gegensatz zu S-Invarianten keine „Invarianteninterpretation“ besitzen. Trotzdem bleiben wir bei dieser Bezeichnung, da es im deutschsprachigen Raum keine andere etablierte Bezeichnung gibt.

**Definition 2.28 (Transitions-Invariante)**

Sei  $\Sigma$  ein S/T-System mit Transitionen  $T$  und mit Inzidenzmatrix  $C$ . Ein T-Vektor  $\underline{j}$  heißt *Transitions-Invariante* (*T-Invariante*) von  $\Sigma$ , wenn  $C \cdot \underline{j} = \underline{0}$  gilt.

Eine Transitions-Invariante  $\underline{j}$  heißt *positiv*, wenn gilt  $\underline{j} > \underline{0}$ . Das S/T-System  $\Sigma$  heißt mit positiven T-Invarianten überdeckt, wenn es eine positive T-Invariante  $\underline{j}$  mit Trägermenge  $T$  gibt.

*Eine positive T-Invariante mit Trägermenge  $T$  wird manchmal auch strikt positiv genannt.*

Nun überlegen wir uns, wie sich eine T-Invariante interpretieren läßt. Die Interpretation ergibt sich unmittelbar aus der Zustandsgleichung: Für zwei Markierungen  $m_1$  und  $m_2$  und ein Schaltwort  $w$  mit  $m_1 \xrightarrow{w} m_2$  gilt  $m_2 = m_1 + C \cdot \#w$ . Für  $C \cdot \#w = \underline{0}$  gilt also  $m_1 = m_2$ . Die Einträge einer T-Invariante  $\underline{j}$  geben also an, wie oft jede Transition in einem Schaltwort  $w$  vorkommen muß, damit das Schaltwort  $w$  die Ausgangsmarkierung reproduziert.

... irgendwann kommt hier noch ein Beispiel

Ein S/T-System kann also nur dann zyklisches Verhalten besitzen, wenn es wenigstens eine positive T-Invariante besitzt:

**Folgerung 2.29 (Interpretation von T-Invarianten)**

Sei  $\Sigma$  ein S/T-System, sei  $m$  eine Markierung von  $\Sigma$  und sei  $w \in T^+$  ein nicht-leeres Schaltwort mit  $m \xrightarrow{w} m$ . Dann besitzt  $\Sigma$  eine positive T-Invariante.

**Beweis:** Aus der Zustandsgleichung (Satz 2.21) folgt unmittelbar  $m = m + C \cdot \#w$ . Also gilt  $C \cdot \#w = \underline{0}$ . Der Vektor  $\#w$  ist also eine T-Invariante; außerdem ist  $\#w$  positiv.  $\square$

Daraus ergeben sich unmittelbar notwendige Bedingungen für die Verklemmungsfreiheit und die Lebendigkeit von beschränkten Systemen. Denn in einem verklemmungsfreien beschränkten System, muß sich zwangsläufig irgendwann eine Markierung wiederholen:

**Folgerung 2.30 (T-Invarianten, Verklemmungsfreiheit und Lebendigkeit)**

Sei  $\Sigma$  ein beschränktes S/T-System (mit endlichem Netz).

1. Wenn  $\Sigma$  keine positive T-Invariante besitzt, dann besitzt der Erreichbarkeitsgraph von  $\Sigma$  keinen unendlichen Pfad; insbesondere ist  $\Sigma$  nicht verklemmungsfrei.

2. Wenn  $\Sigma$  nicht mit positiven T-Invarianten überdeckt ist, dann ist  $\Sigma$  nicht lebendig.

**Beweis:**

1. Wir beweisen die Aussage durch Kontraposition. Wir nehmen an, daß der Erreichbarkeitsgraph einen unendlichen Pfad  $m_0 \rightarrow m_1 \rightarrow m_2 \rightarrow \dots$  besitzt und zeigen, daß dann  $\Sigma$  eine positive T-Invariante besitzt. Wegen der Beschränktheit von  $\Sigma$  gibt es  $i, j \in \mathbb{N}$  mit  $i < j$  und  $m_i = m_j$ . Also gibt es ein nicht-leeres Wort  $w$  mit  $m_i \xrightarrow{w} m_j = m_i$ . Mit Folgerung 2.29 besitzt  $\Sigma$  eine positive T-Invariante.
2. Übung: Blatt 7.

□

Folgerung 2.30 liefert eine notwendige Bedingung für die Verklemmungsfreiheit und eine notwendige Bedingung für die (strukturelle) Lebendigkeit eines S/T-Systems. Im folgenden entwickeln wir eine notwendige und eine hinreichende Bedingung für die strukturelle Lebendigkeit von S/T-Systemen, die wenigstens für eine Teilklasse der S/T-Systeme zusammenfallen.

Zunächst motivieren wir anhand eines Beispiels, daß bei der Bedingung für die strukturelle Lebendigkeit die Anzahl der Konflikte bzw. Konfliktbereiche eine Rolle spielen.

... Beispiel:

**Definition 2.31 (Konfliktbereiche und Transitionsvorbereiche)**

Sei  $\Sigma = (N, W, m)$  ein S/T-System mit Netz  $N = (S, T, F)$ .

1. Eine nicht-leere Teilmenge  $T' \subseteq T$  heißt *Konfliktbereich* von  $\Sigma$  (bzw.  $N$ ), wenn für jedes  $t' \in T'$  und jedes  $t \in T$  mit  $\bullet t' \cap \bullet t \neq \emptyset$  auch  $t \in T'$  gilt.

Ein Konfliktbereich  $T'$  heißt *minimal*, wenn es keinen Konfliktbereich  $T''$  mit  $T'' \subset T'$  gibt.

2. Für jede Transition  $t \in T$  nennen wir die Stellenmenge  $\bullet t$  einen *Transitionsvorbereich* von  $\Sigma$ .

*Offensichtlich ist die Anzahl der Transitionsvorbereiche und die Anzahl der minimalen Konfliktbereiche kleiner als die Anzahl der Transitionen des S/T-Systems.*

*Wenn keine Transition des S/T-Systems einen leeren Vorbereich hat, ist die Anzahl der minimalen Konfliktbereiche kleiner oder gleich der Anzahl der Vorbereiche.*

... graphische Illustration der Begriffe

Erstaunlicherweise liefert uns die Anzahl der minimalen Konfliktbereiche bzw. die Anzahl der Transitionsvorbereiche verglichen mit dem Rang der Inzidenzmatrix eine hinreichende bzw. eine notwendige Bedingung für die strukturelle Lebendigkeit von gewöhnlichen S/T-System: die *Rangbedingung* (das Rangtheorem).

*Zur Erinnerung: Der Rang einer Matrix ist die Anzahl der von Null verschiedenen Zeilen der in Zeilenstufenform transformierten Matrix.  $|T| - \text{Rang}(C)$  ist also die Anzahl der linear unabhängigen Lösungen des linearen Gleichungssystems  $C \cdot \underline{x} = \underline{0}$ .  $|T| > \text{Rang}(C)$  ist also die notwendige Bedingung für die Existenz einer nicht-trivialen Lösung. Damit ist  $|T| > \text{Rang}(C)$  eine notwendige Bedingung für strukturelle Lebendigkeit. Das Rangtheorem verschärft diese Bedingung.*

*Die Idee der Rangbedingung ist nun, daß man für jeden Konflikt im S/T-System eine weitere T-Invariante benötigt, damit das S/T-System lebendig ist. Je nachdem ob die Anzahl der Konflikte über die Anzahl der Konfliktbereiche oder die Anzahl der Transitionsvorbereiche formuliert wird, erhalten wir eine notwendige oder eine hinreichende Bedingung für die strukturelle Lebendigkeit.*

### **Satz 2.32 (Rangtheorem)**

Sei  $\Sigma$  ein gewöhnliches S/T-System, das mit positiven S-Invarianten und mit positiven T-Invarianten überdeckt ist und das keine leeren Transitionsvorbereiche besitzt.

1. Wenn  $\Sigma$  strukturell lebendig ist, dann ist der Rang der Inzidenzmatrix echt kleiner als die Anzahl der Transitionsvorbereiche von  $\Sigma$ .

*Diese Aussage liefert uns also eine notwendige Bedingung für die strukturelle Lebendigkeit eines S/T-Systems.*

2. Wenn der Rang der Inzidenzmatrix echt kleiner ist als die Anzahl der minimalen Konfliktbereiche, dann ist  $\Sigma$  lebendig.

*Diese Aussage liefert uns also eine hinreichende Bedingung für die strukturelle Lebendigkeit eines S/T-Systems.*

Ein Beweis dieses Satzes würde hier zu weit führen. Er kann beispielsweise in [2] nachgeschlagen werden. Für uns genügt es, das Rangtheorem anzuwenden zu können (siehe Übungsblatt 7).

Leider besteht zwischen der notwendigen und der hinreichenden Bedingung des Rangtheorems für S/T-Systeme im allgemeinen eine Lücke. Denn die Anzahl der minimalen Konfliktbereiche ist im allgemeinen echt kleiner als die Anzahl der Transitionsvorbereiche. Wenn die Anzahl der minimalen Konfliktbereiche mit der Anzahl der Transitionsvorbereiche übereinstimmt, dann liefert das Rangtheorem eine notwendige und hinreichende Bedingung für die strukturelle Lebendigkeit eines S/T-Systems. Wir charakterisieren jetzt eine Klasse von S/T-Systemen, für die diese Anzahl übereinstimmt: *free-choice Netze*<sup>8</sup>. Der Einfachheit halber betrachten wir aber nur gewöhnliche S/T-Systeme.

**Definition 2.33 (Free-choice-System)**

Ein gewöhnliches S/T-System  $\Sigma$  mit Transitionen  $T$  heißt *Free-choice-System*, wenn für alle Transitionen  $t_1, t_2 \in T$  mit  $\bullet t_1 \cap \bullet t_2 \neq \emptyset$  gilt  $\bullet t_1 = \bullet t_2$ .

*Man sagt auch, daß  $\Sigma$  die Free-choice-Eigenschaft besitzt.*

... graphische Illustration der Def.

**Lemma 2.34**

Sei  $\Sigma$  ein Free-choice-System ohne Transitionen mit leerem Vorbereich. Dann ist die Anzahl der minimalen Konfliktbereiche gleich der Anzahl der Transitionsvorbereiche.

**Satz 2.35 (Rangtheorem für Free-choice-Systeme)**

Sei  $\Sigma$  ein Free-choice-System ohne Transitionen mit leerem Vorbereich, das von positiven S-Invarianten und von positiven T-Invarianten überdeckt ist. Dann ist  $\Sigma$  genau dann strukturell lebendig, wenn der Rang der Inzidenzmatrix von  $C$  kleiner als die Anzahl der Transitionsvorbereiche ist.

<sup>8</sup> Eine deutsche Bezeichnung für Free-choice-Systeme hat sich bisher nicht durchgesetzt

## 5 Deadlocks und Traps

Im vorangegangenen Abschnitt haben wir gesehen, wie man mit S-Invarianten bestimmte Eigenschaften eines S/T-Systems beweisen kann. Allerdings hat der Einsatz von S-Invarianten seine Grenzen (vgl. Übung 6, Aufgabe 3a und 3c). Insbesondere sind S-Invarianten „blind gegenüber Schlingen“, so daß alle Eigenschaften, die aufgrund einer Schlinge gelten mit S-Invarianten nicht bewiesen werden können. Ebenso sind S-Invarianten „blind gegenüber dem Rückwärtsschalten von Transitionen“.

In diesem Abschnitt werden wir zwei Konzepte vorstellen, die diese blinden Flecke wenigstens teilweise beseitigen: Deadlocks und Traps.

... Beispiel

Manchmal bleibt die gewichtete Markensumme auf einer bestimmten Stellenmenge nicht konstant, aber es ist durch die Struktur des Systems sichergestellt, daß nie alle Marken von diesen Stellenmengen verschwinden. Eine solche Stellenmenge nennen wir *Trap* oder *Falle*.

Umgekehrt gibt es Stellenmengen, die immer unmarkiert bleiben, wenn sie einmal unmarkiert sind. Eine solche Stellenmenge nennen wir *Deadlock*<sup>9</sup>.

### Definition 2.36 (Deadlock und Trap)

Sei  $N = (S, T, F)$  ein Netz.

1. Eine Teilmenge  $A \subseteq S$  von Stellen heißt *Trap* (oder *Falle*) von  $N$ , wenn gilt  $A^\bullet \subseteq \bullet A$ .

*D. h. jede Transition  $t$ , die etwas aus  $A$  rausnimmt ( $t \in A^\bullet$ ), tut auch etwas rein ( $t \in \bullet A$ ).*

*Man könnte die Forderung auch wie folgt formulieren: für alle  $t \in T$  gilt:  $\bullet t \cap A \neq \emptyset \Rightarrow t^\bullet \cap A \neq \emptyset$ .*

2. Eine Teilmenge  $A \subseteq S$  von Stellen heißt *Deadlock* (oder *co-Falle*, *Siphon*) von  $N$ , wenn gilt  $\bullet A \subseteq A^\bullet$ .

*D. h. jede Transition  $t$ , die etwas in  $A$  rein tut ( $t \in \bullet A$ ), nimmt auch etwas raus ( $t \in A^\bullet$ ).*

*Man könnte die Forderung auch wie folgt formulieren: für alle  $t \in T$  gilt:  $t^\bullet \cap A \neq \emptyset \Rightarrow \bullet t \cap A \neq \emptyset$ .*

---

<sup>9</sup> Für Deadlocks gibt es keine etablierte deutsche Bezeichnung. Manchmal werden sie Siphons genannt, manchmal co-Fallen.

Offensichtlich gilt: Ein unter einer Markierung markierter Trap bleibt auch in allen Nachfolgemarkierungen markiert. Ein unter einer Markierung unmarkierter Deadlock bleibt auch in allen Nachfolgemarkierungen unmarkiert.

*Eine Stellenmenge  $A$  heißt markiert unter einer Markierung  $m$ , wenn gilt  $\text{supp}(m) \cap A \neq \emptyset$ .*

*Eine Stellenmenge  $A$  heißt unmarkiert unter einer Markierung  $m$ , wenn gilt  $\text{supp}(m) \cap A = \emptyset$ .*

### Folgerung 2.37

Sei  $\Sigma = (N, W, m_0)$  ein S/T-System.

1. Sei  $A$  eine Falle von  $N$ . Wenn  $A$  unter einer Markierung  $m$  markiert ist, dann ist  $A$  unter allen von  $m$  aus erreichbaren Markierungen markiert.

*Für eine initial markierte Falle  $A = \{s_1, s_2, \dots, s_n\}$  schreiben wir dann  $s_1 + s_2 + \dots + s_n \geq 1$ .*

2. Sei  $A$  ein Deadlock von  $N$ . Wenn  $A$  unter einer Markierung  $m$  unmarkiert ist, dann ist  $A$  unter allen von  $m$  aus erreichbaren Markierungen unmarkiert.

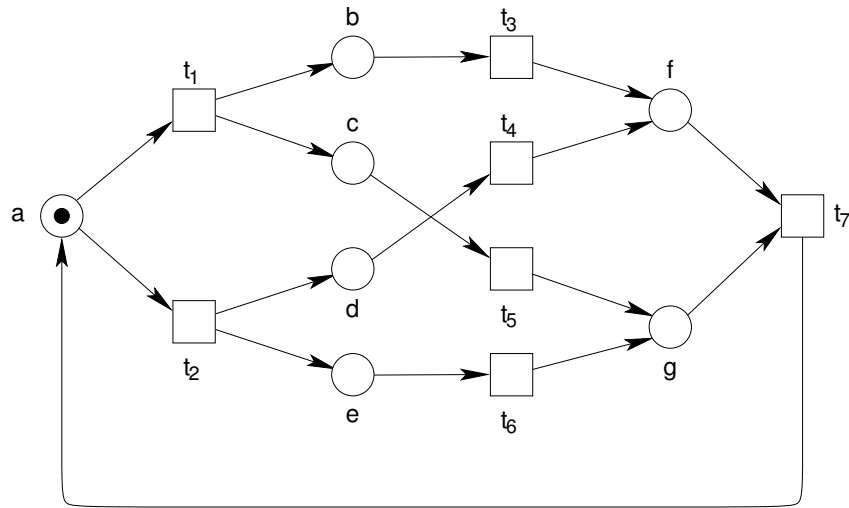
*Für einen initial unmarkierten Deadlock  $A = \{s_1, s_2, \dots, s_n\}$  schreiben wir dann  $s_1 + s_2 + \dots + s_n = 0$ .*

*Allerdings wird es wenige wirklich sinnvolle S/T-Systeme geben, die einen initial unmarkierten Deadlock besitzen.*

Wir können nun Fallen mit S-Invarianten kombinieren, um Eigenschaften von S/T-Systemen zu beweisen (die allein mit S-Invarianten nicht beweisbar waren). Zunächst betrachten wir dazu ein technisches Beispiel (vgl. Übung 6, Aufgabe 3c).

### Beispiel 2.7 (Verifikation mit S-Invarianten und Fallen)

Wir beweisen für das folgende S/T-System, daß keine Markierung  $m$  erreichbar ist, in der sowohl die Stelle  $c$  als auch die Stelle  $d$  markiert sind.



Wir benutzen dazu die folgenden S-Invarianten und Fallen (von deren Gültigkeit man sich durch Nachrechnen leicht überzeugen kann):

$$\begin{aligned} \underline{i_1} &: a + c + e + g = 1 \\ \underline{i_2} &: a + b + d + f = 1 \\ \underline{f_1} &: a + b + e + f + g \geq 1 \end{aligned}$$

Nun beweisen wir, daß in keiner erreichbaren Markierung sowohl auf  $c$  als auch auf  $d$  mind. eine Marke liegt. Wir beweisen dies durch Widerspruch. Wir nehmen also an, daß es eine erreichbare Markierung  $m$  gibt, in der sowohl  $c$  als auch  $d$  markiert sind und führen diese Aussage mit Hilfe der Invarianten und der Falle zum Widerspruch:

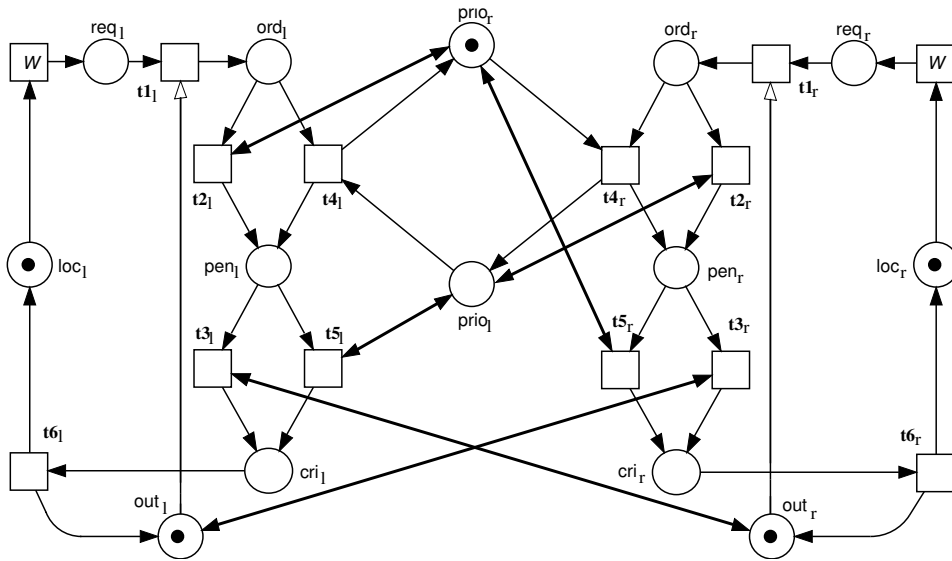
- Wegen  $\underline{i_1}$  sind dann die Stellen  $a, e$  und  $g$  unmarkiert.
- Wegen  $\underline{i_2}$  sind auch die Stellen  $a, b$  und  $f$  unmarkiert.
- Insgesamt sind also die Stellen  $a, b, e, f, g$  unmarkiert.

Dies ist jedoch ein Widerspruch zur Falle  $\underline{f_1}$ , denn mind. eine dieser Stellen ist gemäß der Falle  $\underline{f_1}$  markiert.

Nun wenden wir uns einem realistischen Beispiel zu:

### Beispiel 2.8 (Peterson's Mutex-Algorithmus)

Wir beweisen, daß im folgende S/T-System keine Markierung erreichbar ist, in der sowohl  $cri_l$  als auch  $cri_r$  markiert sind.



Zunächst geben wir wieder die S-Invarianten und Fallen an, die wir im Beweis benutzen werden:

$$\begin{aligned}
 \underline{i_1} &: cri_1 + ord_1 + pen_1 + out_1 = 1 \\
 \underline{i_2} &: cri_r + ord_r + pen_r + out_r = 1 \\
 \underline{i_3} &: prio_1 + prio_r = 1 \\
 f_1 &: pen_1 + prio_1 + ord_r + out_r \geq 1 \\
 f_2 &: pen_r + prio_r + ord_1 + out_1 \geq 1
 \end{aligned}$$

Wir beweisen die Aussage wieder durch Widerspruch. Wir nehmen also an, daß eine Markierung erreichbar ist, in der sowohl  $cri_1$  als auch  $cri_r$  markiert sind.

- Wegen  $\underline{i_1}$  sind in dieser Markierung  $ord_1$ ,  $pen_1$  und  $out_1$  unmarkiert.
- Wegen  $\underline{i_2}$  sind in dieser Markierung  $ord_r$ ,  $pen_r$  und  $out_r$  unmarkiert.
- Insgesamt sind dann also  $ord_1$ ,  $pen_1$ ,  $out_1$ ,  $ord_r$ ,  $pen_r$  und  $out_r$  unmarkiert.
- Wegen  $f_1$  ist dann  $prio_1$  markiert.
- Wegen  $f_2$  ist dann auch  $prio_r$  markiert.
- Insgesamt sind dann also  $prio_1$  und  $prio_r$  markiert.

Dies ist jedoch ein Widerspruch zu  $\underline{i_3}$ .

S-Invarianten in Kombination mit Fallen liefern in vielen Fällen einen überzeugenden Beweis dafür, daß bestimmte Markierungen nicht erreichbar sind. Die Frage ist jedoch, wie man diese Beweise findet. Für kleine Beispiele entwickelt man relativ schnell ein Gefühl für die Wahl der richtigen S-Invarianten, Fallen und die Kombination zu einem Beweis. Tatsächlich kann man dieses Verfahren sogar vollständig automatisieren, da sich Fallen als lineare Ungleichungen formulieren lassen. Die obigen Beweisaufgaben werden dazu als linearen Ungleichungssysteme formuliert (das vom System und der zu beweisenden Eigenschaft abhängt). Diese können dann vollautomatisch gelöst werden. Darauf gehen wir hier aber nicht näher ein.

Neben dem Nachweis von Invarianten liefern Deadlocks und Traps auch notwendige (und teilweise sogar hinreichende) Bedingungen für die Verklemmungsfreiheit und die Lebendigkeit von S/T-Systemen.

**Lemma 2.38 (Deadlocks und Verklemmungen)**

Sei  $\Sigma = (N, W, m_0)$  ein gewöhnliches nicht verklemmungsfreies S/T-System mit mindestens einer Transition. Dann gibt es einen nicht-leeren Deadlock von  $N$ , der unter einer erreichbaren Markierung von  $\Sigma$  unmarkiert ist.

**Beweis:** Da  $\Sigma$  nicht verklemmungsfrei ist, gibt es eine erreichbare Markierung  $m$  von  $\Sigma$ , in der keine Transition von  $\Sigma$  aktiviert ist. Wir definieren  $A = \{s \in S \mid m(s) = 0\}$ . Per Definition ist  $A$  unmarkiert unter einer erreichbaren Markierung.

Es bleibt zu zeigen, daß  $A$  ein nicht-leerer Deadlock von  $N$  ist. Dazu genügt es  $A^\bullet = T$  zu zeigen, denn dann gilt trivialerweise  ${}^\bullet A \subseteq T = A^\bullet$  und  $A \neq \emptyset$  da gemäß Annahme  $T \neq \emptyset$  gilt.

Sei  $t \in T$  eine beliebige Transition von  $\Sigma$ . Da  $\Sigma$  gewöhnlich ist und unter  $m$  nicht aktiviert ist, muß es eine Stelle  $s$  im Vorbereich von  $t$  geben, die in  $m$  nicht markiert ist. Also gilt  $s \in A$  und damit  $t \in A^\bullet$ .  $\square$

Wenn wir nun gewährleisten, daß alle nicht-leeren Deadlocks des S/T-Systems immer markiert bleiben, dann verklemmt das S/T-System niemals. Nun ist die Frage, wie wir das gewährleisten können.

Einfache Idee: Wenn ein Deadlock einen initial markierten Trap enthält, dann bleibt der Deadlock immer markiert.

**Satz 2.39 (Deadlock-Trap-Eigenschaft)**

Sei  $\Sigma$  ein gewöhnliches S/T-System mit mindestens einer Transition. Wenn jeder nicht-leere Deadlock einen initial markierten Trap enthält, dann ist  $\Sigma$

verklemmungsfrei.

**Beweis:** Die Aussage folgt unmittelbar aus Lemma 2.38 und Folgerung 2.37.  
□

**Bemerkung:** *Für Free-choice-Systeme gilt eine viel schärfere Aussage (Commoner's Theorem): Ein Free-choice-System ist genau dann lebendig, wenn jeder nicht-leere Deadlock einen initial markierten Trap enthält.*

## 6 Varianten von Petrinetzen

In diesem Kapitel haben wir die klassische Petrinetztheorie vorgestellt: S-Invarianten, T-Invarianten, Deadlocks und Traps. Diese oder ähnliche Techniken gibt es für verschiedene Varianten von Petrinetzen (mehr oder weniger gut). Für S/T-Systeme lassen sie sich besonders schön formulieren. Deshalb haben wir mit S/T-Systemen begonnen, obwohl sie historisch nicht die erste Variante von Petrinetzen darstellen.

In diesem Abschnitt werden wir einen kurzen Überblick über die verschiedenen Varianten von Petrinetzen geben, die in Theorie und Praxis vorkommen. Dabei belassen wir es jedoch bei einer kurzen Darstellung der jeweiligen Konzepte. Wenn wir alle Varianten im Detail darstellen würden, wären wir damit mehrere Semester beschäftigt (ohne dabei viel zu lernen). Insbesondere können die meisten der folgenden Konzepte beliebig miteinander kombiniert werden – zumindest im Prinzip.

### 6.1 Kapazitäten

In vielen Fällen sind bestimmte Stellen eines S/T-Systems bereits aufgrund der Aufgabenstellung bzw. der vorgegebenen Randbedingung des zu modellierenden Systems beschränkt. Beispielsweise passen auf die Ladefläche eines LKWs nur eine feste Anzahl von Paletten – oder ein bestimmter Puffer eines Systems hat eine feste Obergrenze. Um diese Obergrenze im Petrinetzmodell zu formulieren sehen manche Petrinetzvarianten vor, für jede Stelle eine Obergrenze für die Anzahl der Marken anzugeben. Wir nennen diese Varianten Petrinetze mit *Kapazitäten*.

Technisch werden Kapazitäten für die Stellen durch eine Abbildung  $K : S \rightarrow$

$\mathbb{N} \cup \{\omega\}$  repräsentiert<sup>10</sup>. Für  $K(s) \in \mathbb{N}$  gibt  $K(s)$  die maximale Anzahl der erlaubten Marken auf Stellen  $s$  an. Für  $K(s) = \omega$  ist die Anzahl der Marken auf Stelle  $s$  unbeschränkt.

Ein S/T-System mit Kapazitäten können wir dann durch  $\Sigma = (N, W, K, m_0)$  notieren. Die Schaltregel müssen wir dann so modifizieren, daß nach dem Schalten für keine Stelle ihre Kapazität überschritten ist (die Formulierung dieser Schaltregel ist eine einfache Übungsaufgabe). Außerdem muß natürlich die Anfangsmarkierung die Kapazitäten respektieren. Dazu wird  $m_0 \leq K$  gefordert.

Die Einführung von Kapazitäten ist jedoch nur eine Schreibabkürzung. Denn eine Stelle mit Kapazitätsangabe können wir immer durch das Hinzufügen einer *Komplementstelle* erreichen, die die entsprechende Anzahl von Token enthält, wie dies in Abbildung 2.3 dargestellt ist: die Komplementstelle  $\bar{s}$  ist mit denselben Transitionen verbunden wie die Stelle  $s$  – allerdings laufen die Kanten genau in die entgegengesetzte Richtung. Durch die Anfangsmarkierung  $m_0(s) + m_0(\bar{s}) = 7$  ist gewährleistet, daß auf  $s$  niemals mehr als 7 Marken gelangen (wie die Kapazität im linken Modell vorschreibt).

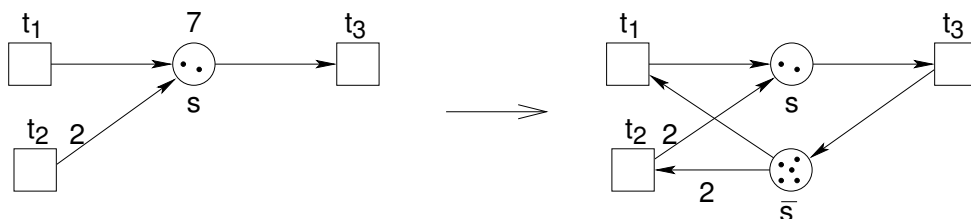


Abbildung 2.3: Modellierung der Kapazität durch eine Komplementstelle

Die Erweiterung von S/T-Systemen um Kapazitäten ist also nur „syntaktischer Zucker“ der die Modellierungsmächtigkeit nicht erhöht. In der Praxis führt der Einsatz von Kapazitäten jedoch oft zu übersichtlicheren Modellen (denn er halbiert die Anzahl der Stellen).

<sup>10</sup> Die Funktion  $K$  ist technisch eine verallgemeinerte Markierung, auch wenn sie konzeptionell nur sehr bedingt etwas mit den verallgemeinerten Markierung des Überdeckungsbaumes zu tun hat.

## 6.2 Elementare Netzsysteme und Bedingungs-/Ereignis-Systeme

Es gibt einige Netzvarianten, die a priori für alle Stellen die Kapazität 1 fordern. Dazu gehören sog. *Elementare Netzsysteme* (EN-Systeme) und *Bedingungs-/Ereignis-Systeme* (B/E-Systeme). Für diese Netzklassen kann man jede Markierung als eine Teilmenge von Stellen auffassen (die Menge der gerade markierten Stellen). Durch diese Einschränkung können wir also die Stellen als Prädikate auffassen, die ihren Gültigkeit dynamisch verändern. Das Prädikat ist (in einer Markierung) gültig, wenn die Stelle markiert ist; es ist ungültig, wenn die Stelle nicht markiert ist.

Neben der genannten Einschränkung besitzen EN-Systeme und B/E-Systeme noch einige weitere Einschränkung, die hauptsächlich „philosophisch“ motiviert sind. Wir beschränken uns hier auf die technischen Einschränkungen: In EN-Systemen können Transitionen einer Schlinge grundsätzlich nicht schalten – Transitionen mit Schlinge sind also überflüssig (dürfen aber vorkommen). In B/E-Systemen sind Schlingen sogar grundsätzlich verboten.

Für die Analyse können wir sowohl EN-Systeme als auch B/E-Systeme als spezielle S/T-Systeme mit Kapazität 1 für alle Stellen auffassen (wobei wir alle Transitionen mit Schlingen aus EN-Systemen löschen).

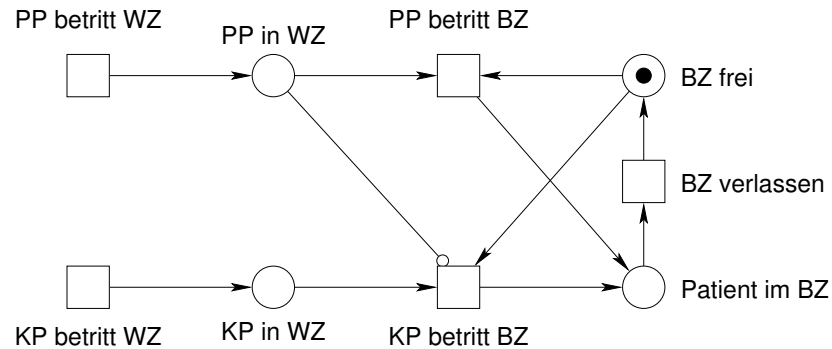
## 6.3 Inhibitorkanten

Beim Modellieren mit Petrinetzen würde man sich manches mal eine Transition wünschen, die nur schaltet, wenn auf einer bestimmten Stelle gerade keine Marke liegt. Für beschränkte Stellen können wir dies mit einer Schlinge zu einer Komplementstelle modellieren, die überprüft, ob alle Marken gerade auf der Komplementstelle vorhanden sind. Für unbeschränkte Stellen gibt es jedoch keine derartige Möglichkeit (das kann man sogar formal beweisen).

Deshalb wurden für Petrinetze sogenannte *Inhibitorkanten* vorgeschlagen. Eine Transition mit einer Inhibitorkante zu einer Stelle kann in einer Markierung nur dann schalten, wenn diese Stelle in der Markierung keine Marke enthält (bezüglich der anderen Kanten gilt weiterhin die übliche Aktivierungsbedingung). Graphisch wird eine Inhibitorkante durch eine Linie mit einem Kreis am Ende dargestellt.

Mit einer Inhibitorkante können wir das Beispiel der Arztpraxis, die Privatpatienten bevorzugt behandelt, wie in Abb. 2.4 dargestellt modellieren. Die Inhibitorkante von der Stelle  $PPinWZ$  zu der Transition  $KPbetrittBZ$  wird

verhindert, daß ein Kassenpatient die Praxis betritt, wenn noch ein Privatpatient wartet.



**Legende:**

PP: Privatpatient  
 KP: Kassenpatient  
 WZ: Wartezimmer  
 BZ: Behandlungszimmer

Abbildung 2.4: Arztpraxis mit Inhibitor-kante

Den Einsatz von Inhibitor-kanten, sollte man sich jedoch sehr gut überlegen.

1. Nicht immer ist in einem realen System operationell überprüfbar, ob die Voraussetzungen für das Schalten einer Transition mit Inhibitor-kante vorliegen. Ein Beispiel dafür ist der Posteingang eines großen Unternehmens: Man kann nicht so einfach überprüfen, ob noch kein Brief eines bestimmten Kunden eingegangen ist.
2. In vielen Fällen erhält man ein besseres Modell, wenn man Inhibitor-kanten unter Ausnutzung von anderen Informationen vermeidet.
3. Für Petrinetze mit Inhibitor-kanten sind viele Fragen nicht mehr entscheidbar, die für S/T-Systeme ohne Inhibitor-kanten entscheidbar sind. Beispielsweise ist nicht entscheidbar, ob ein Petrinetz mit Inhibitor-kanten beschränkt ist.

*Dies werden wir in der Übung beweisen. Die wesentliche Idee dabei ist, daß man mit Petrinetzen mit Inhibitorkanten Registermaschinen simulieren kann (und für Registermaschinen ist nicht entscheidbar, ob sie für eine bestimmte Eingabe terminieren).*

Auf der anderen Seite sind manche Systeme deutlich einfacher durch ein Petrinetz mit Inhibitorkanten zu modellieren als durch eines ohne Inhibitorkanten. Ob man Inhibitorkanten zur Modellierung einsetzt oder nicht erfordert also einen gründlichen Abwägungsprozeß.

## 6.4 Zeitbehaftete Petrinetze

In unseren bisherigen Petrinetzmodellen wird das genaue zeitliche Verhalten eines Systems nicht modelliert. Damit sind Aussagen über das genaue zeitliche Verhalten nicht möglich. Gerade bei Echtzeitsystemen sind zeitliche Aussagen besonders wichtig. Deshalb ist inzwischen eine Vielzahl von Petrinetzvarianten entstanden, die zeitliche Aussagen ermöglichen.

Ganz grob lassen sich zwei verschiedene Ansätze unterscheiden, wie zeitliche Aspekte in Petrinetzen modelliert werden können:

**Zeit-Petrinetze** <sup>11</sup> Die Zeit wird explizit in Form von Schaltdauern, Zeitintervallen, Verweilzeiten angegeben. Die zeitlichen Angaben können sich dabei auf Stellen, Transitionen, Kanten und sogar auf einzelne Marken beziehen.

**Stochastische Petrinetze** Die Zeit wird durch eine (meist kontinuierliche) Wahrscheinlichkeitsverteilung für die Schaltdauern jeder Transition angegeben.

Zeit-Petrinetze erlauben eine sehr detailliertere Beschreibung des zeitlichen Verhaltens eines Systems. Allerdings ist die Analyse des zeitlichen Verhaltens sehr aufwendig (je nach Variante sogar unmöglich). Außerdem hat sich gezeigt, daß sich das Verhalten von Zeit-Petrinetzen durch minimale Änderung der zeitlichen Angaben radikal ändert. Das größte Problem bei Zeit-Petrinetzen ist deshalb die Bestimmung realistischer Zeit-Angaben.

Stochastische Petrinetze sind (zumindest bei den üblichen Wahrscheinlichkeitsverteilungen) robust gegen kleine Änderungen der zeitlichen Angaben.

---

<sup>11</sup> Wir benutzen hier den Begriff Zeit-Petrinetz als einen Oberbegriff. Es gibt jedoch auch eine spezielle Variante von Petrinetzen, für die dieser Begriff benutzt wird. Da wir in diese Details hier nicht einsteigen, sind bei uns Mißverständnisse ausgeschlossen.

Sie erlauben dafür keine so detaillierten Aussagen über das zeitliche Verhalten. Die Analyse ist jedoch für bestimmte Wahrscheinlichkeitsverteilungen (insbes. für die gedächtnislose negative exponentielle Verteilung) effizient möglich (Markovkettenanalyse). Damit sind Aussagen über die durchschnittliche Verweilzeit, die durchschnittliche Durchlaufzeit oder den durchschnittlichen Füllungsgrad möglich.

## 6.5 Nebenläufigkeit und Parallelität

Bisher haben wir als Semantik eines S/T-Systems die Schaltfolgen und den Erreichbarkeitsgraphen kennengelernt. In der sog. *Concurrency-Theory* werden noch viele weitere Semantiken untersucht. Dazu müssen wir S/T-Systeme nicht einmal erweitern; wir definieren nur einige weitere Semantiken. Hier geben wir nur einen Überblick über einige Varianten (der ein Gefühl für die Möglichkeiten und die Motivation für solche „exotischen“ Semantiken geben soll).

*Vielleicht werden wir später der Concurrency-Theory noch ein eigenes Kapitel widmen.*

**Schritte** Bisher sind wir davon ausgegangen, daß Transitionen einzeln schalten. Wenn die Transitionen unabhängig voneinander sind, können jedoch auch mehrere aktivierte Transitionen gleichzeitig schalten. Wir nennen das gleichzeitige Schalten von mehreren Transitionen einen *Schaltschritt*.

Einen Schritt können wir als Transitions-Vektor  $\sigma : T \rightarrow \mathbb{N}$  auffassen, wobei wir im folgenden den leeren Vektor nicht als Schritt auffassen (d.h.  $\sigma > \underline{0}$ ). Analog zu der Definition von  $\bar{t}$  und  $\underline{t}$  für eine Transition  $t$  können wir auch für einen Schritt  $\sigma$  die Markierung  $\bar{\sigma}$  und  $\underline{\sigma}$  definieren:

- $\bar{\sigma} = \sum_{t \in T} \sigma(t) \cdot \bar{t}$ .
- $\underline{\sigma} = C \cdot \sigma$

Damit können wir nun analog zu einzelnen Transitionen die Aktiviertheit eines Schaltschrittes definieren: Ein Schritt  $\sigma$  ist in Markierung  $m$  aktiviert, wenn gilt  $m \geq \bar{\sigma}$  (wir schreiben  $m \xrightarrow{\sigma}$ ). Die Nachfolgemarkierung ergibt sich durch  $m' = m + \underline{\sigma}$  (wir schreiben  $m \xrightarrow{\sigma} m'$ ). Aus den einzelnen Schritten lassen sich dann Schrittschaltfolgen zusammensetzen:  $m_0 \xrightarrow{\sigma_1} m_1 \xrightarrow{\sigma_2} m_2 \xrightarrow{\sigma_3} \dots$

Einen Schritt können wir als Multimengen von Transitionen notieren  $\sigma = [t_1, t_2, \dots, t_n]$ .

Offensichtlich folgt aus  $m \xrightarrow{[t_1, t_2, \dots, t_n]} m'$  unmittelbar  $m \xrightarrow{t_1 t_2 \dots t_n} m'$ , d. h. wenn der Schritt  $[t_1, t_2, \dots, t_n]$  in  $m$  aktiviert ist, kann auch das Schaltwort  $t_1 t_2 \dots t_n$  schalten und führt zur selben Markierung (die Umkehrung der Aussage gilt im allgemeinen nicht).

Insbesondere folgt aus  $m \xrightarrow{[t]} m'$  auch  $m \xrightarrow{t} m'$  (hier gilt sogar die Umkehrung); denn der Schritt  $[t]$  der aus genau einer Transition besteht, entspricht genau dem Schalten der Transition  $t$ .

Deshalb sind unter der Schrittschaltregel dieselben Markierungen erreichbar wie unter der Einzelschrittregel<sup>12</sup>.

Manchmal wird die Schrittschaltregel sogar noch verschärft: Es dürfen nur sog. maximale Schritte schalten. Unter dieser *maximalen Schrittschaltregel* ist ein Schritt  $\sigma$  in Markierung  $m$  nur dann aktiviert, wenn  $m \geq \neg\sigma$  gilt, und wenn es keinen Schritt  $\sigma'$  mit  $\sigma' > \sigma$  und  $m > \neg\sigma'$  gibt.

Die maximale Schrittschaltregel wirft ein paar Fragen auf

1. Ist die Menge der Markierungen eines S/T-Systems, die ausschließlich unter der maximalen Schrittschaltregel erreichbar sind, gleich der Menge der unter der Einzelschrittschaltregel erreichbaren Markierungen?
2. Ist für ein S/T-System entscheidbar, ob die Menge der unter der maximalen Schrittschaltregel erreichbaren Markierungen endlich ist.

Die Antwort auf diese Fragen werden wir uns in der Übung überlegen (Blatt 8 und Blatt 9).

**Prozesse** Mit Hilfe der Schrittschaltregel können wir zum Ausdruck bringen, daß bestimmte Transitionen gleichzeitig schalten. Häufig schalten aber bestimmter Transitionen unabhängig voneinander und nicht gleichzeitig; wir reden dann vom *nebenläufigen* Schalten der Transitionen – Gleichzeitigkeit ist purer Zufall<sup>13</sup>. Hier werden wir nun das Verhalten von S/T-Systemen auf eine Weise formulieren, die dem nebenläufigen Schalten von Transitionen besser Rechnung trägt. Diese Semantik wird *Halbordnungsemantik* genannt, weil

<sup>12</sup> Wenn wir die bisherige Schaltregel von den hier diskutierten Varianten unterscheiden wollen, nennen wir sie *Einzelschrittregel* oder *Einzelschrittschaltregel*.

<sup>13</sup> Die klassische Petrinetphilosophie stellt sich sogar auf den Standpunkt, daß der Begriff der Gleichzeitigkeit nicht sinnvoll gebildet werden kann. Sie lehnt daher die Schrittschaltregel vollkommen ab.

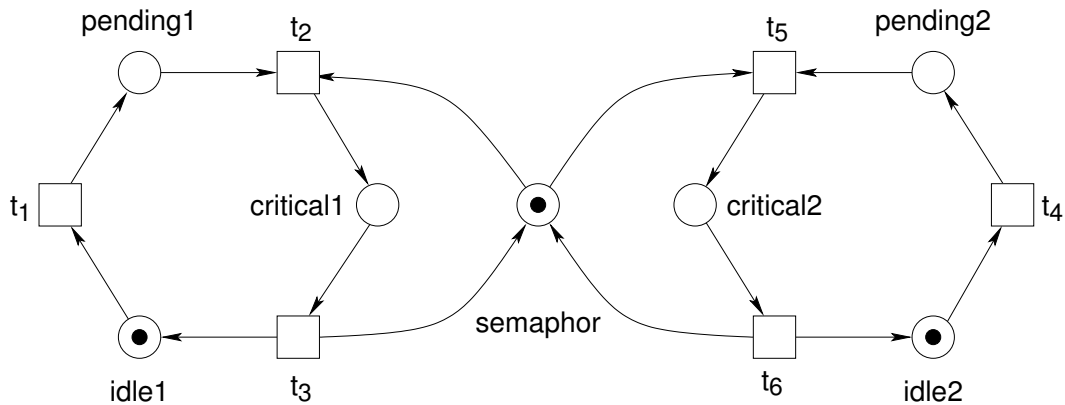


Abbildung 2.5: Der wechselseitige Ausschluß

Abläufe keine Sequenzen, also keine totale Ordnung, mehr sind. Die Abläufe selbst werden *Prozesse* des S/T-Systems genannt.

Wir werden die Prozesse eines S/T-Systems hier nicht formal definieren, sondern nur anhand eines Beispiels vorstellen. Wir betrachten dazu das S/T-System für den wechselseitigen Ausschluß. Eine ausführlichere Behandlung dieses Themas fällt unter das Thema Concurrency-Theory.

In Abb. 2.5 ist nochmals das S/T-System für den wechselseitigen Ausschluß dargestellt. Abbildung 2.6 zeigt einen Prozess dieses S/T-Systems. Der Prozess ist wieder ein Petrinetz, wobei die Stellen und Transitionen des Prozesses mit den Namen der Stellen des ursprünglichen S/T-Systems beschriftet sind, um den Zusammenhang herzustellen – jedes Auftreten einer Transition im Prozess entspricht dem Schalten der entsprechenden Transition des S/T-Systems – wir nennen dieses Auftreten auch ein *Ereignis*. Jede Stelle des Prozesses entspricht einer Marke auf der entsprechenden Stelle des S/T-Systems – wir nennen die Stellen auch eine *Bedingung*. Jedes Ereignis ist mit den Bedingungen verbunden, die es beim Schalten verbraucht bzw. konsumiert. Die Bedingungen auf der linken Seite des Prozesses entsprechen der Anfangsmarkierung des S/T-Systems. Ausgehend von dieser Anfangsmarkierung entwickelt sich der Prozess, wobei jede Bedingung (bis auf die Bedingungen am Anfang) von genau einem Ereignis produziert wird und von genau einem Ereignis konsumiert wird (bis auf die Bedingungen am Ende). Da dieselbe Transition in einem Prozess mehrfach eintreten kann (im Beispiel  $t_1$ ), können im selben Prozess mehrere Ereignisse mit derselben Transition

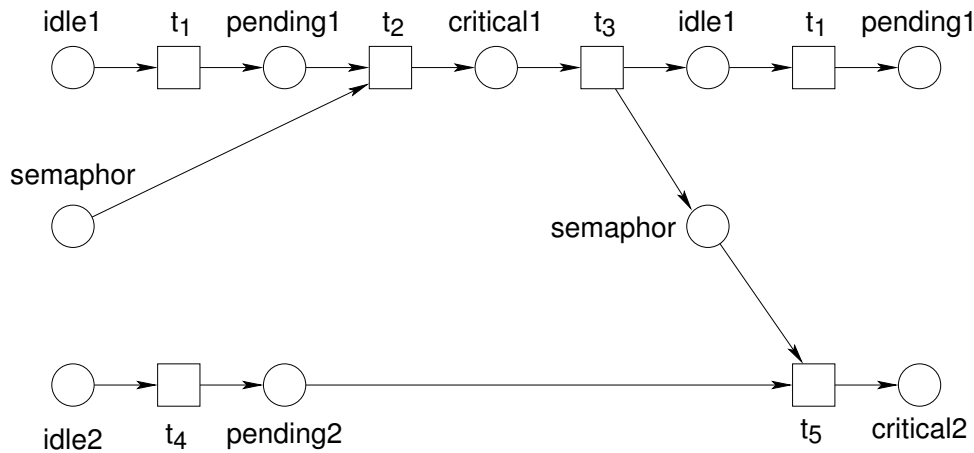


Abbildung 2.6: Ein Prozeß

beschriftet sein. Ebenso können mehrere Bedingungen mit derselben Stelle des S/T-Systems beschriftet sein.

Ein Prozess ist gewissermaßen eine Abwicklung des S/T-Systems ausgehen von der Anfangsmarkierung, wobei wir uns im Falle eines Konfliktes für eine Alternative entscheiden müssen. Deshalb hat ein S/T-System im allgemeinen viele verschiedene Prozesse. Abbildung 2.7 zeigt einen weiteren Ablauf, bei dem der Konflikt um den Semaphor zuerst zugunsten des zweiten Agenten entschieden wurde. Ein Ablauf kann sogar unendlich sein, wenn man die Abwicklung bis ins Unendliche fortsetzt.

Ein Hauptgrund für die Einführung der Prozesse war die „realistischere“ Repräsentation von Nebenläufigkeit. Dies ist in den beiden obigen Prozessen besonders schön am Schalten der Transitionen  $t_1$  und  $t_4$  sichtbar. Den beiden Prozessen sieht man nicht an, welche der beiden Transitionen zuerst eintritt – ihr Schalten ist vollkommen unabhängig voneinander. Die Unabhängigkeit sieht man daran, daß es keine Folge von Kanten zwischen diesen Ereignissen gibt. Die durch die Kanten induzierte Ordnung auf den Bedingungen und Ereignissen ist (wie gesagt) eine Halbordnung. Was durch sie nicht geordnet ist ist nebenläufig zueinander.

Da der Prozess über die Reihenfolge des Eintretens von  $t_1$  und  $t_4$  nichts sagt, ist im zweiten Prozess nicht einmal klar, ob ein Konflikt um den Semaphor vorlag als Transition  $t_5$  geschaltet hat. Nur wenn  $t_1$  vor  $t_4$  geschaltet hätte,

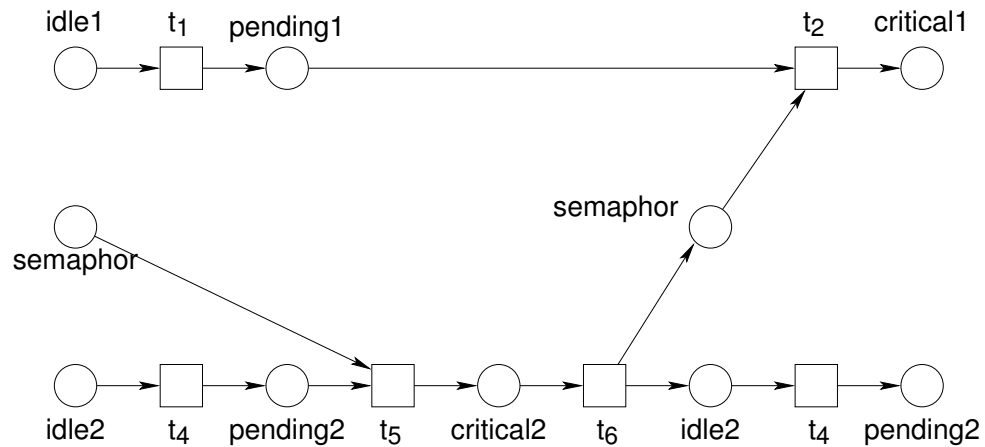


Abbildung 2.7: Ein weiterer Prozeß

hätte ein Konflikt (zwischen  $t_5$  und  $t_2$ ) vorgelegen, sonst nicht. Da der Prozeß aber nichts über die Reihenfolge von  $t_1$  und  $t_4$  aussagt, wissen wir das nicht. Wir nennen das eine *konfuse Situation* (oder kurz *Konfusion*).

Konfusion kommt auch in der Wirklichkeit vor. Sie ist kein Phänomen das erst durch schlechte Modellbildung entstanden ist<sup>14</sup>. Beispiele dafür sind die Regel „Rechts vor Links“ oder die Arztpraxis, die Privatpatienten bevorzugt<sup>15</sup>. Auch dort wird über die Reihenfolge von Ereignissen geredet, die unabhängig voneinander sind, über deren Reihenfolge wir also nichts aussagen können. Meist wird diese Phänomene unter den Teppich gekehrt, indem man Abläufe als reine Sequenzen auffaßt (nebenläufige Ereignisse kommen dann nicht vor); deshalb ist man sich dieser Phänomene und der damit zusammenhängenden Probleme häufig nicht bewußt (in unseren Modelle für die Regel „Rechts vor Links“ und die Arztpraxis haben wir künstlich eine Reihenfolge zwischen den unabhängigen Ereignissen hergestellt). Allerdings hilft das im allgemeinen nicht, die Probleme in der Wirklichkeit zu beseitigen. Man kann beweisen,

<sup>14</sup> Im Gegenteil: Durch schlechte Modellbildung bzw. Abstraktion wird dieses Phänomen oft wegdefiniert, ohne die damit verbundenen Probleme der Wirklichkeit zu lösen.

<sup>15</sup> Ein gerade während der Fußball-WM relevantes Beispiel ist, die Abseitsregel. Denn auch dort wird über die Reihenfolge von Ereignissen geredet, die unabhängig voneinander sind: die Ballabgabe und das Eintreten eines Spielers in eine Abseitsposition (und das ist noch eine starke Vereinfachung). Es ist die Aufgabe des Schiedsrichters, diese Ereignisse (mit Hilfe der Linienrichter) in ein Reihenfolge zu bringen und so zu entscheiden, ob eine Abseitssituation vorlag.

daß bei bestimmten Aufgabestellungen (so zum Beispiel beim wechselseitigen Ausschluß) konfuse Situationen vorkommen müssen. Erst die Halbordnungssemantik ermöglicht es uns, über derartige Phänomene der Wirklichkeit mathematisch präzise Aussagen zu machen, und sie auch bei der Analyse angemessen zu berücksichtigen.

Wir verzichten hier jedoch darauf, diese Mathematik auszuformulieren und die dahinter liegende Philosophie zu vertiefen. In der Praxis können wir durch eine hinreichend feine Zeitskala Konfusion oft vermeiden – ob das immer zweckmäßig ist, ist eine andere Frage.

## 6.6 Netze mit strukturierten Marken

Bisher haben wir die klassischen Petrinetzvarianten vorgestellt. Diese eignen sich, um das Grundprinzip von Petrinetzen, ihre Philosophie, ihre Varianten und die zugehörigen Analysetechniken zu verstehen. Für die Praxis reichen jedoch Petrinetze mit lediglich „schwarzen Marken“ in vielen Fällen nicht aus. In Petrinetzen mit strukturierten Marken können die Marken selbst auch Information tragen. Da man dann Marken voneinander unterscheiden kann, spricht man dann auf von gefärbten Petrinetzen (oder von Coloured Petri Nets<sup>16</sup>).

Wir geben hier keine formale Definition für Netze mit strukturierten Marken, sondern erklären das Prinzip anhand einiger Beispiele.

**Summierung** In Abbildung 2.8 ist ein Petrinetz dargestellt, das die Werte aller Marken auf der Stelle **zahlen** aufaddiert – auf einer einzelnen Marke in Stelle **summe**.

Durch einen Doppelpunkt getrennt haben wir nach dem jeweiligen Stellennamen den Typ der einzelnen Marken angegeben, die auf der Stelle vorkommen dürfen. In unserem Beispiel sind das die ganzen Zahlen (*integer*). Auf jeder Stelle liegt dann eine Multimenge von Marken des entsprechenden Typs. Beispielsweise liegen auf der Stelle **zahlen** die ganzen Zahlen 1, 3, 5, 7 und 23, wobei 3 zweimal vorkommt. Auf der Stelle **summe** liegt genau eine Marke: die Zahl 0.

---

<sup>16</sup> Inzwischen hat sich der Begriff Coloured Petri Net (kurz CPN) jedoch für eine spezielle Variante von Petrinetzen mit strukturierten Marken durchgesetzt. Deshalb benutzen wir als Oberbegriff die Bezeichnung „Netze mit strukturierten Marken“.

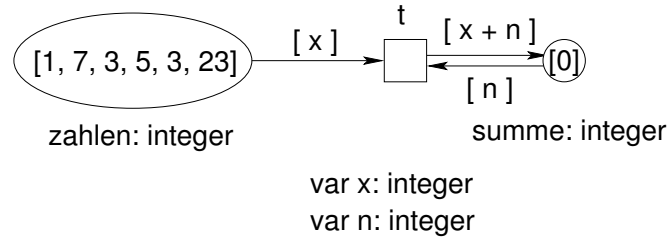


Abbildung 2.8: Berechnung der Summe

*Wir benutzen hier die Notation für Multimengen, die wir bisher für Markierungen benutzt haben:  $[1, 7, 3, 5, 3, 23]$ . Jetzt ist bereits die Markierung einer einzelnen Stelle eine Multimenge von Objekten entsprechenden Typs.*

Welche Marken beim Schalten einer Transition konsumiert und produziert werden, wird wie bisher durch die Kantenbeschriftungen festgelegt. Allerdings lassen wir in den Beschriftungen Variablen zu (deren Typ unter dem Netz angegeben ist). Damit wir sagen können, ob eine Transition schalten kann, werten wir den Ausdruck, der an den entsprechenden Kanten steht, aus. Das Ergebnis ist dann eine Multimenge von Marken (des Typs der entsprechenden Stelle). Damit wir diese Ausdrücke auswerten können, benötigen wir aber die Werte für die entsprechenden Variablen: eine *Variablenbelegung* oder kurz *Belegung*  $\beta$ . Hier sind drei Belegungen  $\beta_1$ ,  $\beta_2$  und  $\beta_3$ , die wir im folgenden noch genauer untersuchen werden:

- $\beta_1(x) = 3, \beta_1(n) = 0$
- $\beta_2(x) = 7, \beta_2(n) = 3$
- $\beta_3(x) = 9, \beta_3(n) = 0$

Für die Auswertung eines Ausdrucks  $e$  unter einer Belegung  $\beta$  schreiben wir dann  $\beta(e)$ . Beispielsweise gilt  $\beta_1([x]) = [3]$  (also die Multimenge, in der die Zahl 3 genau einmal vorkommt). Und es gilt  $\beta_2([x+n]) = [10]$ .

Um zu überprüfen, ob eine gegebene Transition  $t$  für eine gegebene Belegung  $\beta$  in einer Markierung  $m$  aktiviert ist, werten wir zunächst die Ausdrücke an den eingehenden Kanten der Transition aus (mit Belegung  $\beta$ ). Wenn für jede Kante das Ergebnis in der Markierung der entsprechenden Stelle enthalten ist, ist die Transition aktiviert. Wir schreiben dann  $m \xrightarrow{(t,\beta)}$  und sagen auch, daß  $t$  mit *Modus*  $\beta$  in  $m$  aktiviert ist.

*Wir können also nicht mehr über die Aktiviertheit einer Transition reden, sondern nur noch über die Aktiviertheit einer Transition mit einem bestimmten Modus  $\beta$ .*

In der Anfangsmarkierung  $m_0$  unseres Beispielsystems gilt  $\beta_1([x]) = [3]$  und  $\beta_1([n]) = [0]$ . Die erste Multimenge ist in  $[1, 7, 3, 5, 3, 23]$  (der Markierung von **zahlen** enthalten); die zweite Multimenge ist in  $[0]$  der aktuellen Markierung von **summe** enthalten. Also gilt  $m_0 \xrightarrow{(t, \beta_1)}$ .

Dagegen gilt weder  $m_0 \xrightarrow{(t, \beta_2)}$  noch  $m_0 \xrightarrow{(t, \beta_3)}$  (im ersten Fall gilt  $\beta_2([n]) = [3] \not\subseteq [0]$ , im zweiten Fall gilt  $\beta_2([x]) = [9] \not\subseteq [1, 7, 3, 5, 3, 23]$ ).

Beim Schalten einer mit Modus  $\beta$  aktivierten Transition  $t$  werden Marken entsprechend der ausgewerteten Ausdrücke der eingehenden Kanten von den Stellen abgezogen und die Marken entsprechend der ausgewerteten Ausdrücke der ausgehenden Kanten zu den entsprechenden Stellen hinzugefügt. Nach dem Schalten von  $t$  mit Belegung  $\beta_1$  erhalten wir also die Markierung  $[1, 3, 5, 7, 23]$  für die Stelle **zahlen** und die Markierung  $[3]$  für die Stelle **summe** (aus  $[0]$  wird zunächst die 0 entfernt und dann die 3 hinzugefügt).

Die Transition  $t$  in obigem Beispiel entnimmt jeweils eine Marke  $x$  aus der Stelle **zahlen** und summiert diesen Wert auf die bisher berechnete Summe  $n$ , die als eine Marke auf der Stelle **summe** hinterlegt ist. Insgesamt wird so sukzessive der Wert aller Marken auf der Stelle **zahlen** auf die eine Marke auf Stelle **summe** aufaddiert.

**Sieb des Eratosthenes** Um das Prinzip noch etwas zu verdeutlichen, betrachten wir ein weiteres Beispiel: Das Sieb des Eratosthenes zur Bestimmung aller Primzahlen bis zu einer bestimmten Obergrenze  $N$ . Dieses Netz ist in Abbildung 2.9 dargestellt.

Initial kommt auf der Stelle **zahlen** jede Zahl von 2 bis  $N$  genau einmal vor (hier lassen wir nur Marken vom Typ natürliche Zahl zu). Die Transition entnimmt von der Stelle **zahlen** in jedem Falle zwei Zahlen<sup>17</sup>  $x$  und  $n * x$ , also eine Zahl  $x$  und ein Vielfaches  $n * x$ . Die Zahl  $x$  selbst wird jedoch wieder zurückgelegt. Das Vielfache  $n * x$  wird dagegen nicht zurückgelegt – es wird ausgesiebt. So verschwinden aus der Stelle **zahlen** nach und nach alle Zahlen, die ein Vielfaches einer Zahl aus der Stelle **zahlen** sind. Irgendwann sind also alle diese Vielfachen ausgesiebt – übrig bleiben irgendwann nur noch die Primzahlen (dann gibt es auch keine Belegung mehr, unter der die Transition

<sup>17</sup> Die konkreten Werte werden durch den Modus festgelegt.

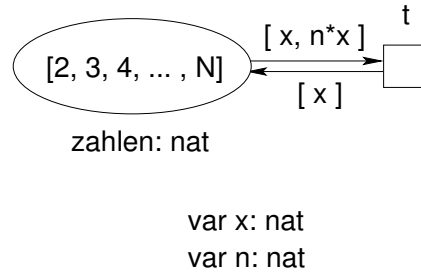


Abbildung 2.9: Sieb des Eratosthenes

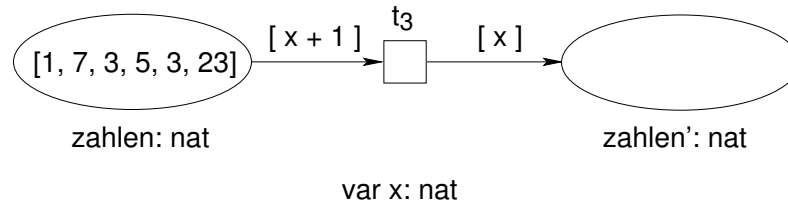


Abbildung 2.10: Vorgänger berechnen

t schalten kann). Dies genau ist die Idee des Algorithmus von Eratosthenes zur Bestimmung der Primzahlen von 2 bis  $N$ .

*Es gibt wohl kaum eine einfachere vollständige formale Formulierung des Prinzips des „Siebs des Eratosthenes“.*

**Subtrahieren ohne Minus** Als letztes Beispiel betrachten wir ein Petri-netz, das alle Zahlen von einer Stelle **zahlen** wegnimmt und um eins vermindert auf die Stelle **zahlen'** hinzufügt. Naiv würde man hier eine Transition mit Kantenbeschriftung  $[x]$  an der eingehenden Kante erwarten und mit der Beschriftung  $[x - 1]$  an der ausgehenden Kante.

Abbildung 2.10 zeigt, daß es ganz ohne Minusoperation geht. Wir schreiben  $[x + 1]$  an die eingehende Kante und  $[x]$  an die ausgehende Kante – auf diese Weise wird die Plusoperation implizit invertiert. Auf diese Weise vermeiden wir das Problem, die Minusoperation für die natürlichen Zahlen definieren zu müssen (was für  $0 - 1$  problematisch wäre).

Auf die gleiche Weise kann man beliebige Operationen invertieren<sup>18</sup>:

<sup>18</sup> Dieser Trick ist ein netter Modellierungstrick; wenn wir das Netz jedoch effizient aus-

Wenn wir Funktion  $f$  invertieren wollen schreiben wir  $[f(x)]$  an die eingehende Kante und  $[x]$  an die ausgehende Kante der entsprechenden Transition.

Damit wollen wir es zunächst mit den Beispielen für Netze mit strukturierten Marken, die oft auch *High-level-Netze* genannt werden, bewenden lassen. Die Beispiele sind zugegebenermaßen sehr einfach, denn jedes Beispiel hatte nur eine Transition. Die Beispiele zeigen jedoch das Prinzip sehr gut. Außerdem haben wir bisher nur Beispiele für sequentielle Algorithmen modelliert (die insbesondere immer terminieren). Wir werden gegen Ende der Vorlesung weitere Modelle für reaktive (und sehr viel komplexere) Systeme kennenlernen (im Abschnitt „Verteilte Algorithmen“).

---

führen wollen empfiehlt sich dieser Trick nicht (zumindest nicht bei komplizierten Operationen), da man das Inverse zum Schalten der Transition richtig raten muß. Ob der Trick zweckmäßig ist oder nicht, hängt also stark vom Zweck des Modells ab.



# Kapitel 3

## Geschäftsprozeßmodellierung

In diesem Kapitel wenden wir uns einem konkreten Anwendungsgebiet zu: der Geschäftsprozeßmodellierung. Wir werden sehen, daß uns die Petrinetztheorie einige Techniken zur Analyse und zur Konstruktion von Geschäftsprozessen liefert. Leider können wir hier nur einen kurzen Ausflug in das Gebiet der Geschäftsprozeßmodellierung machen – eine ausführliche Behandlung des Gebietes könnte mehrere Kurse füllen.

### 1 Überblick und Begriffsbildung

Bevor wir uns den Petrinetztechniken zuwenden, führen wir die wichtigsten Begriffe informell ein.

**Geschäftsprozeß** Unter einem *Geschäftsprozeß* verstehen wir eine Menge von *Aktivitäten*, die in einem Unternehmen nach bestimmten Regeln und auf ein bestimmtes Ziel hin ausgeführt werden.

*Diesen Begriff kann man noch beliebig verfeinern und differenzieren. Wir haben ihn hier bewußt so allgemein wie möglich gefaßt. Ganz bewußt haben wir darauf verzichtet, die informationstechnische Unterstützung zu fordern (das ist genau der Aspekt, der in unserer Begriffsbildung erst beim Workflow hinzu kommt).*

*Oft wird eine Unterscheidung zwischen Geschäftsprozessen, die unmittelbar den Unternehmenszielen dienen („die sind wichtig, weil sie Geld bringen“), und administrativen Geschäftsprozessen („die sind weniger wichtig, weil sie Geld kosten“) getroffen.*

Hier sind ein paar Beispiele für Geschäftsprozesse:

- Kreditvergabe
- Dienstreise
- Bauantrag
- Krankenhausaufenthalt, Patientenaufnahme, Untersuchung, Entlassung, etc.
- Flugzeugentwurf

Diese Beispiele zeigen die Vielfalt der Möglichkeiten. Ihre Dauer kann im Bereich von wenigen Minuten, mehrerer Stunden, bis hin zu Wochen, Monaten oder Jahren liegen. Oft setzt sich ein Geschäftsprozeß aus weiteren Geschäftsprozessen (einer niedrigeren Abstraktionsebene) zusammen.

**Aktivität** Unter einer *Aktivität* versteht man einen in sich abgeschlossenen Arbeitsschritt, der nicht mehr in weitere Teilschritte zerlegt werden kann (er ist atomar).

*Was „in sich abgeschlossen“ bedeutet, ist eine Frage der Abstraktionsebene. Denn natürlich läßt sich jeder Arbeitsschritt in weitere Schritte zerlegen, wenn man nur genau genug hinsieht. Deshalb hätten wir vielleicht besser formulieren sollen „der unter gewissen Gesichtspunkten nicht mehr weiter zerlegt werden sollte“ – aber diese Formulierung ist etwas zu sperrig.*

*Tatsächlich kann das, was auf der einen Abstraktionsebene eine Aktivität ist, auf einer detaillierteren Ebene ein Geschäftsprozeß sein. Beispielsweise ist die Patientenaufnahme oder die Entlassung aus der Sicht des Geschäftsprozesses Krankenhausaufenthalt eine Aktivität; für sich betrachtet sind sie aber wieder ein Geschäftsprozeß, der sich in weitere Aktivitäten zerlegen läßt.*

Beispiele für Aktivitäten sind

- Bonität (eines Kreditnehmers) prüfen
- Kreditvertrag ausfertigen
- Brief schreiben
- Dienstreiseantrag ausfüllen
- Blut abnehmen

**Dokument** Zum Austausch von Information zwischen verschiedenen Aktivitäten eines Geschäftsprozesses und zwischen verschiedenen Geschäftsprozessen dienen *Dokumente*. Dokumente werden durch Aktivitäten erzeugt, benutzt, modifiziert und gelöscht.

*Wir fassen hier den Begriff des Dokuments wieder sehr weit. Selbst ein Telefonanruf (bzw. die entsprechende Gesprächsnotiz) können wir als Dokument ansehen. Im Extremfall können wir auch einen Tupel einer Tabelle einer Datenbank als Dokument auffassen.*

*Wir betrachten hier jedoch Dokumente noch rein konzeptuell und nicht technologisch (XML, HTML, MS Word, ...).*

Hier sind einige Beispiele für Dokumente

- Brief
- Kreditvertrag
- Personalfragebogen
- Ergebnis einer Untersuchung
- CAD-Datei

**Agent und Ressource** Unter einer *Ressource* verstehen wir ein Betriebsmittel, das zur Durchführung einer Aktivität benötigt wird. Wenn es sich bei dem „Betriebsmittel“ um eine Person handelt, sprechen wir jedoch lieber von einem *Agenten*.

Hier sind einige Beispiele für Ressourcen und Agenten:

- Sachbearbeiter
- Geschäftsführer
- Arzt
- Laserdrucker
- Plotter
- Kuvertiermaschine

*In unseren Beispielen haben wir streng genommen gar nicht Agenten und Betriebsmittel angegeben, sondern Rollen. Ein Agent wäre eine konkrete Person, z.B. „Egon Müller“, oder ein konkreter Drucker „lp4711“.*

*In der Praxis werden bei der Modellierung eines Geschäftsprozesse die Agenten und Ressourcen nicht direkt einer Aktivität zugeordnet. Die Zuordnung geschieht über Rollen, die die Ressourcen und Agenten einnehmen können. Der Grund dafür ist, daß sich die Rollen in einem Unternehmen relativ selten ändern (Organisationsstruktur); die konkreten Personen in diesen Rollen dagegen können sich relativ schnell ändern (Mittagspause, Schichtwechsel, Urlaub, Krankheit, Arbeitsplatzwechsel, etc.) Darauf können wir hier jedoch nicht näher eingehen.*

**Aspekte von Geschäftsprozessen** Bei der Analyse und Modellierung von Geschäftsprozessen hat es sich als zweckmäßig erwiesen, bestimmte Aspekte zu unterscheiden und (soweit wie möglich) unabhängig voneinander zu betrachten. Nachfolgend zählen wir die wichtigsten Aspekte auf:

**Informationsbezogene Aspekte** Welche Daten bzw. Dokumente werden benutzt; wie sind die Dokumente konzeptuell aufgebaut.

**Organisatorische Aspekte** Welche Ressourcen und Agenten gibt es (und welche Rollen können sie einnehmen).

**Verhaltensbezogene Aspekte** In welcher Abfolge werden die Aktivitäten ausgeführt.

**Funktionale Aspekte** Welches Ziel wird verfolgt, (oft wird auch die Strukturierung diesem Aspekt zugeordnet), ...

Es gibt noch einige weitere Aspekte, die wir hier nicht weiter betrachten. Insbesondere, wenn Geschäftsprozesse (teilweise oder vollständig) informationstechnisch unterstützt werden sollen, müssen viele weitere Aspekte berücksichtigt werden.

**Workflow** Unter einem *Workflow* verstehen wir die (teilweise oder vollständige) informationstechnische Unterstützung eines Geschäftsprozesses. Dann muß zum Beispiel beschrieben werden, wie die Dokumente repräsentiert sind, welche Aktivität durch welche Applikation unterstützt wird, wie auszuführende Aktivitäten den Agenten zugeteilt werden, etc. Dies alles wird unter dem *operationalen Aspekt* und weiteren *technischen Aspekten*

zusammengefaßt.

Damit wollen wir es mit dem Überblick und der Begriffsbildung bewenden lassen. Im folgenden werden wir nur noch einen Aspekt der Geschäftsprozessmodellierung betrachten: den verhaltensbezogenen Aspekt.

*Petrinetze in Kombination mit anderen Konzepten lassen sich durchaus auch für die Modellierung und Untersuchung anderer Aspekte von Geschäftsprozessen und Workflows einsetzen (bis hin zu ihrer automatischen Ausführung). Eine Betrachtung der anderen Aspekte würde hier jedoch zu weit führen.*

## 2 Workflow-Netze

In diesem Abschnitt betrachten wir eine spezielle Version von Petrinetzen zur Modellierung und Analyse des Verhaltens von Geschäftsprozessen. Insbesondere werden wir eine Technik kennenlernen, mit der wir „vernünftige“ Geschäftsprozessen von „unvernünftigen“ unterscheiden können. Das heißt natürlich nicht, daß ein „vernünftiger“ Geschäftsprozeß korrekt ist; aber absolut blödsinniges Verhalten ist ausgeschlossen. Eine Analogie dazu ist die Normalformtheorie auf dem Gebiet der Relationalen Datenbanken. Die Normalformen (insbes. die 3NF) garantieren, daß bestimmte unerwünschte Anomalien nicht auftreten. Entsprechend schließen „vernünftige“ Geschäftsprozesse anomales Verhalten aus.

Die Motivation hinter der folgenden Definition ist, daß jeder Geschäftsprozeß einen definierten Anfang und ein definiertes Ende hat. Um den Anfang und das Ende zu kennzeichnen, zeichnen wir in einem *Workflow-Netz*<sup>1</sup> eine Anfangsstelle  $i$  (für engl. input) und eine Endstelle  $o$  (für engl. output) aus. Die Anfangsstelle hat einen leeren Vorbereich; sie ist zu Beginn mit einer Marke besetzt. Die Endstelle besitzt einen leeren Nachbereich; sie sollte am Ende mit einer Marke besetzt sein (dazu später mehr).

### Definition 3.1 (Workflow-Netz und Workflow-System)

Ein Netz  $N = (S, T, F)$  ohne isolierte Elemente mit zwei ausgezeichneten Stellen  $i, o \in S$  mit  $\bullet i = \emptyset$  und  $o \bullet = \emptyset$  heißt *Workflow-Netz* (*Wf-Netz*).

<sup>1</sup> Streng genommen sollten wir besser von Geschäftsprozessnetzen oder engl. von business process nets sprechen und nicht von Workflow-Netze, da wir die Geschäftsprozesse ja noch nicht informationstechnisch unterstützen wollen. Der Begriff Workflow-Netz hat sich inzwischen jedoch etabliert.

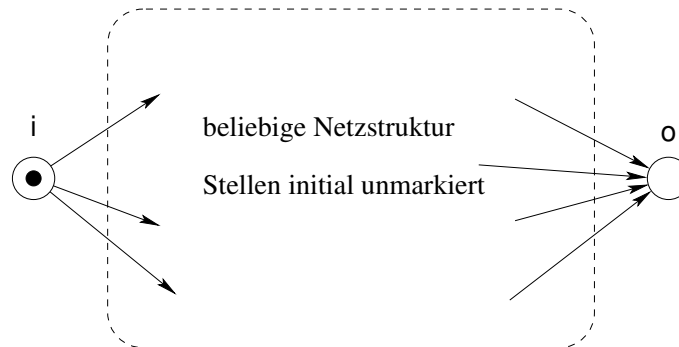


Abbildung 3.1: Schematische Darstellung eines Workflow-Netzes

Für ein Workflow-Netz  $N$  heißt  $\Sigma = (N, W, m_0)$  mit  $W(f) = 1$  für alle  $f \in F$  und mit  $m_0 = [i]$  das zu  $N$  gehörige *Workflow-System*.

Das Workflow-System entsteht aus dem Workflow-Netz, indem wir (ausschließlich) die Anfangsstelle  $i$  markieren und alle Kanten mit 1 beschriften (wir machen also aus dem Workflow-Netz ein gewöhnliches S/T-System mit Anfangsmarkierung  $[i]$ ). Da sich das zugehörige Workflow-System kanonisch aus dem Workflow-Netz ergibt, unterscheiden diese beiden Begriffe im folgenden nicht mehr sehr streng.

Abbildung 3.1 zeigt das allgemeine Schema für ein Workflow-Netz (bzw. das zugehörige Workflow-System). Die Transitionen des Netzes repräsentieren die Aktivitäten des Geschäftsprozesses, die Marken auf den inneren Stellen repräsentieren die Dokumente (ohne deren Inhalt), die zwischen den Aktivitäten ausgetauscht werden.

Abbildung 3.2 zeigt ein (mehr oder weniger) konkretes Beispiel für ein Workflow-Netz. Es handelt sich um das Modell eines Geschäftsprozesses zur Behandlung einer Beschwerde, wobei wir hier nur abstrakte Bezeichnungen für die Transitionen und Stellen benutzen. In der Praxis würden die Transitionen mit der Aktivität bezeichnet, die sie repräsentieren, und die Stellen mit den Dokumenten, die sich darauf befinden.

Ganz unabhängig davon, welche Aufgabe der Workflow aus Abb. 3.2 zu erledigen hat, ist der Workflow schlecht modelliert. Beispielsweise sind die Markierung  $[o, d]$  und  $[o, e]$  erreichbar. Das bedeutet, daß der Workflow beendet ist (weil die Endstelle  $o$  bereits markiert ist), aber es befindet sich noch ein Dokument „im Workflow“ (nämlich auf der Stelle  $d$  oder  $e$ ). Irgend-

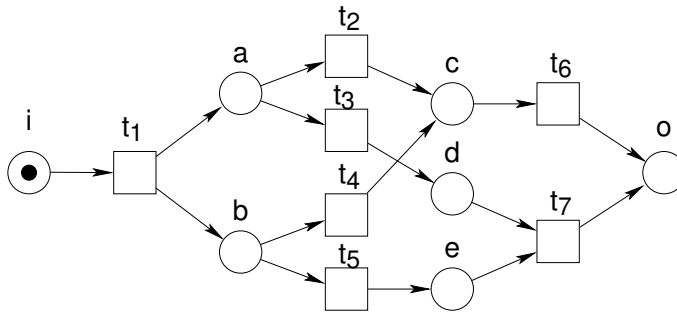


Abbildung 3.2: Ein Workflow-Netz

ein Mitarbeiter hat also noch ein Dokument, das den Workflow betrifft, auf dem Schreibtisch, obwohl der Workflow bereits abgeschlossen ist. Langfristig werden sich also irgendwo immer mehr Dokumente von Geschäftsprozessen, die schon abgeschlossen sind, auf irgendeinem Schreibtisch sammeln.

*Wir gehen hier davon aus, daß das abschließende Archivieren oder Einsortieren von Dokumenten, die dauerhaft (also über den Geschäftsprozeß hinaus) aufbewahrt werden müssen, Teil des Geschäftsprozesses ist.*

Der Geschäftsprozeß aus Abb. 3.2 ist noch in einer weiteren Hinsicht schlecht modelliert. Denn er kann zweimal beendet werden: Es ist die Markierung  $[c, o]$  erreichbar, in der der Geschäftsprozess bereits beendet ist (gemäß unserer Interpretation der Stelle  $o$ ). Dann kann die Transition  $t_6$  schalten und beendet den Geschäftsprozeß noch ein zweites mal. Das widerspricht unserer Auffassung von einem klar definierten Ende eines Geschäftsprozesses. Im folgenden werden wir nun solch schlechtes Verhalten ausschließen. Dazu definieren wir die *vernünftigen* Workflow-Netze.

### Definition 3.2 (Vernünftige Workflow-Netze)

Ein Workflow-Netz  $N = (S, T, F)$  mit zugehörigem Workflow-System  $\Sigma = (N, W, m_0)$  heißt *vernünftig*, wenn die folgenden drei Bedingungen erfüllt sind:

- (i) Für jede Markierung  $m \in [m_0\rangle$  gibt es eine Markierung  $m' \in [m\rangle$  mit  $m(o) \geq 1$ .

*Das heißt, daß der Geschäftsprozeß von jedem Zustand aus beendet werden kann („Terminierung“)*

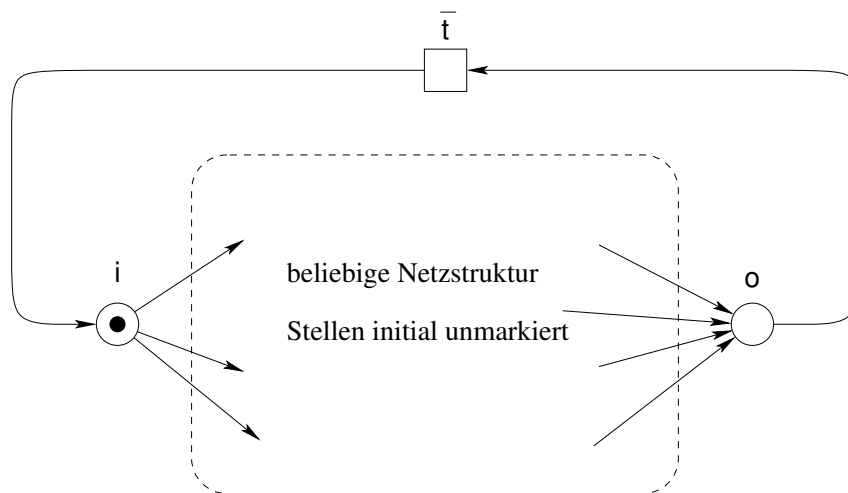


Abbildung 3.3: Schematische Darstellung des erweiterten Workflow-Systems

- (ii) Für jede Markierung  $m \in [m_0\rangle$  mit  $m(o) \geq 1$  gilt  $m = [o]$ .

*Das heißt, wenn der Geschäftsprozeß beendet ist, befinden sich keine Dokumente mehr im Prozeß, und er kann auch nicht noch einmal beendet werden („saubere Terminierung“).*

- (iii) Keine Transition  $t \in T$  ist tot (in  $\Sigma$ ).

*Das heißt, daß jede Aktivität im Geschäftsprozeß, auch irgendwie ausgeführt werden kann („keine überflüssigen Aktivitäten“).*

Die Definition der vernünftigen Workflow-Netze ist eine unmittelbare Übersetzung unserer intuitiven Vorstellung von vernünftig (unter der gewählten Interpretation der Anfangs- und Endstellen und der Interpretation der Marken auf inneren Stellen als Dokumenten). Um so erstaunlicher ist es, daß wir diesen Begriff ganz leicht auf Begriffe zurückführen können, die wir bereits kennen: auf die Beschränktheit und die Lebendigkeit von S/T-Systemen.

Dazu definieren wir zunächst den Begriff des *erweiterten Workflow-Systems*, das in Abb. 3.3 schematisch dargestellt ist. Im erweiterten Workflow-System fügen wir dem Workflow-System lediglich eine neue Transition  $\bar{t}$  hinzu, die die Marke von  $o$  auf  $i$  zurücklegt – sonst ändert sich nichts.

**Definition 3.3 (Erweitertes Workflow-System)**

Sei  $N = (S, T, F)$  ein Workflow-Netz,  $\Sigma = (N, W, m_0)$  das zugehörige Workflow-System und gelte  $\bar{t} \notin S \cup T$ .

Wir definieren  $\bar{N} = (S, \bar{T}, \bar{F})$  mit  $\bar{T} = T \cup \{\bar{t}\}$  und  $\bar{F} = F \cup \{(o, \bar{t}), (\bar{t}, i)\}$ , und wir definieren  $\bar{\Sigma} = (\bar{N}, \bar{W}, m_0)$  mit  $\bar{W}(f) = 1$  für alle  $f \in \bar{F}$ . Das Netz  $\bar{N}$  nennen wir *erweitertes Workflow-Netz* und das S/T-System  $\bar{\Sigma}$  nennen wir *erweitertes Workflow-System* von  $N$ .

**Satz 3.4 (Vernünftige Workflow-Netze)**

Ein Workflow-Netz  $N$  ist genau dann vernünftig, wenn das zugehörige erweiterte Workflow-System  $\bar{\Sigma}$  beschränkt und lebendig ist.

### 3 Workflow-Konstrukte

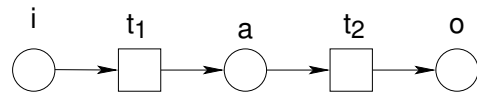
In vielen Fällen wird ein Workflow aus einzelnen *Workflow-Konstrukten* aufgebaut. Die Standard-Konstrukte sind die *Sequenz*, die *Alternative*, die *Wiederholung* und die *parallele Ausführung* (siehe Abb. 3.4). Wenn ein Workflow allein aus diesen Konstrukten aufgebaut ist, dann ist er immer vernünftig. Um das zu zeigen, formalisieren wir den Begriff des Workflow-Konstruktes und der aus Workflow-Konstrukten aufgebauten Workflows formalisieren. Dann werden wir zeigen, wie man neue Workflow-Konstrukte definieren kann, so daß alle daraus aufgebauten Workflows vernünftig sind.

*Aus zeitlichen Gründen ist die folgende Darstellung recht knapp gehalten. Sie würde eigentlich eine etwas ausführlichere Behandlung verdienen. ...*

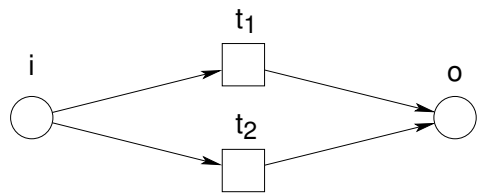
In Abb. 3.4 sind die vier oben genannten Standardkonstrukte dargestellt – technisch gesehen ist ein Workflow-Konstrukt nichts anderes als ein spezielles Workflow-Netz. Da wir uns später nicht auf die vier obigen Konstrukte einschränken wollen, definieren wir nun den Begriff des Workflow-Konstruktes: ein beliebiges Workflow-Netz – zunächst ohne jede Einschränkung. Natürlich ist nicht jedes Workflow-Konstrukt sinnvoll. Was sinnvoll heißen soll, werden wir jedoch erst später definieren.

**Definition 3.5 (Workflow-Konstrukt)**

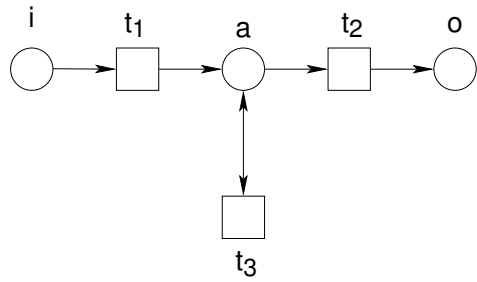
Ein Netz  $N = (S, T, F)$  heißt *Workflow-Konstrukt*, wenn es ein Workflow-Netz ist.



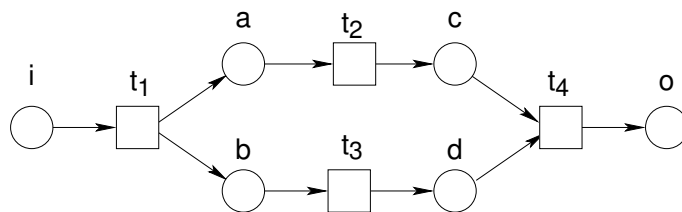
Sequenz



Alternative



Iteration



Parallele Ausfuehrung

Abbildung 3.4: Die vier Standard-Konstrukte für Workflows

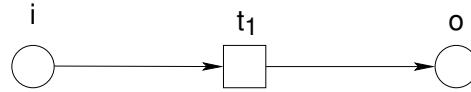


Abbildung 3.5: Das triviale Workflow-Netz

Aus einer Menge von Workflow-Konstrukten können wir nun verschiedene Workflows aufbauen. Dazu gehen wir von dem *trivialen Workflow-Netz* aus, das in Abb. 3.5 dargestellt ist, und ersetzen schrittweise einzelne Transitionen durch ein Workflow-Konstrukt, wie dies in Abb. 3.6 schematisch dargestellt ist. Wenn wir in einem Workflow-Netz  $N$  eine Transition  $t$  durch ein Workflow-Konstrukt  $K$  ersetzen, schreiben wir dafür  $N^{t \rightarrow K}$ . Wenn  $N$  ein Workflow-Netz ist,  $t$  eine Transition des Workflow-Netzes  $N$  ist und  $K$  ein Workflow-Konstrukt, dann ist  $N^{t \rightarrow K}$  auch ein Workflow-Netz.

*Die Ersetzung  $N^{t \rightarrow K}$  kann man natürlich auch präzise mathematisch definieren. Das erfordert aber etwas technischen Aufwand (weil man die Stellen und Transitionen von  $K$  ggf. umbenennen muß). Deshalb sparen wir uns hier eine Formalisierung.*

### Definition 3.6 (Konstruierbare Workflow-Netze)

Seien  $K_1, \dots, K_n$  Workflow-Konstrukte. Ein Workflow-Netz  $N$  heißt mit den Konstrukten  $K_1, \dots, K_n$  *konstruierbar*, wenn es eine Folge von Netzen  $N_0, N_1, \dots, N_k$  gibt, wobei  $N_0$  das triviale Workflow-Netz ist,  $N_k = N$  ist und für jedes  $i$  eine Transition  $t$  von  $N_i$  und ein  $K_j$  existiert, so daß gilt  $N_{i+1} = N_i^{t \rightarrow K_j}$ .

Eine Menge von Workflow-Konstrukten  $\{K_1, \dots, K_n\}$  heißt *vernünftig*, wenn alle daraus konstruierbaren Workflow-Netze vernünftig sind.

Ein Beispiel für eine Menge vernünftiger Workflow-Konstrukte sind die vier Konstrukte aus Abb. 3.4. Das direkt zu beweisen, ist allerdings etwas aufwendig. Deshalb formulieren wir im folgenden eine allgemeine hinreichende Bedingung dafür, daß eine Menge von Workflow-Konstrukten vernünftig ist. Diese hinreichende Bedingung (Folgerung 3.8) gilt für die Standard-Workflowkonstrukte fast trivialerweise.

Offensichtlich ist eine notwendige Voraussetzung für die Vernünftigkeit von  $\{K_1, \dots, K_n\}$ , daß jedes einzelne Workflow-Konstrukt ein vernünftiges Workflow-Netz ist (denn sonst führt bereits die Ersetzung des trivialen Workflows-Netzes durch ein Konstrukt zu einem nicht vernünftigem Workflow-Netz). Das reicht jedoch nicht aus

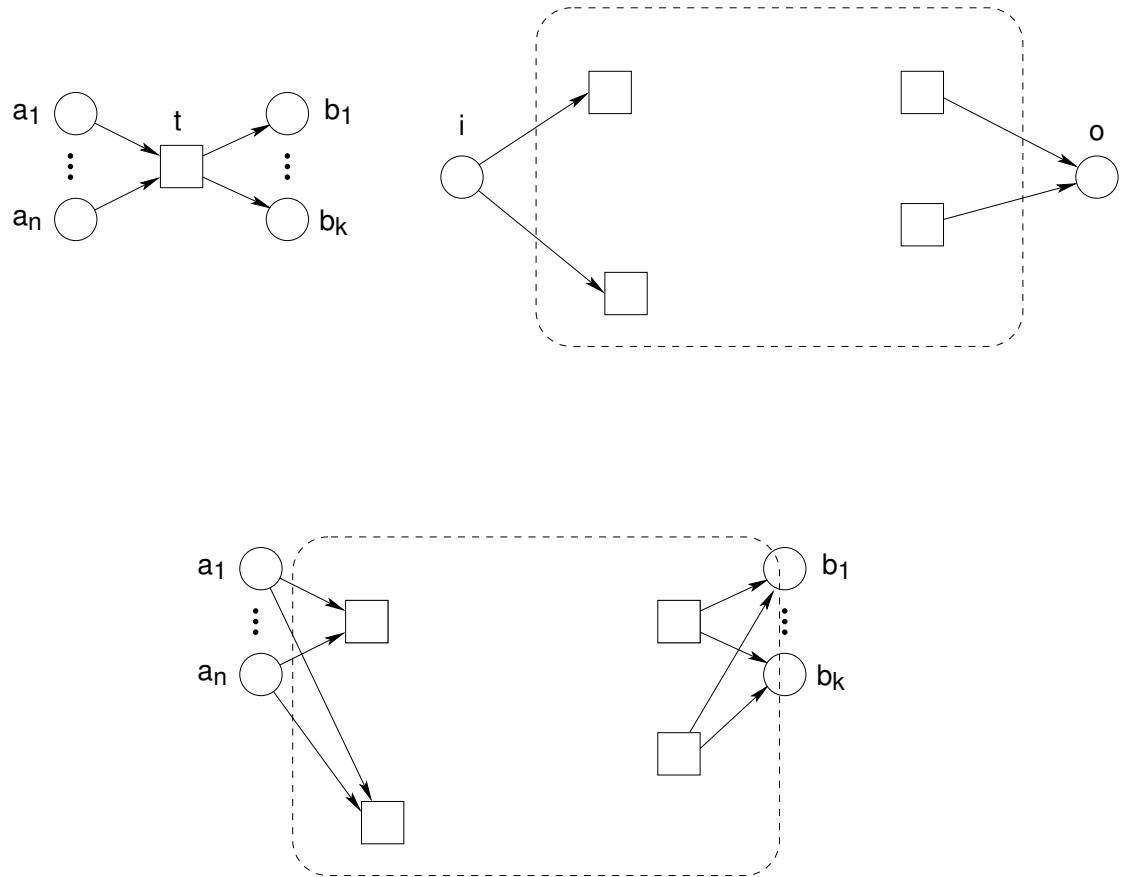


Abbildung 3.6: Ersetzung einer Transition  $t$  durch ein Workflow-Konstrukt  $K$

*Ein Gegenbeispiel dazu werden wir uns in der Übung überlegen.*

Wenn wir allerdings fordern, daß für jedes Workflow-Konstrukt das jeweilige erweiterten Workflow-System nicht nur beschränkt, sondern sogar 1-sicher ist, dann ist  $\{K_1, \dots, K_n\}$  vernünftig. Das folgt aus dem folgenden Satz (den wir hier jedoch nicht beweisen).

**Satz 3.7**

Sei  $\Sigma$  ein gewöhnliches S/T-System, das 1-sicher und lebendig ist und sei  $K$  ein Workflow-Konstrukt, für das das erweiterte Workflow-System auch 1-sicher und lebendig ist. Dann ist  $\Sigma^{t \rightarrow N}$  ebenfalls 1-sicher und lebendig.

Das erweiterte Workflow-System zum triviale Workflow-Netz ist offensichtlich 1-sicher und lebendig. Wenn für jedes Workflow-Konstrukt  $K_i$  das zugehörige erweiterte Workflow-System 1-sicher und lebendig ist, ist damit jedes mit  $\{K_1, \dots, K_n\}$  konstruierbare Workflow-System ebenfalls 1-sicher und lebendig – also insbesondere beschränkt und lebendig, d. h. jedes konstruierbare Workflow-Netz ist vernünftig.

**Folgerung 3.8**

Sei  $\{K_1, \dots, K_n\}$  eine Menge von Workflow-Konstrukten, so daß für jedes  $K_i$  das zugehörige erweiterte Workflow-System 1-sicher und lebendig ist. Dann ist  $\{K_1, \dots, K_n\}$  vernünftig.

Insbesondere folgt daraus, daß die Menge der vier Standard-Konstrukte aus Abb. 3.4 vernünftig (daß die jeweiligen erweiterten Workflow-Systeme 1-sicher und lebendig sind, sieht man auf einen Blick). Wir können aber noch beliebige weitere Workflow-Konstrukte hinzunehmen, wenn deren zugehörige erweiterte Workflow-Systeme 1-sicher und lebendig sind.

...



# Kapitel 4

## Verifikation verteilter Algorithmen

In diesem Kapitel werden wir sehen, wie man Petrinetze zur Modellierung und Verifikation verteilter Algorithmen einsetzen kann. Dabei liegt der Schwerpunkt auf den zugrundeliegenden Konzepten. Diese lassen sich einfacher für S/T-Systeme erläutern als für Netze mit strukturierten Marken. Deshalb beginnen wir mit S/T-Systeme; alle Konzepte lassen sich jedoch auf Netze mit strukturierten Marken übertragen. Wie das geht, werden wir am Ende dieses Kapitels anhand einiger Beispiele andeuten.

Zunächst betrachten wir aber nur S/T-Systeme und die nötigen Erweiterungen.

### 1 S/T-Systemen und ihre Prozesse

... *Die Motivation fehlt noch*

#### **Definition 4.1 (Kausalnetz)**

Ein endliches oder unendliches Netz  $K = (B, E, \prec)$  heißt *Kausalnetz*, wenn

- jede Transition  $e \in E$  einen nicht-leeren Vorbereich und einen nicht-leeren Nachbereich hat (d. h.  $\bullet e \neq \emptyset \neq e\bullet$ ),
- jede Stelle  $b \in B$  unverzweigt ist (d. h.  $|\bullet b| \leq 1$  und  $|b\bullet| \leq 1$ ),
- die transitive Hülle von  $\prec$  keine Zyklen enthält (d. h. wenn  $\prec^+$  irreflexiv ist),

- jede Transition  $e \in E$  nur endlich viele direkte und indirekte Vorgänger besitzt (d.h. die Menge  $\{x \in B \cup E \mid x \prec^+ e\}$  ist für jedes  $e \in E$  endlich).

Die Transitionen eines Kausalnetzes nennen wir *Ereignisse*, die Stellen eines Kausalnetzes nennen wir *Bedingungen*. Die transitive Hülle  $\prec^+$  von  $\prec$  bezeichnen wir im folgenden mit  $\prec$ .

Die Relation  $\prec$  beschreibt die kausalen Abhängigkeiten zwischen den Bedingungen und Ereignissen des Kausalnetzes. Zwei Elemente  $x, y \in B \cup E$  heißen kausal abhängig, wenn gilt  $x \prec y$  oder  $y \prec x$ . Anderenfalls heißen sie kausal unabhängig oder *nebenläufig*.

#### Definition 4.2 (Begriffe für Kausalnetze)

Sein  $K = (B, E, \prec)$  ein Kausalnetz.

- ${}^\circ K = \{b \in B \mid \bullet b = \emptyset\}$  heißt der *Anfang* von  $K$ .
- $K^\circ = \{b \in B \mid b^\bullet = \emptyset\}$  heißt das *Ende* von  $K$ .
- Teilmengen  $Q \subseteq B$  können wir als Markierungen über  $B$  auffassen (mit Einträgen 0 und 1).

Eine Teilmenge  $Q \subseteq B$  heißt *Zustand* von  $K$ , wenn sie in  $K$  vom Anfang  ${}^\circ K$  aus erreichbar ist (d. h. wenn gilt  $Q \in [{}^\circ K)$ ).

**Achtung:** Das Ende eines Ablaufes  $K$  ist für unendliche Kausalnetze im allgemeinen kein Zustand. Für endliche Kausalnetze ist das Ende des Kausalnetzes immer ein Zustand.

Alle Bedingungen eines Zustandes sind paarweise kausal unabhängig.

Die Beziehung zwischen einem Kausalnetz  $K = (B, E, \prec)$  und einem S/T-System  $\Sigma = (N, W, m_0)$  mit Netz  $N = (S, T, F)$  stellen wir her, indem wir die Bedingungen des Kausalnetzes beschriften („labeln“) – und zwar mit den korrespondierenden Stellen des S/T-Systems. Eine Kausalnetz mit einer zum S/T-System passende Beschriftung nennen wir dann Prozeß des S/T-Systems.

*Ursprünglich wurden die Ereignisse eines Prozesses auch noch beschriftet. Nämlich mit den korrespondierenden Transitionen des S/T-Systems. Das ist jedoch nicht notwendig, um Prozesse zu definieren. Deshalb verzichten wir darauf, weil so die Definition technisch etwas einfacher wird.*

**Definition 4.3 (Prozeß eines S/T-Systems)**

Sei  $\Sigma = (N, W, m_0)$  ein S/T-System mit Netz  $N = (S, T, F)$ , sei  $K = (B, E, \prec)$  ein Kausalnetz mit endlichem Anfang  ${}^\circ K$ , und sei  $\lambda : B \rightarrow S$  eine Abbildung.

Für jede endliche Teilmenge von  $Q \subseteq B$  definieren wir  $\lambda(Q)$  als eine Markierung von  $N$  wie folgt:  $\lambda(Q)(s) = |\{b \in Q \mid \lambda(b) = s\}|$  für alle  $s \in S$ .

Ein beschriftetes Kausalnetz  $\pi = (K, \lambda)$  heißt *Prozeß* von  $\Sigma$ , wenn

- der Anfang des Kausalnetzes der Anfangsmarkierung des S/T-Systems entspricht (d. h.  $\lambda({}^\circ K) = m_0$ ) und wenn
- für jedes Ereignis  $e \in E$  des Kausalnetzes eine entsprechende Transition  $t \in T$  des S/T-Systems existiert (d. h.  $\lambda({}^\bullet e) = \bar{t}$  und  $\lambda(e^\bullet) = t^+$ ).

... Hier kommen irgendwann noch ein paar Beispiele für endliche und unendliche Prozesse.

## 2 Systemnetze und ihre Abläufe

Die Prozesse eines S/T-Systems beschreiben das mögliche Verhalten eines S/T-Systems. Allerdings kann ein Prozeß gleich am Anfang stehen bleiben. Das ist normalerweise nicht das, was man von einem System erwartet. Wir führen deshalb zwei weitere Konzepte ein, die verhindern, daß ein Prozeß unmotiviert stehen bleibt, und daß bestimmte Transitionen unfair behandelt werden.

... Motivation wird später noch ergänzt.

... Beispiel für Abläufe die die Progress-Eigenschaft für eine Transition verletzen.

**Definition 4.4 (Progressseigenschaft)**

Sei  $\Sigma = (N, W, m_0)$  ein S/T-System mit Netz  $N = (S, T, F)$ , sei  $\pi = (K, \lambda)$  ein Prozess von  $\Sigma$  mit  $K = (B, E, \prec)$  und sei  $t \in T$  eine Transition von  $\Sigma$ .

Der Prozeß erfüllt die *Progressseigenschaft* bezüglich Transition  $t$ , wenn  $t$  am Ende von  $\pi$  nicht aktiviert ist (d. h. wenn nicht gilt  $\lambda(K^\circ) \geq \bar{t}$ ).

Anderenfalls sagen wir, daß der Prozess  $\pi$  die Progressseigenschaft für  $t$  erfüllt.

In manchen Fällen reicht uns die Progressseigenschaft jedoch noch nicht aus, um bestimmte unerwünschte Prozesse auszuschließen. Beispielsweise muß im System für den wechselseitigen Ausschluß der Semaphore fair an beide Agenten vergeben werden, damit beide Agenten den kritischen Bereich betreten

können, wenn sie dies wollen. Wir zeichnen deshalb bestimmte Kanten (im Beispiel die vom Semaphor zu den entsprechenden Transitionen) als fair aus.

*Die Details der folgenden Definition sind nicht so wichtig. Wichtig ist die Idee die dahinter steckt.*

**Definition 4.5 (Fairnesseigenschaft)**

Sei  $\Sigma = (N, W, m_0)$  ein gewöhnliches S/T-System mit Netz  $N = (S, T, F)$ , sei  $\pi = (K, \lambda)$  ein Prozess von  $\Sigma$  mit  $K = (B, E, \leq)$  und sei  $(s, t) \in F$  eine Kante von  $\Sigma$ .

Der Prozeß erfüllt die *Fairnesseigenschaft* bezüglich der Kante  $(s, t)$ , wenn gilt: Sei  $\{s_1, \dots, s_n\} = \bullet t \setminus s$ . Wenn für jeden Zustand  $Q$  von  $K$  ein von  $Q$  erreichbarer Zustand  $Q'$  existiert mit  $\lambda(Q) \geq [s]$ , dann gilt nicht  $\lambda(K^\circ) \geq [s_1, \dots, s_n]$ .

Für welche Transitionen die Progress-Eigenschaft und für welche Kanten die Fairnesseigenschaft gelten soll, müssen wir natürlich im Systemmodell beschreiben. Deshalb erweitern wir S/T-Systeme um diese beiden Konzepte: eine Progressmenge, d. h. die Menge der Transitionen, für die die Progresseigenschaft gelten soll; und eine Menge von fairen Kanten, für die die Fairnesseigenschaft gelten soll. Die derart ergänzten S/T-Systeme nennen wir Systemnetze.

**Definition 4.6 (Systemnetz und ihre Abläufe)**

$\Sigma = (N, W, m_0, P, A)$  heißt *Systemnetz*, wenn

- $(N, W, m_0)$  ein gewöhnliches S/T-System ist mit Netz  $N = (S, T, F)$ ,
- jede Transition von  $N$  einen nicht-leeren Vorbereich und einen nicht-leeren Nachbereich hat,
- $P \subseteq T$  eine Menge von Transitionen ist (die Progressmenge) und wenn
- $A \subseteq F$  eine Menge von Kanten ist (die fairen Kanten).

Ein Prozess  $\pi = (K, \leq)$  von  $(N, W, m_0)$  heißt *Ablauf* von  $\Sigma$ , wenn er für jede Transition  $t \in P$  die Progresseigenschaft erfüllt und für jede Kante  $f \in A$  die Fairnesseigenschaft erfüllt.

Graphisch heben wir in einem Systemnetz nicht die Progresstransitionen hervor, sondern die Transitionen ohne Progress (d. h. die Transitionen aus  $T \setminus P$ ), da es davon meist wesentlich weniger gibt. Die Transitionen ohne Progress

werden entweder grau schattiert dargestellt oder durch ein fettes W (für engl. weak) gekennzeichnet. Die Menge der fairen Kanten kennzeichnen wir entweder durch eine besondere Pfeilspitze (weiß) oder durch rote oder dickere Darstellung der entsprechenden Kanten.

Die Abläufe eines Systemnetzes bilden dann die Grundlage für die Spezifikation des korrekten Verhaltens. Die Menge der zulässigen Abläufe werden wir durch eine einfache temporale Logik beschreiben.

### 3 Spezifikation

Die Menge der Abläufe beschreiben das Verhalten eines Systemnetzes. Das gewünschte Verhalten eines Systemnetzes können wir nun über die Menge der zulässigen Abläufe spezifizieren. Allerdings können wir diese Menge nicht explizit angeben, da ein Systemnetz im allgemeinen unendlich viele Abläufe besitzen kann. Deshalb führen wir in diesem Abschnitt eine einfache Variante der *temporalen Logik* ein, um die zulässigen Abläufe zu charakterisieren.

Zunächst betrachten wir zwei Eigenschaften unseres Mutex-Beispiels. Diese Logik werden wir dann anschließend formalisieren. Die erste wichtige Eigenschaft des Mutex-Beispiels ist der wechselseitige Ausschluß, d. h. es kommt in keinem Ablauf vor, daß sowohl der Agent 1 als auch der Agent 2 im kritischen Bereich ist. Formal notieren wir das wie folgt:

$$\Box \neg (\text{critical}_1 \wedge \text{critical}_2)$$

Die Box<sup>1</sup>  $\Box$  bedeutet, daß in jedem Zustand jedes Ablaufs die auf die Box folgende Formel  $\neg(\text{critical}_1 \wedge \text{critical}_2)$  gilt. Dabei beziehen sich  $\text{critical}_1$  und  $\text{critical}_2$  auf die entsprechende Stelle des Systems. Sie bedeuten, daß die entsprechende Stelle im jeweiligen Zustand (mit mindestens einer Marke) markiert ist. Insgesamt besagt die Formel also „in jedem Zustand sind  $\text{critical}_1$  und  $\text{critical}_2$  nicht (gleichzeitig) markiert“. Derartige Eigenschaften werden *Invarianten* genannt.

*Streng genommen stehen  $\text{critical}_1$  und  $\text{critical}_2$  für die Anzahl der Marken auf den entsprechenden Stellen im jeweiligen Zustand. Wir benutzen  $\text{critical}_1$  und  $\text{critical}_2$  abkürzend für  $\text{critical}_1 \geq 1$  bzw.  $\text{critical}_2 \geq 1$ .*

*Die Mutex-Eigenschaft wird in Petrinetzen oft etwas anders formuliert:  $\Box \text{critical}_1 + \text{critical}_2 \leq 1$ . Diese Formel fordert aber etwas mehr als*

---

<sup>1</sup>Die Box wird „immer gilt“ oder „always“ gelesen.

$\Box\neg(\text{critical}_1 \wedge \text{critical}_2)$ . Denn in  $\Box\neg(\text{critical}_1 \wedge \text{critical}_2)$  wäre es zulässig, daß mehrere Marken auf  $\text{critical}_1$  oder mehrere Marken auf  $\text{critical}_2$  liegen; dies ist in  $\Box\text{critical}_1 + \text{critical}_2 \leq 1$  ausgeschlossen.

Die zweite Eigenschaft besagt, daß jeder Agent, der den kritischen Bereich gerne betreten möchte, den kritischen Bereich auch irgendwann betreten wird. Dabei wird der Wunsch, den kritischen Bereich zu betreten, durch eine Marke auf der Stelle `pending` angezeigt. Die Eigenschaft drücken wir dann durch

$$\text{pending}_1 \triangleright \text{critical}_1 \quad \text{und} \quad \text{pending}_2 \triangleright \text{critical}_2$$

aus<sup>2</sup>. Der Operator  $\triangleright$  wird *Leadsto-Operator* genannt. Er bedeutet, daß in jedem Zustand eines Ablaufes, in dem die Formel auf der linken Seite des Operators gilt, später im Anlauf ein Zustand folgt, in dem die Formel auf der rechten Seite gilt.

Tatsächlich reichen die beiden Operatoren  $\Box$  und  $\triangleright$  in vielen Fällen aus, um das gewünschte Verhalten eines Systemnetzes zu beschreiben – in Kombination mit Zustandsformel, die bestimmte Zustände charakterisieren.

Im folgenden werden wir nun die temporalen Aussagen und deren Bedeutung formalisieren. Dabei gehen wir von Zustandsformeln aus, die auf Markierungen interpretiert werden (die Zustandsformeln und deren Interpretation werden wir hier jedoch nicht explizit ausformulieren, da dies wie gewöhnlich geschieht).

### 3.1 Zustandsformeln

Eine *Zustandsformel* ist aus Ausdrücken aufgebaut, in denen die Stellen eines Systemnetzes als Variablen vorkommen dürfen. Beispielsweise sind `critical1`, `pending1`, `critical1 + critical2` oder `idle1 + pending2 + critical1` Ausdrücke; aber auch 1 oder `(idle1 + 3) * 7` sind Ausdrücke.

Für eine konkrete Markierung  $m$  kann man einen Ausdruck auswerten. Beispielsweise wird für die Markierung  $m = [\text{idle}_1, \text{idle}_1, \text{critical}_2]$  der Ausdruck `idle1 + pending2 + critical1` zu 3 ausgewertet. Wir schreiben dafür auch (unter leichtem Mißbrauch der Notation)  $m(\text{idle}_1 + \text{pending}_2 + \text{critical}_2) = 3$ . Entsprechend gilt  $m((\text{idle}_1 + 3) * 7) = 35$ .

---

<sup>2</sup> Auch hier steht die Bezeichnung einer Stelle  $s$  in einer Formel wieder abkürzend für  $s \geq 1$ .

Aus zwei Ausdrücken können wir eine Formel bilden, indem wir sie mit einer binären Relation verknüpfen. Beispielsweise ist  $\text{critical}_1 + \text{critical}_2 \leq 1$  eine Zustandsformel (sie verknüpft die Ausdrücke  $\text{critical}_1 + \text{critical}_2$  und 1). Weitere Zustandsformeln erhalten wir durch die üblichen booleschen Verknüpfungen von Zustandsformeln. Für eine gegebene Markierung  $m$  können wir die Gültigkeit einer Zustandsformel wie üblich durch Auswerten der Ausdrücke, Auswertung der entsprechenden Relationen und der entsprechenden booleschen Operatoren überprüfen. Beispielsweise gilt in der Markierung  $m = [\text{idle}_1, \text{idle}_1, \text{critical}_2]$  die Zustandsformel  $\text{idle}_1 + \text{pending}_2 + \text{critical}_1 \leq 5$ . Wenn eine Zustandsformel  $\varphi$  in einer Markierung  $m$  gilt, schreiben wir  $m \models \varphi$ .

Um bestimmte Zustandsformeln etwas kompakter zu schreiben, führen wir einige Schreibabkürzungen ein. Wenn eine Stelle  $s$  in einer Zustandsformel an einer Position auftaucht, an der kein Ausdruck, sondern eine Formel erwartet würde, ersetzen wir  $s$  durch  $s \leq 1$ . Für eine Menge von Stellen  $S' = \{s_1, s_2, \dots, s_n\}$  benutzen wir die Menge  $S'$  abkürzend für die Zustandsformel  $s_1 \wedge s_2 \wedge \dots \wedge s_n$ , die abkürzend für  $s_1 \geq 1 \wedge s_2 \geq 1 \wedge \dots \wedge s_n \geq 1$  steht.

Im folgenden ist es gar nicht so wichtig, wie die Zustandsformeln im einzelnen aufgebaut sind. Wenn es uns zweckmäßig erscheint, können wir also weitere Operationen in Ausdrücken zulassen. Das einzige, was wir im folgenden voraussetzen ist, die Definition der Gültigkeit einer Zustandsformel  $\varphi$  in einer Markierung  $m$ : wenn  $\varphi$  in  $m$  gilt schreiben wir  $m \models \varphi$ . Bei der Definition der Gültigkeit einer Zustandsformel in einem Zustand  $Q$  eines Ablaufes  $\pi = (K, \lambda)$  machen wir uns zunutze, daß der Zustand  $Q$  der Markierung  $\lambda(Q)$  des entsprechenden Systems entspricht; wenn  $\lambda(Q) \models \varphi$  gilt, sagen wir, daß  $\varphi$  in Zustand  $Q$  gilt und wir schreiben dafür auch  $Q \models \varphi$ .

## 3.2 Temporale Formeln

Aus den Zustandsformeln können wir nun die *temporale Formeln* aufbauen. Syntaktisch ist das sehr einfach: für eine Zustandsformel  $\varphi$  ist  $\Box\varphi$  eine temporale Formel. Wir sagen  $\Box\varphi$  ist eine Invarianten-Formel. Für zwei Zustandsformeln  $\varphi$  und  $\psi$  ist  $\varphi \triangleright \psi$  eine temporale Formel. Wir sagen  $\varphi \triangleright \psi$  ist eine Leadsto-Formel.

Wir werden nun definieren, wann eine temporale Formel in einem Ablauf  $\pi = (K, \lambda)$  eines Systemnetzes gilt. Wir beginnen mit der Gültigkeit der

Invarianten-Formeln. Die Formel  $\Box\varphi$  gilt im Ablauf  $\pi$ , wenn für jeden Zustand  $Q$  des Ablaufs die Zustandsformel  $\varphi$  gilt, d. h. wenn in jedem Zustand  $Q$  von  $K$  gilt  $Q \models \varphi$ . Wir schreiben dann  $\pi \models \Box\varphi$ .

Die Formel  $\varphi \triangleright \psi$  gilt im Ablauf  $\pi$ , wenn für jeden Zustand  $Q$  von  $\pi$ , der  $\varphi$  erfüllt in  $\pi$  ein Zustand  $Q'$  erreichbar ist, der  $\psi$  erfüllt. D. h. wenn für jeden Zustand  $Q$  von  $K$  mit  $Q \models \varphi$  ein Zustand  $Q'$  von  $K$  mit  $Q \xrightarrow{*} Q'$  (in  $K$ ) und  $Q' \models \psi$  existiert. Wir schreiben dann  $\pi \models \varphi \triangleright \psi$ .

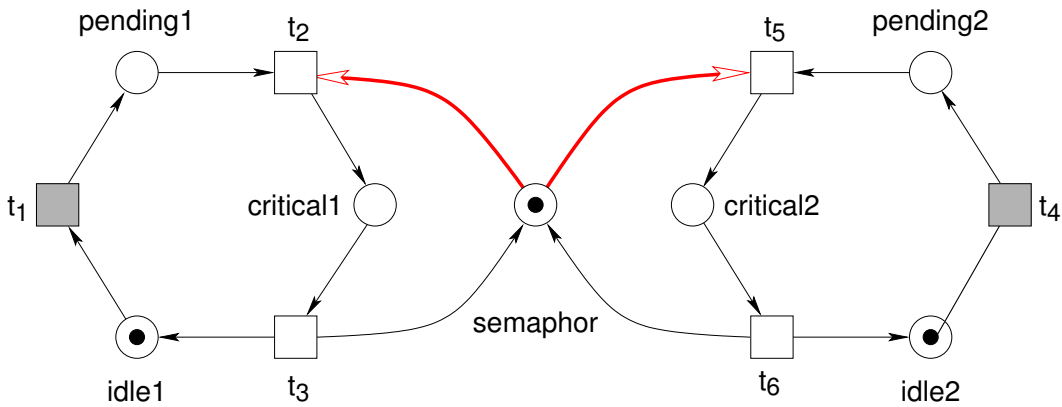
Eine Systemnetz  $\Sigma$  erfüllt eine temporale Formel, wenn jeder Ablauf von  $\Sigma$  die temporale Formel erfüllt. Wir schreiben dann  $\Sigma \models \Box\varphi$  bzw.  $\Sigma \models \varphi \triangleright \psi$ .

*Eine Invarianten-Formel  $\Box\varphi$  gilt in  $\Sigma$  genau dann, wenn für jede in  $\Sigma$  erreichbare Markierung  $m$  gilt:  $m \models \varphi$ . Das kann man sich leicht überlegen – ein formaler Beweis ist jedoch technisch sehr aufwendig, weil er implizit einen Beweis enthält, daß wir den Ablaufbegriff sinnvoll gebildet haben (daß also insbesondere die in allen Abläufen erreichbaren Zustände gerade den erreichbaren Markierungen des Systems entsprechen). Darin gehen die Annahmen für Kausalnetze und die Beschriftungen von Kausalnetzen ein.*

## 4 Verifikation

Nun ist die Frage, wie man sich davon überzeugt, daß ein Systemnetz die spezifizierten Eigenschaften erfüllt. Für unser Mutex-Beispiel aus Abb. 4.1 sieht man die Gültigkeit der Eigenschaft zwar fast auf einen Blick. Für etwas komplexere Systeme ist das meist nicht mehr so offensichtlich. Im folgenden geben wir deshalb einige Regeln zu Verifikation von Eigenschaften an, die rein syntaktisch überprüft werden können.

Zunächst betrachten wir den Beweis für das Mutex-Beispiel. Die einzelnen Regeln und die „Lesart“ der Beweise werden wir danach erläutern. Im Beweis kürzen wir die Stellennamen der Übersichtlichkeit halber ab.



$$\square \neg (\text{critical}_1 \wedge \text{critical}_2)$$

$$\text{pending}_1 \triangleright \text{critical}_1$$

$$\text{pending}_2 \triangleright \text{critical}_2$$

Abbildung 4.1: Das Mutex-System und seine Eigenschaften

- |      |   |                                 |
|------|---|---------------------------------|
| (1)  | $\square c_1 + s + c_2 = 1$                     | S-Invariante                    |
| (2)  | $\square \neg (c_1 \wedge c_2)$                 | Abschwächung (1)                |
| (3') | $\square c_1 \vee s \vee c_2$                   | Abschwächung (1)                |
| (3)  | $\square p_1 \Rightarrow (c_1 \vee s \vee c_2)$ | Abschwächung (1)                |
| (4)  | $c_1 \triangleright s$                          | Progress mit $t_3$              |
| (5)  | $c_2 \triangleright s$                          | Progress mit $t_6$              |
| (6)  | $p_1 \triangleright s$                          | Beweisgraph A mit (3), (4), (5) |
| (7)  | $p_1 \triangleright c_1$                        | Fairness mit $(s, t_2)$ und (6) |

Abbildung 4.2 zeigt den im Beweis benutzen Beweisgraph A.

Der Beweis ist aus Zeilen aufgebaut. Jede Zeile beginnt mit einer Nummer. Dann folgt die in dieser Zeile bewiesene Eigenschaft. Am Ende der Zeile folgt das Argument für die bewiesene Eigenschaft. Die Zeilen (2) und (7) beweisen die gewünschten Eigenschaften des Systems; die Eigenschaft  $\text{pending}_2 \triangleright \text{critical}_2$  beweisen wir nicht, da sie symmetrisch zu (7) bewiesen werden kann. Wir gehen nun zeilenweise durch den Beweis und erläutern die Argumente:

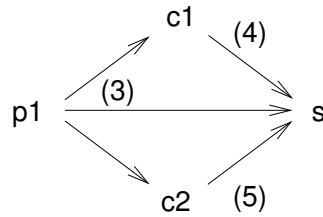


Abbildung 4.2: Das Mutex-System und seine Eigenschaften

- (1) Diese Eigenschaft folgt unmittelbar aus der S-Invarianten des Systems. Die linke Seite der Gleichung ist die S-Invarianten (notiert als symbolische Summe); die rechte Seite der Gleichung ist der Wert der S-Invarianten in der Anfangsmarkierung. Die Gültigkeit dieser Aussage können wir leicht nachrechnen (mit den Techniken aus Kapitel 2 Abschnitt 4.3).
- (2) ist eine Abschwächung der Aussage (1), denn man kann sich leicht überlegen, daß die Implikation  $c_1 + s + c_2 = 1 \Rightarrow \neg(c_1 \wedge c_2)$  gilt (unter der Annahme, daß Stellen nur positiv markiert sein können).
- (3'), (3) folgen beide ebenfalls unmittelbar aus der Aussage (1) durch Abschwächung. Wenn wir wissen, daß immer genau eine Marke auf einer der Stellen  $c_1$ ,  $s$  oder  $c_2$  liegt, dann ist immer mindestens eine der Stellen markiert (3'). In (3) wird diese Aussage mit einer weiteren, zunächst scheinbar unnötigen, Voraussetzung versehen (wir benötigen die Aussage in dieser Form im Beweisgraph A).
- (4) Die Leadsto-Aussage folgt unmittelbar daraus, daß Transition  $t_3$  eine Progress-Transition ist, die zu keiner anderen Transition in Konflikt steht. Wenn also mind. eine Marke auf der Stelle  $c_1$  liegt, muß die Transition im Ablauf vorkommen (sonst würde der entsprechende Prozess die Progresseigenschaft verletzen). Dann produziert das Schalten der Transition eine Marke auf Stelle  $s$  (es produziert auch eine Marke auf  $i_1$ , die benötigen wir aber im weiteren Beweis nicht; deshalb lassen wir sie weg).
- (5) Diese Aussage gilt mit denselben Argumenten wie (4) jedoch für die Progress-Transition  $t_6$ .

- (6) Der Beweisgraph faßt die Aussagen aus (3), (4) und (5) zur Leadsto-Aussage  $p_1 \triangleright s$  zusammen. Wegen (3) gilt  $p_1 \triangleright (c_1 \vee s \vee c_2)$ . Wenn also eine Marke auf  $p_1$  liegt, wissen wir, daß irgendwann eine Marke auf  $c_1$  oder  $s$  oder  $c_2$  liegen wird. Wenn die Marke auf  $s$  liegt, sind wir fertig. Wenn sie auf  $c_1$  oder  $c_2$  liegt, besagen (3) bzw. (4), daß irgendwann eine Marke auf  $s$  gelangen wird. Dies ist kompakt im Beweisgraph A (mit Referenzen auf (3), (4) und (5)) zusammengefaßt.
- (7) Wegen (6) wissen wir nun, daß immer wieder eine Marke auf Stelle  $s$  kommt, solange Stelle  $p_1$  markiert beliebt – also  $t_2$  nicht schaltet. Wegen der Transition  $t_2$  mit der fairen Kante  $(s, t_2)$  muß deswegen irgendwann Transition  $t_2$  schalten; das Schalten markiert die Stelle  $c_1$ . Damit ist die Eigenschaft  $p_1 \triangleright c_1$  bewiesen.

Im folgenden geben wir nun einige Beweisregeln mit präzisen und syntaktisch überprüfbareren Voraussetzungen an, die für eine große Anzahl von Beispielen zum Nachweis der Korrektheit ausreichen.

*Es gibt noch viele weitere Regeln. Hier geht es uns lediglich ums Prinzip. Deshalb verzichten wir darauf, einen vollständigen Satz von Regeln anzugeben.*

*Tatsächlich könnte man zumindest für beschränkte Systeme die Technik des Modelcheckings einsetzen, um die Eigenschaften vollautomatisch zu verifizieren. Das funktioniert jedoch nicht mehr für Systeme mit unendlichem Zustandsraum, die wir im nächsten Abschnitt betrachten. In diesen Fällen sind dann die hier vorgestellten Beweistechniken nötig, da das Modelchecking dann meist nicht mehr funktioniert. Allerdings lassen sich die hier vorgestellten Beweistechniken auch teilweise automatisieren.*

**Beweisregeln für Invarianten** Wir beginnen mit einigen Beweisregeln für Invarianten. Dabei greifen wir auf die bereits bekannte Technik der S-Invarianten und der Fallen zurück:

**S-Invariante** Wenn  $\underline{i} = s_1 + \dots + s_n$  eine S-Invariante ist und für die initiale Markierung  $m_0$  gilt  $m_0 \cdot \underline{i} = k$ , dann gilt  $\Box s_1 + \dots + s_n = k$ .

Die Voraussetzungen der Regel lassen sich für eine gegebene Eigenschaft  $\Box s_1 + \dots + s_n = k$  und ein gegebenes System vollautomatisch überprüfen.

**Trap** Wenn  $\{s_1, \dots, s_n\}$  eine initial markierte Falle des Systems ist, dann gilt  $\Box s_1 + \dots + s_n \geq 1$ .

Wieder lassen sich die Voraussetzungen der Regel vollautomatisch überprüfen.

**Abschwächung** Wenn  $\Box\varphi_1, \dots, \Box\varphi_n$  bereits bewiesene Invarianten des Systems sind und  $(\varphi_1 \wedge \dots \wedge \varphi_n) \Rightarrow \psi$  eine allgemeingültige Zustandsaussage ist, dann gilt im System auch die Aussage  $\Box\psi$ .

In einem Beweis wird die Gültigkeit einer Aussage  $\Box\varphi_i$  durch einen Verweis auf eine vorangegangene Beweiszeile, in der  $\Box\varphi_i$  bewiesen wurde gewährleistet. Zur Überprüfung der Anwendbarkeit der Regel ist also nur noch die Gültigkeit der Implikation  $(\varphi_1 \wedge \dots \wedge \varphi_n) \Rightarrow \psi$  zu gewährleisten. Dazu kann man beispielsweise automatische Theorembeweiser einsetzen.

*Regeln werden oft notiert, indem man die Voraussetzungen über einem vertikalen Strich und die Schlußfolgerung unter den Strich schreibt. Die Regel der Abschwächung kann man dann wie folgt notieren:*

$$\frac{\Box\varphi_1, \dots, \Box\varphi_n, \varphi_1 \wedge \dots \wedge \varphi_n \Rightarrow \psi}{\Box\psi}$$

**Beweisregeln für Leadsto-Eigenschaften** In diesem Abschnitt geben wir nun Beweisregeln für Leadsto-Eigenschaften an:

**Reflexivität** Wenn in einem System  $\Box\varphi \Rightarrow \psi$  gilt, dann gilt auch  $\varphi \triangleright \psi$  (denn in jedem Zustand, in dem  $\varphi$  gilt, gilt insbesondere  $\psi$ ; der Zustand, in dem  $\psi$  gilt, ist also trivialerweise erreichbar).

**Progress** Sei  $t \in P$  eine Progress-Transition des Systems, sei  $S'$  eine Menge von Stellen mit  $\bullet t \subseteq S'$  und sei  $T'$  eine Menge von Transitionen, so daß für jedes  $t' \in T'$  gilt  $\Box S' \Rightarrow \neg \bullet t'$ , dann gilt  $S' \triangleright \bigvee_{\hat{t} \in S' \setminus T'} (S' \setminus \bullet \hat{t} \cup \hat{t} \bullet)$ .

Der Regel liegt die folgende Idee zugrunde: Die Transitionen aus  $T'$  sind nicht aktiviert, wenn  $S'$  gilt. So lange  $S'$  gilt, können diese Transitionen also nicht schalten. Weil  $t$  eine Transition aus der Progressmenge ist und weil  $\bullet t \subseteq S'$  gilt, ist  $t$  aktiviert, wenn  $S'$  gilt. Deshalb muß  $t$  oder eine andere Transition  $\hat{t}$  im Nachbereich von  $S'$  irgendwann schalten, was zu einem der Zustände  $S' \setminus \bullet \hat{t} \cup \hat{t} \bullet$  führt.

Als Argument geben wir im Beweis die Transition  $t$  und die Menge  $T'$  (der nicht aktivierten Transitionen) an. Wenn  $T'$  die leere Menge ist, geben wir  $T'$  nicht explizit an – wie im vorangegangenen Beweis.

Zusätzlich geben wir die Beweiszeilen an, in denen die Gültigkeit der Vorbedingungen  $\Box S' \Rightarrow \neg \bullet t'$  für alle  $t' \in T'$  gezeigt wurde.

**Fairness** Sei  $(s, t) \in A$  eine faire Kante des Systemnetzes und Sei  $S' = \bullet t \setminus \{s\}$  und  $S'^{\bullet} = \{t\}$  (d.h.  $t$  steht über die Stellen  $S'$  mit keiner anderen Transition in Konflikt; ein Konflikt über  $s$  wird jedoch nicht ausgeschlossen). Wenn im Systemnetz  $S' \triangleright s$  gilt, dann gilt auch  $S' \triangleright t^{\bullet}$ .

Dieser Regel liegt die folgende Idee zugrunde: Wenn  $S'$  gilt, sind alle Stellen im Vorbereich von  $t$  außer  $s$  markiert. Wegen  $S' \triangleright s$  wird auch  $s$  immer wieder markiert, solange  $S'$  gilt. Wegen der Fairnessannahme muß also irgendwann die Transition  $t$  schalten, was zu  $t^{\bullet}$  führt.

Als Srgumente geben wir im Beweis die faire Kante und die Beweiszeile an, in der die Eigenschaft  $S' \triangleright s$  bereits gezeigt wurde.

**Beweisgraph** Ein Beweisgraph ist ein zyklenfreier Graph, dessen Knoten Zustandsformeln sind. Der Beweisgraph besitzt einen ausgezeichneten Anfangsknoten  $\varphi_0$  und einen ausgezeichneten Endknoten  $\varphi$ . Wenn für jede Verzweigung des Graphen von einem Knoten  $\varphi'$  zu den Knoten  $\varphi'_1, \dots, \varphi'_n$  die Eigenschaft  $\Box \varphi' \Rightarrow (\varphi'_1 \vee \dots \vee \varphi'_n)$  oder  $\varphi' \triangleright \varphi'_1 \vee \dots \vee \varphi'_n$  gilt, dann gilt auch die Eigenschaft  $\varphi_0 \triangleright \varphi$ .

Als Beweisargument geben wir den Beweisgraphen selbst an, wobei für jede Verzweigung des Graphen ein Verweis auf eine Beweiszeile angegeben ist, in der  $\Box \varphi' \Rightarrow (\varphi'_1 \vee \dots \vee \varphi'_n)$  oder  $\varphi' \triangleright \varphi'_1 \vee \dots \vee \varphi'_n$  bereits bewiesen wurde.

Dies sind alle Regeln, die wir zum Beweis unserer Beispiele benötigen. Wir geben jedoch noch zwei Regeln an, um Beweise abzukürzen. Sie lassen sich jedoch einfach mit Hilfe von Beweisgraphen ausdrücken:

**Transitivität**

$$\frac{\varphi_0 \triangleright \varphi_1, \varphi_1 \triangleright \varphi_2}{\varphi_0 \triangleright \varphi_2}$$

**Disjunktivität**

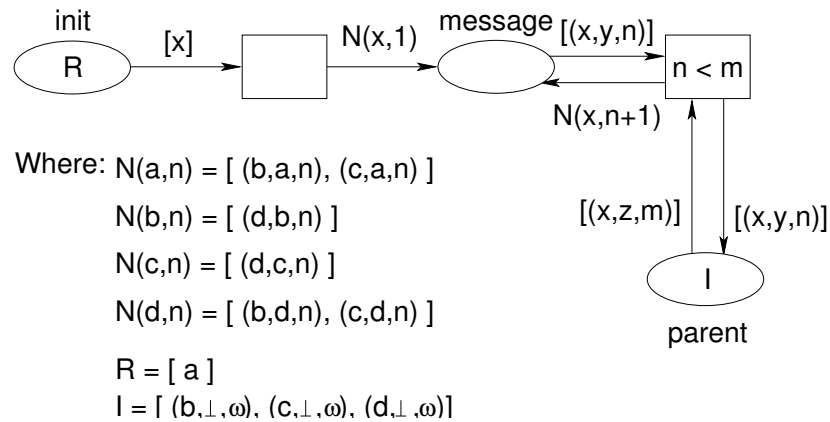
$$\frac{\varphi_0 \triangleright \varphi_1, \varphi_2 \triangleright \varphi_3}{\varphi_0 \vee \varphi_2 \triangleright \varphi_1 \vee \varphi_3}$$

## 5 Verteilte Algorithmen: Einige Beispiele

Zuletzt stellen wir Petrinetzmodelle einiger verteilter Algorithmen vor. Hier benutzen wir eine spezielle Variante von Netzen mit strukturierten Marken: algebraische Petrinetze bzw. algebraische Petrinetzschemata.

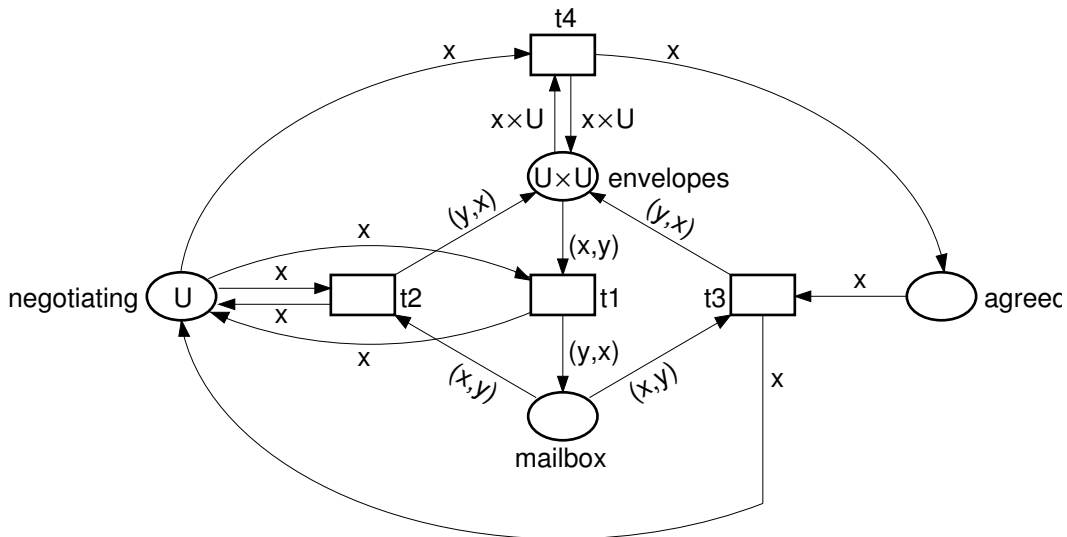
*Hier geben wir nur die Petrinetzmodelle an; eine ausführliche Diskussion der Modelle und der zugrundeliegenden Modellierungsphilosophie findet sich in zwei separaten Arbeiten, die auf der Web-Seite zur Verfügung gestellt werden [9, 4]. Wir geben jeweils die Arbeit und die relevanten Seiten an, wo das Beispiel diskutiert wird.*

*Irgendwann wird der Text hier auch entsprechend ergänzt.*



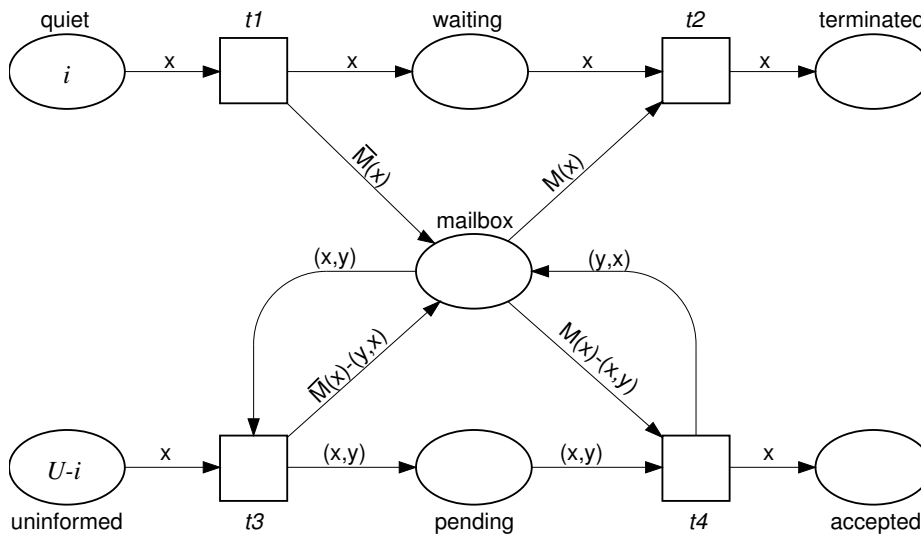
Erklärung des Beispiels: [4] S. 2–5.

Abbildung 4.3: Ein Algorithmus zur Bestimmung des minimalen Abstandes zu einem Wurzelknoten



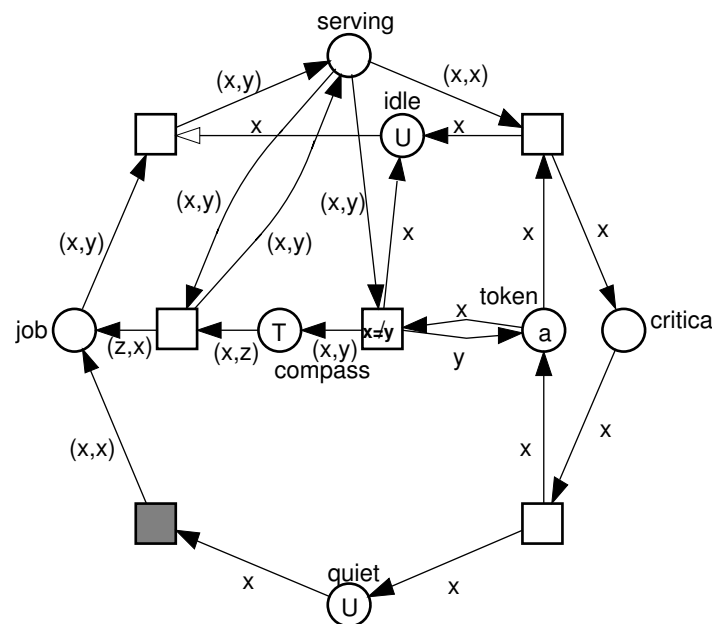
Erklärung des Beispiels: [9] S. 5-9.

Abbildung 4.4: Ein Verhandlungsprotokoll



Erklärung des Beispiels: [9] S. 39-40.

Abbildung 4.5: Der Echo-Algorithmus



Erklärung des Beispiels: [9] S. 62-65.

Abbildung 4.6: Mutex in einem Netzwerk von Agenten

Teil C

Literatur und Index



# Literaturverzeichnis

- [1] BAUMGARTEN, BERND: *Petri-Netze: Grundlagen und Anwendungen*. Spektrum Akademischer Verlag, 2. Auflage, 1996.
- [2] DESEL, JÖRG: *Petrinetze, lineare Algebra und lineare Programmierung*. Nummer 26 in *Teubner-Texte zur Informatik*. Teubner, 1998. 9
- [3] DESEL, JÖRG und JAVIER ESPARZA: *Free Choice Petri Nets*. Cambridge University Press, 1995.
- [4] KINDLER, EKKART: *DAWN for component based systems - Just a different view*. <http://www.upb.de/cs/kindler/Publikationen/copies/components02.pdf>, Juni 2002. 5
- [5] PETERSON, JAMES L.: *Petri Net Theory and the Modeling of Systems*. Prentice-Hall, 1981.
- [6] REISIG, WOLFGANG: *Petrinetze: Eine Einführung*. Studienreihe Informatik. Springer-Verlag, 2 Auflage, 1982.
- [7] REISIG, WOLFGANG: *Elements of Distributed Algorithms — Modeling and Analysis with Petri Nets*. Springer, 1998.
- [8] STARKE, PETER: *Analyse von Petri-Netz-Modellen*. Teubner, 1990.
- [9] WEBER, M., R. WALTER, H. VÖLZER, T. VESPER, W. REISIG, S. PEUKER, E. KINDLER, J. FREIHEIT und J. DESEL: *DAWN: Petrinetzmodelle zur Verifikation Verteilter Algorithmen*. Informatik-Bericht 88, Humboldt-Universität zu Berlin, Dezember 1997. 5, 5, 5, 5

# Index

- Überdeckung, 28
- Überdeckungsbaum, 20, 24
- Übergangsrelation, 17
  
- A-Vektor, 32
- $A \times B$ -Matrix, 33
- Ablauf eines Systemnetzes, 84
- Agent, 69
- aktiviert, 6, 17
- Aktivität, 68
- Algebra
  - lineare, 32
- Anfang eines Kausalnetzes, 82
- Anfangsmarkierung, 5, 15
- Audruck, 86
  
- B/E-Systeme, 53
- Bedingung, 59, 81
- Bedingungs-Ereignis-Systeme, *sie-*  
*he* B/E-Systeme
- Belegung, 62
- beschränkt, 19, 29
- Beweisregeln, 88, 91, 92
  
- co-Falle, *siehe* Deadlock
- Coloured Petri Net, 61
- Commoner's Theorem, 51
  
- Deadlock, 46
- Deadlock-Trap-Eigenschaft, 50
- Dokument, 69
  
- Eigenschaftskatalog, 29
- Einzelstapeltregel, 57
- Element, 12
- Elementare Netzsysteme, *sie-*  
*he* EN-Systeme
- EN-Systeme, 53
- Ende eines Kausalnetzes, 82
- Ereignis, 59, 81
- erreichbare Markierung, 17
- Erreichbarkeitsgraph, 8
- erweitertes Workflow-Netz, *sie-*  
*he* Workflow-Netz
- erweitertes Workflow-System, *sie-*  
*he* Workflow-System
  
- Fairness, 84
- Falle, *siehe* Trap
- Flußrelation, 12
- Folgemarkierung, 17
- Free-choice-System, 45
  
- Gültigkeit
  - einer temporalen Formel, 88
  - einer Zustandsformel, 87
- Geschäftsprozeß, 67
- gewöhnliches S/T-System, 15
- gewichtete Markierung, 36
  
- Halbordnungssemantik, 58
  
- Inhibitorkante, 53
- Invariante, 85

- Inzidenzmatrix, 34
- isoliertes Element, 14
- Jahreszeiten
  - die vier, 4
- Kante, 4, 12
- Kantengewicht, 15
- Kapazität, 16, 51
- Kausalität, 82
- Kausalnetz, 81
- Konfliktbereich, 43
- Konfusion, 60
- konstruierbares Workflow-Netz, 77
- Leadto-Eigenschaft, 86
- lebendig, 29
- Lineare Algebra, 32
- lineares Gleichungssystems, 34
- Marke, 5
- Markierung, 5, 14
  - Addition, 14
  - erreichbare, 17
  - gewichtete, 36
  - Inklusion, 14
  - verallgemeinerte, 23
- Markierungsgleichung, *siehe* Zustandsgleichung
- Matrix, 33
  - transponierte, 33
- maximale Schrittschaltregel, 57
- Modus, 62
- $n$ -sicher, 29
- Nachbereich, 6, 13
- nebenläufig, 58, 82
- Netz, 12
  - mit strukturierten Marken, 61
  - schlichtes, 14
  - Stochastisches, 55
- Nullvektor, 32
- Parikh-Vektor, 33
- Peterson's Mutex-Algorithmus, 48
- Petrinetz, *siehe* Netz
- positive S-Invariante, 40
- Produkt
  - mit einem Skalar, 32
  - Skalar-, 33
- Progress, 83
- Prozes, 83
- Prozess, 58
- Rangbedingung, *siehe* Rangtheorem
- Rangtheorem, 44, 45
  - für Free-choice-Systeme, 45
- Ressource, 69
- S-Invariante, 36
  - positive, 40
  - strikt positive, 40
- S/T-System, 11, 15
  - beschränktes, 19
  - Free-choice-, 45
  - gewöhnliches, 15
  - lebendiges, 29
  - mit Kapazitäten, 51
  - $n$ -sicheres, 29
  - strukturell beschränktes, 29
  - strukturell lebendiges, 29
  - verklemmungsfreies, 29
  - von positiven S-Invarianten überdecktes, 40
- Schalten, 4
- Schaltregel, 17
- Schaltschritt, 56

- Schaltverhalten, 6
- Schaltwort, 17
- schlicht, 14
- Schlinge, 14
- Schrittschaltregel, 56
- Sieb des Eratosthenes, 63
- simultan unbeschränkt, 28, 29
- Siphon, *siehe* Deadlock
- Skalarprodukt, 32, 33
- Spezifikation, 85
- Stelle, 4, 12
  - beschränkte, 29
  - $n$ -sichere, 29
- Stellen-Invariante, *siehe* S-Invariante
- Stellen/Transitions-System, *siehe* S/T-System
- Stochastisches Petrinetz, 55
- strikt positive S-Invariante, 40
- strukturell beschränkt, 29
- strukturell lebendig, 29
- support, *siehe* Trägermenge
- Systemnetz, 84
  
- T-Invariante, 41, 42
- temporale Formel, 87
- temporale Logik, 85
- tot, 29
- Trägermenge, 33
- Transition, 4, 12
  - lebendige, 29
  - tote, 29
- Transitions-Invariante, *siehe* T-Invariante, 42
- Transitionsvorbereich, 43
- transponierte Matrix, 33
- Trap, 46
  
- Unbeschränktheit
  - simultane, 28
  
- Vektor, 32
- Vektoraddition, 32
- verallgemeinerte Markierung, 23
- verklemmungsfrei, 29
- vernünftig, *siehe* Workflow-Netz
- vier Jahreszeiten, 4
- Vorbereich, 6, 13
  
- Workflow, 70
- Workflow-Konstrukt, 75
- Workflow-Netz, 71
  - erweitertes, 75
  - konstruierbares, 77
  - vernünftiges, 73
- Workflow-System, 71
  - erweitertes, 75
  
- Zeit-Petrinetze, 55
- Zustand eines Kausalnetzes, 82
- Zustandsformel, 86
- Zustandsgleichung, 36