KfK 4636
Januar 1990

Eine Einführung in die Grundlagen der Theorie der Höheren Petri-Netze

W. Korczynski, C. Düpmeier, W. Süss
Institut für Datenverarbeitung in der Technik

Kernforschungszentrum Karlsruhe

KERNFORSCHUNGSZENTRUM KARLSRUHE

Institut für Datenverarbeitung in der Technik

KfK 4636

Eine Einführung in die Grundlagen der Theorie der Höheren Petri-Netze

W. Korczynski *)

C. Düpmeier

W. Süss

*)Universität Warschau, Fil. Bialystok

KERNFORSCHUNGSZENTRUM KARLSRUHE GMBH, KARLSRUHE

Kernforschungszentrum Karlsruhe GmbH
Postfach 3640, 7500 Karlsruhe 1

ISSN 0303-4003

An Introduction to the Foundations of the Theory of High Level Petri Nets

ABSTRACT

This paper gives an introduction to the theory of High Level Petri Nets. Beginning with the definition of languages as sets of words over an alphabet, we will develop the concepts of first order formal languages. We will define a formal language over a set of sorts, which can be viewed as set of names for different data types of variables. Therefore we will introduce the concept of algebraic systems in chapter 2.

A High Level Petri Net is first defined in a pure syntactical way over a signature, which defines a formal language of the first order for the annotation of the net. An interpretation of the High Level Petri Net will then define the semantics of the net within a corresponding algebraic system. This approach allows us to connect the theory of ordinary Petri Nets with the theory of Algebraic Specification. Similar approaches can be found in papers of Vautherin ([Vau87a, Mem87a]) and Reisig ([Rei89a]).

The theory of processes and the reachability theory can then be developed similar to the corresponding theories for ordinary Place/Transition Nets. This will be done in the chapters, which will follow the chapter with the basic definitions.

Einführung in die Grundlagen der Theorie der Höheren Petri-Netze

ZUSAMMENFASSUNG

Dieser Bericht gibt eine Einführung in die Theorie der Höheren Petri-Netze. Beginnend mit der Definition einer Sprache als Menge von Wörtern werden die notwendigen Definitionen aus dem Gebiet der abstrakten Sprachen 1. Ordnung vorgestellt. Wir gehen hierbei von einer Definition aus, die eine logische Sprache über einer Menge verschiedener Variablenbereiche, die wir als Variablentypen interpretieren, definiert. Als Grundlage für eine solche Definition führen wir im zweiten Kapitel die algebraischen Systeme ein, die man als Formalisierung des aus der Informatik bekannten Begriffs des abstrakten Datentyps ansehen kann.

Im folgenden Kapitel werden dann Höhere Petri-Netze über Signaturen solcher algebraischer Systeme auf rein syntaktischer Basis eingeführt und in einem zu der Signatur passenden algebraischen System bzgl. ihrer Semantik interpretiert. Dies ist ein Ansatz, der eine "Verheiratung" der Theorie der Algebraischen Spezifikation mit der Petri-Netz-Theorie erlaubt. Wir folgen dabei einer Denkweise, wie man sie in [Vau87a, Mem87a, Kra85a, Rei85a] oder auch [Rei89a] findet.

Die Theorie der Prozesse und die Erreichbarkeitstheorie lassen sich dann analog zum klassischen Fall entwickeln. Ihre Beschreibung folgt im Anschluß an das Kapitel mit den grundlegenden Definitionen der Höheren Petri-Netze.

VORWORT

In den letzten Jahren konnte man eine stürmische Entwicklung von Theorien zur Beschreibung nebenläufiger Systeme beobachten. Zuerst motiviert durch konkrete Fragen innerhalb der Informatik, finden diese Theorien immer mehr Anwendungen auch in anderen Gebieten wie z.B. der allgemeinen Systemtheorie. Die Theorie der Petri-Netze gehört bereits heute zu den klassischen Vertretern solcher Theorien.

Petri-Netze werden dabei als Modellierungshilfsmittel für nebenläufige Systeme verwendet, wobei allerdings in der Regel elementare Netztypen (P/T-Netze) verwendet werden, und in den seltensten Fällen die Modelle rechnergestützt analysiert werden. Am Institut für Datenverarbeitung in der Technik (IDT) des Kernforschungszentrums Karlsruhe (KFK) wird seit einiger Zeit ein Tool zur rechnergestützten Modellierung mit Petri-Netzen entwickelt, das auf einer höheren Form von Netzen, den PrT-Netzen basiert. Diese Form von Petri-Netzen, die von Genrich und Lautenbach ([Gen79a]) eingeführt wurden, verwenden für die Beschriftung des Netzes Bausteine einer logischen Sprache 1. Ordnung. Obwohl man PrT-Netze ohne große Kenntnisse der mathematischen Logik und abstrakten Algebra anwenden kann, basiert ihre Theorie stark auf diesen mathematischen Grundlagengebieten. Da man in den meisten Fachberichten über PrT-Netze Kenntnisse der mathematischen Grundlagen voraussetzt, findet man in der Fachliteratur keine Berichte, die diese für das theoretische Verständnis der PrT-Netze so wichtigen Grundbegriffe in der Weise bereitstellt, wie sie für die Netztheorie benötigt werden.

Dieser Bericht soll das für das Verständis der Theorie der PrT-Netze nötige mathematische Grundwerkzeug bereitstellen und an Beispielen die Verwendung der grundlegenden mathematischen Konstrukte in PrT-Netzen aufzeigen. Dabei sollte dieser Bericht keineswegs ein Handbuch sein, das allumfassend ist. Die grundlegenden Begriffe werden vielmehr kurz und knapp zusammengestellt und auf die Angabe von Beweisen wurde ganz verzichtet. Ein jedes Kapitel abschließender Abschnitt mit Literaturangaben gibt die Arbeiten an, in denen man genauere Bechreibungen der betreffenden Thematik finden kann. Man kann diesen Bericht daher als einen Reiseführer durch die Theorie der Höheren Petri-Netze auffassen, der die wesentlichen Teilaspekte kurz und knapp darstellt und zur weiteren Besichtigung weiterführende Literatur zitiert.

INHALTSVERZEICHNIS

ABBILDUNGEN UND TABELLEN

1 Einleitung und Motivation

Die Modellierung und Simulation von nebenläufigen Systemen wurde in den letzten Jahren immer wichtiger. Aus diesem Grunde entstanden immer neuere Modellierungs- und Simulationstools, welche die Analyse nebenläufiger Systeme erleichtern sollen. Diese Tools basieren auf verschiedenen mathematischen Modellen der Nebenläufigkeit. Eine wichtige Art solcher Modelle sind Petri-Netze. Dabei ist die Theorie der Petri-Netze noch nicht sehr alt, sondern begann in den sechziger Jahren mit einer Arbeit von C. A. Petri. Seit dieser Zeit kann man, besonders in den siebziger Jahren, eine stürmische Entwicklung dieser Theorie beobachten, die schon zu einigen wohldefinierten Teilgebieten der Theorie führte. Einige der wichtigsten sind:

- Die Theorie der sogenannten C/E (Condition - Event) Netze ist der grundlegendste Teil der Theorie. Die ersten Arbeiten über Petri-Netze gehören dieser Richtung an.
- Eine Verallgemeinerung der C/E-Netze sind die Stellen/Transitions - Netze.
- Die Theorie der lose gekoppelten Systeme kann man als eine Art Abhängigkeitstheorie betrachten.
- Die Suche nach immer höheren und komfortableren Beschreibungsmitteln für Bestandteile von Petri-Netzen führte zu einer Reihe von Netzarten, die man unter dem Stichwort *Höhere Petrinetze* zusammenfaßt. Einige dieser höheren Netzarten sind:
  - Prädikat/Transitions-Netze,
  - Relationen-Netze,
  - Produkt-Netze und
  - Coloured-Petri-Netze.

Im Institut für Datenverarbeitung in der Technik (IDT) des Kernforschungszentrums Karlsruhe (KfK) wird seit 3 Jahren ein Modellierungstool (**PROVER**) für komplexe Softwaresysteme mit einem hohen Grad von Nebenläufigkeit entwickelt, das auf Prädikat/Transitions-Netzen basiert. Diese Arbeit soll die grundlegenden mathematischen Begriffe und Sätze aus den Gebieten der universellen Algebra, Logik und Theorie der Petri-Netze bereitstellen, die dem am IDT entwickelten Werkzeug **PROVER** zugrunde liegen. Bevor wir damit beginnen, wollen wir uns zunächst einmal an einem Beispiel in kleinen Schritten vergegenwärtigen, welche mathematischen Hilfsmittel wir benötigen.

Als einführendes Beispiel wählen wir ein typisches Problem aus dem Bereich der Betriebssystemtheorie und/oder der Programmierung von Realzeitsystemen.
Ein modernes Timesharing System besteht aus einer Vielzahl selbstständiger Abläufe, sogenannte Prozesse oder auch Tasks, die auf verschiedenen Arten und Weisen miteinander kooperieren müssen, um gemeinsame Resourcen, wie z.B. Speicher, Drucker und Bandgeräte, zu nutzen, ohne sich gegenseitig zu stören oder die Konsistenz von Daten des Systems zu beeinträchtigen. Das Betriebssystem stellt hierzu eine Reihe von Mechanismen bereit, die es erlauben, daß sich Prozesse an einer einzelnen Resource serialisieren und daß Zugriffe auf Resourcen, die zum Schutz vor Korruption zu einem Zeitpunkt nur von einem Prozess belegt werden dürfen, unter gegenseitigem Ausschluß stattfinden. Häufig verwendet man weiter Mechanismen zur Synchronisation der Abläufe von Prozessen untereinander, die auf den Austausch von Nachrichten unter den Prozessen basieren. Man spricht dann von Interprozeßkommunikation. Wir wollen uns diese Interprozeßkommunikationsmechanismen und ihre Anwendung einmal in einigen Beispielen genauer ansehen.

Der Einfachheit halber betrachten wir nur zwei Prozesse A und B, die untereinander Nachrichten austauschen, wie dies das folgende Beispiel verdeutlicht.

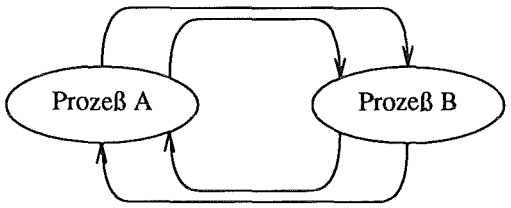


Abb. 1–1 : Kommunikation zwischen zwei Prozessen

Im einfachsten Fall sendet der Prozeß A an einer definierten Stelle in seinem Ablauf eine Nachricht an Prozeß B, die dieser an einer wohldefinierten Stelle in seinem Ablauf entgegennimmt. Prozeß B wartet dabei solange, bis Prozeß A die Nachricht schickt. Die beiden Prozesse sind dann über das Senden und Empfangen der Nachricht in ihren Abläufen an definierten Stellen synchronisiert.

Zur Modellierung des Verhaltens der Prozesse ist es nun günstig, wenn man die Aktion "Kommunikation" von dem Objekt der Kommunikation "die Nachricht" logisch trennt. Die Aktion teilen wir weiter auf in zwei Aktionen "Senden" des Prozeß A und "Empfangen" des Prozeß B. Die Aktionen "Senden" und "Empfangen" verdeutlichen wir in dem folgenden Diagramm über Kästen, die wir *Transitionen* nennen. Den Austausch der Nachricht symbolisieren wir durch einen Übertragungskanal zwischen den Aktionen, den wir durch einen Kreis darstellen und *Stelle* nennen wollen. Pfeile verbinden die *Transitionen* und die *Stelle* miteinander und geben die Richtung an, in die die Nachricht fließt.

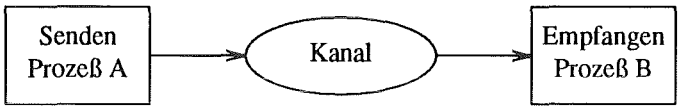


Abb. 1–2 : Ein erstes Petri-Netz

Wir können dieses Bild als ein erstes informelles Petri-Netz auffassen und halten die Bestandteile dieses Netzes fest.

Bestandteile eines Petri-Netzes

Die wesentlichen Bauelemente eines Petri-Netzes sind *Transitionen*, die "Aktionen" darstellen, *Stellen*, die passive Informationen tragen und Pfeile, die *Stellen* und *Transitionen* verbinden.

Wir wollen jetzt unser Modell der Kommunikation zwischen dem Prozeß A und dem Prozeß B noch etwas verallgemeinern. Dazu stellen wir uns vor, daß beide Prozesse innerhalb einer Endlosschleife in jedem Zyklus eine Operation ("Erzeugen bei A" bzw. "Verarbeiten bei B") durchführen, wobei stets Prozeß A seine Operation beendet haben muß, bevor Prozeß B seine Operation ausführen kann. Wir synchronisieren daher die Arbeit der beiden Prozesse dadurch, daß A am Ende seiner Operation in einer "Senden" Aktion B eine Nachricht über die erfolgte Operation "Erzeugen" schickt, während B am Beginn der Schleife in einer "Empfangen" Aktion auf die Nachricht von A wartet, bevor er die Operation "Verbrauchen" durchführt. Man spricht bei dieser Form von Kooperation von einem Erzeuger/Verbraucher-System, für das das folgende Bild ein Modell darstellt.

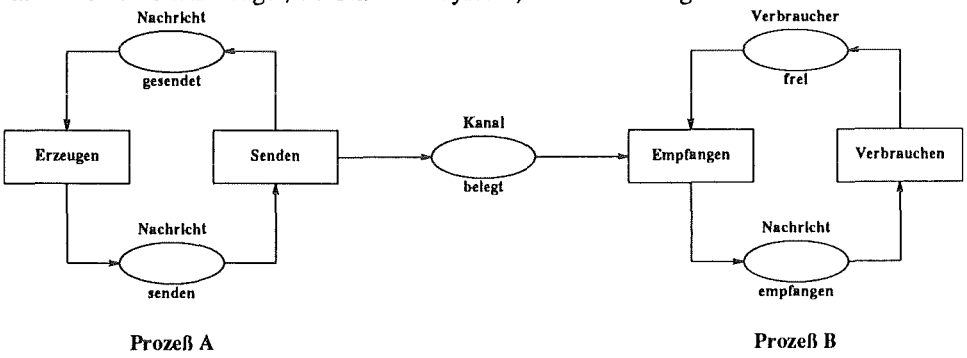


Abb. 1–3 : Erzeuger/Verbraucher-System

In dem Bild haben wir die Aktivität des Prozeß A in zwei Teilaktivitäten "Erzeugen" und "Senden" und die des Prozeß B in die Teilaktivitäten "Empfangen" und "Verbrauchen" zerlegt. Eine *Stelle* "Nachricht senden" nimmt die Nachricht auf, die Prozeß A an Prozeß B sendet. Eine *Stelle* "Nachricht gesendet" nimmt eine "innere" Information auf, ob die "Senden" Aktivität durchgeführt wurde. Entsprechend nimmt eine *Stelle* "Nachricht empfangen" im Prozeß B die Information auf, daß die Nachricht empfangen wurde, und eine *Stelle* "Verbraucher frei" nimmt eine innere Information auf, daß die Verarbeitenoperation durchgeführt wurde und damit auf eine neue Nachricht von B gewartet wird. Die *Stelle* "Kanal belegt" nimmt die gesendete Nachricht auf, die dann empfangen werden kann.

Zur Veranschaulichung des Ablaufes der einzelnen Aktivitäten und dem Austausch der Nachrichten, stellen wir die Nachricht(en) nun in dem Bild als schwarze Punkte, sogenannte *Marken*, dar, die beim Ablauf (*Schalten*) einzelner Aktivitäten von *Stelle* zu *Stelle* wandern. Die Verteilung der *Marken* zu einem Zeitpunkt (eine *Markierung* des Netzes) gibt den Zustand des Netzes zu diesem Zeitpunkt wieder.

Im Ausgangszustand des Systems befinde sich eine *Marke* auf der *Stelle* "Nachricht senden" und eine weitere *Marke* auf der *Stelle* "Verbraucher frei". Diese Ausgangsstellung zeigt dann an, daß Prozeß A bereits einen Erzeugungsschritt durchgeführt hat und nun die Nachricht senden muß, während Prozeß B auf die Nachricht wartet. Das folgende Bild gibt die Ausgangsmarkierung wieder:

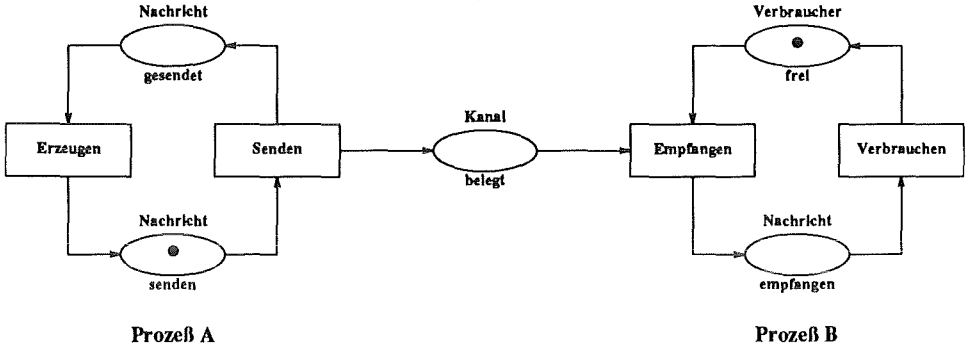


Abb. 1–4 : Ausgangsmarkierung unseres Erzeuger/Verbraucher-Systems

Wenn der Prozeß A nun die Nachricht sendet (Man spricht dann vom *Schalten der Transition* "Senden") geht das Netz in einen neuen Zustand über, der durch die folgende *Markierung* dargestellt werden kann.

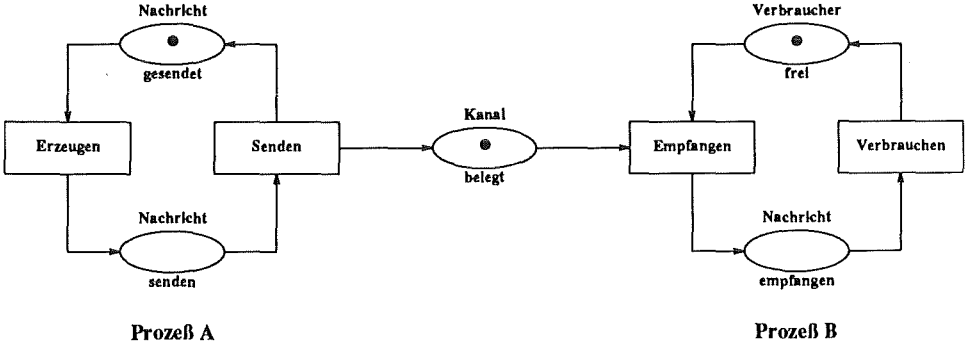


Abb. 1–5 : Markierung nach dem Schalten der Transition Senden

Die *Marke* auf der *Stelle* "Kanal belegt" zeigt nun an, daß eine Nachricht von Prozeß A gesendet wurde, die auf dem Kanal bereit zur Abholung durch B liegt. Die *Marke* auf der *Stelle* "Nachricht gesendet" gibt an, daß die Nachricht gesendet wurde, und B damit eine weitere "Erzeugen" Aktivität durchführen kann. In diesem Zustand des Netzes können nun unabhängig voneinander die beiden *Transitionen* "Erzeugen" oder "Empfangen" schalten. Wenn "Empfangen" schaltet, geht das Netz in einen Zustand über, der durch die folgende Markierung gegeben ist.

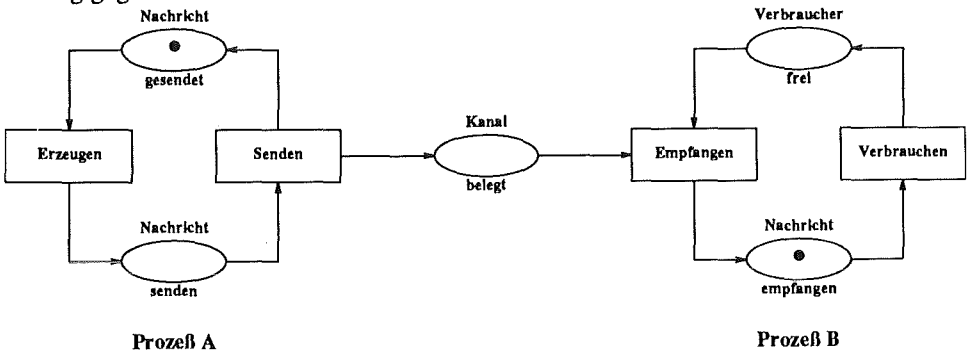


Abb. 1–6 : Markierung nach dem Schalten der Transition Empfangen

Die *Marke* auf der *Stelle* "Nachricht empfangen" zeigt an, daß die Nachricht von Prozeß B empfangen wurde. Nun kann die Transition "Verbrauchen" schalten. Das Netz geht dann in den folgenden Zustand über:

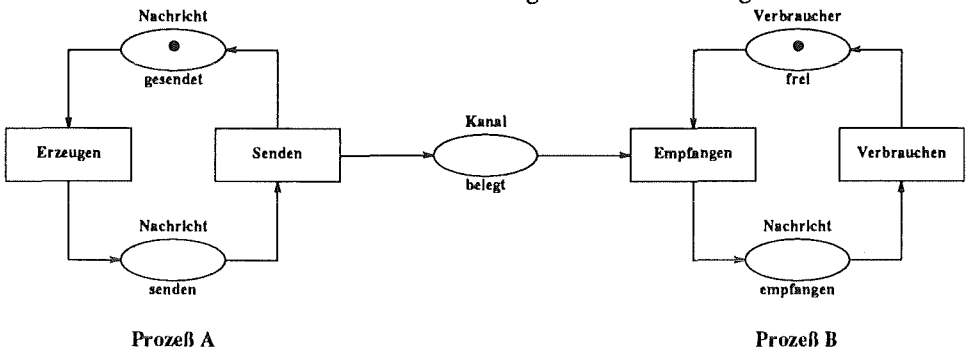


Abb. 1–7 : Markierung nach dem Schalten der Transition Verbrauchen

Die *Marke* auf der Stelle "Verbraucher frei" zeigt nun an, daß die Aktivität "Verbrauchen" durchgeführt wurde und Prozeß B auf den Eingang einer neuen Nachricht wartet. Schaltet nun noch die *Transition* "Erzeugen", so geht das Netz wieder in den Ausgangszustand über.

Beim Schalten einer Transition ändert sich die *Markierung* des Netzes in einer wohldefinierten Form. Die Regel, die diesen Übergang beschreibt, nennt man *Schaltregel.*

Schaltregel

> Beim Schalten einer Transition werden von den Inputstellen einer Transition alle Marken abgezogen und auf die Outputstellen der Transition je eine Marke gelegt.

Interessant ist es, daß wir die *Stellen* in unserem Beispielnetz und deren Beschriftungen auch noch anders deuten können: Die Beschriftungen der Stellen können wir als Aussagen mit einem Wahrheitswert **true** oder **false** interpretieren, wobei die Aussage **true** ist, wenn eine *Marke* auf der *Stelle* liegt, und andernfalls **false** ist. Man nennt diese Form von Netzen daher *C/E-Netze* (*Condition/Event-Netze*).

In unserem bisherigen Modell sind die Prozesse A und B in ihrer Arbeit so synchronisiert, daß Prozeß A an der "Senden" Aktivität warten muß, bis Prozeß B die Nachricht abgeholt hat. Man kann sich nun aber durchaus Situationen vorstellen, in denen Prozeß A bis zu einem gewissen Grade vorarbeiten darf, d.h.: A führt mehrere "Erzeugen Senden" Schritte durch, bevor B einen "Empfangen Verbrauchen" Schritt ausführt. Weiter kann man sich denken, daß B in einer Verbrauchsoperation mehrere von A in einer "Erzeugen" Operation bereitgestellte Objekte verbraucht. Zur Realisation so eines Modells erlauben wir, daß mehrere *Marken* auf einer Stelle liegen dürfen, wobei wir jeder *Stelle* eine natürliche Zahl, die *Kapazität der Stelle*, zuordnen, die angibt, wieviele *Marken* maximal auf dieser *Stelle* liegen dürfen. Weiter führen wir natürliche Zahlen als *Beschriftungen* von Kanten ein, die angeben, wieviele *Marken* beim *Schalten einer Transition* gleichzeitig von dieser *Stelle* abgezogen bzw. auf diese *Stelle* gelegt werden. Die *Schaltregel* ändern wir dann dahingehend, daß diese *Beschriftung* des Netzes respektiert wird. Wir erhalten damit ein Modell unserer Situation, wie sie typischer Weise das folgende Bild zeigt:

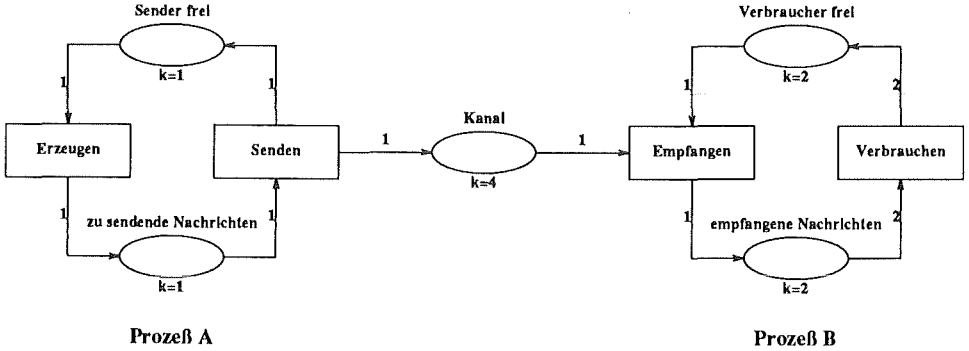


Abb. 1–8 : Erzeuger/Verbraucher-System als Stellen/Transitions-Netz

Der Prozeß A kann nun bis zu vier Nachrichten auf die *Stelle* "Kanal" (mit Kapazität 4) legen, bevor Prozeß B Nachrichten abholen muß, damit A bei der nächsten "Senden" Operation nicht warten muß. Weiter arbeitet die "Verbrauchen" Operation nur, wenn zwei Nachrichten empfangen wurden (Kantenbeschriftung 2) und wir können dies so interpretieren, daß sie stets zwei von A erzeugte Objekte in einem Schritt verbraucht. Netze mit solchen *Beschriftungen* nennt man *Stellen/Transitions-Netze.*

Zum Abschluß wollen wir uns noch eine weitere Verallgemeinerung der Eigenschaften von *Transitionen, Stellen, Marken* und *Beschriftungen* eines Netzes ansehen, die in den Bereich der "Höheren Netze" führt. Dazu stellen wir uns vor, daß wir nun nicht mehr nur einen Prozeß A vor uns haben, sondern eine Reihe von Prozessen $a_1,a_2,\ldots,a_n$, die irgendwelche Daten ermitteln. Diese Prozesse erstellen aus den von Ihnen ermittelten Daten nach den Prozessen unterscheidbare Datenpakete $d_1,d_2,\ldots,d_n$, die dann über einen Kanal zu einem weiteren Prozeß *b* gesendet werden, der diese weiterbearbeitet (z.B. in einer Datenbank ablegt).

Wir wollen nun in unserem System sowohl zwischen den einzelnen Prozessen aus der Menge $Process=\{a_1,a_2,..,a_n,b\}$ als auch zwischen den verschiedenen Datenpaketen $Daten=\{d_1,d_2,..,d_n\}$ unterscheiden können. Da wir nun unterscheidbare Nachrichten haben, wobei mehrere von Ihnen gleichzeitig auf dem Kanal liegen können, ist uns auch die Reihenfolge nicht mehr gleichgültig, mit denen diese Nachrichten von Prozeß *b* empfangen werden. In Programmen löst man dieses Problem häufig dadurch, daß man eine lineare Liste (Warteschlange) einführt, in die der Sender die Nachrichten in einer bestimmen Weise einordnet und der Empfänger nach einer bestimmten Methode abholt. Das bekannteste Beispiel hierfür ist eine FIFO (First in First out) Warteschlange, in der ein Sender eine Nachricht an das Ende der Liste anfügt und der Empfänger jeweils die erste Nachricht in der Liste abholt.

Mathematisch läßt sich eine lineare Liste mit Elementen aus unserer Nachrichtenmenge *Daten* am einfachsten als Folge der Gestalt $d_{i_1}d_{i_2}\cdots d_{i_k}$ beschreiben, wobei $d_{i_j}\in Daten$ für alle $j=1,2,...,k$ ist. So eine Folge bezeichnet man dann auch als ein *Wort* über dem *Alphabet Daten.* Die Menge aller Folgen $Warteschlange=\{d_{i_1}d_{i_2}..d_{i_k} \mid d_{i_j}\in Daten, 1\le j\le k \text{ und } k\in IN\}$ stellt dann die Menge aller linearen Listen mit Elementen aus *V* dar und ist eine *Sprache* über dem *Alphabet Daten.*

Das Einhängen von Datenpaketen in eine Warteschlange läßt sich durch eine Operation der Gestalt $ein:Daten\times Warteschlange\rightarrow Warteschlange,(d,w)\rightarrow wd$ beschreiben. Das Aushängen eines Datenpaketes aus der Warteschlange beschreiben wird durch zwei Operationen $aus:Warteschlange\rightarrow Daten, w=d_{i_1}d_{i_2}..d_{i_k}\rightarrow d_{i_1}$ und $rest:Warteschlange\rightarrow Warteschlange, w=d_{i_1}d_{i_2}..d_{i_k}\rightarrow d_{i_2}d_{i_3}\cdots d_{i_k}$. Die erste Operation *aus* gibt uns das auszuhängende Element (bei FIFO das erste) und die zweite Operation *rest* gibt uns die Liste, die nach dem Aushängen verbleibt. Weiter nehmen wir noch an, daß wir zwei Operationen $daten:Process\rightarrow Daten$, $daten(a_i)=d_i$ und $process:Daten\rightarrow Process$, $process(d_i)=a_i$ für alle $i=1,..,n$ haben, die eindeutig Prozessen Datenpakete und umgekehrt Datenpaketen Prozesse zuordnen. Des weiteren nehmen wir an, daß wir eine nullstellige Operation (nullstellige Operationen kann man als Konstanten auffassen) $empf:\emptyset\rightarrow Process$, $empf\equiv b$ haben, die für den Prozeß *b* steht. Das Tupel $DMS=((Process,Daten,Warteschlange),(ein,aus,rest,daten,process,empf))$ ist dann eine *Algebra* mit *Trägermengen Process, Daten* und *Warteschlange* und Operationen *ein, aus, rest, daten, process* und *empf.*

Wie können wir nun die mathematischen Beschreibungen zur Modellierung unseres Problems nutzen? Die Kanten des Netzes beschriften wir mit Variablen, deren Werte entweder Elemente aus der Menge *Daten* (Datenpakete),*Process* (Prozesse) oder *Warteschlange* (Warteschlangen) sind. $PR=\{pr,pr_1,pr_2,...\}$ sei die Menge der Variablen mit Werten aus *Process*, $D=\{d,d_1,d_2,...\}$ sei die Menge der Variablen mit Werten aus *Daten* und $W=\{w_1,w_2,..\}$ sei die Menge der Variablen mit Werten aus *Warteschlange.* Auf den Stellen sollen jeweils als Marken nur Prozesse (Elemente aus *Process*), Daten (Elemente aus *Daten*) oder Warteschlangen (Elemente aus *Warteschlange*) liegen, was wir durch Angabe eines *Prädikats* "Prozeß", "Warteschlange" oder "Daten" als Beschriftung der Stellen festlegen. Weiter ordnen wir den Transitionen im Netz "logische Formeln" bestehend aus den Variablen aus *PR*, *D* und *W*, den Operationssymbolen (Als Symbole wählen wir hier in der Einführung die Zeichen, mit denen wir die Operationen selbst bezeichnen) der Algebra *DMS*, sowie aussagenlogischen Operatoren, wie das logische UND, das logische ODER und die Identität, den Existenzquantor und den Allquantor, zu. Diese "logischen Formeln" beschreiben dann das Schaltverhalten der Transitionen in dem Sinne, daß bei geeigneter Interpretation der Variablen, Operations- und Relationssymbole in der Algebra *DMS* das Schaltverhalten der Transition durch die Formel definiert ist. Die Transitionsinschrift der Transition "Senden" soll dann die folgende Gestalt haben:

$$F_S: \qquad \exists w \in W: w_1 = ein(w, n_1) \wedge pr_1 = process(n_1)$$

d.h.: Hier wird eine Aussage formuliert, daß eine Warteschlange $w \in W$ existiert, sodaß die Identität zwischen Warteschlangen $w_1 = ein(w, n_1)$ erfüllt ist, wobei *ein* die Operation des Einhängens eines Datenpaketes, gegeben durch die Variable $n_1 \in V$, in eine Warteschlange, gegeben durch $w \in W$, ist, und daß die Identität $pr_1 = process(n_1)$ zwischen Prozessen gilt. Die Transitionsinschrift der Transition "Empfangen" soll analog die Gestalt haben:

$$F_{EF}: \qquad n_2 = aus(w_1) \wedge pr_2 = empf()$$

d.h.: Es soll die Identität zwischen Datenpaketen $n_2 \in V$ und $aus(w_1)$ mit $w_1 \in W$ gelten, wobei *aus* die Operation des Aushängens eines Datenpaketes aus einer Warteschlange ist, und die Identität $pr_2 = empf()$ zwischen Prozessen erfüllt sein soll. Die Transition "Erzeugen" soll weiter die Inschrift

$$F_E: \qquad n_1 = daten(pr_1)$$

und die Transition "Verbrauchen" die Inschrift

$$F_V: \qquad pr_2 = empf()$$

haben. Man beachte dabei, daß wir die Art des Verbrauchens hier nicht exakt beschreiben, da es uns vorwiegend um die Kommunikation geht.

Die Schaltregel des Netzes formulieren wir in der Weise, daß eine Transition dann schalten kann, wenn es eine Substitution der Variablen innerhalb der Formel einer Transition gibt, sodaß einmal die Formel zu **true** ausgewertet wird und zweitens bei Variablen, die an Inputkanten der Transition als Beschriftung verwendet werden, der Wert innerhalb der Substitution als Marke auf der zugehörigen Inputstelle vorhanden ist. Beim Schalten werden die Marken dann von den Inputstellen abgezogen und auf die Outputstellen die Marken gelegt, die als Werte der Variablen, die als Outputkantenbeschriftung der zugehörigen Kante auftreten, bei Substitution gegeben sind. Damit können wir unser Modell durch das Bild 1-9 darstellen:

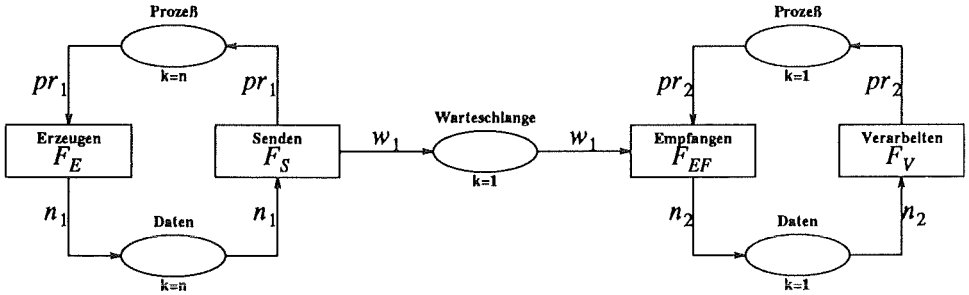


Abb. 1–9 : Beispiel für ein Prädikat/Transitions-Netz

Wir wollen uns einmal das Schalten einzelner Transitionen des Netzes ansehen. Dazu nehmen wir an, daß wir die folgende Ausgangsmarkierung des Netzes haben.

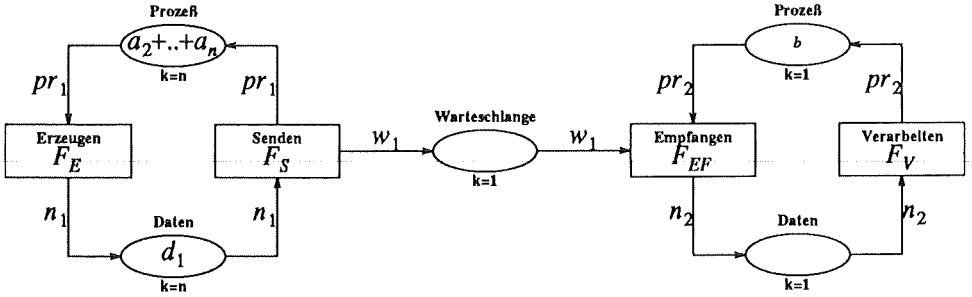


Abb. 1–10 : Das Prädikat/Transitions-Netz mit Anfangsmarkierung

Wenn wir als Variablensubstitution für n_1 d_1, für w d_2d_3, für w_1 $d_2d_3d_1$ und für pr_1 a_1 wählen, so ist die Formel F_S bei Einsetzen der Substitution erfüllt, da *ein* ja als Konkatenation $ein(d_2d_3, d_1) = d_2d_3d_1$ definiert ist. Das Netz geht dann nach dem Schalten der Transition "Senden" in die Markierungssituation in Bild 1-11 über:

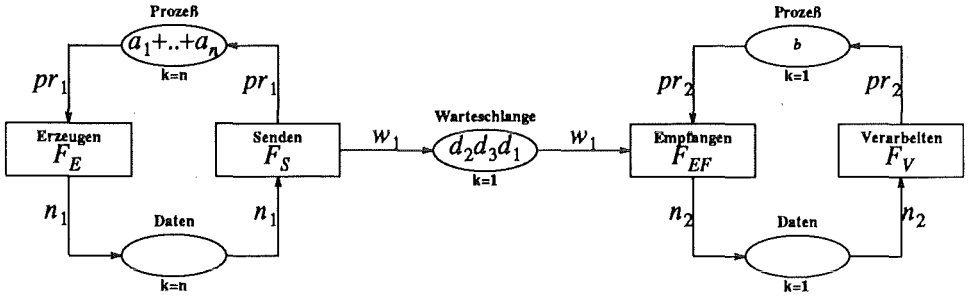


Abb. 1–11 : Das Prädikat/Transitions-Netz nach Schalten der Transition Senden

Jetzt kann die Transition "Empfangen" mit der Substitution $d_2d_3d_1$ für w_1, d_2 für n_2 und b für pr_2 schalten. Das Netz geht dann in die Markierung in Bild 1.12 über:

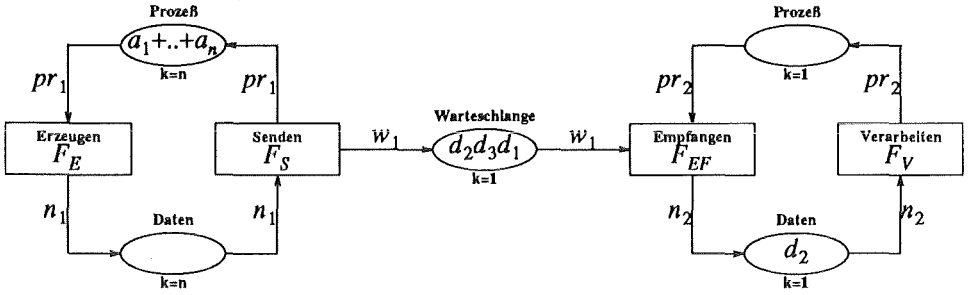


Abb. 1–12 : Das Prädikat/Transitions-Netz nach Schalten der Transition Empfangen

An dieser Schaltfolge sieht man deutlich, daß es sich in unserem Modell nicht um eine reine Modellierung einer FIFO Warteschlange handelt. Die freie Variable w in der Transitionsformel von "Senden" hat vielmehr die Wirkung, daß hier eine beliebige Warteschlange substituiert werden kann und nicht nur die Warteschlange, die bereits auf der Stelle "Warteschlange" liegt. Dies liegt daran, daß die freie Variable w in keiner der Inputkantenbeschriftungen vorkommt. Entsprechend wird beim Aushängen eines Elementes der Warteschlange durch "Empfangen" nicht etwa die Warteschlange auf der Stelle "Warteschlange" verändert, da diese Stelle keine Outputkante der Transition "Empfangen" ist. Um eine reine FIFO Warteschlange zu modellieren, müssen wir zwei zusätzliche Kanten von "Warteschlange" nach "Senden" und von "Empfangen" nach "Warteschlange" als Inputkante für "Senden" und Outputkante für "Empfangen" einführen. Wenn diese als Kantenbeschriftungen die Variable w_2 tragen, müssen wir als Formeln der Transitionen nun definieren:

$$F_S: \qquad w_1{=}ein(w_2,n_1)\wedge pr_1{=}process(n_1)$$

und

$$F_{EF}: \qquad n_2{=}aus(w_1)\wedge w_2{=}rest(w_1)\wedge pr_2{=}empf()$$

Das Bild 1-13 zeigt das so modefizierte Netz.

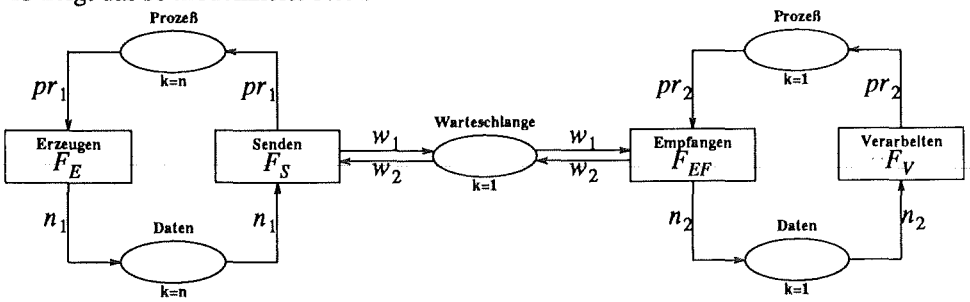


Abb. 1–13 : Modell einer FIFO-Nachrichtenübertragung

Wir wollen unser Modell eines Daten Management Systems zum Abschluß noch in dem Sinne verallgemeinern, daß wir nun davon ausgehen, daß Prozeß b nun nicht mehr nur ein Datenpaket sondern zwei Datenpakete gleichzeitig verarbeiten kann. Hierzu fügt er in der Operation "Empfangen" gleichzeitig zwei Datenpakete aus und verarbeitet zwei Datenpakete in der Transition "Verarbeiten".

Um dies modellieren zu können, müssen wir nun die Kanten von "Empfangen" zur Stelle "Daten" und von der Stelle "Daten" zur Transition "Verarbeiten" mit einer Multisumme $<n_1>+<n_2>$ beschriften. Die Transitionsinschrift von "Empfangen ändert sich zu:

$$F_{EF} \qquad n_1=aus(w_1) \wedge n_2=aus(rest(w_1)) \wedge pr_2=empf() \wedge w_2=rest(rest(w_1))$$

Bild 1-14 verdeutlicht das Modell.

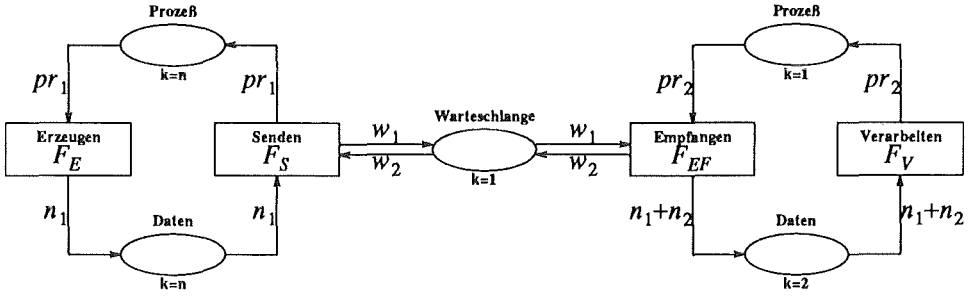


Abb. 1–14 : Beispiel eines Netzes mit Multisummen an den Kanten

Wenn die Transition "Empfangen" schaltet, werden aus der Warteschlange auf der Stelle "Warteschlange" zwei Datenpakete entfernt und auf die Nachstelle "Daten" gelegt (sofern zwei in der Warteschlange vorhanden sind). Transition "Verarbeiten" zieht stets zwei Datenpakete von der Vorstelle "Daten" ab.

Das bisher entwickelte Petri-Netz-Modell des Daten Management Systems zeigt noch einige gravierende Nachteile. Im bisherigen Modell liegen alle wesentlichen Informationen, die das Schalten einer Transition bestimmen, in den Transitionsinschriften. Insbesondere der Fluß der individuellen Objekte durch das Netz ist dadurch nur schwer erkennbar, denn welche Objekte von Stellen abgezogen bzw. auf Stellen gelegt werden, wird durch die Formeln der Transitionen bestimmt.

Günstiger wäre es für das Verständnis des Netzes, wenn die Kantenbeschriftungen des Netzes und die vorhandenen Marken auf den Stellen das Wie eines Schaltvorganges bestimmen, während die Transitionsinschrift nur darüber entscheidet, ob eine Transition bei gegebenem Wie schalten kann. Dies läßt sich dadurch erreichen, daß man Formeln der Gestalt $w_1=ein(w_2,n)$ in Transitionsinschriften vermeidet und stattdessen anstatt mit w_1 die zugehörige Kante gleich mit $ein(w_2,n)$ beschriftet. Konstrukte der Gestalt $ein(w,n)$, die aus Variablensymbolen und Operationssymbolen (Operatoren) nach definierten Regeln der Anwendung von Operatoren auf ihre Argumente gebildet werden, bezeichnet man als *Terme*. Unsere Netzbeschriftung läßt sich wesentlich übersichtlicher gestalten, wenn wir *formale Summen von Termen* als Beschriftung von Kanten zulassen. Die Formeln der Transitionen reduzieren sich dann zu der trivialen Formel **true**.

Das folgende Bild zeigt unser Daten Management System unter der Ausnutzung von Termbeschriftungen an den Kanten.

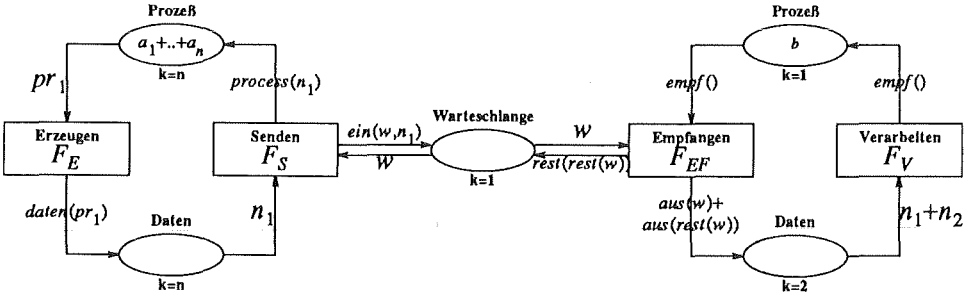


Abb. 1–15 : Netz mit Termbeschriftungen an den Kanten

Wenn wir einmal die Umgebungen der Transitionen ansehen, so sehen wir nun, daß das Wie eines Schaltvorganges eindeutig durch die Kantenbeschriftungen und die auf den Umgebungsstellen vorhandenen Marken bestimmt ist. Der Fluß der individuellen Objekte durch das Netz wird deutlich sichtbar. Bei einem Schaltvorgang der Transition "Senden" wird z.B. ein Datenpaket von der Vorstelle "Daten" abgezogen, in die Warteschlange w auf der Stelle "Warteschlange" durch die Operation $ein(w,n)$ eingefügt und die so entstandene neue Warteschlange wieder auf die Stelle "Warteschlange" zurückgelegt. Der Prozeß, der das Datenpaket erzeugt hat, wird wieder auf die Nachstelle "Prozeß" zurückgelegt, was anzeigt, daß er einen Erzeugungszyklus beendet hat. Die Transitionsinschriften reduzieren sich nun zu der trivialen Formel **true**, weil die Transitionsinschriften aus den vorhergehenden Beispielen nur das Wie nicht aber Ob Bedingungen enthielten.

Wir wollen die allgemeine Struktur des Modells im letzten Beispiel einmal zusammenfassen. Das Modell läßt sich beschreiben durch ein 4-Tupel $N=(P,T,F,A)$ mit:

- P und T sind die Mengen der Stellen und Transitionen,
- $F \subseteq P \times T \cup T \times P$ ist eine Relation in der Menge $X=P \cup T$,
- $A: P \cup T \cup F \rightarrow$ *Formeln* $\cup$ *Formale Summen von Termen* ist eine Funktion, die Stellen und Transitionen auf *Formeln* und Kanten auf *Formale Summen von Termen* abbildet, d.h: $A(P \cup T) \subseteq$ *Formeln* und $A(F) \subseteq$ *Formale Summen von Termen*,

Solche Strukturen werden *Höhere Petri-Netze* genannt. Wenn wir nur Variablen als Kantenbeschriftungen zulassen, kann man im Sinne von Genrich/Lautenbach von *Prädikat/Transitions-Netzen* sprechen. In einer Interpretation kann man Netze der obigen Gestalt mit Termen als Kantenbeschriftung als *Coloured Petri-Netze* im Sinne von Jensen auffassen.
In den folgenden Kapiteln wollen wir die Hilfsmittel bereitstellen, die nötig sind, solche *Höheren Petri-Netze* formal zu beschreiben.

2 Elementare Begriffe

In diesem Abschnitt beschreiben wir die elementaren Begriffe der Theorie der formalen Sprachen, die wir im folgenden benötigen. Dabei werden wir Sprachen als Mengen von endlichen Folgen von Symbolen (Wörtern) betrachten. Genauer definieren wir:

Definition 2.1 : (Wort)

Es sei A eine endliche, nichtleere Menge. Jede endliche Folge w (d.h. eine Funktion von der Gestalt $w:[1..n]\rightarrow A$ mit [1..n] die Menge der ersten n natürlichen Zahlen) wird ein *Wort* über dem Alphabet A genannt. Die Mächtigkeit der Menge [1..n] , d.h. die Zahl |[1..n]| = n wird die *Länge des Wortes* w genannt und stets mit $l(w)$ bezeichnet.
Mit dem Symbol /w/ bezeichnen wir die Menge $w([1..n])=\{a\in A \mid \exists_{i\in[1..n]} a=w(i)\}$, d.h. die Menge der Elemente der Folge w. Mit $dom(w)(=[1..n])$ bezeichnen wir die *Urbildmenge* des Wortes (der Funktion) w.

Betrachten wir einmal ein paar Beispiele für Wörter:

Beispiel 2.1

Es sei $A=\{\bullet,+\}$. Die nachstehenden Folgen sind Wörter über dem Alphabet A:

a) $w:[1..3]\rightarrow A$ mit $w(1)=w(2)=\bullet, w(3)=+$, $l(w)=3$

b) $w':[1..4]\rightarrow A$ mit $w'(1)=w'(2)=w'(3)=w'(4)=\bullet$, $l(w')=4$

c) $w'':\emptyset\rightarrow A$, d.h. die leere Funktion, $l(w)=0$

Die Folgen des nächsten Beispiels sind keine Wörter über A:

Beispiel 2.2

a) $w'':GER(\mathbb{N})\rightarrow A$ mit GER(IN) - die Menge der geraden natürlichen Zahlen und

$$w''(k)=\begin{cases}\bullet & \text{für } k \text{ dividierbar durch } 4\\ + & \text{für } k \text{ nicht dividierbar durch } 4\end{cases}$$

Der Definitionsbereich GER(IN) der Folge ist zwar eine Untermenge der Menge IN der natürlichen Zahlen, er ist aber nicht von der Gestalt [1..n] und damit ist die Folge kein Wort.

b) Sei A wie unter Beispiel 2.1.

$w''':[1..4]\rightarrow A'$ *mit* $A'=A\cup\{@\}$ *und* $w'''(1)=w'''(2)=\bullet$, $w'''(3)=+$, $w'''(4)=@$

Da das Symbol @ der Menge A nicht angehört, ist die Funktion w''' kein Wort über dem Alphabet A sondern über dem Alphabet $A'=A\cup\{@\}=\{\bullet,+,@\}$.

Wörter schreibt man häufig in der Form w(1)w(2)...w(k), wobei man hiermit die Linearigkeit der "normalen" Schreibweise ausnützt. Ein typisches Beispiel hierfür sind die natürlichen Zahlen.

Beispiel 2.3

Es sei $A = \{0,1,...,9\}$ die Menge der Ziffern. Jede endliche Folge $z:[1..k]\rightarrow A$, d.h. jede Standarddarstellung einer natürlichen Zahl ist ein Wort über A. Die Wörter $w:[1..3]\rightarrow A$ mit $w(1)=w(2)=w(3)=5$ und $w':[1..2]\rightarrow A$ mit $w'(1)=3$, $w'(2)=1$ lassen sich in der Form *555* und *31* schreiben.

Die oben beschriebene Schreibweise ist natürlich für das leere Wort $\emptyset:\emptyset\rightarrow A$ nicht möglich, denn man müßte "nichts" auf "keiner" Stelle schreiben. Das leere Wort wird daher stets durch ein spezielles Symbol bezeichnet. In dieser Arbeit wird hierzu das Symbol ε (der griechische Buchstabe "Epsilon") benutzt.

Die Menge aller Wörter über dem Alphabet A wird stets mit dem Symbol A^* bezeichnet. A^+ bezeichnet die Menge aller Wörter mit mindestens einem Zeichen über dem Alphabet A.

Definition 2.2 : (Sprache)

Die Untermengen der Menge A^* werden *Sprachen* über dem Alphabet A genannt.

Untermengen von A^* lassen sich auf vielfache Art und Weise beschreiben, wie die folgenden Beispiele zeigen:

Beispiel 2.4

Es sei $A=\{\bullet,+\}$ wie in Beispiel 2.1. Die Sprache $L\subseteq A^*$ wird auf folgender Weise definiert:

$$L=\{w\in A^* \mid l(w)\leq 2\}.$$

L besteht aus allen Wörter über A, deren Länge nicht größer als 2 ist. Es ist dann leicht zu sehen, daß gilt:

$$L=\{\varepsilon,\bullet,+,\bullet\bullet,\bullet+,+\bullet,++\}.$$

Die Sprache in dem obigen Beispiel ist endlich und wird mit Hilfe der Längenfunktion definiert. Leider sind die meisten in der Mathematik vorkommenden Sprachen weder endlich noch so einfach definiert.

Beispiel 2.5

Betrachten wir eine Sprache, die in der Theorie der natürlichen Zahlen benutzt wird. Es sei $A = \{0,'\}$. Die Sprache LN sei wie folgt induktiv definiert:

1. Das Wort 0 gehört der Sprache LN an.
2. Wenn ein Wort w der Sprache LN angehört, dann gehört auch das Wort w' dieser Sprache an.

Die obigen Sätze betrachten wir als Konstruktionsregeln, die uns erlauben, neue Wörter der Sprache aus schon bekannten Wörtern der Sprache zu bilden. Um festzustellen, ob ein gegebenes Wort der Sprache angehört oder nicht, muß man feststellen, ob es von dem Wort 0 mit den obigen Regeln konstruierbar ist. Dies ist nicht schwer, da jedes Wort dieser Sprache von der Gestalt 0''''' , d.h. der Gestalt "Null mit eine Menge von ' - Zeichen, ist."

Es ist leicht zu sehen, daß für jede nicht leere Menge A das Tripel $(A^*, \bullet, \varepsilon)$, wobei $\bullet$ die Operation der Konkatenation (Hintereinanderschaltung von Wörtern) darstellt, ein Monoid ist, d.h. daß für alle Elemente $a,b,c\in A^*$ gilt:

(1) $(a\bullet b)\bullet c=a\bullet(b\bullet c)$

(2) $\varepsilon\bullet a=a\bullet\varepsilon=a$

Im Folgenden werden oft Homomorphismen von solchen Monoiden betrachtet. Wie man weiß, läßt sich jede Abbildung $f:A\to B^*$ eindeutig zu einem Monoid-Homomorphismus $f^{\#}:A^*\to B^*$ fortsetzen, indem man definiert: $f^{\#}(\varepsilon)=\varepsilon$ und für jedes Wort $w\in A^*$ und jedes $a\in A$ sei $f^{\#}(wa)=f^{\#}(w)f(a)$.

Beispiel 2.6

Es sei $A\subseteq B$ eine Untermenge der Menge B und $\phi_{B,A}$ eine Abbildung der Gestalt:

$$\phi_{B,A}:A\to B^* \quad mit \quad \phi_{A,B}(x)=\begin{cases} x & für\ x\in A \\ \varepsilon & für\ x\notin A \end{cases}$$

$\phi_{A,B}$ erzeugt dann einen Homomorphismus $\phi^{\#}_{A,B}:A^*\to B^*$, der aus jedem Wort $w\in A^*$ alle Symbole der Menge $B\backslash A$ auslöscht.

Analog zur Operation der Konkatenation (Zusammensetzung) von Wörtern kann man eine Konkatenation von Sprachen definieren.

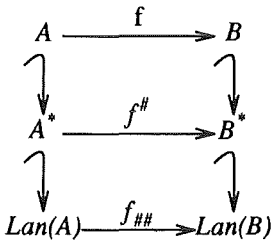
Definition 2.3

Es seien $L,L'\subseteq A^*$ Sprachen über dem Alphabet A. Wir definieren dann:

$$L\bullet L'=\{w\bullet w' \mid w\in L\wedge w'\in L'\}$$

Die Konkatenation von Sprachen wird mit demselben Symbol bezeichnet, wie die Konkatenation von Wörtern. Dies führt aber zu keinen Mißverständnissen, da ja aus dem Kontext jederzeit ersichtlich ist, um welche Operation es sich handelt.

Es ist leicht zu sehen, daß für jede Menge A das Tripel $Lan(A)=(pow(A),\circ,\{\varepsilon\})$ ein Monoid ist, und daß jede Funktion $f:A\to B$ über die oben schon erwähnte Erweiterung $f^{\#}:A^{*}\to B^{*}$ eindeutig zu einem Homomorphismus $f^{\#\#}:Lan(A)\to Lan(B)$ erweitert werden kann.



Diese Eigenschaft werden wir im folgenden noch öfters ausnutzen.

Bezeichnungen

Sei w ein Wort über einem Alphabet V und $A:V\to$**Set** eine Funktion (Die Werte von A sind Mengen über einem bestimmtem Universum U).

(a) Wir bezeichnen mit dem Symbol A^w die Menge

$$A^w=\begin{cases}\varnothing & wenn\ w=\varepsilon\\ A(a_1)\times A(a_2)\times\ldots\times A(a_k) & wenn\ w=a_1a_2\ldots a_k\end{cases}$$

(b) Mit $A(w)$ bezeichnen wir das Bild von w unter dem Homomorhismus $A^{\#}$, d.h.
$A(w)=A(a_1)\cdots A(a_n)$ falls $w=a_1\cdots a_n$.

(c) Für ein Tupel (w_1,w_2) mit $w_1,w_2\in V^{*}$ bezeichnen wir mit $A((w_1,w_2))$ das Tupel $(A(w_1),A(w_2))$.

Beispiel 2.7

Es sei A eine Abbildung $A:[1..3]\to$**Set** *mit* $A(1)=\mathbb{N}, A(2)=\mathbb{Z}, A(3)=\mathbb{R}$. Dann gilt: $A^{113}=\mathbb{N}\times\mathbb{N}\times\mathbb{R}$.

Eine analoge Schreibweise wird auch für Funktionen und Relationen verwendet, d.h. für jede Abbildung $A:S\to$**Func** ($A:S\to$**Rel**), wobei **Func (Rel)** - die Menge aller Funktionen (Relationen) über einem bestimmten Universum U ist, bezeichnen wir für jedes Wort $w\in A^{*}$, $w=s_1s_2,\ldots,s_k$ mit A^w die Funktion (Relation) $A^w=A(s_1)\times A(s_2)\times,\ldots,\times A(s_k)$.

Beispiel 2.8

Es sei $S=[1..3]$. Die Funktion A sei gegeben durch:

$$A:S\to\textbf{Func},\ A(i):\mathbb{R}\to\mathbb{R}\ mit\ A(i)(x)=(i+1)x\ für\ alle\ x\in\mathbb{R}\ und\ i\in S.$$

Die Abbildung A^{13} ist dann von der Gestalt: $A^{13}:\mathbb{R}\times\mathbb{R}\to\mathbb{R}\times\mathbb{R}$, $A^{13}:(x,y)\to(2x,4x)$.

Die Wörter der Gestalt $A(w)$ ($w=a_1a_2\cdots a_k$) werden manchmal auch explizit als Folgen geschrieben , d.h. wir schreiben $A(a_1),A(a_2),\ldots,A(a_k)$ anstatt $A(w)=A(a_1)A(a_2)\ldots A(a_k)$.

Definition 2.4 : (Präfix)

Ein Wort $u:[1..m]\to A$ ist ein Präfix des Wortes $v:[1..n]\to A$ dann und nur dann, wenn $m\le n$ und $u=v\mid_{[1..m]}$, d.h. wenn $m\le n$ und für $i\in[1..m]$ $u(i)=v(i)$ ist.

Für das nächste Kapitel benötigen wir noch die folgende Definition:

Definition 2.5 : (Typ einer Operation)

(a) Sei $A:V \to Set$, $w \in V^+$, $a \in V$ und eine Funktion $f:A^w \to A^a$ gegeben. Der *Typ von f* ist dann definiert als das Tupel $typ(f) = (A(w),A(a))$.

(b) Sei $A:V \to Set$, $w \in V^+$, $a \in V$ und eine Relation $r \subseteq A^w$ gegeben. Der *Typ von r* ist dann definiert als $typ(r) = A(w)$.

Bezeichnung

Seien A und B Mengen. Mit $[\, A \to B \,]$ bezeichnen wir die Menge aller Abbildungen der Gestalt $f:A \to B$.

2.1 Hinweise zur Literatur

Die in diesem Kapitel beschriebenen Begriffe der Theorie der formalen Sprachen kann man in fast jedem Buch über die Grundlagen der Informatik finden. Sie werden in der Informatik bei verschiedenen Gelegenheiten benötigt (z.B. Compilerbau, Theorie der Programmiersprachen und in der Künstlichen Intelligenz). Formaler werden diese Begriffe in der Mathematik innerhalb der Theorie der formalen Sprachen betrachtet. Eine klassische Arbeit ist in diesem Sinne das Buch von Salomaa "Formal Languages"([Sal73a]).

3 Algebraische Systeme

Algebraische Systeme werden als mathematische Modelle für eine Vielzahl von Teilaspekten der realen Welt verwendet. Sie sind mathematische Modelle in dem Sinne, daß sie in einer mathematisch exakten Form beschrieben werden. Der zu modellierende Teil der Realität wird durch eine Mengenfamilie (manchmal auch durch eine Menge), die sogenannte Trägermengenfamilie (Trägermenge), dargestellt. Die Eigenschaften dieser (kleinen) Welt werden durch Relationen auf den Trägermengen beschrieben. Dabei können einige von diesen Relationen funktionell (d.h. sie sind rechtseindeutig) sein und werden dann als Operationen bezeichnet.

Für die Beschreibung ist daher ein Tupel (Mengenfamilie A, Operationen u. Relationen auf der Familie A) sinnvoll. Die Frage, ob eine solche Darstellung der Realität wirklich adequat ist, gehört zu den grundlegenden Problemen der Philosophie und wird in vielen philosophischen Arbeiten (z.B. die Arbeiten Lukasiewicz oder Wittgenstein) untersucht. In der Mathematik werden die algebraischen Systeme als rein mathematisch abstrakte Objekte betrachtet ohne zu fragen, ob sie genau die Realität beschreiben. Auch wenn man bezweifelt, daß algebraische Systeme die Realität adequat wiedergeben, sind sie dennoch in der Praxis von großem Nutzen.

3.1 Definition algebraischer Systeme

Es sei im folgenden S eine nichtleere Menge.

Definition 3.1 : (Signatur)

Eine *Signatur über S* ist ein 5-Tupel $SIG=(S,\Omega,\Pi,typ_\Omega,typ_\Pi)$, für das gilt:

(1) S, Ω, Π sind Mengen, die wir Menge der *Sortennamen*, Menge der *Operatoren* und Menge der *Prädikate* nennen wollen,

(2) $typ_\Omega : \Omega \to S^*\times S$ und $typ_\Pi : \Pi \to S^+$ sind Funktionen, wobei wir für $\omega\in\Omega$ $typ_\Omega(\omega)$ den *Typ des Operators* ω und für $\pi\in\Pi$ $typ_\Pi(\pi)$ den *Typ des Prädikates* π nennen wollen.

Für eine Signatur $SIG=(S=\{s_1,s_2,\cdots,s_n\},\Omega=\{\omega_1,\omega_2,\cdots,\omega_k\},\Pi=\{\pi_1,\pi_2,\cdots,\pi_l\},typ_\Omega,typ_\Pi)$ mit $typ_\Omega(\omega_i)=(w_i,s_i)$ $i=1,2,..,k$, $w_i\in S^*$, $s_i\in S$ und $typ_\Pi(\pi_i)=(w_j')$ $i=1,2,..,l$, $w_j'\in S^+$ schreiben wir in Beispielen häufig auch in Form einer Pseudosprache

```
SIG Name =
        sorts= s1, s2, s3, ..., sn;
        ops=   ω1 : w1→s1,
               ω2 : w2→s2,
                 ............
                 ............
               ωk : wk→sk;
        preds= π1 : w1',
               π2 : w2',
                 ..............
                 ..............
               π1 : w1';
SIGEND.
```

Dabei steht Name als Platzhalter für den Namen der Signatur. Diese Schreibweise verdeutlicht sehr anschaulich die Bedeutung der Typen von Operatoren und Prädikaten als Angabe für die Argument- und Wertebereiche der durch eine Interpretation gegebenen zugehörigen Operationen und Relationen. Signaturen sind rein syntaktische (sprachliche Objekte), denen wir später über eine "Interpretation der sprachlichen Konstrukte" innerhalb eines "algebraischen Systems" eine semantische Bedeutung zukommen lassen. Dazu müssen wir zunächst einmal definieren, was wir unter einem algebraischen System verstehen:

Definition 3.2 : (algebraisches System)

Ein *algebraisches (relationales) System* über der Signatur $SIG=(S,\Omega,\Pi,typ_\Omega,typ_\Pi)$ ist ein Tripel $ALS=(A_S, A_\Omega, A_\Pi)$ mit den Eigenschaften:

(1) $A_S : S\rightarrow$**Set** , $A_\Omega : \Omega\rightarrow$**Func** und $A_\Pi : \Pi\rightarrow$**Rel** sind injektive Abbildungen.

(2) Für jeden Operator $\omega\in\Omega$ gilt: $typ(A_\Omega(\omega)) = A_S(typ_\Omega(\omega))$.

(3) Für jedes Prädikat $\pi\in\Pi$ gilt: $typ(A_\Pi(\pi)) = A_S(typ_\Pi(\pi))$.

Die Elemente der Mengen $\{X\in$ **Set** $\mid\exists_{s\in S}A_S(s)=X\}$, $\{X\in$ **Func** $\mid\exists_{\omega\in\Omega}A_\Omega(\omega)=X\}$ und $\{X\in$ **Rel** $\mid\exists_{\pi\in\Pi}A_\Pi(\pi)=X\}$ werden *Trägermengen, Operationen* bzw. *Relationen* des Systems *ALS* genannt.

Bemerkung 3.1

Die Abbildungen A_S, A_Ω und A_Π werden manchmal auch in der Gestalt $(A_s)_{s\in S}$, $(o_\omega)_{\omega\in\Omega}$ und $(r_\pi)_{\pi\in\Pi}$ geschrieben. Dann schreibt man das algebraisches System (A_S,A_Ω,A_Π) in der Gestalt $((A_s)_{s\in S},(o_\omega)_{\omega\in\Omega},(r_\pi)_{\pi\in\Pi})$. Wenn die Mengen S, Ω und Π endlich sind, werden sie oft durch die Anfangsbereiche der Menge der natürlichen Zahlen ersetzt. Im Fall $S = [1..k]$, $\Omega = [1..m]$ und $\Pi = [1..n]$ schreibt man das algebraische System (A_S, A_Ω, A_Π) dann auch in der Gestalt $(A_1,\ldots,A_k,o_1,\ldots,o_m,r_1,\ldots,r_n)$ mit $A_S(i)=A_i$ für $i\in[1..k]$, $A_\Omega(j)=o_j$ für $j\in[1..m]$, und $A_\Pi(l)=A_p$ für $l\in[1..n]$.

Es gibt keine Standardschreibweise für die Werte einer mehrstelligen Operation (also einer Funktion) $\bullet:A^w\rightarrow A^s$ auf einem Tupel $(a_1,\cdots,a_k)\in A^w$. Je nach Situation werden diese vielmehr in sehr verschiedenen Formen beschrieben. Bei einer zweistelligen Operation $\bullet:A^{s_1}\times A^{s_2}\rightarrow A^s$ schreibt man den Wert z.B. in der Gestalt $\bullet(a_1,a_2)$. Man nennt dies dann *Präfixnotation.* Häufiger verwendet man hierzu aber die sogenannte *Infixnotation* $a_1\bullet a_2$. Weiter tritt die *Postfixnotation* $(a_1,a_2)\bullet$ auf. Die Wahl der Notation ist durch nichtmathematisch bedingte (z.B. traditionelle) Gründe bestimmt und wird sehr selten explizit vorgegeben.

Die Begriffe Signatur und algebraisches System sind sehr stark verbunden. In dieser Arbeit wird der Begriff der Signatur als grundlegender angesehen als der Begriff des Systems. Ein System wird als ein System **über** einer Signatur definiert. Es mag merkwürdig erscheinen, daß ein System durch eine Signatur **und** ein Tripel (A_S, A_Ω, A_Π) von Abbildungen definiert ist, denn oftmals sieht man umgekehrt ein algebraisches System als ein Tripel von Folgen

$$S=((A_1,A_2\cdots A_k),(o_1,o_2\cdots o_m),(r_1,r_2\cdots r_n))$$

definiert, für die gilt:

(1) $(A_1,A_2,\ldots,A_k)$ ist eine Folge von Mengen (die Folge der Trägermengen des Systems S),

(2) $(o_1,o_2,\ldots,o_m)$ ist eine Folge von Operationen (die Folge der Operationen des Systems S), wobei jedes Element dieser Folge eine Funktion von der Gestalt $o_i:A^{p_1}\times A^{p_2}\times\cdots\times A^{p_i}\rightarrow A^{p_{i+1}}$ $(i \le m)$ ist, und

(3) $(r_1,r_2,\ldots,r_n)$ ist eine Folge von Relationen (die Folge der Relationen des Systems S), wobei jedes Element dieser Folge eine Relation von der Gestalt $r_j\subseteq A^{p_1}\times A^{p_2}\times\cdots\times A^{p_i}$ $(i \le n)$ ist.

Man kann nun die Signatur eines solches Systems definieren, indem man den Mengen $A_1,A_2,\ldots,A_k$, den Operationen $o_1,o_2,\ldots,o_m$ und Relationen $r_1,r_2,\ldots,r_n$ Namen gibt. Wenn eine solche Benennung durch die Zuordnungen $A_i\rightarrow s_i$, $o_j\rightarrow\omega_j$, $r_l\rightarrow\pi_l$ gegeben ist, dann ist die Signatur des Systems ein 5-Tupel der Gestalt

$$(S=\{s_1,s_2,\ldots,s_k\},\Omega=\{\omega_1,\omega_2,\ldots,\omega_m\},\Pi=\{\pi_1,\pi_2,\ldots,\pi_n\},typ_\Omega,typ_\Pi),$$

wobei die Typfunktionen $typ_\Omega:\Omega\rightarrow S^*\times S$ und $typ_\Pi:\Pi\rightarrow S^+$ in der folgenden Weise definiert sind:

(a) Für jedes $i\in[1..m]$ ist $typ_\Omega(\omega_i) = typ(o_i)$ und

(b) für jedes $j\in[1..n]$ ist $typ_\Pi(\pi_j) = typ(r_j)$.

Hier ist die Signatur durch ein System **und** eine Benennung der Trägermengen des Systems bestimmt. Man kann natürlich die Klasse aller betrachteten Trägermengen (sie ist in jeder praktischen Anwendung endlich) mit natürlichen Zahlen über die Zuordnung $A_i\rightarrow i$ benennen und damit eine **kanonische Gestalt** der Signatur erhalten. Dieses Verfahren ist aber sehr unpraktisch, denn der Name einer Menge hängt dann von ihrer Stellung in der Darstellung ab, d.h. dieselbe Menge könnte sehr verschieden benannt werden. Normalerweise sind aber die Namen der Mengen konstant, und die Menge $\mathbb{R}$ der reellen Zahlen wird z.B. unabhängig vom Kontext als **die**

Menge der reellen Zahlen bezeichnet. Nach dem Auswahlaxiom kann man zwar die Menge aller Mengen des betrachteten Universum wohlordnen, was suggerieren könnte, daß es eine wirklich günstige kanonische Gestalt der Signatur gäbe. Das Auswahlaxiom gibt aber kein Verfahren für die Konstruktion dieser Ordnung, sodaß man nicht weiß, wie diese Ordnung aussehen könnte. Aus all diesen Gründen ist es sinnvoller den Begriff der Signatur als fundamentaler gegenüber dem Begriff des algebraischen Systems anzusehen. Betrachten wir einmal als ein Beispiel das algebraische System, das wir in der Einleitung benutzt haben.

Beispiel 3.1

Sei

```
SIG DMS =
    sorts= Process, Daten, Warteschlange, IN;
    ops=   ein:Warteschlange Daten → Warteschlange,
           aus:Warteschlange → Daten,
           rest:Warteschlange → Warteschlange
           laenge:Warteschlange → IN,
           daten:Process → Daten,
           process:Daten → Process,
           empf:∅ → Process,
           len:∅ → IN;
    preds= isprocess:Process,
           isdaten:Daten,
           isWarteschlange:Warteschlange,
           lt:IN IN;
SIGEND
```

eine Signatur. Das algebraische System $ALS(DMS)=(A_S,A_\Omega,A_\Pi)$ über der Signatur *DMS* definieren wir dann durch:

(1) $A_S(Process)=\{a_1,a_2,...,a_n,b\}$ sei eine Menge von Prozessen.

(2) $A_S(Daten)=\{d_1,d_2,...,d_n\}$ sei eine Menge von Datenpaketen.

(3) $A_S(Warteschlange)=A_S(Daten)^*$ sei die Menge der linearen Warteschlangen mit Elementen aus $A_S(Daten)$.

(4) $A_S(\mathbb{N})$ sei die Menge der natürlichen Zahlen.

(5) $A_\Omega(empf)\equiv b$ sei die Konstante b.

(6) $A_\Omega(len)\equiv 2$ sei die Konstante $2\in\mathbb{N}$.

(7) $A_\Omega(ein):A_S(Warteschlange)\times A_S(Daten)\rightarrow A_S(Warteschlange)$ sei definiert durch $A_\Omega(ein)(w,d)=wd$.

(8) $A_\Omega(aus):A_S(Warteschlange)\rightarrow A_S(Daten)$ sei definiert durch $A_\Omega(aus)(d_1d_2..d_k)=d_1$.

(9) $A_\Omega(rest):A_S(Warteschlange)\rightarrow A_\Omega(Warteschlange)$ sei definiert durch $A_\Omega(rest)(d_1d_2..d_k)=d_2..d_k$.

(10) $A_\Omega(laenge):A_S(Warteschlange)\rightarrow A_S(\mathbb{N})$ sei definiert durch $A_\Omega(laenge)(w)=l(w)$.

(11) $A_\Omega(daten):A_S(Process)\rightarrow A_S(Daten)$ sei definiert durch $A_\Omega(daten)(a_i)=d_i$ für alle $i=1,..,n$.

(12) $A_\Omega(process):A_S(Daten)\rightarrow A_S(Process)$ sei definiert durch $A_\Omega(process)(d_i)=a_i$ für alle $i=1,..,n$.

(13) $A^\Pi(isprocess)=A_S(Process)$.

(14) $A^\Pi(isdaten)=A_S(Daten)$.

(15) $A^\Pi(isWarteschlange)=A_S(Warteschlange)$.

(16) $A_\Pi(lt)=\leq$, wobei $\leq$ die "Kleiner-Gleich" Relation auf den natürlichen Zahlen ist.

Die Signatur *DMS* beschreibt die Syntax unseres Beispiels aus der Einleitung, wobei wir allerdings die Sorte IN, die Operation *laenge* und das Prädikat *lt* (lower then) neu eingeführt haben. *ALS(DMS)* definiert (wie wir später noch sehen werden) die Semantik der Signatur *DMS* als Beschriftung unseres Beispielnetzes.

Definition 3.3 : (Algebren)

Die algebraischen Systeme mit leerer Menge der Relationen werden *Algebren* genannt. Wir notieren Algebren daher einfach als Tupel $ALG=(A_S,A_\Omega)$ oder bezeichnen eine Algebra mit $ALG(SIG)$, wenn die Signatur der Algebra aus dem Kontext nicht ersichtlich ist.

Beispiel 3.2

Es sei *ALG* das Tupel $ALG=(\mathbb{R},C^0_{\mathbb{R}};\bullet,+)$ mit:

$\bullet:\mathbb{R}\times C^0_{\mathbb{R}}\to C^0_{\mathbb{R}}$ sei die Multiplikation von reellen Funktionen mit reellen Zahlen und

$+:C^0_{\mathbb{R}}\times C^0_{\mathbb{R}}\to C^0_{\mathbb{R}}$ sei die komponentweise Addition von Funktionen.

Das Tupel kann dann als eine Algebra der Signatur $SIG=(S=\{s_1,s_2\},\Omega=\{\omega_1,\omega_2\},\Pi=\emptyset,typ_\Omega,typ_\Pi)$ betrachtet werden, wobei $typ(\omega_1)=(s_1s_2,s_2)$ und $typ(\omega_2)=(s_2s_2,s_2)$ ist. Die Folge *supp(ALG)* der Trägermengen der Algebra ist die erste Komponente des Tupels $(\mathbb{R},C^0_{\mathbb{R}})$, die Folge *oper(ALG)* der Operationen der Algebra ist die zweite Komponente des Tupels $(\bullet,+)$. Die Folge *rel(ALG)* der Relationen ist nach Definition leer. Jeder Vektorraum ist ein Beispiel einer Algebra von diesem Typ.

Definition 3.4 : (isomorphe Signaturen)

Signaturen $SIG=(S,\Omega,\Pi,typ_\Omega,typ_\Pi)$ und $SIG'=(S',\Omega',\Pi',typ_{\Omega'},typ_{\Pi'})$ werden isomorph genannt dann und nur dann, wenn es drei Bijektionen $\phi:S\to S'$, $\gamma:\Omega\to\Omega'$ *und* $\chi:\Pi\to\Pi'$ gibt, sodaß gilt:

(1) für jedes $\omega\in\Omega$ ist: $\phi^{\#}(typ_\Omega(w)) = typ_{\Omega'}(\gamma(\omega))$ und

(2) für jedes $\pi\in\Pi$ ist $\phi^{\#}(typ_\Pi(w)) = typ_{\Pi'}(\chi(\omega))$.

Bemerkung 3.2

Es ist leicht zu sehen, daß die Relation $\Sigma \tilde{} \Sigma'$ dann und nur dann, wenn Σ und Σ' isomorph sind, eine Äquivalenzrelation ist.

Definition 3.5 : (Ähnlichkeit von algebraischen Systemen)

Algebraische Systeme *ALS* und *ALS'* sind ähnlich dann und nur dann, wenn ihre Signaturen isomorph sind.

Es ist leicht zu sehen, daß die oben definierte Ähnlichkeitsrelation für Algebraische Systeme eine Äquivalenzrelation ist.

Die Signaturen *SIG* und *SIG'* sind isomorph dann und nur dann, wenn der einzige Unterschied zwischen ihnen in der Wahl der Symbole liegt. Der Unterschied zwischen ähnlichen Systemen ist dagegen viel tiefer, denn die entsprechenden Trägermengen, Operationen oder Relationen können sehr "unähnlich" sein.

Beispiel 3.3

Die Algebren $\mathbf{Z} = (\mathbb{Z},+)$ und $\mathbf{R}^+ = (\mathbb{R},+,/)$ mit $+:\mathbb{Z}\times\mathbb{Z}\to\mathbb{Z}$ und $/:\mathbb{R}^+\times\mathbb{R}^+\to\mathbb{R}^+$ als Standardaddition der ganzen Zahlen und Division der positiven reellen Zahlen sind ähnlich. Die Mächtigkeit der Menge $\mathbb{Z}$ ist aber z.B. nicht gleich der Mächtigkeit der Menge $\mathbb{R}^+$ ($|\mathbb{Z}| \neq |\mathbb{R}^+|$). Außerdem sind die Eigenschaften der Addition andere als die Eigenschaften der Division (die Addition ist z.B. kommutativ und die Division nicht).

Wir wollen am Ende dieses Unterkapitels noch in ein paar Worten die Intuition erläutern, die zu den Begriffen des algebraischen Systems und der Signatur führt.

Um mathematische Modelle für verschiedene Teilaspekte der "realen Welt" zu gewinnen, gruppiert man die "Objekte dieser Welt" in Mengen verschiedener Sorten so, daß die Beziehungen zwischen ihnen als Relationen auf den Mengen dargestellt werden können. Dabei faßt man aus technischen Gründen einige dieser Relationen, die durch ihre Funktionalität gekennzeichnet sind, als *Funktionen* auf. Unser Modell des ausgewählten Fragments der "realen Welt" sieht damit so aus:

Reale Objekte	Beziehungen zwischen Objekten	
$(A_1, \ldots, A_k,$	$f_1, \ldots, f_m,$	$r_1, \ldots, r_n)$
Mengen	Funktionen	Relationen ohne Funktionen

Die Folge $(A_1, ..., A_k, f_1, ..., f_m, r_1, ..., r_n)$ stellt dann das gesuchte Modell dar.

Weiter sollte man aber angeben, von welchem *Typ* die Funktionen $f_1, ..., f_m$ und die Relationen $r_1, ..., r_n$ sind, d.h. man sollte ihre Definitions- und Wertebereiche angeben. Dies könnte einfach dadurch geschehen, daß man die Funktionen explizit in der Gestalt $f_1:A_{i_1}\times A_{i_2}\times\ldots\times A_{i_{p_i}}\rightarrow A_{i_0}$ mit $i_0, i_1, ..., i_{p_i} \leq k$ angibt.

Diese Schreibweise ist aber sehr aufwendig und im allgemeinen nicht gut lesbar. Sie ist weiter eindeutig durch das Wort $(i_1, ..., i_{p_i}, i_0)$ bestimmt, sodaß man den Typ der Funktion auch als das Wort kodieren kann. Die Zeichen, aus denen diese Wörter gebildet werden, sind dabei die *Namen* der Mengen $A_{i_1}, ..., A_{i_{p_i}}$ und A_0.

Entsprechend kann man auch den Typ einer Relation definieren.

In dieser Hinsicht enthält die Signatur eines Systems genau die Typinformationen der Funktionen und Relationen eines algebraischen Systems, die man nur mit Hilfe der Sortennamen (Mengennamen) ausdrücken kann.

3.2 Polynome und Relationen in algebraischen Systemen

Wir wollen hier zuerst an den Begriff der *Produktklammernoperation* erinnern. Für eine Familie $(f_i:X\rightarrow Y_i)_{i\in I}$ von Funktionen mit demselben Definitionsbereich X definiert die Zuordnung $<f_I>:x\rightarrow(f_i(x))_{i\in I}$ eine Abbildung $<f_I>:X\rightarrow\underset{i\in I}{\times} Y_i$ der Menge X in die Produktmenge $\underset{i\in I}{\times} Y_i$. Wir nennen diese Abbildung die Produktklammernoperation.

Beispiel 3.4 : (Beispiel zur Produktklammernoperation)

Eine Gerade in der Ebene wird oft durch eine zweielementige Familie von Funktionen

$$G = (t \rightarrow kt + l, t \rightarrow mt + n)$$

beschrieben. Diese Familie bestimmt eindeutig die Abbildung $g : t \rightarrow (kt + l, mt + n)$, die gerade der Wert der oben erwähnten Produktklammernoperation angewandt auf die Familie G ist. Wir haben also

$$g = < G >.$$

Es sei $ALS=((A_s)_{s\in S},(o_i)_{i\in I},(r_j)_{j\in J})$ ein algebraisches System. Alle Funktionen, die aus Projektionen der Gestalt $pr_{i_0}:\underset{i\in I}{\times} A_i\rightarrow A_{i_0}$, $i_0\in I$ und Operationen des algebraischen Systems *ALS* mit der Hilfe der Produktklammeroperation, dem Produkt einer Familie von Funktionen und der Operation der Zusamensetzung von Funktionen gewonnen werden können, heißen Polynome des algebraischen Systems *ALS*. Präziser definieren wir:

Definition 3.6 : (Polynome eines algebraischen Systems)

Die Menge aller Polynome eines algebraischen Systems $ALS=(A_S,A_\Omega,A_\Pi)$ ist die kleinste Menge *POL(ALS)*, die die folgenden Bedingungen erfüllt:

(1) Für jedes Wort $u:[1..n]\rightarrow S$ mit $u\in S^*$ und jedes $i\in dom(u)$ ist die Funktion $pr_i^u:A_S^u\rightarrow A_S^{u(i)}$ mit der Zuordnung

$$(x_1,x_2,\cdots,x_{i-1},x_i,x_{i+1},\cdots,x_n)\rightarrow x_i\in A_S^{u(i)}$$

ein Element der Menge *POL(ALS)*.

(2) Für jedes $i\in I$, jedes Wort $u\in S^*$, jede Operation $A_\Omega(\omega)$ mit $typ_\Omega(\omega)=(w,s)$ und jede Familie $(p_j^u:A_S^u\rightarrow A_S^{w(j)})_{j\in dom(w)}$ mit $p_j^u\in POL(ALS)$ ist die Funktion $A_\Omega(\omega)\circ<p^u_{dom(w)}>:A_S^u\rightarrow A_S^s$ mit der Zuordnung

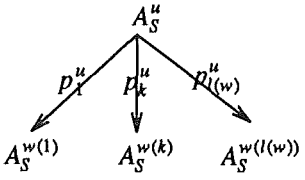
$$(x_1,x_2,..,x_{n_i})\rightarrow A_\Omega(\omega)(p_1^u(x_1,..,x_{n_i}),p_2^u(x_1,..,x_{n_i}),\ \cdots\ ,p^u_{l(w)}(x_1,...,x_{n_i}))$$

ein Element der Menge *POL(ALS)*.

Bedingung (1) heißt mit anderen Worten, daß für jedes Wort $u=s_1s_2\cdots s_m\cdots s_{n_i}$ die Projektion

$$A_S^{s_1}\times A_S^{s_2}\times\ldots\times A_S^{s_m}\times\ldots\times A_S^{s_{n_i}}\to A_S^{s_i}$$
$$(x_1,x_2,\ldots,x_m,\ldots,x_{n_i})\to x_m$$

der Menge *POL(ALS)* angehört. Bedingung (2) heißt, daß für jedes Wort $u\in S^*$, jedes $\omega\in\Omega$ mit $typ_\Omega(\omega)=(w,s)$ und jede Familie



von Polynomen mit $w=s_1\cdots s_m$ die Zusammensetzung

$$A_S^u \xrightarrow{<p^u_{dom(w)}>} A_S^{s_1}\times\ldots\times A_S^{s_k}\times\ldots\times A_S^{s_m} \xrightarrow{o_i} A_S^s$$

der Menge *POL(ALS)* angehört.

Beispiel 3.5

Betrachten wir ein algebraisches System $S_A=(I,O,S,\lambda,\mu)$ mit den Trägermengen I für Input, O für Output, S für State und den Operationen $\lambda:I\times S\to S$ als Inputoperation und $\mu:I\times S\to O$ als Outputoperation (z.B. ein Mealy-Automat, d.h. einen deterministischen Pushdown Automaten mit Outputfunktion). Es sei nun $w=IIIS$. Die Funktion

$$p=\mu(pr_3^w,\lambda(pr_2^w,\lambda(pr_1^w,pr_4^w)))$$

ist ein Polynom des Systems, denn es gilt:

- Die Projektionen pr_3^w, pr_2^w, pr_1^w und pr_4^w sind nach (1) Polynome des Systems S_A.
- Die Abbildung $p_1=\lambda\circ<pr_1^w,pr_4^w>:I\times I\times I\times S\to S$ ist ein Polynom des Systems auf Grund des vorherigen Punktes und (2).
- Die Abbildung $\mu\circ<pr_3^w,p_1>:I\times I\times I\to O$ ist ein Polynom auf Grund des vorhergehenden Punktes und (2).

Alle "Inputstellen" des Diagrammes (in der Zeichnung mit Doppelkreisen markiert) in Abb. 3.2-1 sind mit Funktionen der Gestalt $A^w\to A^s$, $w\in\{I,O,S\}^*$ und $s\in\{I,O,S\}$, oder Operationen des Systems markiert. Die Kästen des Diagrammes stellen die Produktklammernoperationen und die Kompositionen der einzelnen Funktionen dar.

Das Polynom p beschreibt das Verhalten des Automats S_A bei der Eingabe des letzten von zwei Inputsymbolen. Wenn der Anfangszustand des Automaten S_A s_0 ist und am Eingang des Automaten das Wort *ab* ansteht, wird das letzte am Ausgang des Automaten erhaltene Symbol *y* gleich dem Wert des Polynoms *p* angewandt auf das Tripel (b,a,s_0) sein. Es gilt also: $y=p(b,a,s_0)=\mu(b,\lambda(a,s))$.

Definition 3.7 : (einfache Relationen)

Die Menge der einfachen Relationen des Systems *ALS* ist die kleinste Menge *REL(ALS)*, die der folgenden Bedingung genügt:

- für jedes j∈J, jedes Wort $u=s_1s_2\cdots s_n$ mit $u\in S^*$, jedes $\pi\in\Pi$ mit $typ_\Pi(\pi)=w$ und jede Familie von Polynomen $(p_l^u:A_S^u\to A_S^{w(l)})_{l\in dom(w)}$ des Systems *ALS* gehört die Relation $A_\Pi(\pi)\circ<p^u_{dom(w)}>\subseteq A_S^u$ der Menge *REL(ALS)* an.

Die obige Definition kann man sich auch folgendermaßen erklären: Für jede Familie $(p_i^u:A_S^u\to A^{w(l)})$ gehört die Folge α der Relation $A_\Omega(\omega)\circ<p^u_{len(u_j)}>$ dann und nur dann der Menge A_S^u an, wenn die Folge $\alpha_p=(p_{w(1)}(\alpha),p_{w(2)}(\alpha),\ldots,p_{w(l(w))}(\alpha))$ der Menge (Relation) $A_\Omega(\omega)$ angehört. Es ist hierbei merkwürdig, daß die Relation $A_\Omega(\omega)\circ<p>$ von einem anderem Typ als die Relation $A_\Omega(\omega)$ ist. Typische Beispiele für solche

Relationen sind Gleichungen und Ungleichungen.

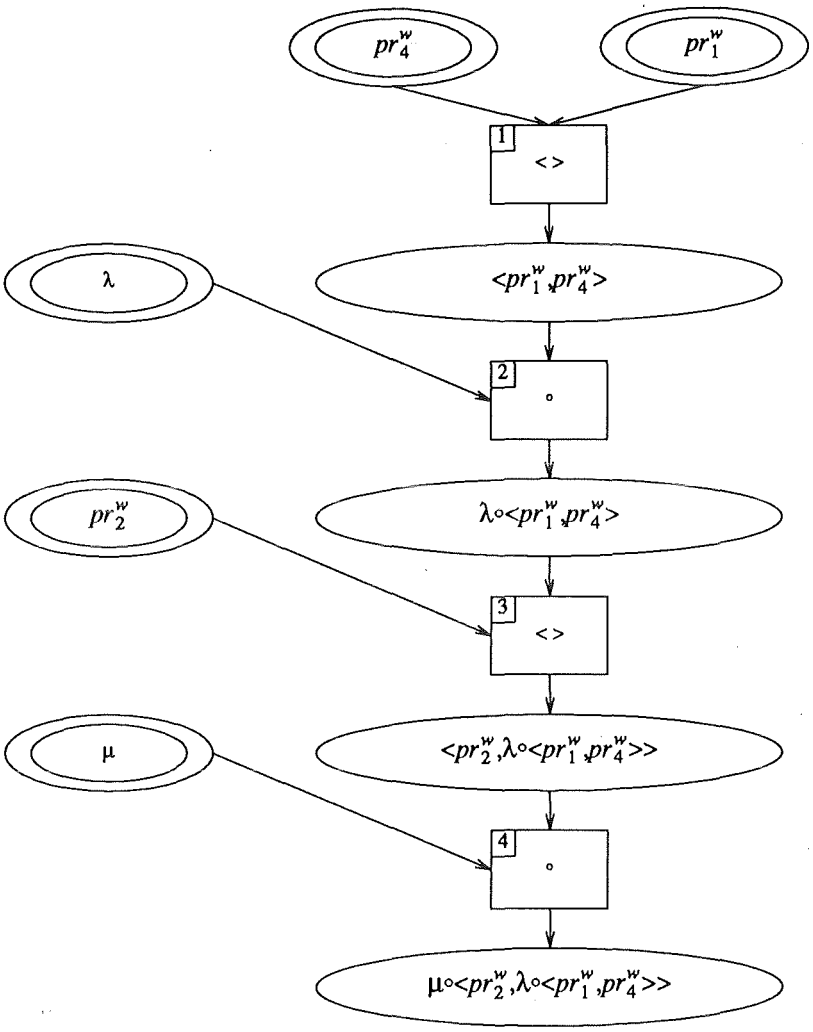


Abb. 3.2–1 : Konstruktion von Polynomen

Beispiel 3.6

Betrachten wir einmal das System $S_{\mathbb{R}}=(\mathbb{R};+,\bullet,\phi;\leq)$ mit den folgenden Eigenschaften:

(1) $\mathbb{R}$ sei die Menge der reellen Zahlen,

(2) $+:\mathbb{R}\times\mathbb{R}\to\mathbb{R}$ sei die Standardaddition reeller Zahlen,

(3) $\bullet:\mathbb{R}\times\mathbb{R}\to\mathbb{R}$ sei die Standardmultiplikation reeller Zahlen,

(4) $\phi:\emptyset\to\mathbb{R}$ sei die Nulloperation und

(5) $\leq\subseteq\mathbb{R}\times\mathbb{R}$ sei die übliche Kleiner-Gleich Ordnungsrelation auf den reellen Zahlen.

Die Relation $r\subseteq\mathbb{R}\times\mathbb{R}\times\mathbb{R}$, mit $(x,y,z)\in r$ per Definition genau dann wenn $x+y\leq x\bullet z+y$, ist eine einfache Relation des Systems $S_{\mathbb{R}}$, denn es gilt:

(1) Die Funktionen $p_1(x,y,z)=x+y$ und $p_2(x,y,z)=x\bullet z+y$ sind Polynome des Systems $S_{\mathbb{R}}$.

(2) Da $\leq\subseteq\mathbb{R}\times\mathbb{R}$ eine Relation des Systems $S_{\mathbb{R}}$ ist, und r von der Gestalt $\leq\circ<p_1,p_2>$ ist, ist r eine einfache Relation von $S_{\mathbb{R}}$.

Wir beschreiben jetzt eine Konstruktion, die in den folgenden Kapiteln eine sehr große Rolle spielen wird.
Es sei $SIG=(S,\Omega,\Pi,typ_\Omega,typ_\Pi)$ eine Signatur und $V_S:S\to\mathbf{Set}$ eine Abbildung.

Definition 3.8 : (Die Sprache der Terme und Typ von Termen)

Die *Sprache der Terme* der Signatur *SIG* mit Variablen aus den Mengen $(V_S(s))_{s\in S}$ $TERM(SIG,V_S)$ und der *Typ von Termen* $typ(t)$ seien wie folgt rekursiv definiert:

(1) für jedes $s\in S$ und jedes $v\in V_S(s)$ gehört das Wort v der Sprache $TERM(SIG,V_S)$ an und es ist $typ(v)=s$,

(2) für jedes $\omega\in\Omega$ mit $typ(\omega)=(w,s)$ und jede Folge $t_1,t_2,\ldots,t_k\in TERM(SIG,V_S)$ mit $typ(t_1)typ(t_2)...typ(t_k)=w$ gehört das Wort $t=\omega(t_1,t_2,\ldots,t_k)$ der Sprache $TERM(SIG,V_S)$ an und es ist $typ(t)=s$.

Nach der obigen Definition ist jede Variable (d.h. ein Element einer der Mengen V_s , $s\in S$) ein Term des Typs s und die Konkatenation von einem Operator ω und einer Folge von Termen, die vom Typ $typ_\Omega(\omega)$ ist, ergibt auch einen Term vom Typ $typ_\Omega(\omega)$.

Definition 3.9 : (Algebra der Terme)

Sei $SIG=(S,\Omega,\Pi,typ_\Omega,typ_\Pi)$ eine Signatur, $V_S=(V_s)_{s\in S}$ eine Familie von Variablen und $TERM(SIG,V)$ die zugehörige Menge von Termen.

(a) Für jedes $s\in S$ bezeichnen wir mit $TERM_s(SIG,V)$ die Menge aller Terme vom Typ s.

(b) Für jeden Operator $\omega\in\Omega$ mit $typ_\Omega(\omega)=(w=s_1\ldots s_k,s)$ definieren wir die Operation

$$\bar{\omega}:TERM_{s_1}(SIG,V)\times TERM_{s_2}(SIG,V)\times...\times TERM_{s_k}(SIG,V)\rightarrow TERM_s(SIG,V)$$

durch $\bar{\omega}(t_1,...,t_k)=\omega(t_1,\ldots,t_k)$. Man beachte, daß dann gilt: $typ(\bar{\omega})=A_S(typ_\Omega(\omega))$.

(c) Die *Algebra der Terme* der Signatur *SIG* mit Variablen aus der Familie $VALG(SIG,V)=(A_S,A_\Omega)$ sei wie folgt definiert:

(i) $A_S:S\rightarrow$ **Set** sei definiert durch $A_S(s)=TERM_s(SIG,V)$ für alle $s\in S$.

(ii) $A_\Omega:\Omega\rightarrow$ **Func** sei definiert durch $A_\Omega(\omega)=\bar{\omega}$ für alle $\omega\in\Omega$.

Betrachten wir zur Definition der Terme ein Beispiel:

Beispiel 3.7

Es sei die Signatur $SIG=(S,\Omega,\Pi,typ_\Omega,typ_\Pi)$ gegeben, wobei wir als Operationssymbole + und • für (ss,s)-Operatoren , – für einen (s,s)-Operator und als Relationssymbol δ für eine (ss)-Relation nehmen. Die Mengen Ω und Π haben dann die Gestalt: $\Omega=(+,\bullet,-)$ und $\Pi=(\delta)$. Da die Menge der Sorten einelementig ist, brauchen wir nur eine Sorte von Variablen und die Abbildung $V_S:S\rightarrow$ **Set** läßt sich durch Angabe von $V_S(s)=\{x,y,z\}$ eindeutig beschreiben. Wir haben also nur die drei Variablen x, y, z von der Sorte s.

Weiter müssen wir noch die Schreibweise für die Wörter der Sprache $TERM(SIG,V)$ festlegen. Wir übernehmen hierbei die Konventionen, daß wir zweistellige Operationen und Relationssymbole in *Infixnotation* und einstellige Operationen in *Präfixnotation* schreiben. Damit erscheinen die Symbole +, • und δ zwischen ihren Argumenten und das – Symbol vor seinem Argument. Weiter benutzen wir runde Klammern, um Teilausdrücke eindeutig zu kennzeichnen. Das Wort $w=(x+y)z+x$ ist dann ein Term der Signatur *SIG*, da gilt:

(1) x und y sind Variable der Signatur *SIG* und damit nach Bed. (1) der Definition 3.8 Terme,

(2) aus (1) und der Bedingung (2) der Definition 3.8 folgt, daß das Wort $x+y$ ein Term von *SIG* ist,

(3) aus der Bedingung (1) der Definition 3.8 folgt: z ist ein Term der Signatur *SIG*,

(4) aus (2), (3) und der Bedingung (2) der Definition 3.8 folgt dann, daß $(x+y)z$ ein Term ist,

(5) aus (1) und (4) folgt mit der Bedingung (2) der Definition 3.8 , daß $(x+y)z+x$ ein Term der Signatur *SIG* ist.

(6) aus (1) und (4) folgt mit der Bedingung (2) der Definition 3.8 , daß $(x+y)z+x$ ein Term der Signatur *SIG* ist.

Das Bild 3.2-2 verdeutlicht die Situation.

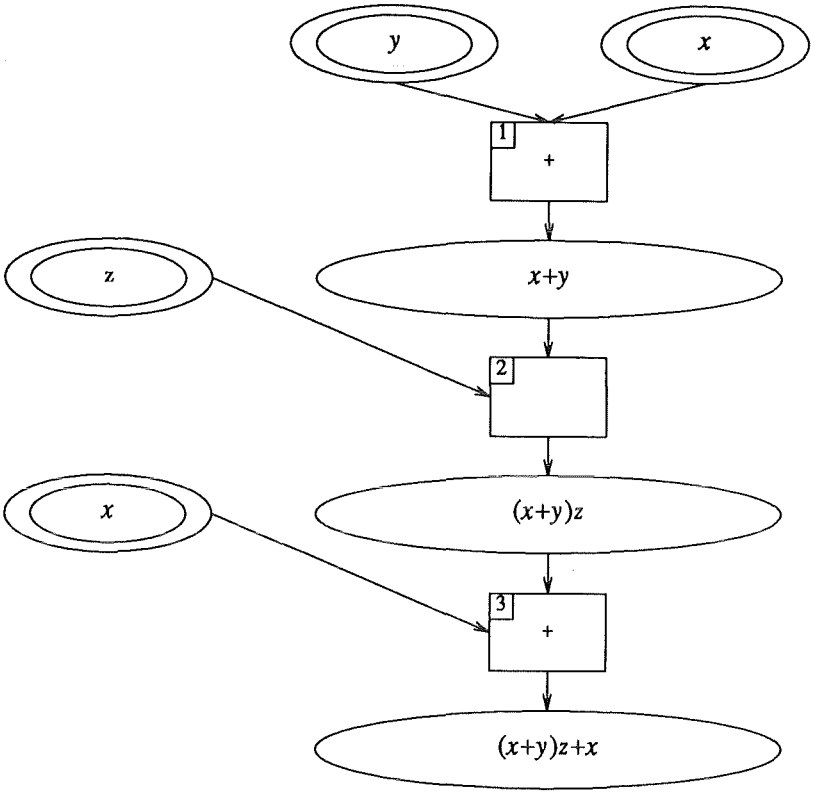


Abb. 3.2–2 : Konstruktion von Termen

Die Konstruktionsregeln (1) und (2) aus Definition 3.8 werden im letzten Beispiel zum "Bau" des Terms w ausgenutzt. Oftmals befindet man sich aber in genau der umgekehrten Situation, daß man ein Wort gegeben hat und feststellen muß, ob dieses Wort ein Term einer zugrundeliegenden Sprache ist oder nicht. Formaler muß man hierzu zeigen, daß gilt:

(a) Als erstes müssen die Symbole des Wortes w entweder Variablensymbole, Operationssymbole, Relationssymbole oder die speziellen Symbole (,) sein. Es muß also gelten:

$$/w/\subseteq(\bigcup_{s\in S}V_S(s))\cup\Omega\cup\Pi\cup\{(,)\}.$$

(b) Wenn die Bedingung (a) erfüllt ist, muß man weiter zeigen, daß das Wort w nach den Regeln (1) und (2) aufgebaut ist. Hierzu muß man das Wort w so zerlegen, daß man bei jedem Zerlegungsschritt nur eine der Bedingungen (1) und (2) anwendet, und am Ende nur noch Variablen der Sprache übrig hat. Wenn man eine solche Zerlegung gefunden hat, ist w ein Term aus $TERM(SIG,V_S)$ und sonst nicht.

Es ist hier wertvoll zu wissen, daß im Fall der Existenz einer solchen Zerlegung diese schon eindeutig bestimmt ist. Formal schreibt man Zerlegungen häufig in der folgenden Form auf:

Notation

Man definiert Konstruktionsregeln auf Grund der Bedingungen (1) und (2), indem man spezielle syntaktische Variablen für Teilausdrücke in Termen einführt (wir schreiben diese im folgenden griechisch) und diese dann mit den Operatoren und Relationen in der folgenden Weise als Regel darstellt:

$$\frac{\alpha_1,\alpha_2,\ldots,\alpha_k}{\chi(\alpha_1,\alpha_2,\ldots,\alpha_k)}$$

heißt, daß sich mit Hilfe des Operators χ aus den Teilausdrücken α_1, α_2, ..., α_k der Term $\chi(\alpha_1,\alpha_2,\ldots,\alpha_k)$ bilden läßt, wobei χ ein k-stelliger Operator vom entsprechenden Typ sein muß.

Betrachten wir dies einmal an unserem vorangegangenen Beispiel:

Beispiel 3.8

Die Konstruktionsregeln für die Sprache $TERM(SIG,V_S)$ aus dem Beispiel 3.7 kann man dann folgendermaßen schreiben:

(K_x) x ist ein Term der Signatur SIG.

(K_y) y ist ein Term der Signatur SIG.

(K_z) z ist ein Term der Signatur SIG.

(K_+) $\frac{\varsigma_1,\varsigma_2}{\varsigma_1+\varsigma_2}$, d.h. Sind ς_1 und ς_2 Terme, so ist auch das Wort $\varsigma_1+\varsigma_2$ ein Wort.

($K_\bullet$) $\frac{\varsigma_1,\varsigma_2}{\varsigma_1\bullet\varsigma_2}$, d.h. Sind ς_1 und ς_2 Terme, so ist auch das Wort $\varsigma_1\bullet\varsigma_2$ ein Wort.

(K_-) $\frac{\varsigma_1}{-\varsigma_1}$, d.h. Ist ς_1 ein Term, so ist auch das Wort $-\varsigma_1$ ein Wort.

Die Analyse des Wortes $w=(x+y)z+x$ zeigt die Abb. 3.2-3

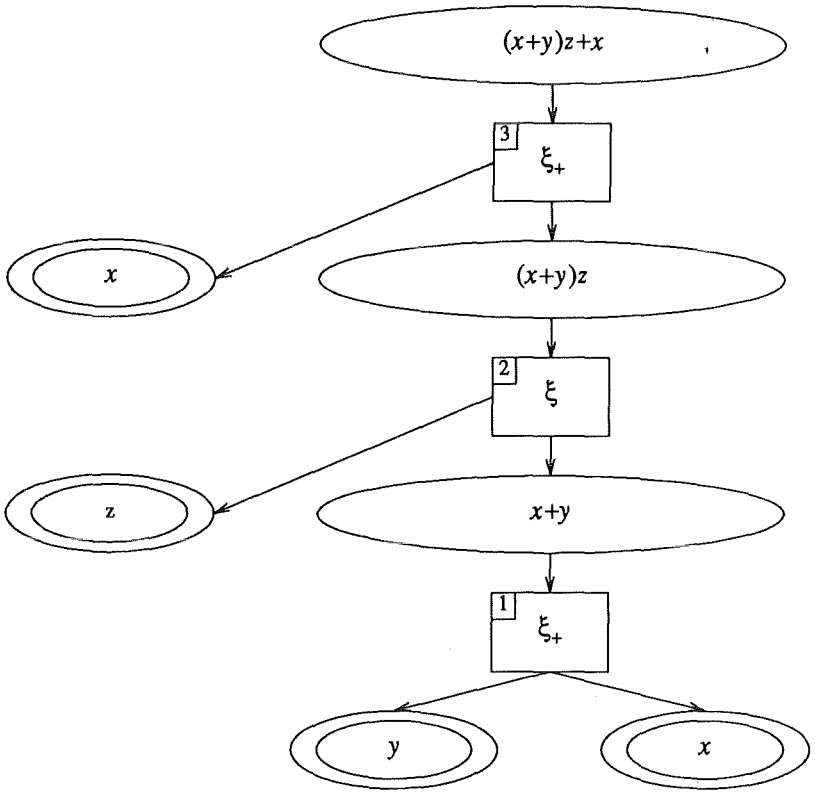


Abb. 3.2–3 : Zerlegung des Worts $(x+y)z+x$

Damit dieses Verfahren wirklich deterministisch anwendbar ist, müßte auf jeder Zerlegungsstufe genau bekannt sein, welche der verschiedenen Zerlegungsregeln auf dieser Stufe anzuwenden ist. In dem obigen Beispiel ist dies sehr einfach zu entscheiden, da wir implizit durch Klammerung des Teilausdrucks $x+y$ und durch das Vorhandensein der normalen Operatorpräzedenz $\bullet$ vor +, eine Eindeutigkeit der Auswahl der Zerlegungsregel nahelegen.

Um die Eindeutigkeit einer Zerlegung gewährleisten zu können, muß man in der Tat eine Präzedenz zwischen den einzelnen Zerlegungsregeln und/oder eine Klammerung von Teilausdrücken einführen. In der Regel ordnet man daher Operatoren einer Sprache explizit eine Priorität zu, die die Präzedenz der Operatoren untereinander beschreibt. Bei der Konstruktion von Termen bildet man dann zunächst die Teilausdrücke, die mit Operatoren niedrigerer Priorität gebildet werden und schreitet zu denen mit höherer Priorität fort. Teilausdrücke, die aus Operatoren niedrigerer Priorität bestehen, und mit Operatoren höherer Priorität verknüpft werden, setzt man

dann in Klammern, wobei den Klammern eine höhere Priorität als sämtlichen Operatoren zugeordnet ist. Bei der Analyse verfährt man dementsprechend umgekehrt.

Zur Beschreibung eines Operators muß man dann neben dem Typ des Operators (d.h. das Paar (w_i,s_i)) die Priorität des Operators angeben. Man nennt so eine Beschreibung dann auch das *Format des Operators*.

Beispiel 3.9

Die Formate der Operatoren aus den vorhergehenden Beispielen können z.B. so aussehen:

\+ hat das Format (ss,s), Infixnotation, Priorität 1,

• hat das Format (ss,s), Infixnotation, Priorität 2,

– hat das Format (s,s), Präfixnotation, Priorität 3.

Zusätzlich verlangen wir:

(,) haben die Priorität 4, d.h. Klammern haben die höchste Priorität.

Unser Beispielterm $(x+y)\bullet z+x$ ist dann eindeutig interpretierbar.

Die oben beschriebenen Präzedenzregeln für die Operatoren geben mehr Informationen über die Sprache $TERM(SIG,V_S)$, wie sie in der Signatur *SIG* enthalten sind. In Standardbeispielen, wie dem von uns gewähltem Beispiel, kann man diese zusätzlichen Informationen ohne weiteres als implizit gegeben voraussetzen. In weniger bekannten Fällen muß man allerdings diese zusätzliche Information explizit angeben. Die Gesamtheit aller benötigten Informationen über die Variablen, die Operatoren und Prädikate eines nach der Konstruktion in Definition 3.8 erhaltenen Systems nennt man oft die *Spezifikation des Systems*.

Beispiel 3.10

Eine mögliche Spezifikation für unser System $TERM(SIG,V)$ aus dem Beispiel 3.9 könnte folgendermaßen aussehen:

Variablen: x,y,z vom Typ s,

Operatoren:			
Bezeichnung	Typ(Stelligkeit)	Schreibweise	Priorität
+	(ss,s)	Infixnotation	1
•	(ss,s)	Infixnotation	2
-	(s,s)	Präfixnotation	3

Prädikatssymbole		
Bezeichnung	Typ(Stelligkeit)	Schreibweise
δ	(ss)	Infixnotation

Neben der Sprache der Terme wollen wir nun noch eine Sprache $FORM(SIG,V_S)$ definieren, deren Elemente wir die elementaren Formeln der Signatur *SIG* nennen werden.

Definition 3.10 : (elementare Formeln einer Signatur)

$FORM(SIG,V_S)$ sei die kleinste Sprache, die die folgende Bedingung erfüllt:

für jedes $\pi\in\Pi$, jede Folge $(t_1,t_2,...,t_k)$ von Termen mit $typ_\Pi(\pi)=typ(t_1)...typ(t_k)$ gehört das Wort $\pi((t_1,t_2,...,t_k))$ der Sprache $FORM(SIG,V_S)$ an.

Die Elemente dieser Sprache werden *elementare Formeln der Signatur SIG* genannt.

Manchmal ist es günstig, eine Untersprache von *atomaren Formeln* einer Signatur *SIG* zu betrachten. Eine atomare Formel ist ein Wort der Gestalt $\pi(x_1,..,x_k)$ mit π und k wie in Definition 3.10. Man kann dies so ausdrücken, daß atomare Formeln die elementaren Formeln sind, die nur aus Variablen, Prädikatssymbolen und Klammern konstruiert werden.

Die Elemente der Vereinigung $\cup_{s\in S}TERM_s(SIG,V_S)\cup FORM(SIG,V_S)$ nennt man *Ausdrücke der Signatur SIG*.

Notation

Wir werden im Folgenden häufig Tupel verwenden, deren Komponenten Elemente von einer Sorte sind. Daher führen wir hierfür eine einfachere Notation ein, wo dies zu keinen Mißverständnissen führt, und schreiben für ein Tupel von Termen $(t_1,t_2,\dots,t_k)$ einfach **t** und für ein Tupel von Variablen $(x_1,x_2,\dots,x_n)$ einfach **x**.

Die Definition des algebraischen Systems ist dadurch sehr kompliziert, daß wir Elemente verschiedener Sorten betrachten. In der Theorie betrachtet man häufig nur algebraische Systeme mit einer Sorte. Die Abbildung $A_S:S\to\mathbf{Set}$ ist dann eindeutig durch die Menge $A_S(s)$ bestimmt. Wir schreiben daher in diesem speziellen Fall vereinfachend $ALS=(A,(o_i)_{i\in I},(r_j)_{j\in J})$ mit $o_i:A^{n_i}\to A$ und $r_j\subseteq A^{m_j}$ für ein algebraisches System *ALS* über der Signatur $SIG=(S=\{s\},\Omega,\Pi,typ_\Omega,typ_\Pi)$. Die Zahlen n_i und m_j werden dann die *Stelligkeiten* der Operationen o_i bzw. der Relationen r_j genannt. Die Typfunktionen eines solchen Systems sind eindeutig durch die zwei Folgen $(n_i)_{i\in I}$ und $(m_j)_{j\in J}$ bestimmt.

Die Terme einer Signatur *SIG* kann man als Bezeichnung für die Polynome eines algebraischen Systems mit der Signatur *SIG* interpretieren, d.h. wenn wir die Operatoren eines Termes $t\in TERM(SIG,V_s)$ als Namen für zugehörige Funktionen eines algebraischen Systems deuten, so können wir den Term als Bezeichnung für ein Polynom des algebraischen Systems auffassen, das man erhält, wenn wir sukzessive die Operatoren durch die zugehörigen Funktionen ersetzen. Wir wollen diese Idee nun genauer beschreiben.

Definition 3.11 : (Auswertung von Termen)

Sei $SIG=(S,\Omega,\Pi,typ_\Omega,typ_\Pi)$ eine Signatur, $V_S:S\to\mathbf{Set}$ eine Familie von Variablen, $TERM(SIG,V_S)$ die Menge der Terme der Signatur *SIG* mit Variablen der Famile V_S und $ALG=(A_S,A_\Omega)$ eine Algebra über der Signatur *SIG*.

(a) Eine *Auswertung der Variablen* V_S in ALG ist eine Familie von Abbildungen $\nu=(\nu_s:V_S(s)\to A_S(s))_{s\in S}$.

(b) $Val(N_S,A_S)$ sei die Menge aller Auswertungen $\nu=(\nu_s:V_S(s)\to A_S(s))_{s\in S}$ von Variablen aus der Familie V_S in der Familie A_S.

(c) Eine Auswertung der Terme $t\in TERM(SIG,V_S)$ ist eine Abbildung $\bar\nu_T:TERM(SIG,V_S)\to\bigcup_{s\in S}A_S(s)$, die wie folgt rekursiv definiert ist.

(i) Für $t=x\in V_S(s)$ ist $\bar\nu_T(t)=\nu_s(t)$.

(ii) Für einen Term $t=\omega(t_1,t_2,\dots,t_k)$ ist $\bar\nu_T(t)=A_\Omega(\omega)(\bar\nu_T(t_1),\bar\nu_T(t_2),\dots,\bar\nu_T(t_k))$.

(d) Auswertungen von Variablen der Gestalt $\nu=(\nu_s:V_S(s)\to ALG(SIG,V_S))_{s\in S}$ bezeichnet man auch als (Term)-Substitutionen.

Definition 3.12 : (Realisation der Terme)

Sei $SIG=(S,\Omega,\Pi,typ_\Omega,typ_\Pi)$ eine Signatur, $V_S:S\to\mathbf{Set}$ eine Familie von Variablen, $TERM(SIG,V_S)$ die Menge der Terme der Signatur *SIG* mit Variablen der Famile V_S und $ALG=(A_S,A_\Omega)$ eine Algebra über der Signatur *SIG*. Die durch *ALG* bestimmte *Realisation der Terme über der Signatur SIG* ist die Abbildung

$$\mathbf{R}_T:TERM(SIG,V_S)\to[Val(V_S,A_S)\to\bigcup_{s\in S}A_S(s)],$$

die durch die folgenden Regeln eindeutig bestimmt ist:

Für jeden Term $t\in TERM(SIG,V_S)$ ist

$$\mathbf{R}_T(t)=t_{\mathbf{R}}:Val(V_S,A_S)\to\bigcup_{s\in S}A_S(s)$$

definiert durch

$$t_{\mathbf{R}}(\nu)=\bar\nu_T(t)\quad\text{für alle }\nu\in Val(V_S,A_S).$$

Es ist bei den obigen Voraussetzungen bemerkenswert, daß jede Interpretation (und damit auch jede Realisation) durch Angabe des entsprechenden algebraischen Systems (oder der Algebra) eindeutig bestimmt ist. In diesem Sinne kann man von *der* (eindeutig bestimmten) Realisation der Terme in einem algebraischen System (in einer Algebra) sprechen.

3.3 Hinweise zur Literatur

Der Begriff des algebraischen Systems wird in vielen Büchern beschrieben. Algebraische Systeme werden oft auch relationale Systeme oder Modelle genannt und vor allem im Zusammenhang mit verschiedenen Anwendungen der Logik betrachtet. Dabei werden meistens nur Systeme mit einer Trägermenge verwendet, weil man alle wesentlichen Eigenschaften algebraischer Systeme auch in diesem Spezialfall untersuchen kann und Systeme mit mehreren Trägermengen weitaus komplexer sind. Leider sind aber die meisten in der Praxis vorkommenden Systeme auf natürliche Weise Systeme mit mehreren Trägermengen. Aus der Sicht der Praxis ist es daher besser, vielsortige algebraische Systeme zu betrachten. Diese Situation ist besonders bei den Programmiersprachen sichtbar, da hier verschiedene Variablensorten auf Grund der verschiedenen Variablentypen einer Sprache gegeben sind.

Eine ähnliche Situation haben wir, wenn es um Algebren geht. Man kann auf die Betrachtung algebraischer Systeme überhaupt verzichten und nur Algebren mit mehreren Trägermengen betrachten, indem man die charakteristischen Funktionen der Relationen statt den Relationen als Operationen einführt. Dies führt zwar zu einer viel einfacheren Theorie, ist aber in der Praxis sehr selten gewünscht. So haben z.B. die meisten Programmiersprachen Prädikat(ssymbol)e wie $\leq,=,<$.

Als Einführung in die generelle Theorie der universellen Algebra kann das Buch von G. Grätzer, "Universal Algebra"([Gr83a]), dienen. Es enthält viele Übungen und ein umfangreiches Literaturverzeichnis. Eine Abhandlung der universellen Algebra auf abstrakteren Niveau findet man in P. M. Cohn, "Universal Algebra"([Coh65a]). Eine sehr einfache Einführung in die elementaren Grundbegriffe der Algebra gibt H. Lugowski in "Grundzüge der universellen Algebra"([Lug76a]).

Es gibt viele verschiedene Formalisierungen des Begriffs des algebraischen Systems. Wir benutzen hier die Definitionen im Sinne von [Man76a, Gr83a, Lug76a].

Die Konstruktion der Termalgebra und einige ihrer Eigenschaften werden genauer von Grätzer ([Gr83a]) und Manes ([Man76a]) beschrieben. Die Beschreibung von Manes ist dabei sehr kompakt und abstrakt, während die von Grätzer einfacher erscheint. Für vielsortige Algebren wird diese Konstruktion weiter in ([Lug76a]) beschrieben.

Eine sehr interessante Beschreibung der Substitutionen und deren Eigenschaften kann man bei Manes ([Man76a]) finden. Sie werden weiterhin in fast jeder Arbeit aus dem Gebiet der theoretischen Grundlagen der Informatik beschrieben. Als Beispiel diene [Sal70a].

4 Logik

4.1 Formale Sprachen

Wir werden im folgenden bei der Realisation von formalen Sprachen 1. Ordnung Algebren einer speziellen Gestalt benötigen, die man logische Matrizen nennt.

Definition 4.1 : (logische Matrizen)

Sei $LSIG=(S_0=\{s_0\},\Omega_0=L_0\cup L_1\cup L_2,typ_{\Omega_0})$ mit $L_0=\{\mathsf{v},e_1,\cdots,e_k\}$, $L_1=\{o^1,\cdots,o^l\}$, und $L_2=\{o_1,\cdots,o_m,\rightarrow\}$ eine Signatur mit $typ_{\Omega_0}(\mathsf{v})=(\varepsilon,s_0)$, $typ_{\Omega_0}(e_i)=(\varepsilon,s_0)$ für alle $i=1,..,k$, $typ_{\Omega_0}(e^i)=(s_0,s_0)$ für alle $i=1,..,l$, $typ_{\Omega_0}(o_i)=(s_0s_0,s_0)$ für alle $i=1,..,m$ und $typ_{\Omega_0}(\rightarrow)=(s_0s_0,s_0)$. Eine Algebra $ALG(LSIG)=(A_{S_0},A_{\Omega_0})$ mit Signatur *LSIG* heißt *logische Matrix*, wenn gilt:

Die Operation $A_{\Omega_0}(\rightarrow)$ erfüllt die folgenden Bedingungen:

für alle $a,b,c\in A_{S_0}(s_0)$ gilt:

(i) $aA_{\Omega_0}(\rightarrow)a=A_{\Omega_0}(\mathsf{v})$

(ii) $aA_{\Omega_0}(\rightarrow)b=A_{\Omega_0}(\mathsf{v})\wedge bA_{\Omega_0}(\rightarrow)c=A_{\Omega_0}(\mathsf{v})\Rightarrow aA_{\Omega_0}(\rightarrow)c=A_{\Omega_0}(\mathsf{v})$

(iii) $aA_{\Omega_0}(\rightarrow)b=A_{\Omega_0}(\mathsf{v})\wedge bA_{\Omega_0}(\rightarrow)a=A_{\Omega_0}(\mathsf{v})\Rightarrow a=b$

(iv) $aA_{\Omega_0}(\rightarrow)A_{\Omega_0}(\mathsf{v})=A_{\Omega_0}(\mathsf{v})$

Im folgenden werden wir mit dem Symbol *LSIG* stets eine Signatur, wie sie in Definition 4.1 definiert ist, bezeichnen, d.h. unter *LSIG* verstehen wir stets eine Signatur mit nur einer Sorte, endlich vielen nullstelligen Operatoren, wobei ein Operator (stets als v bezeichnet) ausgezeichnet ist, endlich vielen einstelligen Operatoren und endlich vielen zweistelligen Operatoren, wobei wiederum ein zweistelliger Operator (im folgenden stets mit $\rightarrow$ bezeichnet) ausgezeichnet ist. Die Menge der nullstelligen Operatoren werden wir im Zusammenhang mit einer Signatur *LSIG* stets mit L_0, die Menge der einstelligen Operatoren mit L_1 und die Menge der zweistelligen Operatoren stets mit L_2 bezeichnen.

Wenn wir eine Relation $\tilde{}$ auf die folgende Weise definieren:

$$a \tilde{} b \text{ dann und nur dann wenn } a\,A_{\Omega_0}(\rightarrow)\,\mathrm{b} = A_{\Omega_0}(\mathsf{v}),$$

so stellt die Bedingung (i) die Reflexivität, die Bedingung (ii) die Transitivität und die Bedingung (iii) die Antisymmetrie dieser Relation dar, und definiert damit eine Halbordnung auf $A_{S_0}(s_0)$. Immer wenn wir über Ordnungseigenschaften (z.B. Schranken) in logischen Matrizen sprechen, stellen wir uns diese Ordnung auf der Algebra vor.

Wie wir etwas später noch sehen werden, benutzen wir logische Matrizen, um Formeln einer formalen Sprache bei einer Realisation "logische Werte" zuzuordnen. Diese "logischen Werte" werden wir als Elemente der Trägermenge einer logischen Matrix ansehen, wobei der ausgezeichnete Wert v (von veritas) für "wahr" steht. Im Fall einer klassischen Logik wird die logische Matrix die folgende Gestalt haben.

Beispiel 4.1

Sei $SIGBOOL=(S_0=\{Bool\},\Omega_0=\{true,false,\neg,\vee,\wedge,\rightarrow\},typ_{\Omega_0})$ eine Signatur mit einer Sorte *Bool*, nullstelligen Operatoren *true, false*, dem einstelligen Operator $\neg$ und den zweistelligen Operatoren $\vee$, $\wedge$ und $\rightarrow$. Die Algebra $BOOL=ALG(SIGBOOL)=(A_{S_0},A_{\Omega_0})$ sei wie folgt definiert:

(1) Die Trägermenge sei $A_{S_0}(Bool)=\{0,1\}$.

(2) Die nullstellige Operation $A_{\Omega_0}(true)$ sei definiert durch $A_{\Omega_0}(true)\equiv 1$.

(3) Die nullstellige Operation $A_{\Omega_0}(false)$ sei definiert durch $A_{\Omega_0}(false)\equiv 0$.

(4) $A_{\Omega_0}(\neg)$ sei über die Wahrheitstabelle

$A_{\Omega_0}(\neg)$	0	1
	1	0

definiert. Hierbei sind die Argumentwerte der Operation oberhalb des horizontalen Striches und darunter die zugehörigen Werte nach Anwendung der Operation angegeben.

(5) Die zweistelligen Operationen $A_{\Omega_0}(\vee)$ (oder) und $A_{\Omega_0}(\wedge)$ (und) seien über die Wahrheitstabellen

$A_{\Omega_0}(\vee)$	0	1
0	0	1
1	1	1

$A_{\Omega_0}(\wedge)$	0	1
0	0	0
1	0	1

definiert. Dabei stellen wir uns das linke Argument der Operationen auf der linken Seite des vertikalen Striches und das rechte Argument oberhalb des horizontalen Striches vor. Der zu einem Paar von Argumentwerten gehörende Tabelleneintrag gibt dann den Wert der Operation auf dem Argumentwertepaar an.

(6) Analog definieren wir $A_{\Omega_0}(\rightarrow)$ über die Wahrheitstabelle

$A_{\Omega_0}(\rightarrow)$	0	1
0	1	1
1	0	1

Die so gegebene logische Matrix bezeichnen wir als *klassische boolesche Matrix.* Wir verwenden sie bei Auswertungen von Formeln einer klassischen Logik.

Definition 4.2 : (Alphabet 0. Ordnung)

Sei $LSIG=(S_0=\{s_0\},\Omega_0=L_1\cup L_2\cup L_2,typ_{\Omega_0})$ eine Signatur wie in Definition 4.1 und V und U Mengen, die paarweise disjunkt untereinander und zu den Mengen von *LSIG* sind.

(a) $ALPH^0 = (V, L_0, L_1, L_2, U)$, heißt dann *Alphabet 0. Ordnung* über der Signatur *LSIG.*

(b) Im Kontext von (a) nennen wir die Mengen V, L_0, L_1, L_2 und U Menge der *Aussagenvariablen*, Menge der *Aussagenkonstanten*, Menge der *einstelligen Konnektoren*, Menge der *zweistelligen Konnektoren* und Menge der *Hilfszeichen* des Alphabets $ALPH^0$.

Aussagenvariablen werden wir im folgenden mit den Symbolen p, q, r , s, ... bezeichnen. Weiter setzen wir voraus, daß die Menge U stets die Hilfszeichen "(" und ")" enthält. Die Elemente der Vereinigung $AL=V\cup L_0\cup L_1\cup L_2\cup U$ werden die Symbole des Alphabets $ALPH^0$ genannt.
Bei einigen Anwendungen des Begriffs des Alphabets 0. Ordnung ist die Reihenfolge wichtig, in der die Elemente in den Mengen V, L_0, L_1, L_2 und U erscheinen. Wir werden in solchen Situationen o.B.d.A. V, L_0, L_1, L_2, U nicht als Mengen sondern als Folgen auffassen.

Die obige Definition ist verschieden von der Definition des Alphabets im Kapitel über Sprachen, denn das Alphabet wird hier nicht als eine Menge, sondern als Folge von Mengen definiert. Der Unterschied ist jedoch nicht wesentlich, da die Folge per Definition endlich ist, und man dann die Vereinigung der Mengen der Folge als ein Alphabet im Sinne des Kapitels über Sprachen auffassen kann. Andererseits läßt sich jede Menge A so zerlegen, das sie in fünf Äquivalenzklassen zerfällt, die man dann als Folge im Sinne der Definition 4.2 anordnen kann, und die die Bedingungen (1)-(4) dieser Definition erfüllen.

Es sei $ALPH^0 = (V, L_0, L_1, L_2, U)$ ein Alphabet über der Signatur *LSIG.* Wenn wir die Menge V der Aussagenvariablen als Variablen der Sorte s_0 über der Signatur *LSIG* auffassen, so können wir die Menge der Terme $TERM(LSIG,V)$ definieren. Diese Terme werden *Aussagenformeln* über dem Alphabet $ALPH^0$ genannt und mit den griechischen Buchstaben π, ψ, ρ, ... bezeichnet. Da wir es allerdings nur mit null-, ein- und zweistelligen Operatoren zu tun haben, ist es üblich, die ein- bzw. zweistelligen Operatoren in Präfix- und Infixnotation zu

schreiben. Analog zu der Definition 3.8 definieren wir die Menge der Aussagenformeln daher auf die folgende Weise:

Definition 4.3 : (Aussagenformeln)

Die Menge der *Aussagenformeln* über einem Alphabet $ALPH^0=(V,L_0,L_1,L_2,V)$ ist die kleinste (im Sinne der Mengeninklusion) Menge $FORM(ALPH^0)$, die folgende Bedingungen erfüllt:

(1) $V \subseteq FORM(ALPH^0)$.

(2) $L_0 \subseteq FORM(ALPH^0)$.

(3) $L_1\ FORM(ALPH^0) \subseteq FORM(ALPH^0)$.

d.h. für jedes Element $o \in L_1$ der Menge L_1 und jedes Element $x \in FORM(ALPH^0)$ gehört das Wort $o(x)$ zu der Menge $FORM(ALPH^0)$.

(4) $(FORM(ALPH^0))L_2(FORM(ALPH^0)) \subseteq FORM(ALPH^0)$

d.h. für jedes Element $o \in L_2$ der Menge L_2 und alle Elemente $x,y \in FORM(ALPH^0)$ der Menge $FORM(ALPH^0)$ gehört das Wort $(x)o(y)$ zu der Menge $FORM(ALPH^0)$.

Die Bedingung (3) besagt, daß die Menge (Sprache) $FORM(ALPH^0)$ abgeschlossen unter der (Links)Konkatenation mit Elementen der Menge (Sprache) L_1 ist. Die Bedingung (4) sagt aus, daß die Menge (Sprache) $FORM(ALPH^0)$ abgeschlossen unter der Operation $(x,y,o) \rightarrow (x)o(y)$ mit $x,y \in FORM(ALPH^0)$ und $o \in L_2$ ist. Wir benutzen dabei in der Definition die Präfixschreibweise für einstellige Operatoren und die Infixschreibweise für zweistellige Operatoren. Man beachte weiter, daß die Klammern um die Argumente der Operatoren nötig sind, um in zusammengesetzten Aussagenformeln eine eindeutige Zugehörigkeit der Argumente der Operatoren zu den Operatoren zu gewährleisten. Das folgende Beispiel verdeutlicht die Konstruktion von Aussagenformeln.

Beispiel 4.2

Sei die Signatur *SIGBOOL* wie in Beispiel 4.1 gegeben und $ALPH^0 = (V, L_0, L_1, L_2, U)$ ein Alphabet 0. Ordnung über *SIGBOOL*. Weiter gelte:

a) Die Menge der Aussagenvariablen sei gegeben durch: $V=\{p, q, r, \dots\}$.

b) Die Menge der Hilfszeichen sei $U=\{'(', ')'\}$.

Die folgenden Zeichenfolgen stellen dann Aussagenformeln über dem Alphabet $ALPH^0$, d.h. Elemente der Menge $FORM(ALPH^0)$ dar:

$$p \quad q \quad true \quad false \quad \neg(p) \quad \neg(true) \quad \neg(\neg(p)) \quad (p)\rightarrow(false) \quad \neg(p)\wedge q$$

$$(\neg(true))\rightarrow(\neg(\neg(p))) \quad ((p)\rightarrow(false))\vee(\neg(p)\wedge q)$$

Das Alphabet $ALPH^0$ ist gerade das Alphabet der klassischen Logik. Die vorgestellten Aussagenformeln sind somit Aussagenformeln der klassischen Logik.

Definition 4.4 : (Sprache 0. Ordung)

Es sei $ALPH^0$ ein Alphabet 0. Ordnung. Das Paar $L=(ALPH^0, FORM(ALPH^0))$ nennt man dann eine *Sprache 0. Ordnung*.

Wie wir im Kapitel über Algebra hinter der Definition 3.8 beschrieben haben, definiert jede Sprache 0. Ordnung $(ALPH^0,FORM(ALPH^0))$ oder besser gesagt die Menge $FORM(ALPH^0)$ der Aussagenformeln dieser Sprache eine Algebra mit der Operatorenmenge $L_0 \cup L_1 \cup L_2$. Diese Algebra wird die Algebra der Aussagenformeln über dem Alphabet $ALPH^0$ genannt.

Formeln einer Sprache 0. Ordnung können wir als Terme einer Signatur *LSIG* mit Variablen aus einer Menge V auffassen. Wenn wir nun anstatt der Variablenmenge V als Variablenmenge die Menge der elementaren Formeln $FORM(SIG,V_S)$ einer Signatur *SIG* mit Variablen aus einer Familie V_S einsetzen, so nennt man die Menge der Terme $TERM(LSIG,FORM(SIG,V_S))$ die *Menge der offenen Formeln* einer Sprache 1. Ordnung. Wenn wir zusätzlich noch Quantoren einführen, kommen wir zu Formeln einer Sprache 1. Ordnung.

Definition 4.5 : (Sprache der 1. Ordnung)

Es sei *LSIG* wie unter 4.1 gegeben, $SIG=(S,\Omega,\Pi,typ_\Omega,typ_\Pi)$ eine Signatur und $V_S^f:S\to$**Set**, $V_S^b:S\to$**Set**, $Q:S\to$**Set** seien Abbildungen, für die gilt:

- V_S^f, V_S^b sind Injektionen.

(*) Für alle $s\in S$ gilt: $V_S^f(s)\cap Q(s)=V_S^b(s)\cap Q(s)=V_S^f(s)\cap V_S^b(s)=\emptyset$.

Wir nennen dann im Kontext dieser Definition die Elemente von $V_S^f(s)$ *freie Variablen* der Sorte s, die Elemente der Menge $V_S^b(s)$ *gebundene Variablen* der Sorte s und die Elemente der Menge $Q(s)$ *Quantoren* der Sorte s

(a) $FORM(LSIG,SIG,V_S,Q)$ sei die kleinste Menge, die die folgenden Bedingungen erfüllt:

(1) $FORM(SIG,V_S)\subseteq FORM(LSIG,SIG,V_S,Q)$

(2) Für jedes Element $o\in L_1$ der Menge L_1 und jedes Element $\alpha\in FORM(LSIG,SIG,V_S,Q)$ gehört das Wort $o(\alpha)$ der Menge $FORM(LSIG,SIG,V_S,Q)$ an.

(3) Für alle $\alpha,\beta\in FORM(LSIG,SIG,V_S,Q)$ und jedes Element $o\in L_2$ gehört das Wort $(\alpha)o(\beta)$ der Menge $FORM(LSIG,SIG,V_S,Q)$ an.

(4) Für jedes $s\in S$, jedes Element $v\in V_S^b(s)$, jedes $x\in V_S^f(s)$, jedes Element $q\in Q(s)$ und jede Formel $\alpha\in FORM(LSIG,SIG,V_S,Q)$ gehört das Wort $q_v\alpha(x/v)$, wobei $\alpha(x/v)$ das Wort ist, das aus dem Wort α dadurch entsteht, daß man jedes Vorkommen des Symbols x durch das Symbol v ersetzt, der Menge $FORM(LSIG,SIG,V_S,Q)$ an.

(b) Die Folge der Gestalt $ALPH^1=ALPH=(V_S^f,V_S^b,\Omega,\Pi,L_0,L_1,L_2,Q,U)$ nennen wir *Alphabet 1. Ordnung* über den Signaturen LSIG und SIG mit Variablen aus der Familie V_S und Quantoren aus der Familie Q.

(c) Das Tripel $L^1=L=(ALPH,TERM(SIG,V_S),FORM(LSIG,SIG,V_S,Q))$ wird eine *Sprache 1. Ordnung* über dem *Alphabet 1. Ordnung ALPH* genannt.

(d) Elemente der Menge $FORM(LSIG,SIG,V_S,Q))$ nennen wir Formeln der Sprache L.

Die Bedingung (2) der Definition 4.5 (a) bedeutet analog der Bedingung (3) der Definition 4.3, daß die Menge $FORM(LSIG,SIG,V_S,Q)$ abgeschlossen bzgl. der (Links)Konkatenation mit Elementen der Menge L_1 ist. Die Bedingung (3) besagt, daß analog zur Bedingung (4) der Definition 4.3 die Menge $FORM(LSIG,SIG,V_S,Q)$ abgeschlossen bzgl. der "Infixkonkatenation" mit Elementen der Menge L_2 ist. In Bedingung (4) geben wir eine Konstruktionsregel an, die die Verwendung von Quantoren in Formeln bestimmt. Diese sieht vor, daß zunächst eine freie Variable (die nicht unbedingt in der Formel vorkommen braucht) innerhalb einer Formel (Wort) α überall durch eine gebundene Variable ersetzt werden muß, und dann das resultierende Wort mit einem Präfix versehen werden kann, das aus einem Quantor und der gebundenen Variablen als Index dieses Quantors besteht. Man beachte dabei, daß nach dieser Regel eine gebundene Variable ohne vorangestellten Quantor nur dann vorkommen darf, wenn sie vorher im Wort als Index eines Quantors auftritt. Unsere Konstruktion beschreibt man oft auch etwas ungenau mit den Worten: Eine Variable wird durch einen Quantor gebunden.

Sei $L=(ALPH,TERM(SIG,V_S),FORM(LSIG,SIG,V_S,Q))$ eine Sprache 1. Ordnung über dem Alphabet 1. Ordnung $ALPH=(V_S^f,V_S^b,\Omega,\Pi,V,L_0,L_1,L_2,Q,U)$. Jede Formel $\alpha\in FORM(LSIG,SIG,V_S,Q)$ ist dann ein Wort über dem Alphabet $\mathbf{Al}=\bigcup_{s\in S}(V_S^f(s)\cup V_S^b(s))\cup L_o\cup L_1\cup L_2\cup U$.

Wir werden im folgenden nur formale Sprachen 1. Ordnung mit *Gleichheitszeichen* betrachten, d.h. obwohl wir in der Menge der Prädikate nicht explizit Gleichheitszeichen aufführen werden, gehen wir immer davon aus, daß wir für jede Sorte $s\in S$ ein binäres Prädikat *Gleichheitszeichen* vom Typ (ss) vorliegen haben, das wir unabhängig von der Sorte stets mit dem Zeichen '=' bezeichnen werden. Dabei werden wir bei Anwendung des Gleichheitszeichen in Beispielen dieses in der Infixnotation schreiben.

Wir wollen nun ein Beispiel für eine formale Sprache 1. Ordnung angeben, die wir bereits in der Einleitung informell verwendet haben.

Beispiel 4.3

Sei die Signatur

```
SIG DMS =
        sorts= Process, Daten, Warteschlange, ℕ;
        ops=   ein:Warteschlange Daten → Warteschlange,
               aus:Warteschlange → Daten,
               rest:Warteschlange → Warteschlange
               laenge:Warteschlange → ℕ,
               daten:Process → Daten,
               process:Daten → Process,
               empf:∅ → Process,
               len:∅ → ℕ;
        preds= isprocess:Process,
               isdaten:Daten,
               isWarteschlange:Warteschlange,
               lt:ℕ ℕ;
SIGEND
```

und die Signatur *LSIG* wie in Beispiel 4.1 gegeben.

a) Sei $V^f_{Process}=\{pr,pr_1,pr_2,pr_3,\dots\}$ die Menge der freien Variablen der Sorte *Process*.

b) Sei $A^f_{Daten}=\{n,n_1,n_2,n_3,\dots\}$ die Menge der freien Variablen der Sorte *Daten*.

c) Sei $A^f_{Warteschlange}=\{w,w_1,w_2,w_3,\dots\}$ die Menge der freien Variablen der Sorte *Warteschlange*.

d) Sei $A^b_{Daten}=\{\nu_1,\nu_2,\nu_3,\dots\}$ die Menge der gebundenen Variablen der Sorte *Daten*.

e) Sei $A^b_{Warteschlange}=\{\omega,\omega_1,\omega_2,\omega_3,\dots\}$ die Menge der gebundenen Variablen der Sorte *Warteschlange*.

f) Die Menge der Quantoren enthalte nur die Quantoren $Q=\{\forall,\exists\}$.

g) Die Menge der Hilfszeichen sei $U=\{'(',')',',',\}$.

LDMS sei die Sprache 1. Ordnung über den Signaturen *DMS* und *LSIG* mit Variablen aus der Familie V und Quantoren aus Q. Die folgenden Zeichenfolgen stellen dann Formeln der Sprache *LDMS* dar:

(1) $\exists_{\omega\in V^b_{Warteschlange}}\ w_1=ein(\omega,n_1)\wedge pr_1=process(n_1)$

(2) $n_2=aus(w_1)\wedge pr_2=empf()$

(3) $n_1=daten(pr_1)$

(4) $pr_2=empf()$

(5) $n_1=aus(w_1)\wedge n_2=aus(rest(w_1))\wedge pr_2=empf()\wedge w_2=rest(rest(w_1))$

Diese Formeln werden gerade in der Einleitung bei den Beispielen verwendet.

Bezeichnung

(a) Mit dem Symbol var(α) bezeichnen wir das Bild der Formel α unter dem Homomorphismus $\phi^{\#_{Al,B}}: \mathbf{Al}^* \to B^*$ mit $B=\bigcup_{s\in S} V^f_S(s)$. Dieser löscht gerade aus dem Wort $\alpha\in\mathbf{Al}^*$ alle Symbole aus, die keine freien Variablen sind.

(b) Für jedes Wort u von Variablen aus den Mengen $V^f_S(s)$ $(s\in S)$ bezeichnen wir mit dem Symbol $typ(u)$ das Bild des Wortes u unter dem Homomorphismus $f^{\#}$, der durch die folgende Abbildung erzeugt wird:

$$f: x\to s \quad \text{mit} \quad x\in V^f_S(s)$$

und für jede Formel $\alpha\in FORM(LSIG,SIG,V_S,Q)$ definieren wir **typ**(α) = **typ**(**var**(α)).

(c) Für jeden Ausdruck α (d.h. ein Term oder eine Formel) definieren wir:

$$fvar_s(\alpha) = \{x \in V_S^f(s) \mid x \text{ kommt im Wort } \alpha \text{ vor}\}$$
$$bvar_s(\alpha) = \{x \in V_S^b(s) \mid x \text{ kommt im Wort } \alpha \text{ vor}\}$$
$$fvar(\alpha) = \bigcup_{s \in S} fvar_s$$
$$bvar(\alpha) = \bigcup_{s \in S} bvar_s$$

Beispiel 4.4

Es sei eine Signatur $SIG=(S=\{s_1,s_2\},\Omega=\{+\},\Pi=\{\leq\},typ_\Omega,typ_\Pi)$ mit $typ_\Omega(+)=(s_1s_2,s_1)$ und $typ_\Pi(\leq)=(s_1s_1)$ gegeben. Betrachten wir nun die Formel $\alpha \in FORM(LSIG,SIG,V_S,Q)$ mit

$$\alpha = \exists_{\xi \in X}\ \forall_{\eta \in X} : \xi + x \leq y + (\eta + x), \qquad \text{mit } y \in V_S(s_1),\ x \in V_S(s_2)$$

Nach der Definition der Abbildung $\phi_{Al,B}$ haben wir $\mathbf{var}(\alpha) = xyx$. Der Typ der Formel α ist gegeben durch: $\mathbf{typ}(\alpha) = \mathbf{typ}(\mathbf{var}(\alpha)) = \mathbf{typ}(xyx) = s_2s_1s_2$.

4.2 Realisation

Im Folgenden setzen wir voraus, daß für jedes $s \in S$ $Q(s) = \{\forall, \exists\}$ ist.

Wenn wir eine Sprache 1. Ordnung $L=(ALPH,TERM(SIG,V_S),FORM(LSIG,SIG,V_S,Q))$ über den Signaturen SIG und $LSIG$ mit Variablen aus der Familie V_S und Quantoren aus Q haben, so definiert jede Algebra $ALG(SIG)=(A_S,A_\Omega)$ über der Signatur SIG in kanonischer Weise eine Realisation der Terme aus $TERM(SIG,V_S)$ (siehe 3.12).

Im Fall einer "klassischen zweiwertigen" Logik definiert in analoger Weise jedes algebraische System $ALS(SIG)=(A_S,A_\Omega,A_\Pi)$ über der Signatur SIG nicht nur eine Realisation der Terme aus $TERM(SIG,V_S)$, sondern auch eine Realisation der elementaren Formeln aus $FORM(SIG,V_S)$, indem man jedes Prädikat $\pi \in \Pi$ als die zugehörige Relation $A_\Pi(\pi)$ interpretiert. Diese Form der Interpretation von elementaren Formeln ist bei "mehrwertigen" Logiken nicht möglich. Man kann nun aber im Fall einer "klassischen" Logik jede Relation $\rho=A_\Pi(\pi)$ in äquivalenter Weise durch ihre charakteristische Funktion $\rho_\pi: A_S^{typ_\Pi(\pi)} \to \{0,1\}$, $\rho_\pi(\mathbf{a})=1$ falls $\mathbf{a} \in \rho$ sonst 0, beschreiben. Diese Interpretation von Prädikaten durch charakteristische Funktionen (anstatt von Relationen) läßt sich nun in natürlicher Weise auf mehrwertige Logiken übertragen, wenn wir die Menge $\{0,1\}$ durch die Menge der "logischen Werte" einer mehrwertigen Logik ersetzen. Wir definieren daher:

Definition 4.6 : (Realisierung der elementaren Formeln einer Sprache 1. Ordnung)

Seien SIG, $LSIG$ und V_S wie in vorhergehenden Definitionen gegeben. $ALG(SIG)=(A_S,A_\Omega)$ sei eine Algebra zu der Signatur SIG und $ALG(LSIG)=(B_{S_0},B_{\Omega_0})$ sei eine logische Matrix zu der Signatur $LSIG$. Weiter sei $\rho_\Pi=(\rho_\pi: A_S^{typ_\Pi(\pi)} \to B_{S_0})_{\pi \in \Pi}$ eine Familie von Funktionen.

(a) Für jede Auswertung von freien Variablen $\nu=(\nu_s: V_S^f(s) \to A_S(s))_{s \in S}$ ist dann eine *Auswertung von elementaren Formeln* $\bar{\nu}_F: FORM(SIG,V_S) \to B_{S_0}$ durch folgende Regel eindeutig definiert:

> Ist $\alpha=\pi(t_1,t_2,\ldots,t_k)$ mit $\pi \in \Pi$, $t_1,t_2,\ldots,t_k \in TERM(SIG,V_S)$ und $typ_\Pi(\pi)=typ(t_1)\cdots typ(t_k)$, so ist $\bar{\nu}_F(\alpha)=\rho_\pi(\bar{\nu}_T(t_1),\ldots,\bar{\nu}_T(t_k))$, wobei $\bar{\nu}_T$ die in Definition 3.11 definierte Auswertung der Terme aus $TERM(SIG,V_S)$ ist, die durch ν eindeutig bestimmt ist.

(b) Die durch die Algebren $ALG(SIG)$ und $ALG(LSIG)$ und die Abbildungsfamilie ρ_Π eindeutig bestimmte *Realisation der elementaren Formeln* ist die Abbildung $\mathbf{R}_{EF}: FORM(SIG,V_S) \to [Val(V_S^f,A_S) \to B_{S_0}]$, die für jede Formel $\alpha \in FORM(SIG,V_S)$ durch $\mathbf{R}_{EF}(\alpha)=\alpha_\mathbf{R}$ mit $\alpha_\mathbf{R}(\nu)=\bar{\nu}_F(\alpha)$ für jede Auswertung $\nu \in Val(V_S^f,A_S)$ definiert ist.

Beispiel 4.5

Sei $SIG=(S=\{r,w\},\Omega=\{+,\bullet,\vec{+}\},\Pi=\{\wedge\},typ_\Omega,typ_\Pi)$ eine Signatur, für die gilt:

(1) $+$ ist ein Operator des Typs (rr,r),

(2) $\bullet$ ist ein Operator des Typs (rw,w),

(3) $\vec{+}$ ist ein Operator des Typs (ww,w) und

(4) $\wedge$ ist ein Prädikat des Typs (ww).

Da alle Operatoren und das Prädikat zweistellig sind, schreiben wir sie im folgenden in Infixschreibweise.

Das algebraische System $ALS(SIG)=(A_S,A_\Omega,A_\Pi)$ sei weiter wie folgt definiert:

(a) $A_S(r)$ sei die Menge der reellen Zahlen $\mathbb{R}$.

(b) $A_S(w)$ sei die Menge der zweidimensionalen Vektoren $\mathbb{R}^2$ über dem Körper $\mathbb{R}$.

(c) $A_\Omega(+)$ sei die übliche Addition reeller Zahlen in $\mathbb{R}$.

(d) $A_\Omega(\bullet)$ sei die übliche skalare Multiplikation von Vektoren mit reellen Zahlen.

(e) $A_\Omega(\vec{+})$ sei die übliche komponentenweise definierte Addition von Vektoren im $\mathbb{R}^2$.

(f) $A_\Pi(\wedge)$ sei die übliche komponentenweise definierte Ordnungsrelation im $\mathbb{R}^2$.

Weiter seien *SIGBOOL* und die klassische boolesche Matrix *BOOL* wie in Beispiel 4.1 gegeben.

Die charakteristische Funktion $\rho_\wedge:\mathbb{R}^2\times\mathbb{R}^2\to\{0,1\}$ der Ordnungsrelation $A_\Pi(\wedge)$ sei wie folgt definiert: Für alle $(x,y)\in\mathbb{R}^2\times\mathbb{R}^2$ ist $\mathbb{R}_\wedge(x,y)=1$ falls $xA_\Pi(\wedge)y$ gilt, sonst *0*. $\rho_\wedge$ definiert uns dann in Verbindung mit der kanonischen Realisation $\mathbf{R}_T$ der Terme in der Algebra $ALG(SIG)=(A_S,A_\Omega)$ eine eindeutig bestimmte Realisation der elementaren Formeln. Wenn wir z.B. $V_S(r)=\{x,y,...\}$ und $V_S(w)=\{\vec{x},\vec{y},...\}$ wählen, so können wir die elementare Formel $\alpha=(x\bullet\vec{y}\vec{+}\vec{z})\wedge\vec{x}$ in einer Auswertung ν mit partieller Zuordnung $x\to 3$, $\vec{y}\to(1,1)$, $\vec{z}\to(2,1)$ und $\vec{x}\to(3,3)$ wie folgt auswerten:

$$\alpha_\mathbf{R}(\nu)=\rho_\wedge(3A_\Omega(\bullet)(1,1)A_\Omega(\vec{+})(2,1),(3,3))=\rho_\wedge((3,3)A_\Omega(\vec{+})(2,1),(3,3))=\rho_\wedge((5,4),(3,3))=0$$

da *5* nicht kleiner oder gleich *3* ist.

Man beachte, daß $\rho_\wedge$ gemäß unserer Konstruktion als charakteristische Funktion von $A_\Pi(\wedge)$ schon eindeutig durch das algebraische Systen $ALS(SIG)$ bestimmt ist. Dies liegt daran, daß wir die Signatur *SIGBOOL* in unserem Beispiel durch die klassische boolesche Matrix *BOOL* interpretiert haben, die nur die zwei logischen Werte *0* und *1* besitzt. Wir haben es also mit einer zweiwertigen Logik zu tun. In so einem Fall können wir uns stets sowohl die Terme als auch die elementaren Formeln eindeutig durch Angabe eines algebraischen Systems über der zugehörigen Signatur realisiert denken.

Wir wollen noch als ein weiteres Beispiel das Beispiel 4.3 fortführen.

Beispiel 4.6 : (Fortführung von Beispiel 4.3)

Die Sprache *LDMS* sei wie im Beispiel 4.3 gegeben und *BOOL* sei die klassische boolesche Matrix über der Signatur *SIGBOOL*, wie sie in Beispiel 4.1 gegeben ist. Das algebraische System *ALS(DMS)* aus dem Beispiel 3.1 definiert dann in eindeutiger Weise eine Realisation der Terme und elementaren Formeln der Sprache *LDMS*.

Die elementare Formel $\alpha=$*laenge*(w) *lt len* können wir bei der Auswertung ν mit partieller Zuordnung $w\to d_1d_2d_3$ durch

$$\alpha_\mathbf{R}(\nu)=\rho_{lt}(l(d_1d_2d_3),2)=\rho_{lt}(3,2)=0$$

auswerten, da *3* nicht kleiner gleich *2* in den natürlichen Zahlen ist.

Aufbauend auf der Realisation der Terme und elementaren Formeln einer Sprache 1. Ordnung können wir nun in natürlicher Weise eine Realisation der Formeln der Sprache definieren. Für die Realisation der offenen Formeln müssen wir uns nur vor Augen halten, daß eine logische Matrix $ALG(LSIG)$ über der Signatur *LSIG* eine Interpretation der Operatoren aus L_0, L_1 und L_2 über die zugehörigen Operationen in $ALG(LSIG)$ nahelegt.

Wir müssen dann nur noch die Quantoren $\forall$ und $\exists$ interpretieren. Hierzu werden wir $\forall$ als *Allquantor* und $\exists$ als *Existenzquantor* in jeder Trägermenge interpretieren. Wir definieren also:

Definition 4.7 : (Realisation der Formeln einer Sprache 1. Ordnung)

Seien *SIG*, *LSIG*, V_S, Q und L wie in der Definition 4.5 gegeben. $ALG(SIG)=(A_S,A_\Omega)$ sei eine Algebra über der Signatur *SIG* und $ALG(LSIG)=(B_{S_0},B_{\Omega_0})$ eine logische Matrix über *LSIG*.

Weiter sei $\rho_\Pi=(\rho_\pi:A_S^{typ_\Pi(\pi)}\to B_{S_0}(s_0))_{\pi\in\Pi}$ eine Familie von Funktionen und $\mathbf{R}_T:TERM(SIG,V_S)\to[Val(V_S^f,A_S)\to\bigcup_{s\in S}A_S(s)]$ und $\mathbf{R}_{EF}:FORM(SIG,V_S)\to[Val(V_S^f,A_S)\to B_{S_0}(s_0)]$ die durch *ALG(SIG)*, *ALG(LSIG)* und ρ_Π eindeutig bestimmten Realisationen der Terme und elementaren Formeln der Sprache *L*.

Die durch *ALG(SIG)*, *ALG(LSIG)* und ρ_Π eindeutig bestimmte *Realisation der Formeln* der Sprache *L* $\mathbf{R}_F:FORM(LSIG,SIG,V_S,Q)\to[Val(V_S^f,A_S)\to B_{S_0}(s_0)]$ ist wie folgt gegeben:

Sei $\alpha\in FORM(LSIG,SIG,V_S,Q)$ eine beliebige Formel und $\nu=(\nu_s:V_S^f(s)\to A_S(s))_{s\in S}$ im folgenden eine beliebige Auswertung der freien Variablen. Dann ist $\mathbf{R}_F(\alpha)=\alpha_\mathbf{R}:Val(V_S^f,A_S)\to B_{S_0}(s_0)$ wie folgt induktiv definiert:

(i) Wenn $\alpha=\pi\in FORM(SIG,V_S))$ eine elementare Formel ist, dann ist $\alpha_\mathbf{R}=\mathbf{R}_{EF}(\alpha)$.

(ii) Wenn $\alpha=e_i\in L_0$ eine Aussagenkonstante ist, dann ist $\alpha_\mathbf{R}=B_{\Omega_0}(e_i)$.

(iii) Wenn α eine Formel der Gestalt $o^i(\beta)$ $(i\le l)$ mit $o^i\in L_1$ ist und der Wert $\beta_\mathbf{R}(\nu)$ definiert ist, dann ist $\alpha_\mathbf{R}(\nu)=B_{\Omega_0}(o^i)(\beta_\mathbf{R}(\nu))$.

(iv) Wenn α eine Formel der Gestalt $\beta o_i\gamma$ $(i\le m)$ mit $o_i\in L_2$ ist, und die Werte $\beta_\mathbf{R}(\nu)$ und $\gamma_\mathbf{R}(\nu)$ definiert sind, dann ist $\alpha_\mathbf{R}(\nu)=B_{\Omega_0}(o_i)(\beta_\mathbf{R}(\nu),\gamma_\mathbf{R}(\nu))$.

(v) Wenn α eine Formel der Gestalt $\forall_\xi\beta$ ist, so gilt:

(1) wenn die Variable $\xi\in A_S^b$ in der Formel β nicht vorkommt, dann ist $(\forall_\xi\beta)_\mathbf{R}(\nu)=\beta_\mathbf{R}(\nu)$;

(2) wenn die Variable ξ in der Formel β vorkommt, dann ist

$$(\forall_\xi\beta)_\mathbf{R}(\nu)=inf_{j\in V_S^f(s)}\{\beta_\mathbf{R}(x/j)\}(\nu),$$

wobei $\beta(\xi/x)$ eine Formel ist, die aus der Formel β durch die Substitution

$$\xi\to x \text{ mit } x\in A_S^f(s)\ \wedge\ x \text{ kommt in } \beta \text{ nicht vor}$$

entsteht und *inf* die Operation der unteren Schranke in der Algebra *ALG(LSIG)* ist.

(vi) Wenn α eine Formel der Gestalt $\exists_\xi\beta$ ist, so gilt:

(1) wenn die Variable $\xi\in A_S^b$ in der Formel β nicht vorkommt, dann ist $(\exists_\xi\beta)_\mathbf{R}(\nu)=\beta_\mathbf{R}(\nu)$;

(2) wenn die Variable ξ in der Formel β vorkommt, dann ist

$$(\exists_\xi\beta)_\mathbf{R}(\nu)=sup_{j\in V_S^f(s)}\{\beta_\mathbf{R}(x/j)\}(\nu)$$

wobei $\beta(\xi/x)$ eine Formel ist, die aus der Formel β durch die Substitution

$$\xi\to x \text{ mit } x\in A_S^f(s)\ \wedge\ x \text{ kommt in } \beta \text{ nicht vor}$$

entsteht und *sup* die Operation der oberen Schranke in der Algebra *ALG(LSIG)* ist.

Definition 4.8 : (Realisation einer Sprache 1. Ordnung)

Seien *L*, $\mathbf{R}_T$ und $\mathbf{R}_F$ wie in Definition 4.7 gegeben. Das Paar $\mathbf{R}=(\mathbf{R}_T,\mathbf{R}_F)$ nennen wir dann eine *Realisation der Sprache L.*

Da aus dem Kontext stets ersichlich ist, welche der Abbildungen $\mathbf{R}_T$ oder $\mathbf{R}_F$ anzuwenden ist, werden wir im folgenden kurz $\mathbf{R}$ für $\mathbf{R}_T$ oder $\mathbf{R}_F$ schreiben.

Folgerung 4.1

Es seien die Voraussetzungen wie in Definition 4.7 gegeben. Für jede Auswertung $v=(v_s:V_S^f(s)\rightarrow A_s(s))_{s\in S}$ und jede Substitution $u=(u_s:V_S^f(s)\rightarrow TERM_s(SIG,V_S))_{s\in S}$ gilt:

$$\alpha_{\mathbf{R}}(\mathbf{u} \circ \mathbf{v}) = (\mathbf{u}^{\#}(\alpha))_{\mathbf{R}}(\mathbf{v}).$$

Den Beweis kann man bei Rasiowa ([Ras68a]) nachlesen.

Beispiel 4.7 : (Fortsetzung von Beispiel 4.6)

Wir wollen einmal die Formel

$$\alpha=\exists_{\omega\in V^b_{Warteschlange}} w_1=ein(\omega,n_1)$$

bzgl. der Auswertung v von Variablen, die partiell durch $w_1\rightarrow d_1d_2d_3$ und $n_1\rightarrow d_3$ gegeben ist, auswerten. Wenn wir die gebundene Variable ω durch d_1d_2 ersetzen, erhalten wir:

$$\wedge_{\mathbf{R}}(=_{\mathbf{R}}(d_1d_2d_3,A_\Omega(ein)(d_1d_2,d_3)),=_{\mathbf{R}}(a_3,A_\Omega(process)(d_3)))=\wedge_{\mathbf{R}}(1,1)=1$$

Die Formel α wird damit zu *1* (true) ausgewertet, d.h. es gilt: $\alpha_{\mathbf{R}}(v)=1$.

4.3 Erfüllbarkeit

Definition 4.9

Es sei $\mathbf{R}=(\mathbf{R}_T,\mathbf{R}_F)$ eine Realisation der Sprache $L=(ALPH,TERM(SIG,V_S),FORM(LSIG,SIG,V_S,Q))$ und $A=(A,\mathsf{v},e_1,e_2,..,e_k,o^1,o^2,..,o^l,o_1,o_2,..,o_m,\rightarrow)$ die zugehörige logische Matrix. Weiter sei $v=(v_s:V_S^f(s)\rightarrow A_S(s))_{s\in S}$ eine Auswertung der Variablen aus V_S und α eine Formel.

Die Auswertung v *erfüllt* die Formel α dann und nur dann, wenn $\alpha_{\mathbf{R}}(v) = \mathsf{v}$ gilt.

Definition 4.10

Eine Formel $\alpha\in FORM(LSIG,SIG,V_S,Q)$ heißt *erfüllbar* in der Realisation $\mathbf{R}$ dann und nur dann, wenn es eine Auswertung $v = (v_s : V_S^f(s) \rightarrow A_S(s))_{s\in S}$ gibt, die die Formel α erfüllt.

Definition 4.11

Eine Realisation $\mathbf{R}$ der Sprache L heißt ein *L-Modell* der Formel α dann und nur dann, wenn jede Auswertung $v = (v_s : V_S^f(s) \rightarrow A_S(s))_{s\in S}$ die Formel α erfüllt.

Definition 4.12

Eine Realisation $\mathbf{R}$ der Sprache L heißt ein *L-Modell von einer Menge* F_L der Formeln der Sprache, wenn sie ein L-Modell von jeder Formel dieser Menge ist.

Definition 4.13

Eine Formel $\alpha\in FORM(LSIG,SIG,V_S,Q)$ der Sprache L heißt *wahr* dann und nur dann, wenn jede Realisation der Sprache ein L-Modell der Formel ist.

4.4 Hinweise zur Literatur

In dem Kapitel Logik wird ziemlich stark algebraisiert. Wir folgen hier der algebraischen Denkweise von Rasiowa ([Ras74a]). Alle Definitionen in dem Abschnitt kommen in dieser Arbeit vor, in der man auch nähere Informationen über die betrachteten Systeme finden kann.

Wir haben hier eine ziemlich allgemeine Auffassung der Logik gewählt, weil wir uns nicht auf eine Beschreibung von Elementen eines Petri-Netzes (d.h. eine Beschreibung von Stellen, Transitionen und Kanten des Netzes) beschränken wollen. Ein nebenläufiges System kann man auf viele verschiedene Weisen, auch mit verschiedenen Logiken beschreiben, und wir wollen immer frei sein eine solche Beschreibung zu wechseln.

Die logischen Matrizen werden von Lukasiewicz und Tarski eingeführt und in vielen Arbeiten (vor allem in den Arbeiten über mehrwertige Logiken) beschrieben. Sie spielen dieselbe Rolle wie die klassische zweielementige boolesche Algebra für die klassische Aussagenrechnung, d.h. die Elemente der Trägermenge einer solchen Algebra sind eine Art von Wahrheitswert. Man kann sagen, daß die Aussagen in solchen Algebren im gewissen Sinne interpretiert werden (so wie die Aussagen der klassischen Aussagenrechnung wahr oder falsch sind).

Sprachen (von beliebiger Ordnung) kann man immer als Trägermengen der entsprechenden absolut freien Algebren betrachten. Dieser Gesichtspunkt ist immer da sehr günstig, wo man über verschiedene Homomorphismen (z.B. Realisation, Substitution) spricht. Wir verzichten hier auf eine genaue Beschreibung von diesen Algebren, weil sie logisch sehr einfach, aber technisch sehr kompliziert sind. Eine genaue Beschreibung und die Beweise von vielen Eigenschaften dieser Algebren kann man z.B. in [Ras74a] finden.

Wenn man über Sprachen der Ordnung größer als 0 spricht, muß man sich entscheiden, ob man für freie und gebundene Variablen dieselben oder verschiedene Symbole benutzt. Wir haben den zweiten Fall gewählt, weil wir nicht über Schlußregeln sprechen wollen und für die anderen Betrachtungen diese Schreibweise günstiger erscheint.

Alle "semantischen" Begriffe wie Realisation, Erfüllbarkeit, Modell etc. werden klassisch definiert. Diese Definition kann man in jedem Buch aus dem Gebiet der Logik finden. Einige dieser Bücher sind:

- Ein heute schon klassisches Buch über formale Logik und Metamathematik, das besonders gut für Leute, die sich mit mathematischen Grundlagen der Informatik beschäftigen wollen, geeignet ist, ist [Ras68a]. Alle in diesem Abschnitt vorkommenden Begriffe werden in dem Buch genau beschrieben.
- Vor allem den nichtklassischen Logiken gewidmet ist [Ras74a]. Der erste Teil enthält aber eine umfangreiche Einführung in die Theorie der implikativen booleschen und Post- Algebren.

5 Multimengen

Definition 5.1 : (Multimenge)

Sei A eine Menge.

(a) Eine Funktion $\mu : A \to \mathbb{N}$ heißt eine *Multimenge über A.*

(b) Die Menge aller Multimengen über A bezeichnen wir mit *MULT(A).*

Auf den Multimengen wollen wir die folgenden Operationen definieren:

Definition 5.2

Es seien μ, μ' Multimengen über der Menge A und n sei eine natürliche Zahl.

(1) $\mu \leq \mu' \Leftrightarrow \forall_{s \in A} \mu(s) \leq \mu'(s)$

(2) $(\mu + \mu')(s) = \mu(s) + \mu'(s)$

(3) $(\mu \cdot \mu')(s) = \mu(s) \cdot \mu'(s)$

(4) $(n\mu)(s) = n\mu(s)$

(5) $(\mathbf{max}(\mu,\mu'))(s) = max(\mu(s),\mu'(s))$

(6) $(\mathbf{min}(\mu,\mu'))(s) = min(\mu(s),\mu'(s))$

(7) $(\mu - \mu')(s) = \mu(s) \dot{-} \mu'(s)$ $\mathbb{N}$ wobei $\dot{-} : \mathbb{N} \times \mathbb{N} \to$ die Operation der beschränkten Subtraktion ist, d.h. für $k,m \in \mathbb{N}$ ist $k \dot{-} m = \begin{cases} k - m & wenn\ k \geq m \\ 0 & wenn\ k < m \end{cases}$

(8) Für $\mu_1 + \mu_2 + ... + \mu_k$ schreiben wir abkürzend $\sum_{i \in [1..k]} \mu_i$

Die Algebra $\mathrm{Mult} = (\mathbb{N}, MULT(A), \cdot, +, 0)$ mit $0 : A \to \mathbb{N}, 0(s)=0$ kann man in den Modul $(\mathbb{Z}, [A \to \mathbb{Z}], \cdot, +, \bar{0})$ mit der üblichen (komponentenweise) definierten Multiplikation (der Funktionen durch ganze Zahlen) und Addition (der ganzzahligwertigen Funktionen) einbetten. Diese Einbettung respektiert die Subtraktion der Funktionen nicht. Diese Einbettung wird oft auf diese Weise ausgenutzt, daß man eine Multimenge $\mu : A \to \mathbb{N}$ als eine "formale lineare Kombination" der Gestalt $\Sigma\mu(s)<s>$ darstellt. Solche Zeichenfolgen können wir als ein Spezialfall von formalen Summen auffassen.

Zu einer Multimenge $\mu = \Sigma a_i(t_i) <t_i>$ definieren wir $|\mu| = \{t_i \mid a_i(t_i) \neq 0\}$

Um über Multimengen "sprechen zu können", braucht man eine Sprache, in der man entsprechende Bezeichnungen hat. In dieser Sprache sollte man die Möglichkeit haben, über verschiedene Operationen und Relationen zwischen Multimengen zu sprechen, und man sollte auch Bezeichnungen (Symbole) für diese haben. Man sollte deshalb sagen, mit welchen Symbolen (Zeichen) diese Multimengen und die entsprechenden Operationen bezeichnet werden, und wie man diese Bezeichnungen zusammensetzen darf. Formale Summen stellen eine solche Sprache dar. Wir wollen sie in der folgenden Definition näher erläutern.

Definition 5.3 : Formale Summen

Sei $SIG = (S,\Omega,\Pi,typ_\Omega,typ_\Pi)$ eine Signatur, $V=V_S=(V_s)_{s \in S}$ eine Familie von Variablen der Sorten $s \in S$ und *TERM(SIG,V)* die Menge der Terme der Signatur *SIG* mit Variablen aus der Familie *V*.

(a) Die Menge der formalen Summen über *TERM(SIG,V)* ist die kleinste Menge *FSM(SIG,V)*, für die gilt:

(1) für alle $t \in TERM(SIG,V)$ ist die Zeichenfolge $<t>$ aus $FSM(SIG,V)$,

(2) für beliebige $\omega_1,\omega_2 \in FSM(SIG,V)$ ist die Zeichenfolge $\omega_1+\omega_2$ aus $FSM(SIG,V)$.

(b) Wir wollen noch einige abkürzende Schreibweisen einführen:

(1) $<t> + .. + <t>$ (wobei $<t>$ n mal wiederholt wird, mit $n \in \mathbb{N}$) schreiben wir abkürzend als $n<t>$.

(2) Für $\omega_1 + \omega_2 + ... + \omega_k$ schreiben wir abkürzend $\sum_{i \in [1..k]} \omega_i$.

(c) Den Typ einer formalen Summe wollen wir rekursiv definieren durch:

(1) für $<t> \in FSM(SIG,V)$ ist $typ(<t>) = \{typ(t)\}$,

(2) für $\sum_{i\in I} \omega_i \in FSM(SIG,V)$ ist $typ(\sum_{i\in I} \omega_i) = \bigcup_{i\in I} typ(\omega_i)$

(d) Sei $\mu = \sum_{i\in I} a_i<t_i>$ eine formale Summe. Dann ist $fvar(\mu)=\bigcup_{i\in I} fvar(t_i)$.

(e) Sei $\omega = \sum_{i=1}^{k} a_i<t_i> \in FSM(SIG,V)$. Dann heißt $card(\omega) = \sum_{i=1}^{k} a_i$ die Kardinalität von ω.

Die Realisation einer formalen Summe wollen wir in der nächsten Definition beschreiben.

Definition 5.4 : Realisation einer formalen Summe

Sei $SIG = (S,\Omega,\Pi,typ_\Omega,typ_\Pi)$ eine Signatur, $V=V_S=(V_s)_{s\in S}$ eine Familie von Variablen der Sorten $s\in S$, $TERM(SIG,V)$ die Menge der Terme der Signatur SIG mit Variablen aus der Familie V und $ALS(SIG)=(A_S, A_\Omega)$ ein algebraisches System. Die Realisation der formalen Summe $FSM(SIG,V)$ in dem algebraischen System $ALS(SIG)$ ist eine Abbildung $\mathbf{R}_{FSM}:SIG \to (\bigcup_{s\in S} A_S(s)\to I\!N)^{I\!N}$, die rekursiv wie folgt definiert ist:

(i) Für $<t>$ ist $\mathbf{R}_{FSM}(<t>) = \overline{<t>}$ definiert als die Abbildung $\overline{<t>} : \bigcup_{s\in S} A_S(S) \to I\!N$ mit

$$\overline{<t>}(x) = \begin{cases} 1 & falls\ x=\mathbf{R}_T(t) \\ 0 & sonst \end{cases}$$

(ii) Für $\omega_1 + \omega_2$ ist $\mathbf{R}_{FSM}(\omega_1 + \omega_2) = \mathbf{R}_{FSM}(\omega_1) + \mathbf{R}_{FSM}(\omega_2)$, wobei das + auf der rechten Seite für die Addition von Multisummen steht.

Die formalen Summen von Termen sind Ausdrücke deren "Argumente" Terme sind. Semantisch sollen sie Eigenschaften von Eigenschaften (von Elementen aus bestimmten Mengen) ausdrücken. Die Situation ist hier ähnlich zu der, die bei der Betrachtung von dem Prinzip der mathematischen Induktion vorkommt. Bei diesem Prinzip hat man nicht nur Eigenschaften von natürlichen Zahlen, sondern auch Eigenschaften von diesen Eigenschaften, wie z.B. die Vererblichkeit. Deshalb erfordert die Betrachtung von formalen Summen das Einführen von einer neuen Art von Variablen für Terme, die nicht die Eigenschaften von Elementen aus entsprechenden Mengen, sondern die Eigenschaften von Relationen zwischen diesen Elementen bezeichnen.
Zur Beschreibung solcher Strukturen benötigt man bereits eine Sprache 2. Ordnung. Wir wollen hier diese Dinge nicht weiter besprechen, sondern nur auf dieses unangenehme Merkmal der Theorie der Multimengen hinweisen. Eine gute Einführung in die Problematik der Sprachen höherer Ordnung kann man z.B. in [Her76a] oder [Mal76a] finden.

Das Wort "Multimenge" ist relativ neu. Die Semantik, d.h. die Bedeutung des Wortes ist ein wenig paradox. Eine Multimenge besteht aus Elementen die nicht immer paarweise unterscheidbar sind. Es kann also passieren, daß zwei verschiedene Elemente derselben Multimenge im gewissen Sinne "identisch" sind. In dieser Situation sagt man auch, daß ein Element mit einer *Vielfachheit* größer als 1 in der Multimenge vorkommt. Diese zwei Gesichtspunkte führen zur Definition einer Multimenge als eine Äquivalenzrelation oder als eine Funktion. Beide Definitionen sind eng verbunden. Für jede Äquivalenzrelation (Multimenge als Äquivalenzrelation) $r \subseteq A \times A$ wird die entsprechende Funktion (Multimenge als Funktion) durch die folgende Zuordnung definiert:

$$A/r \to I\!N_o$$
$$[x] \to card([x])$$

Diese Zuordnung ordnet jeder Äquivalenzklasse $[x]_r$, d.h. jedem Element der Faktormenge A/r ihre Mächtigkeit zu. So ist für jede Multimenge als Äquivalenzrelation die entsprechende Multimenge als Funktion eindeutig definiert. Damit ist die Äquivalenz der beiden Definitionen in dieser Richtung klar.

Die andere Richtung der Äquivalenz ist nicht so einfach ersichtlich. Die einfachste Lösung scheint hierbei zu jedem Paar (x,m(x)), das dieser Multimenge als Funktion angehört, eine Menge A_x zu suchen, die die Bedingung $card(A_x) = m(x)$ erfüllt und diese Menge dann als Äquivalenzklasse zu nehmen. Man sollte dabei beachten, daß für verschiedene $x,y \in dom(m)$ die entsprechenden Mengen (Äquivalenzklassen) disjunkt sind. Diese Bedingung ist auch leicht zu erfüllen, wenn für $x \neq y$ die Mengen A_x und A_y nicht disjunkt sind. Dann kann man sie durch die Produktmengen $\{x\} \times A_x$ und $\{y\} \times A_y$ ersetzen.

Das Problem liegt hier aber darin, daß es für jedes x ∈ dom(m) unendlich viele Mengen A_x geben kann, die die Bedingung card(A_x) = m(x) erfüllen, und daß es unter ihnen keine vernünftige kanonische (d.h. eine die für alle Elemente x auf dieselbe Weise mit demselben Verfahren bestimmt werden kann) gibt. Man kann also nicht sagen, daß es für jede Multimenge als Funktion m: dom(m) → $\mathbb{N}_0$ nur eine durch diese Funktion bestimmte Multimenge als Äquivalenzrelation gibt, die für jedes x ∈ dom(m) die Bedingung card($[x]_r$) = m(y) erfüllt. Hierbei ist [x] die dem Element y zugeordnete Äquivalenzklasse. In diesem Sinne ist die Definition der Multimenge als Äquivalenzklasse "besser" als diese, die sie als Funktion bezeichnet.

In der Petri-Netz-Theorie wird aber eine Multimenge stets als eine Funktion definiert, weil die "Rechnungen" mit diesen Funktionen wesentlich einfacher als mit Äquivalenzrelationen sind.

5.1 Hinweise zur Literatur

Multimengen kann man als Funktionen mit Werten in der Menge $\mathbb{N}$ der natürlichen Zahlen oder als Äquivalenzrelation betrachten. Vom logischen Standpunkt aus ist natürlich der zweite Fall viel einfacher, gibt aber keine guten "Rechenmechanismen". Die Darstellung der Multimengen als Funktionen erlaubt es aber, sie als Elemente eines Moduls oder eines Vektorraums zu betrachten und viele Methoden der linearen Algebra zu verwenden. Wir haben hier diese Auffassung gewählt, weil wir auf Multimengen verschiedene "Rechenoperationen" durchführen müssen.

Multimengen werden in verschiedenen Gebieten (nicht nur in der Informatik) verwendet. Die Theorie wurde aber in keinem separaten Buch beschrieben. Wir geben hier zwei Arbeiten an, die nach unserer Meinung eine gute Einführung sind. Diese sind:

- Eine Arbeit über die Anwendung von Multimengen in der Automatentheorie ist [Der79a].
- Eine Arbeit mit Anmerkungen zu Multimengen ist [Hic80a].

6 Petri Netze

Petri Netze waren eines der ersten mathematischen Modelle der Nebenläufigkeit. Die Einfachheit der Theorie, die keine große Mathematikkenntnis erfordet, macht aus Petri Netzen ein gutes Werkzeug für Ingenieure, die sich mit verschiedenen Aspekten der nebenläufigen Systeme beschäftigen. In diesem Kapitel beschreiben wir die elementaren, grundlegenden Begriffe der Theorie der sogenannten Prädikat-Transitions-Netze.

Definition 6.1 : (Petri Netze)

Ein *Petri Netz* ist ein Tupel $N = (P,T,F,K)$, wobei P und T Mengen von *Stellen* und *Transitionen* mit der folgenden Eigenschaft sind:

(i) $P \cap T = \emptyset$

(ii) $F \subseteq P \times T \cup T \times P$

Weiterhin ist $K : P \rightarrow \mathbb{N}$ eine Funktion, die jeder Stelle $p \in P$ eine Kapazität $K(p) \in \mathbb{N}$ zuordnet.

Man veranschaulicht sich ein Petri-Netz, indem man die Stellen als Kreise, die Transitionen als Rechtecke und die Elemente der Flußrelation $F \subseteq P \times T \cup T \times P$ als gerichtete Kanten darstellt. Die Kapazitäten werden als Anschriften an die Stellen geschrieben. Betrachten wir hierzu das Beispiel aus der Einleitung.

Beispiel 6.1

Sei $P=\{p_1,p_2,p_3,p_4,p_5\}$ die Menge der Stellen, $T=\{t_1,t_2,t_3,t_4\}$ die Menge der Transitionen und

$$F=\{(t_1,p_2),(p_2,t_2),(t_2,p_1),(p_1,t_1),(t_2,p_3),(p_3,t_2),(p_3,t_3),(t_3,p_3),(t_3,p_5),(p_5,t_4),(t_4,p_4),(p_4,t_3)\}$$

die Flußrelation. Die Kapazitätsfunktion sei gegeben durch $K(p)=2$ für $p=p_5$ und $K(p)=n$ für $p=p_1,p_2$ und $K(p)=1$ für p_3,p_4. Das folgende Bild zeigt dann das Netz in der Veranschaulichung:

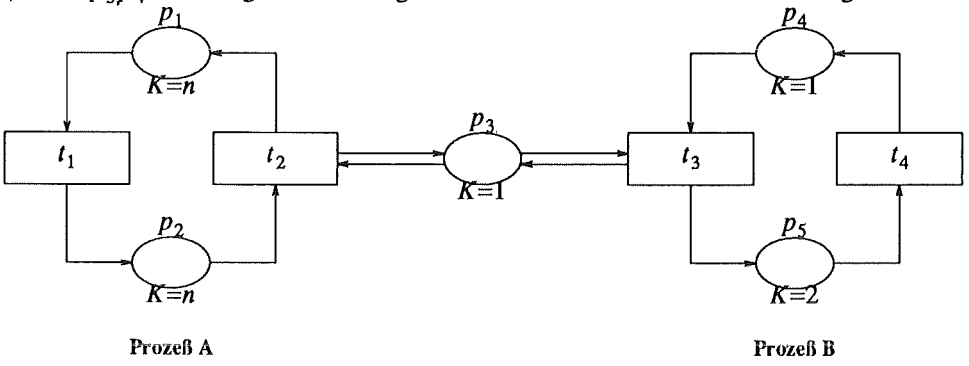


Abb. 6–1 : Petri-Netz des Beispiels 6.1

Definition 6.2

Es sei N = (P,T,F,K) ein Netz. Für jedes Element $x \in P \cup T$ definieren wir

$$^\bullet x = \{y \in P \cup T \mid yFx\} \ , \ x^\bullet = \{y \in P \cup T \mid xFy\}$$

Wenn $x \in T$ ist, schreiben wir auch vor(x) statt $^\bullet x$ und nach(x) statt $x^\bullet$. Die Elemente der Mengen vor(t) bzw. nach(t) werden entsprechend Vor - bzw. Nachstellen der Transition t genannt.

Im Folgenden werden nur solche Petri Netze betrachtet, die die folgenden Bedingungen erfüllen:

(i) $P \cup T \neq \emptyset$

(ii) Die Mengen P und T sind endlich.

Wie wir bereits in der Einleitung beschrieben haben, wünscht man sich in der Praxis Netze, die eine problemnahe Modellierung erlauben. Da der Anwender es mit konkreten Objekten und Abläufen auf diesen Objekten zu tun hat, wünscht er sich eine Netzform, in der "individuelle Objekte" aus seiner Problemwelt und die zugehörigen Aktionen zur Modellierung verwendet werden. Dies führte zu einer Reihe von Petri-Netzformen mit individuellen Marken und Beschriftungen.

Wir wollen zunächst informell beschreiben, wie wir die Anforderungen an eine solche Netzform auf den Begriff des algebraischen Systems zurückführen können. Die grundlegende Philosophie hierbei ist, daß wir die Marken des Netzes als Elemente der Trägermengen eines algebraischen Systems auffassen, und die Bewegung der Marken als Fluß der zugehörigen Elemente des algebraischen Systems durch das Netz definieren. Wenn wir nun ein algebraisches System $S=(A_1,..,A_k,f_1,...f_k,r_1,\ldots,r_n)$ haben, so wird in der Regel nur eine Teilfolge der Folge von Trägermengen Marken (z.B. Warteschlangen) und der Rest der Trägermengen Attribute von Marken (z.B. die Länge einer Warteschlange) beschreiben. Es ist daher günstig, die Folge der Trägermengen $A_1,...,A_k$ in zwei Teilfolgen $\mathbf{M}=M_1,...,M_{k_1}$ von Marken und $\mathbf{B}=B_1,\ldots,B_{k_2}$ von Attributen von Marken zu zerlegen.

Nach unserer Philosophie muß nun eine Markierung jeder Stelle eines Petri-Netzes $N=(P,T,F,K)$ eine Anzahl von Elementen aus der Folge **M** zuordnen, wobei wir durchaus zulassen, daß von einem Element mehrere Kopien gleichzeitig auf einer Stelle liegen können. Die Markierung einer Stelle ist daher eine *Multimenge* über **M**, wobei wir in diesem Kontext **M** als die Vereinigung der Mengen aus **M** auffassen. Eine Markierung des Netzes ist damit eine Funktion der Gestalt $M:P\rightarrow MULT(\mathbf{M})$.

Die Dynamik eines Systems wird nun weiter durch eine Veränderung der Markierung des zugehörigen Petri-Netz-Modells durch "Schalten von Transitionen" des Netzes beschrieben. Das Schalten einer Transition wollen wir dabei als Fluß von Marken innerhalb der Umgebung der Transition betrachten. Daher läßt sich ein Schaltvorgang dadurch charakterisieren, welche Marken von jeder Vorstelle $p\in vor(t)$ der Transition abfließen, und welche Marken in jede Nachstelle $p\in nach(t)$ hineinfließen. Es stellt sich nun die Frage, wie wir die Mengen der Marken, die von einer Stelle abfließen bzw. in eine Stelle hineinfließen, beschreiben können. Hierzu bietet sich in natürlicher Weise eine Beschriftung der gerichteten Kanten an, um den Zu- und Abfluß von Marken auch bildlich sichtbar zu machen. Die Beschriftung der Kante gibt dabei an, welche Multimenge von Marken in einem konkreten Schaltvorgang von der zugehörigen Stelle abzuziehen bzw. auf die zugehörige Stelle zu legen ist. Die Marken, die abzuziehen sind bzw. auf Stellen gelegt werden, sollten dabei über die Operationen des algebraischen Systems, auf Grundlage des aktuellen Zustandes in der Umgebung einer Transition (welche Marken auf den Vor- und Nachstellen vorhanden sind), bestimmbar sein. Die Kantenbeschriftung sollte daher eine Multimenge $\phi_F(p,t)$ bzw. $\phi_F(t,p)$ über die Polynome des algebraischen Systems sein. Bei Einsetzen eines konkret durch den aktuellen Zustand in der Umgebung einer Transition bestimmten Schaltmodus v (Einsetzen eines Urbildwertes des Polynoms in Form eines Tupels aus Marken, die auf den Umgebungsstellen liegen) erhalten wir dann die Multimenge $\phi_F(p,t)(v)$ bzw. $\phi_F(t,p)(v)$ der Marken, die von der Stelle p abzuziehen bzw. auf die Stelle p zu legen sind. Die Beschriftung der Kanten ist damit eine Abbildung der Gestalt $A_F:F\rightarrow POL(S)$.

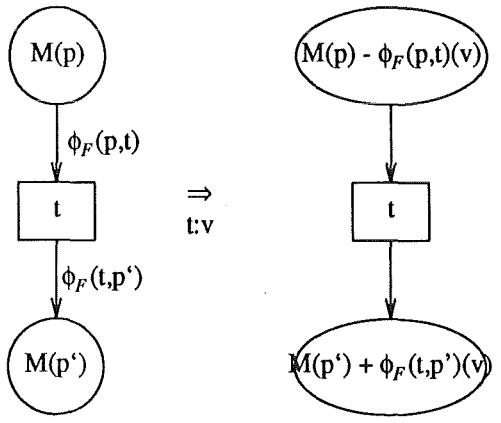


Abb. 6–2 : Schaltvorgang als Fluß von Multimenge von Marken

Einen Schaltvorgang kann man dann als ein Tripel $(M,t:v,M')$ auffassen, wobei die auf M nach Ausführung der Transition t im Modus v folgende Markierung M' durch $M'(p)=M(p)-\phi_F(p,t)(v)+\phi_F(t,p)(v)$ gegeben ist.

Das bislang entwickelte Modell hat nun noch einige Unzulänglichkeiten. Zum einen haben wir noch nicht erklärt, nach welchem Verfahren wir einen Schaltmodus (Festlegung der Urbildwerte, die in die Polynome eingesetzt werden) bestimmen. Hierzu ordnen wir jeder Transition eine Relation bzgl. des algebraischen Systems S so zu, daß ein erlaubter Schaltmodus genau durch solche Tupel von Marken, die auf Umgebungsstellen

liegen, gegeben ist, die die zur Transition gehörige Relation erfüllen. Die Beschriftung der Transitionen ist damit eine Abbildung der Gestalt $A_T{:}T{\to}REL(S)$.

Eine weitere Unzulänglichkeit liegt bei der Behandlung von Stellen. In unserem Modell können wir Stellen als "Speicherplätze" für die individuellen Marken ansehen. An solche Speicherplätze werden in der Praxis nun in der Regel weitere Anforderungen gestellt. So kann einer Stelle, die eine Warteschlange beinhaltet, eine Relation zugeordnet sein, die festlegt, daß die Länge der betreffenden Warteschlange eine fest vorgegebene Grenze nicht überschreiten darf. Die Formulierung solcher Bedingungen können wir einfach dadurch ermöglichen, daß wir auch Stellen mit einer Relation aus $REL(S)$ beschriften. Die Beschriftung von Stellen ist dann eine Abbildung $A_P{:}P{\to}REL(S)$.

Bei der Realisierung unseres Konzeptes müssen wir allerdings noch einige Konsistenzbedingungen beachten. So muß jeder Stelle implizit ein "Typ", der bestimmt, welche Trägermengen potentielle Marken für diese Stelle liefern dürfen, zugeordnet sein. Die zu dieser Stelle gehörenden Kanten dürfen dann nur mit Polynomen eines zu der Stelle passenden "Typs" beschriftet sein. Man wird daher eine Konsistenzbedingung der Gestalt $typ(A_F(f))=typ(A_P(p))$ für $f=(p,t)$ bzw. $f=(t,p)$ haben.

Während wir in der vorausgehenden Motivation unser Netz-Modell auf Grundlage einer informellen semantischen Betrachtungsweise entwickelt haben, werden wir in der folgenden formalen Realisierung des Konzeptes das Netz-Modell zunächst rein syntaktisch über einer formalen Sprache 1. Ordnung definieren und dann in der Weise interpretieren, daß die Realisierung der Syntax über ein algebraisches System ein Netz-Modell gemäß der obigen Motivation liefert. Wir beginnen daher mit der folgenden syntaktischen Definition der Beschriftung eines Netzes:

Definition 6.3 : (Netzbeschriftung)

Es sei $N=(P,T,F,K)$ ein Petri Netz, $SIG=(S,\Omega,\Pi,typ_\Omega,typ_\Pi)$ eine Signatur, $V_S : S \to$**Set** eine Familie von Variablen und $L=(ALPH,TERM(SIG,V_S),FORM(ALPH^0,SIG,V_S,Q))$ eine Sprache der ersten Ordnung über SIG mit Variablen aus V_S. Die Beschriftung des Netzes N in der Sprache L ist ein Tripel $A_N=(A_P, A_T, A_F)$, wobei gilt:

(i) $A_P : P \to FORM(ALPH^0,SIG,V_S,Q)$, $A_T : T \to FORM(ALPH^0,SIG,V_S,Q)$, $A_F : F \to FSM(SIG,V_S)$ seien Abbildungen.

(ii) Für jede Stelle $p\in P$ existiert ein $s\in S$ und ein $v\in V_S$, so daß $fvar(A_P(p))=\{v\}$ ist und für jedes Tupel $(p,t)\in F$ bzw. $(t,p)\in F$ gilt: $typ(A_F(p,t))=\{s\}$ bzw. $typ(A_F(t,p))=\{s\}$.

Die Bedingung (i) ist klar. Die Bedingung (ii) beschreiben wir ein wenig genauer. Die Stellen eines Netzes können wir gemäß unserer Motivation als "Speicherzellen" von Elementen einer gewissen Sorte interpretieren. In jeder solchen Stelle dürfen sich, bezüglich einer Interpretation des Netzes, nur Multimengen über Elementen dieser Sorte befinden, die die Formel $A_P(p)$ erfüllen. Die Elemente von dieser Multimenge erhält man als die Werte einer Realisation der Terme (angewandt auf eine Auswertung der Variablen), die in einer formalen Summe $A_F(t,p)$ vorkommen. Es ist also klar, daß die Bedingung $typ(A_F(t,p))=\{s\}$ für jede Stelle $p\in P$ und jede Kante $(t,p)\in F$ erfüllt werden muß.

Definition 6.4 : (PrT-Netz)

Für jedes Netz $N=(P,T,F,K)$ und jede Beschriftung $A_N=(A_P, A_T, A_F)$ des Netzes N wird das Paar (N,A_N) ein PrT -Netz genannt.

Gemäß unserer Definition der Beschriftung muß die kleinste formale Sprache $L^0((N,A_N))$, die ein Prt-Netz beschreibt, die Bedingungen $A_P(S)\subseteq L^0((N,A_N))$, $A_F(S)\subseteq L^0((N,A_N))$ und $A_T(S)\subseteq L^0((N,A_N))$ erfüllen. Diese Sprache ist daher von der 2. Ordnung, da die Beschriftung von Kanten über formale Summen nicht in einer Sprache 1. Ordnung formuliert werden kann. Wegen der Komplexität einer Formalisierung von Konzepten einer Sprache 2. Ordnung werden wir aber die Aspekte 2. Ordnung in unserem Netz-Modell nicht genauer formalisieren, sondern nur in ihrer intuitiv verständlichen Form verwenden.

Als Beispiel für ein PrT-Netz versehen wir unser Netz aus Beispiel 6.1 mit einer geeigneten Beschriftung.

Beispiel 6.2

Sei $N=(P,T,F,K)$ das Petri-Netz aus dem Beispiel 6.1. *LDMS* sei die Sprache 1. Ordnung aus dem Beispiel 4.2. Wir ordnen dem Netz nun die folgenden Beschriftungen zu:

(1) Die Beschriftung der Transitionen sei stets die Formel *true*.

(2) Die Beschriftung der Stellen p_1 und p_4 sei gegeben durch die Formel $isprocess(pr)$.

(3) Die Beschriftung der Stellen p_2 und p_5 sei gegeben durch die Formel $isdaten(n)$.

(4) Die Beschriftung der Stelle p_3 sei gegeben durch die Formel $iswarteschlange(w)$.

(5) Die Beschriftung der Kanten sei gegeben durch $A_F(t_1,p_2)=<daten(pr_1)>$, $A_F(p_2,t_2)=<n_1>$, $A_F(t_2,p_1)=<process(n_1)>$, $A_F(p_1,t_1)=<pr_1>$, $A_F(t_2,p_3)=<ein(w,n_1)>$, $A_F(p_3,t_2)=<w>$, $A_F(p_3,t_3)=<w>$, $A_F(t_3,p_3)=<rest(rest(w))>$, $A_F(t_3,p_5)=<aus(w)>+<aus(rest(w))>$, $A_F(p_5,t_4)=<n_1>+<n_2>$, $A_F(t_4,p_4)=<empf()>$ und $A_F(p_4,t_3)=<empf()>$.

Das Bild 6.2 veranschaulicht dann das beschriftete Netz.

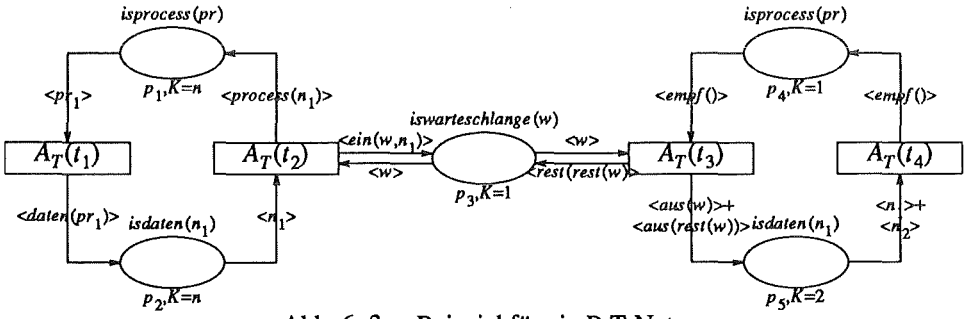


Abb. 6–3 : Beispiel für ein PrT-Netz

Wir wollen nun definieren, was eine Markierung eines PrT-Netzes ist.

Definition 6.5 : (Markierung eines Netzes)

Sei (N,A_N) ein PrT-Netz mit Signatur $SIG = (S,\Omega,\Pi,typ_\Omega,typ_\Pi)$ und Variablen aus der Familie $V_S : S \rightarrow$**Set**.

(a) Eine Abbildung $M : P \rightarrow FSM(SIG,V_S)$ heißt eine Markierung des PrT-Netzes (N,A_N), falls gilt:

(i) $typ(M(p)) = typ(A_P(p))$ für alle $p \in P$,

(ii) $fvar(M(p)) = \emptyset$ für alle $p \in P$.

(b) Sei $M_0 : P \rightarrow FSM(SIG,V_S)$ eine Markierung. Das Tripel (N,A_N,M_0) heißt PrT-System mit Anfangsmarkierung M_0.

Betrachten wir ein Beispiel für ein PrT-Netz mit einer Markierung.

Beispiel 6.3

Wir betrachten wieder das Netz $N=(P,T,F,K)$ aus dem Beispiel 6.1 mit der in Beispiel 6.2 definierten Beschriftung. Eine Anfangsmarkierung ist dann durch $m(p_1)=<a_1>+\cdots+<a_n>$, $m(p_4)=b$ gegeben, wenn wir alle weiteren Stellen mit der formalen Summe 0, die wir nie explizit angeben werden, versehen denken. Das folgende Bild zeigt das Netz mit Anfangsmarkierung.

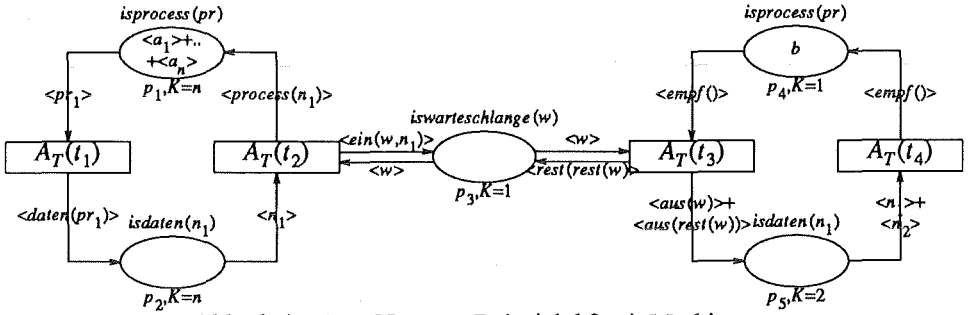


Abb. 6–4 : PrT-Netz aus Beispiel 6.3 mit Markierung

Definition 6.6

Es sei $N = (P,T,F,K)$ ein Petri-Netz und A_N eine Beschriftung des Netzes N in einer Sprache L. Für alle $t \in T$ ist

$$fvar(t) = \bigcup_{p \in P} fvar(A_F(t,p)) \cup \bigcup_{p \in P} fvar(A_F(p,t))$$

$$hang(t) = fvar(A_t(t)) - fvar(t)$$

Die Menge $fvar(t)$ ist also die Menge aller freien Variablen in den Beschriftungen der Eingangs- und Ausgangskanten einer Transition t. Um die dynamischen Eigenschaften eines PrT-Systems definieren zu können, müssen wir die Beschriftung und Markierung des Systems auswerten können. Hierzu ist eine Realisation der Sprache notwendig.

Definition 6.7 : (realisiertes PrT-System)

Es sei (N,A_N,M_0) ein PrT-System über der Signatur $SIG = (S,\Omega,\Pi,typ_\Omega,typ_\Pi)$ und $ALS(SIG) = (A_S,A_\Omega,A_\Pi)$ ein algebraisches System über SIG. Eine *Realisation von* (N,A_N,M_0) mit einer Realisierung der Terme in dem algebraischen System $ALS(SIG)$ ist dann definiert durch:

(a) Eine Realisierung $\mathbf{R} = (\mathbf{R}_T, \mathbf{R}_F)$ der Formeln des Netzes mit einer Realisierung der Terme über das algebraische System *ALS(SIG)*.

(b) Eine Realisierung der formalen Summen auf den Kanten in dem algebraischen System *ALS(SIG)*.

(c) Eine Realisierung der Anfangsmarkierung M_0 in dem algebraischen System *ALS(SIG)*.

(d) Für alle $p \in P$ mit $M_0(p) = \sum_{i=1}^{n} a(\tau_i) <\tau_i>$, $\tau_i \in TERM(SIG,V_S)$ für alle $i=1,..,n$, und $fvar(A_P(p)) = \{v\}$ gilt:

$$(A_P(p))_{\mathbf{R}}(v \rightarrow \tau_i) = \mathsf{v} \text{ für alle i=1..n.}$$

Ein PrT-System mit einer Realisierung nennen wir auch *realisiertes PrT-System.*

Den Begriff des realisierten Netzes kann man sich grob in der folgenden Weise vorstellen. Ein PrT-Netz ist zunächst einmal ein Graph, dessen Knoten und Kanten mit verschiedenen Zeichenfolgen (ohne einer semantischen Bedeutung) beschriftet sind. Die Beschriftungen sind damit rein syntaktische Objekte, denen keine semantische Bedeutung zukommt. Erst durch eine Realisation wird jeder Beschriftung auch eine semantische Bedeutung zugeordnet, wobei Knotenbeschriftungen als auswertbare Formeln bzgl. einer Interpretation der einzelnen Prädikate als Relationen, der Operatoren als Funktionen, usw. oder formale Summen als Multimengen aufgefaßt werden.

Ein realiertes PrT-Netz kann man sich also als einen bipartiten Graphen vorstellen, dessen Knoten und Kanten nicht mit rein syntaktischen Zeichenfolgen sondern mit Relationen und Multimengen (von Funktionen) beschriftet werden, so daß der Netzbeschriftung eine semantische Bedeutung zukommt. Dies entspricht unserer Motivation am Anfang des Kapitels. Die Frage ist nun, wie wir die Schaltbedingung des Netzes formulieren. Zum einen sollte die Markierungssituation in der Umgebung einer Stelle so sein, daß bei einer Substitution der Variablen innerhalb der Formel der Transition, die die in der Umgebung vorhandenen Marken auf den Stellen berücksichtigt, die Formel der Transition zu "true" ausgewertet wird. Zum anderen sollte die Transition bei jeder Substitution, die die obige Bedingung erfüllt, schalten können. Dies führt zu den folgenden Definitionen.

Definition 6.8 : (Aktivierbarkeit von Schritten)

Sei (N,A_N) ein realisiertes PrT-Netz mit Signatur $SIG = (S,\Omega,\Pi,typ_\Omega,typ_\Pi)$ und Variablen aus der Familie $V_S : S \rightarrow$**Set**. Sei weiter $ALS(SIG) = (A_S,A_\Omega,A_\Pi)$ ein algebraisches System über SIG und **R** eine Realisierung der zum Netz gehörigen Sprache mit einer Realisierung der Terme über $ALS(SIG)$.

(a) Die Multimenge $\mu = \sum_{t\in T} a_t\, t$ ist in einer Markierung $M : P \rightarrow FSM(SIG,V_S)$ des Netzes (N,A_N) im Modus v aktivierbar, dann und nur dann, wenn v eine Auswertung $v = (v_s : V_S^f(s) \rightarrow A_S(s))_{s\in S}$ ist, für die gilt:

(1) Für alle $t\in T$ mit $a_t\neq 0$ gilt: Für jede Variable $x \in fvar(A_T(t))$ gibt es eine Stelle $p\in {}^\bullet t^\bullet$, so daß $v(x)\in |(M(p))_{\mathbf{R}}|$ gilt.

(2) Für alle $t\in T$ mit $a_t\neq 0$ gilt: $A_T(t)_{\mathbf{R}}\,(v) = v$

(3) Für alle $p\in P$ gilt: $\sum_{t\in T} a_t\,(A_F(p,t))_{\mathbf{R}}(v) \leq (M(p))_{\mathbf{R}}$

(4) Für alle $p\in P$ gilt: $card(M(p)) - \sum_{t\in T} a_t\, card(A_F(p,t)) + \sum_{t\in T} a_t\, card(A_F(t,p)) \leq K(p)$

(5) Für alle $t\in T$ mit $a_t\neq 0$ und $p \in nach(t)$ mit $A_F(t,p) = \sum_{i\in I} a_i(\tau_i) <\tau_i>$, $\tau\in TERM(SIG,V_S)$, gilt: $(A_P(p))_{\mathbf{R}}(\tau_i(v))=v$ für alle $i\in I$.

(b) Die Multimenge $\mu = \sum_{t\in T} a_t\, t$ heißt in einer Markierung $M : P \rightarrow FSM(SIG,V_S)$ des Netzes (N,A_N) aktivierbar, genau dann wenn es eine Auswertung $v = (v_s : V_S(s) \rightarrow A_S(s))_{s\in S}$ gibt, so daß sie in dieser Markierung im Modus v aktivierbar ist.

(c) Sei $\mu = \sum_{t\in T} a_t t$ eine Multisumme von Transitionen. μ heißt *Schritt*, wenn eine Markierung M aus der Menge aller Markierungen des Netzes N existiert, so daß μ unter M aktivierbar ist. Die Menge aller Schritte bezeichnen wir mit $\mathbf{S}_N$.

Mit der Definition der Aktivierbarkeit einer Multimenge können wir nun die Schrittregel angeben, die den dynamischen Ablauf in einem Netz bestimmt.

Definition 6.9 : (Schrittregel)

Sei (N,A_N) ein PrT-Netz mit Signatur $SIG = (S,\Omega,\Pi,typ_\Omega,typ_\Pi)$ und Variablen aus der Familie $V_S : S \rightarrow$**Set**. Sei weiter $ALS(SIG) = (A_S,A_\Omega,A_\Pi)$ ein algebraisches System über SIG und **R** eine Realisierung des Netzes mit einer Realisation der Terme über $ALS(SIG)$. $v = (v_s : V_S(s) \rightarrow A_S(s))_{s\in S}$ sei eine Auswertung und $M : P \rightarrow FSM(SIG,V_S)$ eine Markierung des Netzes (N,N_N).

$\mu = \sum_{t\in T} a_t t$ sei ein in der Markierung M im Modus v aktivierbarer Schritt. Dann nennt man das Tripel (M, μ, M'), wobei $M' : P \rightarrow FSM(SIG,V_S)$ eine Markierung des Netzes (N,A_N) ist, die durch die Formel

$$\forall_{p\in P}\ M'(p) = M(p) - \sum_{t\in T} a_t\, A_F(p,t)(v) + \sum_{t\in T} a_t\, A_F(t,p)(v)$$

definiert ist, einen *Schaltvorgang des Netzes*.

Betrachten wir auch an dieser Stelle ein Beispiel.

Beispiel 6.4

Wenn wir das PrT-System aus dem Beispiel 6.3 über die Algebra $ALG(DMS)$ aus dem Beispiel 3.1 wie in Beispiel 4.3 und 4.6 realisieren, kann die Transition t_1 in dem Mode $pr_1 \rightarrow a_1$ schalten, da die Formel $A_T(t_1)$ stets *true* ist und a_1 als Marke auf der Stelle p_1 vorhanden ist. Das Netz geht dann in die Markierung im Bild 6.4 über.

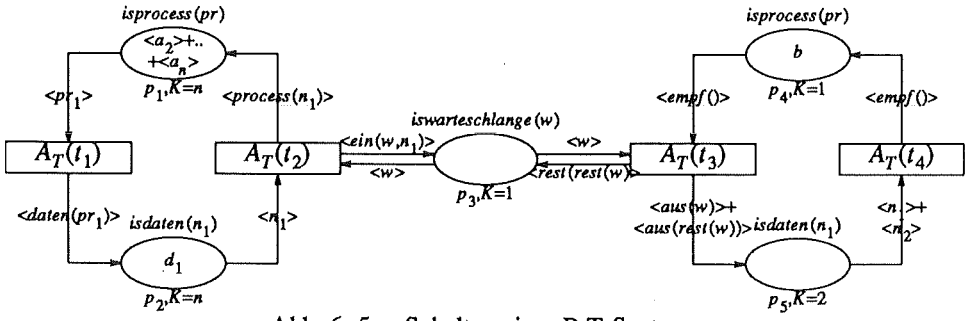


Abb. 6–5 : Schalten eines PrT-Systems

Das Verhalten von Petri-Netzen wird durch das (eventuell nebenläufige) Schalten von Multimengen von Transitionen bestimmt. Die Hauptprobleme, die dabei untersucht werden, sind die folgenden:

(1) Erreichbarkeitsproblem
Zu untersuchen ist hier die Klasse aller Markierungen, die von einer gegebenen Markierung aus erreicht werden können.

(2) Sprach-theoretische Probleme
Diese Probleme sind im gewissen Sinne ähnlich zu den Problemen, die in der klassischen Theorie der formalen Sprachen und Automaten betrachtet werden. Sehr grob kann man sagen, daß es um eine Art der Charakterisierung der Klasse aller möglichen Prozesse in einem Netz geht.

(3) Verschiedene Invarianzprobleme
Man kann nach solchen Eigenschaften der Markierungen (oder Prozesse) fragen, die durch die Erreichbarkeitsrelation (oder eine "Unterrelation" einer Erreichbarkeitsrelation) erhalten bleiben.

Eine andere Art von Problemen, die im praktischen Einsatz der Modellierung von Systemen von größerer Bedeutung ist, sind verschiedene Fragen nach der Transformation von Petri-Netzen. Dasselbe System kann man auf viele verschiedene Weisen mit Petri-Netzen beschreiben. Hier ist es sehr wichtig zu wissen, wie die entsprechenden Eigenschaften des Systems (des entsprechenden PrT-Modells) transformiert werden.
Eine sehr natürliche Frage ist hier z.B. die Frage nach der Erhaltung von Erreichbarkeitsrelationen, d.h. die Frage, ob aus der Tatsache, daß eine Markierung M_2 von einer anderen Markierung M_1 in einem Netz N erreichbar ist, auch folgt, daß ihr Bild M_2' vom Bild M_1' der Markierung M_1 aus erreichbar ist.
Sehr viele dieser Probleme können auf die lineare Algebra zurückgeführt werden, wo sie mit Hilfe der für diese Disziplin charakteristischen Methoden effektiv gelöst werden können. Wir werden in den folgenden Kapiteln einige der eben angesprochenen Probleme genauer erörtern.

6.1 Hinweise zur Literatur

Petri Netze wurden von C.A. Petri im Jahre 1962 als eine Art von Modell der Kommunikation eingeführt. Seit den siebziger Jahren kann man eine starke Entwicklung der Theorie beobachten, und in den letzten Jahren sind auch einige Bücher über Petri Netze erschienen.

- Ein sehr leicht geschriebenes Buch über grundlegende Begriffe der Theorie der Petri Netze, das als eine gute Einführung in die Theorie dienen kann, ist [Rei86a].
- Ein klar und sehr "systematisch" geschriebenes Buch ist [Sta85a]. Beim Lesen ist die Kenntnis der Theorie der formalen Sprachen und ein wenig "mathematische Kultur" erforderlich.
- Eine Beschreibung der PrT-Netze nach Genrich und Lautenbach findet man in [Gen79a, Gen81a] und [Gen86a].
- Eine weitere Form von Höheren Petri-Netzen stellen die Coloured Netze von J. Jensen dar, die man z.B. in [Jen81a] und [Jen81b] findet.
- J. Vautherin und G. Memmi beschreiben in [Vau87a] und [Mem87a] ein Konzept, daß die Petri-Netz-Theorie mit der Theorie der algebraischen Spezifikation verknüpft. Einen ähnlichen Ansatz verfolgt W. Reisig in [Rei89a].

Eine sehr gute Übersicht über den "Arbeitsstand" in der Theorie von Petri Netzen kann man in den unregelmäßig von der GMD herausgegebenen Bibliographien (Berichte der GMD) finden.

7 Prozesse

Das Wort Prozeß wird auf verschiedene Art und Weise verstanden. Grob kann man sagen, daß ein Prozeß als eine Beschreibung der Änderungen eines Systems aufgefaßt werden kann. Welche Aspekte dieser Änderungen durch diese Beschreibung abgebildet werden, hängt von konkreten Auffassungen ab. In Systemen, die durch Petri Netze modelliert werden, sind das "kausale Aspekte der Änderungen". Prozesse werden hier durch eine spezielle Art von Petri Netzen modelliert. Die Flußrelation in solchen Netzen stellt gerade die kausale Abhängigkeit der Änderungen des betrachteten Systems dar. Diese Relation hat hier zwei wichtige Eigenschaften, die sie von der Flußrelation der "normalen" Petri Netze unterscheidet.

1. Sie ist nicht umkehrbar, d.h. es wird eine "Richtung der Kausalität" definiert. Diese Richtung entspricht ungefähr dem Begriff des "Zeitpfeiles" im Zeitraum und wird durch eine Halbordnung bestimmt. Man kann sagen, daß die Kausalität in Prozessen zyklusfrei ist.
2. Wir wollen einen Prozeß als eine "ex post" Beschreibung der Änderungen eines Systems betrachten. Der Prozeß soll also etwas, was schon geschehen ist, darstellen. Man kann daher annehmen, daß diese Beschreibung keine Konflikte enthält.

Die obigen Bedingungen führen zu der folgenden Definition des Netzes, das eine Darstellung eines Prozesses sein soll.

Definition 7.1 : (Occurence Netz)

Sei $N = (P,T,F,K)$ ein Petri Netz. N ist ein *Occurence Netz*, falls

1. $F^+ \cap id = \emptyset$ (N ist azyklisch: $(x,y) \in F^+ \Rightarrow (y,x) \in F^+$)
2. $\forall_{p \in P} \; | {}^\bullet p | \leq 1 \wedge | p^\bullet | \leq 1.$

Da die Kapazität der Stellen eines Occurence Netzes für uns nur eine untergeordnete Bedeutung hat, lassen wir die Kapazitätsfunktion des öfteren auch weg und schreiben nur $N = (P,T,F)$ für ein Occurence Netz.

Die von uns betrachteten Prozesse sind immer *Prozesse in einem System*, wir wollen keine "Prozesse an sich" betrachten. Der Zusammenhang zwischen einem Prozeß und einem System wird durch eine Abbildung der entsprechenden Netze bestimmt. Aus vielen Gründen ist es günstig den Prozeß als eine solche Abbildung zu definieren.

Definition 7.2 : (Semiprozesse)

Es seien (N,A_N) und $(N',A_{N'})$ PrT-Netze. Eine Abbildung $spro:X(N) \rightarrow X(N')$ heißt ein Semiprozeß in dem Netz N', dann und nur dann wenn gilt:

(1) N ist ein Occurence Netz

(2) $spro(P(N)) \subseteq P(N')$ und $spro(T(N)) \subseteq T(N')$

(3) für $x,y \in X(N)$ gilt $(x,y) \in F \Leftrightarrow (spro(x),spro(y)) \in F'$

(4) für beliebiges $t \in T$ ist die Begrenzung von $spro$ auf die Menge ${}^\bullet t^\bullet$ eine Bijektion von ${}^\bullet t^\bullet$ auf ${}^\bullet spro(t)^\bullet$.

(5) für jedes $p \in P(N)$ gilt: $K(p) = K'(spro(p))$ und $A_P(p) \Rightarrow A'_P(spro(p))$

(6) für jedes $t \in T(N)$ gilt: $A_T(t) \Rightarrow A'_T(spro(t))$

(7) für jedes Paar $(x,y) \in F(N)$ ist: $A_F(x,y) = A'_F(spro(x),spro(y))$

Wir erläutern die Intuition der obigen Bedingungen ein wenig. Bedingung (1) besagt, daß der Definitionsbereich eines Semiprozesses ein "abstrakter Prozeß an sich" ist. Bedingung (2) garantiert, daß die Stellen und Transitionen des Occurence Netzes N als Vorkommen von entsprechenden Stellen und Transitionen des Netzes N' interpretiert werden können. Analog garantiert die Bedingung (3), daß die Kausalität in dem Occurence Netz N als die Kausalität in dem entsprechenden Netz N' interpretiert werden kann. Die Bedingungen (5) und (6) besagen nur, daß die in dem Netz N erlaubten Markierungen auch in dem Netz N' erlaubt sind, und daß die Schaltbedingungen in dem Netz N auch in dem Netz N' gelten. Die letzte Bedingung drückt aus, daß die Transitionen im Occurence Netz N genauso schalten, wie es in dem Netz N' vorgeschrieben wird.

Beispiel 7.1

Um den Begriff des Semiprozesses näher zu erläutern, wollen wir zu einem Netz N' ein geeignetes Occurence Netz N und eine Abbildung $spro{:}X(N) \rightarrow X(N')$ mit den obigen Eigenschaften angeben. Als Beispielnetz N' betrachten wir ein Netz ähnlich wie in dem Beispiel aus dem vorhergehenden Kapitel.

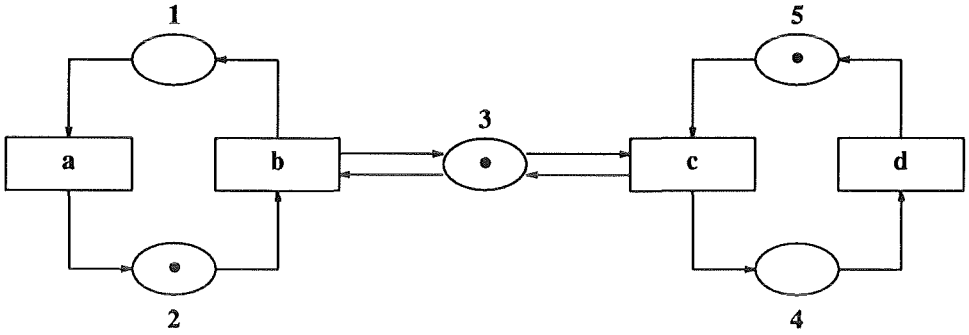


Abb. 7–1 : Ein Netz N'

Wir betrachten nun das folgende Occurence Netz N und geben dann eine geeignete Abbildung *spro* an.

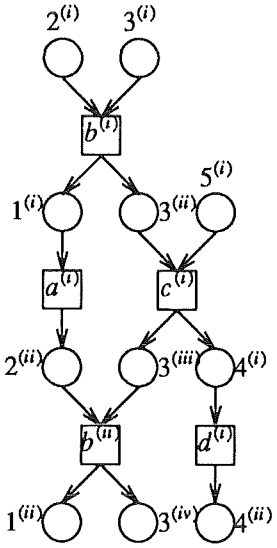


Abb. 7–2 : Ein Occurence Netz N

Die Abbildung *spro* sei nun folgendermaßen definiert:

$$
\text{spro}: \quad
\begin{array}{ll}
2^{(i)} \rightarrow 2 & \\
2^{(ii)} \rightarrow 2 & \\
3^{(i)} \rightarrow 3 & a^{(i)} \rightarrow \text{a} \\
3^{(ii)} \rightarrow 3 & b^{(i)} \rightarrow \text{b} \\
3^{(iii)} \rightarrow 3 & b^{(ii)} \rightarrow \text{b} \\
3^{(iv)} \rightarrow 3 & c^{(i)} \rightarrow \text{c} \\
4^{(i)} \rightarrow 4 & d^{(i)} \rightarrow \text{d} \\
4^{(ii)} \rightarrow 4 & \\
5^{(i)} \rightarrow 5 &
\end{array}
$$

Um zu zeigen, daß *spro* ein Semiprozeß in dem Netz N' ist, muß man die sieben Bedingungen aus Definition 7.2 nachweisen.

(1) N ist ein Occurence Netz, weil N offensichtlich azyklisch ist, und jede Stelle nur höchstens eine Transition als Vorgänger bzw. Nachfolger hat.

(2) Die Abbildung *spro* wurde gerade so gewählt, daß $spro(P(N)) \subseteq P(N')$ und $spro(T(N)) \subseteq T(N')$ gilt.

(3) Durch die Indizierung der Stellen und Transitionen im Netz N wird die Gültigkeit dieser Bedingung sofort klar.

(4) Diese Bedingung wollen wir an einem Beispiel erläutern. Nehmen wir die Transition $c^{(i)}$ des Netzes N. Die Menge ${}^\bullet c^{(i)\bullet}$ ist gerade die Menge $\{3^{(ii)}, 5^{(i)}, 3^{(iii)}, 4^{(ii)}\}$. Zu zeigen ist nun, daß $spro({}^\bullet c^{(i)\bullet}) = {}^\bullet spro(c^{(i)})^\bullet$ ist. $spro({}^\bullet c^{(i)\bullet})$ ist die Menge $\{3,4,5\}$ denn $spro(3^{(ii)})=p(3^{(iii)})=3$, $spro(4^{(ii)})=4$ und $spro(5^{(i)})=5$. Wie man aus dem Netze N' sieht, ist ${}^\bullet spro(c^{(i)})^\bullet = {}^\bullet c^\bullet = \{3,4,5\}$ also gleich $spro({}^\bullet c^{(i)\bullet})$, was wir zeigen wollten.

Die Gültigkeit der Bedingungen (5) bis (7) kann man sehr leicht nachprüfen, worauf wir aber verzichten wollen.

In realen Prozessen von PrT-Netzen dürfen einige Transitionen in manchen Situationen mehrfach schalten. Man kann sagen, daß in diesen Prozessen die Transitionen mit gewissen Mehrfachheiten (Multiplicity) vorkommen. Um diese Eigenschaften auszudrücken führen wir an Stelle des Occurence Netzes N ein T-Multinetz ein.

Definition 7.3 : (T-Multinetze)

Ein T-Multinetz ist ein Paar der Gestalt $N^{oc} = (N,\mu)$, so daß das Tripel $N=(P,T,F)$ ein Occurence Netz und $\mu : T \rightarrow \mathbb{N}$ eine Multimenge ist.

Definition 7.4 : (Schicht eines T-Multinetzes)

Es sei N^{oc} ein T-Multinetz. Eine Schicht in N^{oc} ist die Beschränkung der Multimenge $X(N^{oc})$ auf eine maximale Antikette der Ordnung $(X(N^{oc}), F(N^{oc})^*)$, d.h. eine Multimenge $(X_0, \mu_0) \subseteq (X(N^{oc}), \mu)$ mit der Eigenschaft, daß X_0 eine maximale Antikette in der Ordnung $(X(N^{oc}), F(N^{oc})^*)$ ist und für $x \in X_o$ die Gleichung $\mu_o(x) = \mu(x)$ gilt.

Beispiel 7.2

Wir wollen in diesem Beispiel die Begriffe T-Multinetz und Schicht verdeutlichen. Ein Beispiel eines T-Multinetzes N^{oc} ist das folgende:

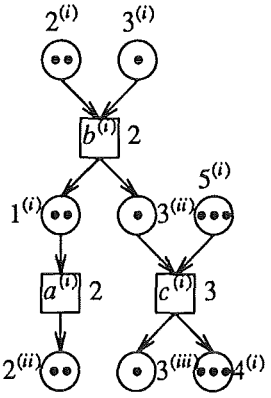


Abb. 7–3 : Ein T-Multinetz N^{oc}

Die Mehrfachheiten der Transitionen sind dabei $\mu(a^{(i)})=2$, $\mu(b^{(i)})=2$ und $\mu(c^{(i)})=3$.
Für folgende Mengen X_0 ist die Multimenge (X_0, μ_0) eine Schicht:

$X_0=\{1^{(i)}, c^{(i)}\}$, $X_0=\{1^{(i)}, 3^{(ii)}, 5^{(i)}\}$, $X_0=\{a^{(i)}, c^{(i)}\}$ und $X_0=\{2^{(ii)}, 3^{(ii)}, 5^{(i)}\}$

(Natürlich gibt es noch andere Schichten in dem T-Multinetz N^{oc}.)

Eine Schicht (X_0, μ_0) nennt man *T-Schicht*, wenn $X_0 \subseteq T(N^{oc})$ und *P-Schicht*, wenn $X_0 \subseteq P(N^{oc})$ ist.

Beispiel 7.3

In dem Beispiel 7.2 ist die Schicht mit $X_0=\{a^{(i)}, c^{(i)}\}$ eine T- Schicht und die Schichten mit $X_0=\{1^{(i)}, 3^{(ii)}, 5^{(i)}\}$ und $X_0=\{2^{(ii)}, 3^{(ii)}, 5^{(i)}\}$ sind P-Schichten.

Für jede Schicht (X_0, μ) definieren wir:

$$Pre(X_0, \mu) = \bigcup_{t \in X_0 \cap T(N^{oc})} {}^\bullet t \cup (X_0 \cap P(N^{oc}))$$

$$Post(X_0, \mu) = \bigcup_{t \in X_0 \cap T(N^{oc})} t^\bullet \cup (X_0 \cap P(N^{oc})).$$

Es ist leicht zu sehen, daß für jede Schicht (X_0, μ) $Pre(X_0, \mu)$ und $Post(X_0, \mu)$ aus Stellen des Netzes N besteht.

Beispiel 7.4

Wir geben nun für verschiedene Schichten die Mengen *Pre* und *Post* an. Der Einfachheit halber benutzen wir dazu eine Tabelle.

(X_0,μ)	$Pre(X_0, \mu)$	$Post(X_0, \mu)$
$\{a^{(ii)}, 3^{(ii)}, 5^{(i)}\}$	$\{1^{(i)}, 3^{(ii)}, 5^{(i)}\}$	$\{2^{(ii)}, 3^{(ii)}, 5^{(i)}\}$
$\{1^{(ii)}, 3^{(ii)}, 5^{(i)}\}$	$\{1^{(ii)}, 3^{(ii)}, 5^{(i)}\}$	$\{1^{(ii)}, 3^{(ii)}, 5^{(i)}\}$
$\{a^{(i)}, c^{(i)}\}$	$\{1^{(i)}, 3^{(ii)}, 5^{(i)}\}$	$\{2^{(ii)}, 3^{(iii)}, 4^{(i)}\}$

Definition 7.5 : (markiertes T-Multinetz)

Es sei N^{oc} ein T-Multinetz, $SIG = (S,\Omega,\Pi,typ_\Omega,typ_\Pi)$ eine Signatur, $V_S : S \rightarrow$ **Set** eine Familie von Variablen und $M : P(N) \rightarrow FSM(SIG,V_S)$ eine Markierung von N^{oc}. Dann nennt man das Paar (N^{oc},M) ein *markiertes T-Multinetz*.

Definition 7.6 : (Prozesse)

Es sei (N^{oc},M) ein markiertes T-Multinetz. Ein Prozeß in einem PrT-Netz $N' = (P',T',F')$ ist eine Abbildung $pro:X(N) \rightarrow X(N')$, die die folgenden Bedingungen erfüllt:

(i) $pro:X(N) \rightarrow X(N')$ ist ein Semiprozeß in dem Netz N'.

(ii) für jede T-Schicht (X_0, μ) ist das Tripel $(M|_{Pre(X_0, \mu)}, (X_0, \mu), M|_{Post(X_0, \mu)})$ ein Schritt im Netz N^{oc}.

Im Folgenden werden wir Prozesse auch in der Gestalt $pro:N^{oc} \rightarrow N'$ schreiben.

Beispiel 7.5

Zur Verdeutlichung des Prozeßbegriffes benutzen wir das Netz N' des Beispieles 7.1, allerdings mit anderer Markierung.

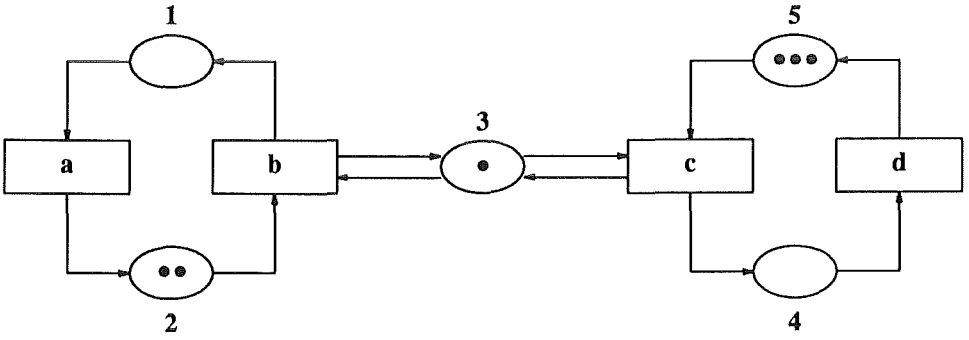


Abb. 7–4 : Das Netz N'

Das T-Multinetz das wir betrachten wollen übernehmen wir aus Beispiel 7.2.

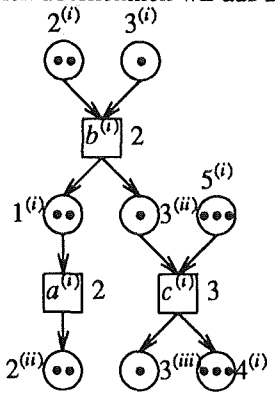


Abb. 7–5 : Ein T-Multinetz N^{oc}

Die Abbildung *pro* definieren wir wie folgt:

$$
\text{p}: \quad
\begin{array}{lcl}
1^{(i)} & \rightarrow & 2 \\
2^{(i)} & \rightarrow & 2 \\
2^{(ii)} & \rightarrow & 2 \\
3^{(i)} & \rightarrow & 3 \\
3^{(ii)} & \rightarrow & 3 \\
3^{(iii)} & \rightarrow & 3 \\
4^{(i)} & \rightarrow & 4 \\
5^{(i)} & \rightarrow & 5
\end{array}
\qquad\qquad
\begin{array}{lcl}
a^{(i)} & \rightarrow & \text{a} \\
b^{(i)} & \rightarrow & \text{b} \\
c^{(i)} & \rightarrow & \text{c}
\end{array}
$$

Man kann durch Nachprüfen der einzelnen Bedingungen leicht sehen, daß *pro* ein Prozeß ist.

Die Menge aller Prozesse im Sinne der Definition 7.4 wird mit *BEH(N')* bezeichnet. Ihre Elemente sind alle *potentiell möglichen* Prozesse, die in dem Netz laufen können. Um die Klasse der Prozesse, die in einem *konkretem System* wirklich laufen dürfen, zu beschreiben, muß man die Klasse der in dem System möglichen (erlaubten) Situationen definieren. Diese werden im Netz durch die Klasse aller seiner Markierungen charakterisiert. Diese Klasse kann auf verschiedene Art und Weise beschrieben werden. Man kann sie z.B. explizit als eine Menge der Markierungen deren Elemente Stück für Stück aufgezählt werden definieren, man kann sie mit einer Menge von Formeln beschreiben oder durch eine induktive Definition geben. Wenn eine solche Menge von in dem System erlaubten Markierungen gegeben ist, dann kann man die Klasse der in *dem* System real möglichen Prozesse als die Klasse aller Prozesse deren "Markierungen dieser gegebenen Klasse der erlaubten Situationen angehören" bezeichnen.

Beispiel 7.6

Wir betrachten den Prozeß *pro* aus dem Beispiel 7.5. In dem Netz *N'* seien die erlaubten Markierungen gekennzeichnet durch die Bedingung, daß auf allen Stellen höchstens drei Token liegen dürfen. Unter dieser Bedingung sind alle P-Schichten des Prozesses erlaubte Markierungen und damit der Prozeß *pro* real möglich bzw. korrekt.

Wir erklären den Begriff des korrekten Prozesses ein wenig genauer. Es sei $pro : N^{oc} \rightarrow N'$ ein Prozeß im Netz *N'*. Das Bild *pro(A)* von jeder P-Schicht $A \subseteq P(N)$ des Prozesses *pro* (des Netzes *N'*) ist eine Teilmenge der Menge *P(N')* aller Stellen des Netzes *N'*. Wir wollen sie als die Menge auf der eine Situation, d.h. eine Markierung im Netz *N*, definiert ist betrachten. Die obige Bedingung wollen wir in einer Definition genauer formulieren. Dazu sei *N'* ein Prt-Netz und M_N eine Klasse von Markierungen des Netzes *N'* (die Klasse aller im Netz *N'* erlaubter Situationen).

Definition 7.7 : (Korrektheit von Prozesse)

Sei N^{oc} ein T-Multinetz, N' ein PrT-Netz, $SIG = (S,\Omega,\Pi,typ_\Omega,typ_\Pi)$ eine Signatur und $V_S : S \rightarrow$**Set** eine Familie von Variablen. Ein Prozeß $pro{:}N^{oc} \rightarrow N'$ ist *im Sinne* der Markierungsklasse M_N korrekt dann und nur dann wenn für jede P-Schicht $P_0 \subseteq P(N)$ die Markierung $M_{P_0} : P(N') \rightarrow FSM(SIG,V_S)$ mit $\forall_{p \in P_0}\ M_{P_0}(pro(p)) = M(p)$ der Klasse M_N angehört.

Beispiel 7.7

Ein Beispiel für einen korrekten Prozeß haben wir schon im Beispiel 7.6 gesehen. Nun wollen wir ein Beispiel für einen nicht korrekten Prozeß geben. Wir benutzen dazu wieder den Prozeß aus Beispiel 7.5 und definieren die erlaubten Markierungen des Netzes N' dadurch, daß wir alle Markierungen erlauben bei denen nicht mehr als zwei Token auf der Stelle 4 liegen. Unter dieser Markierungsklasse ist der Prozeß nicht mehr korrekt, da die P-Schicht $X_0=\{2^{(ii)}, 3^{(iii)}, 4^{(i)}\}$ auf keine erlaubte Markierung abgebildet werden kann.

Die Klasse M_N wird oft als die Klasse aller Markierungen die von einer gegebener Markierung erreichbar sind definiert. In dem Fall kann man die Definition wesentlich vereinfachen.

7.1 Algebra der Prozesse

In der Menge *BEH(N)* aller Prozesse eines Systems kann man auf natürliche Weise Operationen definieren, die eine algebraische Struktur in dieser Menge bestimmen.

Definition 7.8

Es seien $pro{:}N^{oc} \rightarrow N'$ ein Prozeß in dem PrT-Netz $N' = (P',T',F')$ und $N^{oc}_{\min}$ bzw. $N^{oc}_{\max}$ die Mengen der minimalen bzw. maximalen Elemente der Halbordnung $(X(N^{oc}), F(N^{oc})^*)$. Wir definieren

$$begin(pro) : (N^{oc}_{\min},\emptyset,\emptyset,\emptyset,M\mid_{N^{oc}_{\min}}) \rightarrow N' \text{ und } end(pro) : (N^{oc}_{\max},\emptyset,\emptyset,\emptyset,M\mid_{N^{oc}_{\max}}) \rightarrow N',$$

so daß für $s \in N^{oc}_{\min}$ $begin(pro)(s) = pro(s)$, und für $s \in N^{oc}_{\max}$ $end(pro)(s) = pro(s)$.

In der Definition 7.8 stellen die Tupel $(N^{oc}_{\min},\emptyset,\emptyset,\emptyset,M\mid_{N^{oc}_{\min}})$ und $(N^{oc}_{\max},\emptyset,\emptyset,\emptyset,M\mid_{N^{oc}_{\max}})$ markierte T-Multinetze dar, bei denen die Mengen T,F und die Multimenge μ leer sind.

Beispiel 7.8

In dem T-Multinetz aus Beispiel 7.5 wollen wir die Prozesse *begin(pro)* und *end(pro)* verdeutlichen.

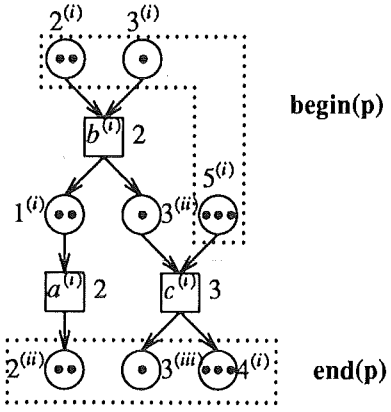


Abb. 7.1–1 : *begin(pro)* und *end(pro)* eines T-Multinetzes

So ordnet die Operation *begin* jedem Prozeß *pro* seine Beschränkung auf die minimale *Maximalkette*, d.h. auf den minimalen Schnitt des Prozesses *pro* und die Operation *end* seine Beschränkung auf den maximalen Schnitt zu. Jetzt können wir eine neue Operation auf Prozesse definieren. Wenn das "Ende" eines Prozesses p identisch mit dem "Anfang" eines anderen Prozesses *pro'* ist dann kann man einen neuen Prozeß bilden, der aus dem Prozeß *pro* gefolgt von dem Prozeß *pro'* gebildet wird. Die formale Definition ist in dem Fall folgende.

Definition 7.9 : (Sequentielle Zusammensetzung von Prozessen)

Es seien $(N_1^{oc},M_1) = (P_1, T_1, F_1, \mu_1, M_1)$ und $(N_2^{oc},M_2) = (P_2, T_2, F_2, \mu_2, M_2)$ zwei markierte T-Multinetze. Weiterhin seien $pro_1{:}N_1^{oc} \rightarrow N'$ und $pro_2{:}N_2^{oc} \rightarrow N'$ Prozesse in einem PrT-Netz $N' = (P',T',F')$. Dann ist

$$pro_1 pro_2{:}N_{12}^{oc} = (N_{12}, M_{12}) = (P_{12}, T_{12}, F_{12}, \mu_{12}, M_{12}) \rightarrow N'$$

ein Prozeß in N' der folgendermaßen definiert ist:

a) wenn $X(N_1) \cap X(N_2) = min(N_2) = max(N_1) \wedge \mu_1|_{N_1 \cap N_2} = \mu_2|_{N_1 \cap N_2}$ dann ist:

$P_{12} = (P_1 - \min(N_1)) + S_2$
$T_{12} = T_1 + T_2$
$F_{12} = (F_1 - F_1|_{T_{max} \times P_{max}}) \cup F_2 \cup \bar{F}_{12}$ wobei $\bar{F}_{12} = \{((t_1,1),(p_2,2)) \mid (t_1,p_2) \in F_1\}$
$\mu_{12} = \mu_1 + \mu_2$
$M_{12} = (M_1 - M_1|_{N_{1^{min}}}) + M_2$

b) sonst ist $pro_1 pro_2$ nicht definiert.

Diese Operation klebt den Prozeß pro_1 mit dem Prozeß pro_2 zusammen, indem sie die maximalen Elemente der Ordnung $(X(N_1),F_1^*)$ mit minimalen Elementen der Ordnung $(X(N_2),F_2^*)$ identifiziert. Diese Operation wird die sequentielle Zusammensetzung von Prozesse genannt und mit dem Symbol "•", das oft weggelassen wird, bezeichnet.

Beispiel 7.9

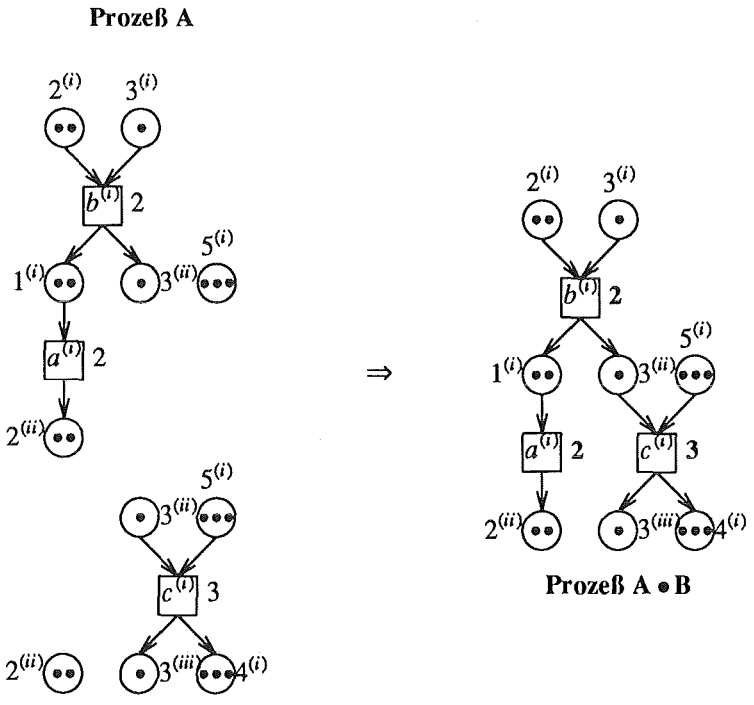


Abb. 7.1–2 : Das Zusammenkleben zweier Prozesse

Definition 7.10 : (Parallele Zusammensetzung von Prozessen)

Es seien $(N_1^{oc},M_1)=(P_1,T_1,F_1,\mu_1,M_1)$ und $(N_2^{oc},M_2)=(P_2,T_2,F_2,\mu_2,M_2)$ zwei markierte T-Multinetze. Weiterhin seien $pro_1 : N_1^{oc} \rightarrow N'$ und $pro_2 : N_2^{oc} \rightarrow N'$ Prozesse in einem PrT-Netz $N'=(P',T',F')$. Die parallele Zusammensetzung der Prozesse pro_1 und pro_2 ist

(a) die mengentheoretische Vereinigung $pro_1 \cup pro_2 : N_1^{oc} \cup N_2^{oc} \rightarrow N$, wenn folgendes gilt:

(i) $X(N_1^{oc}) \cap X(N_2^{oc}) = \emptyset$, und

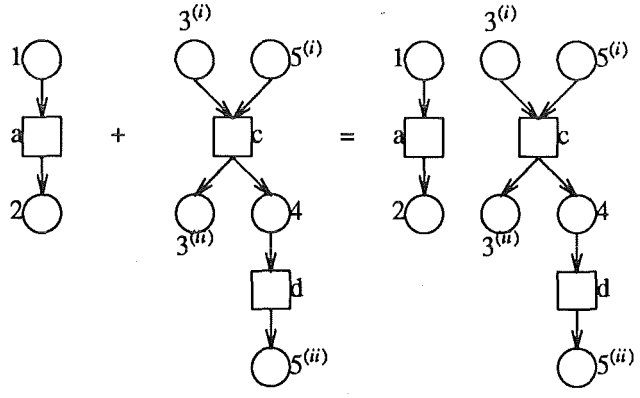
(ii) $pro_1(X(N_1^{oc})) \cap pro_2(X(N_2^{oc})) = \emptyset$.

(b) nicht definiert sonst.

Die parallele Zusammensetzung der Processe pro_1 und pro_2 wird mit dem Symbol $pro_1 + pro_2$ bezeichnet. Das folgende Beispiel verdeutlicht diese parallele Zusammensetzung.

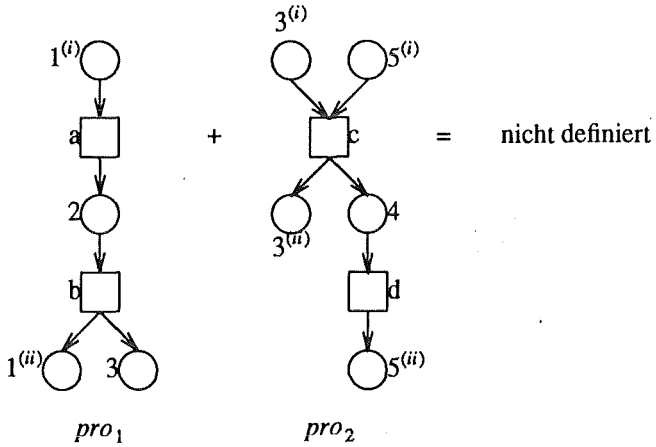
Beispiel 7.10

Betrachten wir wieder das Netz aus dem Beispiel 7.1. In diesem System sei die Klasse der erlaubten Markierungen die Klasse aller Markierungen die von der Markierung $M_0(1) = M_0(3) = M_0(5) = \bullet$ erreichbar sind. In diesem System ist dann



pro_1 pro_2 $pro_1 + pro_2$

eine korrekte parallele Zusammensetzung von zwei Prozessen pro_1 und pro_2 zu $pro_1 + pro_2$, während

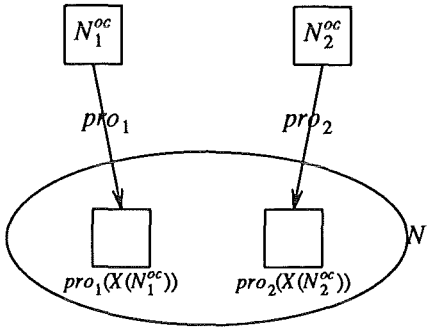


pro_1 pro_2

zwei Prozesse zeigt, die man nicht Zusammensetzen kann. Bei diesem Beispiel ist in Prozeß pro_2 das Bild der Stellen $3^{(i)}$ und $3^{(ii)}$ identisch mit dem Bild der Stelle *3* im Prozeß pro_1, was die Bedingung (ii) der Definition 7.10 verletzt.

Wir illustrieren die obigen Operationen graphisch wie folgt. Die Definitionsbereiche der Prozesse (die Occurrence Netze sind) stellen wir als Rechtecke dar. Die Richtung der Kausalität ist von oben nach unten.

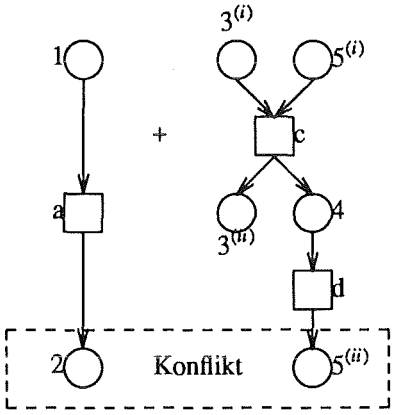
Die oberen bzw. unteren Seiten der Rechtecke stehen für die Operationen *begin* und *end*. Die Prozesse selbst werden durch Pfeile dargestellt.



Die (partielle) Algebra *(BEH(N), begin , end ,* ∘ *, +)* hat eine unangenehme Eigenschaft. Es kann sein, daß eine (sequentielle oder paralle) Zusammensetzung von zwei, im Sinne einer Klasse von Markierungen eines Netzes N, korrekte Prozesse nicht mehr korrekt sind im Sinne dieser Klasse.

Beispiel 7.11

Betrachten wir wieder das System aus dem vorhergehenden Beispiel. Setzen wir voraus, daß wir die Menge aller möglichen Markierungen durch die Bedingung, entweder die Stelle 2 oder die Stelle 5 ist in einer Markierung markiert, beschränken. In diesem Fall ist die parallel Zusammensetzung der Prozesse pro_1 und pro_2 nicht mehr möglich, da der Schnitt $\{2, 5^{(ii)}\}$ auf keine erlaubte Markierung abgebildet werden kann.



Wir wollen dieses Problem hier nicht betrachten. Entscheidend sind dabei die Eigenschaften der Klasse $\mathcal{M}_N$ der im Netz N erlaubten Markierungen. Einige Lösungen des Problems für spezielle Klassen von Petri Netzen kann man in [Kor80a, Kor88a] oder [Win82a] finden. Wenn man (was stets der Fall ist) mit Prozessen arbeiten will, die im Sinne einer bestimmten Klasse von Markierungen korrekt sind, dann muß man zum Teil (a) der obigen Definitionen noch die folgende Bedingung hinzufügen.

(iii) Jede Schicht des Prozesses $pro_1 pro_2$ bzw. $pro_1 + pro_2$ gehört der Klasse $\mathcal{M}_N$ aller Markierungen eines Netzes N an.

Die Klasse aller in einem PrT-Netz möglichen Prozesse wird das *Verhalten* des Netzes N genannt und mit dem Symbol *BEH(N)* bezeichnet.

Die Untersuchung von den Eigenschaften der Klasse aller in einem PrT-Netz möglichen Prozesse ist eine der wichtigsten Aufgaben der Theorie von Petri Netzen. Sie ist auch von großer praktischer Bedeutung. Wir beschreiben hier einige von vielen existierenden Möglichkeiten dieses Problem anzugehen.

a) Spurentheorie

Diese Auffassung basiert auf dem Begriff der binären Unabhängigkeitsrelation von Ereignissen (Events) in einem nebenläufigen System. Die grundlegende Ideologie dieser Auffassung ist die Folgende. Betrachten wir einmal einen Beobachter eines von einem PrT-Netz modellierten Systems. Nehmen wir an, der Beobachter muß alle Vorkommen der Transitionen im System (d.h. die einzig beobachtbare Änderung des Systems) registrieren.
Wie diese Registrierung bestimmt ist, ist hier irrelevant. Man kann sich z.B. vorstellen, daß jede Transition nach jedem Vorkommen eine Nachricht zu dem Beobachter sendet, oder daß dieser jedes Vorkommen einer Transition als eine Art von "Blitz" sieht. Der Beobachter registriert diese Vorkommen sequentiell. Als Resultat erhält er eine Folge $£_1 = t_1, t_2, \cdots, t_n$ von Vorkommen von Transitionen, die sein Bild des Verhaltens des Systems wiederspiegelt. Diese Folge ist nichts anderes als eine lineare Wohlordnung. Nun nehmen wir an es gibt mehrere solche Beobachter $£_1, £_2, \cdots, £_m, \cdots$ Jeder von ihnen hat sein eigenes Bild, seine eigene Registrierung von denselben Änderungen desselben Prozessen in dem betrachteten Netz. Nehmen wir an wir hätten folgende Folgen:

$$\begin{aligned} £_1 &= t_{11}, t_{12}, \cdots, t_{1n} \\ £_2 &= t_{21}, t_{22}, \cdots, t_{2n} \\ &\vdots \\ £_m &= t_{m1}, t_{m2}, \cdots, t_{mn} \\ &\vdots \end{aligned}$$

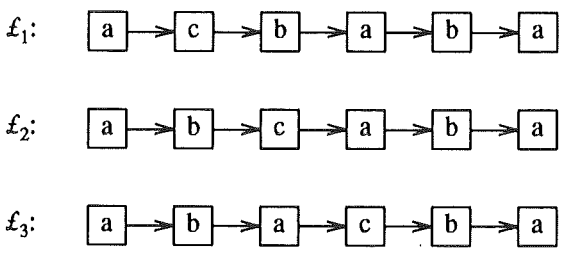
Nun haben wir ein Problem. Welche von diesen Beobachtungen sind "wahr", d.h. welche beschreibt die richtige Anordnung der Vorkommen der Transitionen, also die richtigen Ereignisse im System. Diese Registrierungen können sich stark voneinander unterscheiden, es kann z.B. sein, daß der Beobachter $£_i$ das j-te Vorkommen von der Transition t_0 vor dem k-ten Vorkommen der Transition t_1 registriert hat, aber ein anderer Beobachter $£_j$ genau die umgekehrte Reihenfolge, d.h. t_1 vor t_0 beobachtet hat. Haben beide Beobachter dieselbe Glaubwürdigkeit, so gibt es keinen Grund nur eine dieser Beobachtungen als wahr und die andere als falsch anzunehmen. In diesem Fall nehmen wir an, daß diese Vorkommen von Transitionen kausal unabhängig sind. Man kann hier nicht feststellen, welche von ihnen die Ursache und welche das Resultat ist. (Wenn alle Beobachter das i-te Vorkommen von t_0 vor dem j-ten Vorkommen von t_1 registriert haben, dann scheint es vernünftig zu glauben, daß dieses i-te Vorkommen von t_0 eine Ursache des j-ten Vorkommens von t_1 ist, weil man kein "Gegenbeispiel" finden konnte.)

In diesem Fall zeigt sich jede lineare Anordnung von beobachteten Vorkommen von Transitionen als sinnlos. Man kann sich aber auf eine gemeinsame Beobachtung einigen: für je zwei Vorkommen von Transitionen t_0, t_1 kommt t_0 **vor** t_1 genau dann wenn von allen Beobachtern $£_1, £_2, \cdots, £_m, \cdots$ genau dieselbe Reihenfolge festgestellt wurde, sonst sind sie *unvergleichbar*. Die obige Regel wurde mathematisch von E. Marczewski als ein Satz über die Erweiterung von Halbordnungen zur linearen Ordnung formuliert. Wir erhalten für die Beschreibung des Verhaltens eines Systems die Halbordnung:

(1) $\gamma_{£_1, £_2, \ldots £_m, \ldots} = F_{£_1} \cap .. \cap F_{£_m} \cap ..$ mit $F_{£_i} = t_{i1}, t_{i2}, \cdots, t_{in}$ ..

Beispiel 7.12

Wir wollen in unserem Beispiel von drei Betrachtern $£_1, £_2, £_3$ und Transitionen a,b und c ausgehen. Die drei Beobachter beobachten folgende Reihenfolge des Schaltens der Transitionen a,b und c:



Die aus diesen Betrachtungen resultierenden Mengen $F_{£_i}$ von Relationen sind:

$$F_{£_1} = \{(a_1,c_1), (a_1,b_1), (a_1,a_2), (a_1,b_2), (a_1,a_3), (\mathbf{c}_1,\mathbf{b}_1),$$

$$(\mathbf{c}_1,\mathbf{a}_2), (\mathrm{c}_1,\mathrm{b}_2), (\mathrm{c}_1,\mathrm{a}_3), (\mathrm{b}_1,\mathrm{a}_2), (\mathrm{b}_1,\mathrm{b}_2), (\mathrm{b}_1,\mathrm{a}_3), (\mathrm{a}_2,\mathrm{b}_2), (\mathrm{a}_2,\mathrm{a}_3), (\mathrm{b}_2,\mathrm{a}_3)\}$$

$$F_{£_2} = \{(a_1,b_1), (a_1,c_1), (a_1,a_2), (a_1,b_2), (a_1,a_3), (\mathbf{b}_1,\mathbf{c}_1),$$

$$(b_1,a_2), (b_1,b_2), (b_1,a_3), (\mathbf{c}_1,\mathbf{a}_2), (\mathrm{c}_1,\mathrm{b}_2), (\mathrm{c}_1,\mathrm{a}_3), (\mathrm{a}_2,\mathrm{b}_2), (\mathrm{a}_2,\mathrm{a}_3), (\mathrm{b}_2,\mathrm{a}_3)\}$$

$$F_{£_3} = \{(a_1,b_1), (a_1,a_2), (a_1,c_1), (a_1,b_2), (a_1,a_3), (b_1,a_2),$$

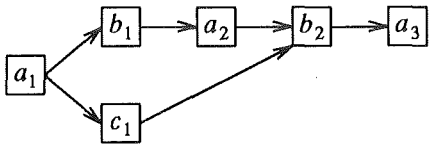
$$(\mathbf{b}_1,\mathbf{c}_1), (\mathrm{b}_1,\mathrm{b}_2), (\mathrm{b}_1,\mathrm{a}_3), (\mathbf{a}_2,\mathbf{c}_1), (\mathrm{a}_2,\mathrm{b}_2), (\mathrm{a}_2,\mathrm{a}_3), (\mathrm{c}_1,\mathrm{b}_2), (\mathrm{c}_1,\mathrm{a}_3), (\mathrm{b}_2,\mathrm{a}_3)\}$$

Damit haben wir die Halbordnung

$$\gamma_{£_1,£_2,£_3} = F_{£_1} \cap F_{£_2} \cap F_{£_3} = \{(a_1,b_1), (a_1,a_2), (a_1,c_1), (a_1,b_2),$$

$$(a_1,a_3), (b_1,a_2), (b_1,b_2), (b_1,a_3), (a_2,b_2), (a_2,a_3), (c_1,b_2), (c_1,a_3), (b_2,a_3)\}$$

Zur besseren Veranschaulichung wurden alle Paare, die nicht in jeder Menge $F_{£_i}$ liegen, fett gedruckt. Weiterhin bedeutet z.B. a_3 das dritte Auftreten der Transition a. Die graphische Darstellung dieser Halbordnung zeigt folgendes Bild.



Die obige Halbordnung (1) definiert auf natürliche Weise eine Unabhängigkeitsrelation in der Menge aller Transitionen die in dieser Halbordnung auftreten. Transitionen t_0 und t_1 sind in der Halbordnung $\gamma_{£_1,..£_m,..}$ unabhängig, genau dann wenn alle Vorkommen der Transition t_0 von allen Vorkommen der Transition t_1 unabhängig sind.

Die Halbordnung (1) kann man als eine Beschreibung eines im System laufenden Prozesses betrachten. Nun nehmen wir an, daß die Beobachter $£_1, £_2, \cdots, £_m, \cdots$ **alle** Prozesse im System beobachtet haben. Wir haben also eine Menge

(2) $BEOB(N) = (H_i)_{i \in J}$

von Halbordnungen der Vorkommen von Transitionen, die als Bilder der im System laufenden Prozesse aufgefaßt werden können.

Nun sind zwei verschiedene Situationen möglich:

1) je zwei Transitionen t_0, t_1 sind entweder in jeweils zwei Prozessen pro_i, pro_j (i,j ∈ J) unabhängig oder abhängig.

2) es gibt Transitionen t_0, t_1 und Prozesse pro_i, pro_j ($i,j \in J$), so daß in dem Prozeß pro_i die Transitionen t_0, t_1 unabhängig und im Prozeß pro_j abhängig sind.

Welche Situation eintritt, hängt von der Menge der Beobachter und dem beobachteten System ab. Liegt die erste Möglichkeit vor, kann man über eine Unabhängigkeitsrelation in der Menge der Transitionen des Systems sprechen. Zwei Transitionen sind unabhängig, wenn es ein Prozeß des Systems gibt, in dem sie unabhängig sind. Diese Relation ist natürlich symmetrisch und irreflexiv. Sie braucht aber nicht transitiv zu sein. Die grundlegende Behauptung der Spurentheorie ist, daß diese Unabhängigkeitsrelation eine Basis für die Berechnung von der Menge der in dem System möglichen Prozesse ist. Die Denkweise ist hier die Folgende. Es sei A eine Menge (Alphabet) und $I_0 \subseteq A \times A$ sei eine irreflexive und symmetrische Relation in A. In der Menge A^* aller Wörter über dem Alphabet A definieren wir die Relation $I \subseteq A^* \times A^*$ folgendermaßen: für die Wörter $w,w' \in A^*$ ist

$$I(w,w') \Leftrightarrow \exists_{w_0,w_1 \in A^*} \exists_{a,b \in A} (a,b) \in I_0 \wedge w = w_0 a b w_1 \wedge w' = w_0 b a w_1$$

d.h. die Wörter w und w' sind in dieser Relation genau dann wenn der einzige Unterschied zwischen ihnen die Reihenfolge von zwei benachbarten Symbolen ist.

Beispiel 7.13

Die Alphabet A habe die Gestalt A=(a,b,c,d,e). Die Relation $I_0 \subseteq A \times A$ habe die folgende Gestalt:

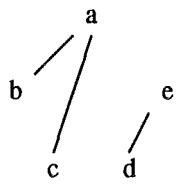


Abb. 7.1–3 : Beispiel für eine Relation I_0

Die folgenden Wörter w und w' sind in der oben definierten Relation I:

w	w'
abcdade	bacdade
adcedbe	adcdebe
bac	bca

während die folgenden Wörter w und w' nicht in dieser Relation I sind:

w	w'
adcdabe	acddabe
cdbecda	cdbceda
abd	adb

Zu dem Wort w=abdade wollen wir die Menge der ähnlichen Elemente $[w]$ angeben. Diese Menge hat dann die Gestalt $[w]$={abdade, badade, abdaed}.

Nun sei $\tilde{}_I$ die kleinste Äquivalenzrelation in der Menge A, die die Relation I enthält. Die Äquivalenzklasse $[w]_I$ des Wortes w für diese Relation ist die Menge aller Wörter die man aus dem Wort w durch Transposition von zwei benachbarten, unabhängigen Elementen erhalten kann. Jedes Element dieser Klasse enthält also nur Permutationen von denselben Elementen und kann als entsprechende Halbordnung dargestellt werden. Diese Halbordnung ist gerade der Durchschnitt aller ihrer elementaren, linearen Ordnungen. Solche Klassen nennt man Spuren und ihre Elemente heißen sequentielle Repräsentanten von diesen Spuren. Falls aus dem Kontext hervorgeht, welche Relation man meint, schreiben wir an Stelle von $[w]_I$ nur $[w]$. Die Relation $\tilde{}_I$ hat eine sehr wichtige Eigenschaft. Sie bleibt unter der Operation der Konkatenation erhalten, d.h. für beliebige Wörter w_1, w_1', w_2, $w_2' \in A^*$ ist $w_1 \tilde{}_I w_1' \wedge w_2 \tilde{}_I w_2' \Rightarrow w_1 w_2 \tilde{}_I w_1' w_2'$

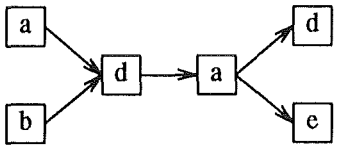
Diese Eigenschaft erlaubt es wie normal eine Quotientoperation $\circ : A^*|_{\tilde{}_I} \times A^*|_{\tilde{}_I} \to A^*|_{\tilde{}_I}$ durch die Zuordnung $[w] \circ [w'] = [ww']$ zu definieren.

Beispiel 7.14

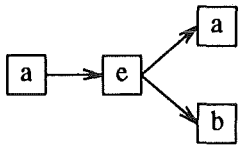
Wenn wir noch einmal obiges Beispiel betrachten, so war die Ähnlichkeitsklasse von w=abdade die Menge $[w]$={abdade, badade, abdaed}. Die Äquivalenzklasse von w ist aber die Menge $[w]_I$ = {abdade, badade, abdaed, badaed}, da folgende Elemente in Relation zueinander stehen:

$$\text{abdade} \overset{I}{\text{———}} \text{badade} \overset{I}{\text{———}} \text{abdaed}$$

Wegen der Transitivität der Relation $\tilde{}_I$ ist damit auch das Wort badaed in der Äquivalenzklasse $[w]_I$.
Die graphische Darstellung dieser Klasse, d.h. die Spur hat die folgende Gestalt.



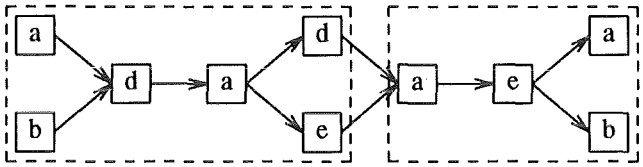
Betrachten wir nun ein anderes Wort w'=aeab. Die zugehörige Äquivalenzklasse ist die Menge $[w']_I$ = {aeab, aeba} mit der Spur



Wendet man auf diese beiden Äquivalenzklassen den Quotientoperator "o" an, so erhält man das Folgende:

$$[w] \circ [w'] = [ww'] = [abdadeaeab] = \{abdadeaeab, abdadeaeba,$$
$$badadeaeab, badadeaeba, abdaedaeab, abdaedaeba, badaedaeab, badaedaeba\}$$

Die dazugehörige Spur zeigt das nächste Bild.



Diese Operation kann man standardmaßig auf die Untermengen der Menge $A^* |_{I}$ fortsetzen, indem man definiert

$$L \circ L' = \{ [w] \circ [w'] \mid [w] \in L \wedge [w'] \in L' \}$$

Es zeigt sich weiter, daß die so definierte Operation stetig ist, d.h. daß für jede Sprache L und jede Familie $(L_i)_{i \in M}$ von Sprachen, die folgende Gleichung gilt: $L \circ (\cup_{i \in I} L_i) = \cup_{i \in I} (L \circ L_i)$ Die Stetigkeit der Operation "o" erlaubt die Verwendung der sogenannten "Fixpunktmethoden", die manchmal eine Sprache als die Lösung von einer solchen Gleichung, bzw. einem solchen Gleichungssystem, darstellen dürfen.

Die oben beschriebene Auffassung ist sehr streng auf die sprachtheoretischen Methoden eingestellt. Spuren (Sprachen) werden als (Klassen von) Äquivalenzklassen von Wörtern definiert und entsprechende Operationen als Quotient-Operationen in entsprechenden Algebren. Um diese Methode für Petri-Netze anwenden zu können, muß man ein Verfahren finden, das für jedes Netz genau eine Unabhängigkeitsrelation erlaubt und mit dem man eine Sprache finden kann, die man durch diese Relation "dividiert", um eine entsprechende Spurensprache zu definieren. Die Unabhängigkeitsrelation I wird in dem Fall in der Menge aller Transitionen des Netzes N durch die Bedingung $I_N(t_1, t_2) \Leftrightarrow {}^\bullet t_1^\bullet \cap {}^\bullet t_2^\bullet = \emptyset$ definiert.

Beispiel 7.15

In dem Beispiel 7.1 sind die Transitionen a und c, b und d sowie a und d unabhängig.

Zwei Transitionen t_1, t_2 sind unabhängig, wenn sie keine "gemeinsame Stelle" haben.

Definition 7.11 : (Verhalten eines Netzes)

Es sei (N, A_N, M_0) ein PrT-System. Die (Spuren) Sprache $SS(N) = \{ [w]_{I_N} \mid w$ ist eine Schaltfolge des Netzes $N \}$ wird das Verhalten des Netzes N genannt.

Grob kann man eine Spur $[w] \in SS(N)$ als Resultat des Vergessens über allen Stellen des Definitionsbereiches N^{oc} eines Prozesse $pro: N^{oc} \rightarrow N$ betrachten.

Beispiel 7.16

Die Spur des Prozesses aus Beispiel 7.5 ist die Menge $\{b^{(i)}a^{(i)}c^{(i)}, b^{(i)}c^{(i)}a^{(i)}\}$.

Formaler kann man eine Abbildung $ps: BEH(N') \rightarrow P(N')$ definieren indem man jedem Prozeß $pro: N^{oc} \rightarrow N'$ die Halbordnung $H(pro) = (T(N), f^* |_{T(N)})$ zuordnet. Die Menge aller linearer Erweiterungen der Ordnung $H(pro)$ ist genau die dem Prozeß pro entsprechende Spur.

Es sei jetzt eine Spur $\sigma = [w]$ des PrT-Systems (N,A_N,M_0) gegeben. Sei $(T_0, \leq)$ die der Spur [w] entsprechende Ordnung. Wir zeigen wie man aus dieser Ordnung einen Prozeß des Systems (N,A_N,M_0) rekonstruieren kann. Zuerst "rekonstruieren" wir ein "bloßes" Petri-Netz $N=(P,T,F)$ ohne irgendeine Markierung und danach erzeugen wir eine Markierung des Netzes, die der gegebenen Anfangsmarkierung entspricht.

Um ein Occurence Netz von einer Spur zu konstruieren muß man:

(1) die Mengen von Minimal- und Maximalstellen des gesuchten Netzes finden.

(2) zwischen je zwei aufeinanderfolgende Elemente der Ordnung $(T_0, \leq)$ (d.h. Elemente $x,y \in T_0$ mit der Eigenschaft, daß y ein Nachfolger von x ist) eine Stelle einschreiben.

Das Verfahren verläuft folgendermaßen:

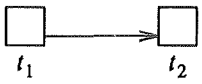
(1) Die Menge der Minimalstellen des gesuchten Occurence Netzes ist genau die Menge

$$BEG(\sigma) = \{ p \in P \mid \exists_{t_0 \in T_0} \exists_{t \in T}\ t_0=(t,1) \wedge p \in {}^\bullet t \wedge t_0 \textit{ ist ein Minimalelement in } (T_0,\leq)\}$$

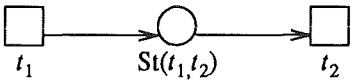
und die Menge der Maximalelemente ist die Menge

$$END(\sigma) = \{ p \in P \mid \exists_{t_0 \in T_0} \exists_{t \in T}\ t_0=(t,n_{\max}) \wedge p \in {}^\bullet t \wedge t_0 \textit{ ist ein Maximalelement in } (T_0,\leq)\}$$

(2) Zwischen beliebigen Elementen $t_1,t_2 \in T$, mit t_2 Nachfolger von t_1, schreiben wir eine Stelle, die mit $St(t_1,t_2)$ bezeichnet wird. Grob kann man sagen, daß der Pfeil



durch



ersetzt wird.

(3) Die Flußrelation definieren wir folgendermaßen:

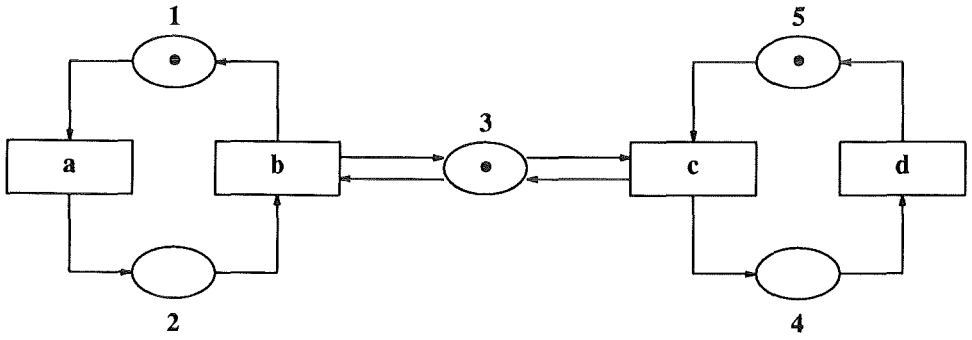
(i) für $p \in BEG(\sigma)$ gilt: $(p,(t,1)) \in F \Leftrightarrow p \in {}^\bullet t$

(ii) für $p \in END(\sigma)$ gilt: $(p,(t,n)) \in F \Leftrightarrow p \in t^\bullet$, wobei $n = \max \{ i \in \mathbb{N} \mid (t,i) \in T_0 \}$

(iii) für $t_1, t_2 \in T$ gilt: $St(t_1, t_2)\ F\ t_2 \wedge t_1\ F\ St(t_1, t_2)$

Beispiel 7.17

Betrachten wir das folgende Netz N'.



Eine zu dem Verhalten des Netzes gehörende Spur ist σ = [acbdacdb]

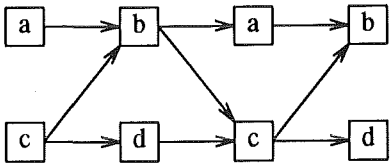


Abb. 7.1–4 : Die Spur σ

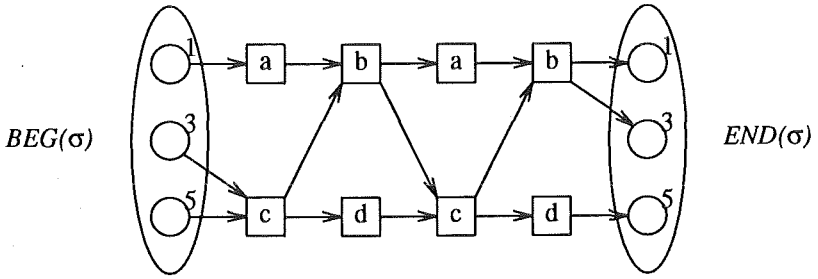
1. Schritt

$BEG(\sigma) = \{ p \in P \mid \exists_{t_0 \in T_0} \exists_{t \in T}\ t_0=(t,1) \wedge p \in {}^\bullet t \wedge t_0$ *ist ein Minimalelement in* $(T_0,\leq)\}$

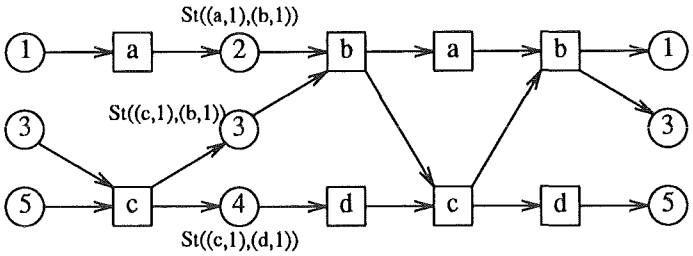
$= {}^\bullet a \cup {}^\bullet c = \{1\} \cup \{3,5\} = \{1,3,5\}$

$END(\sigma) = \{ p \in P \mid \exists_{t_0 \in T_0} \exists_{t \in T}\ t_0=(t,n_{max}) \wedge p \in {}^\bullet t \wedge t_0$ *ist ein Maximalelement in* $(T_0,\leq)\}$

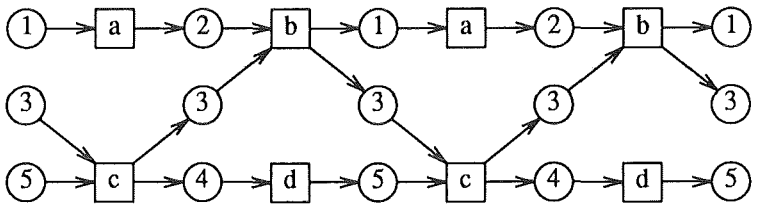
$= b^\bullet \cup d^\bullet = \{1,3\} \cup \{5\} = \{1,3,5\}$



2. Schritt



n. Schritt



Es ist leicht zu sehen, daß ein so erhaltenes Netz N(σ) ein Occurence Netz ist. Wir müssen nun noch eine Markierung M_0 und eine Abbildung *pro* : $X(N(\sigma)) \rightarrow X(N')$ finden, so daß *pro* : $(N(\sigma),M_0) \rightarrow N'$ ein Prozeß ist.

Eine der fundamentalsten Eigenschaften der Petri Netze ist, daß es in jedem Netz $N=(P,T,F)$ *zwischen je zwei aufeinanderfolgenden Transitionen* t_1,t_2 höchstens eine Stelle gibt. Bezeichnen wir diese Stelle durch $St(t_1,t_2)$. Die gesuchte Abbildung *pro* : $X(N(\sigma)) \rightarrow X(N')$ kann man jetzt folgendermaßen definieren :

$$pro(x) = \begin{cases} pr_1(x) & wenn\ x = \bar{t_i} = (t_i, n_i) \\ St(t_1, t_2) & wenn\ x = St(t_1 bar, \bar{t_2})\ mit\ \bar{t_1} = (t_1, n_1),\ \bar{t_2} = (t_2, n_2) \end{cases}$$

Es ist leicht zu sehen, daß sie die Bedingungen (i) und (ii) der Definition 7.6 erfüllt, und daß jede T-Schicht im Netz $N(\sigma)$ ein maximaler Schnitt ist. In der Menge aller dieser Schnitte führen wir eine Ordnung ein, indem wir $\gamma_1 \ll \gamma_2 \Leftrightarrow Pre(\gamma_2) = Post(\gamma_1)$ definieren. Die Ordnung " « " ist eine Halbordnung in der Menge aller Schnitte (T- Schnitte) des Netzes $N(\sigma)$, die ein kleinstes und ein größtes Element hat.
Es sei M_o die Anfangsmarkierung des Netzes $N' = (P', T', F')$. Die Idee der Konstruktion der Markierung $M : P(N(\sigma)) \rightarrow FSM(SIG, V_S)$ ist klar. Die Stellen des Netzes $N(\sigma)$ sind Urbilder von Stellen des Netzes N'. Sie haben die Eigenschaft, daß alle Urbilder von derselben Stelle $p \in P(N')$ *vergleichbar* sind, d.h. für jede Stelle $p_1, p_2 \in P(N(\sigma))$ gilt $pro(p_1) = pro(p_2) \Rightarrow p_1 F^* p_2$ oder $p_2 F^* p_1$.

Die Menge $pro^{-1}(p') = \{ p \in P(N(\sigma)) \mid pro(p) = p' \}$ bildet also bezüglich der Ordnungsrelation F^* eine Kette. Weiter ist diese Menge endlich (jede Spur ist endlich, weil sie als Durchschnitt von Mengen endlicher Ordnungen - Wörter - definiert ist).

Diese Menge kann man also auf natürliche Weise mit natürlichen Zahlen numerieren. Bei der Numerierung entspricht das i-te Element der Menge (Ordnung) $pro^{-1}(p)$ dem i-ten Vorkommen der Stelle p in dem Netz $N(\sigma)$. die gesuchte Markierung $M : P(N(\sigma)) \rightarrow FSM(SIG, V_S)$ konstruieren wir schrittweise wie folgt:

(*) Jedes erste Vorkommen $p \in P(N)$ einer Stelle $p' \in P(N')$ wird genau so wie sein Bild in dem Netz N' markiert d.h. $M(p) = M_0(pro(p))$.

Die so erhaltene Markierung bezeichnen wir mit dem Symbol M_0. In der Markierung M_0 sollte das kleinste Element der Menge aller maximalen Schnitte des Netzes $N(\sigma)$ aktivierbar sein. Durch das Schalten von diesem Schnitt bekommt man eine Markierung M_1 in dem der einzige Schnitt γ_1 mit der Eigenschaft daß $Pre(\gamma_1) = Post(\gamma_0)$ aktivierbar sein sollte. Es sei M_2 die Markierung, die man aus der Markierung M_1 durch das Schalten von dem Schnitt γ_1 erhält. In dieser Markierung sollte (wieder der einzige) Schnitt γ_2 mit der Eigenschaft, daß $Pre(\gamma_2) = Post(\gamma_1)$ aktivierbar sein, so daß man eine neue Markierung M_3 erhält etc. Formal definiert man eine Folge $M_0, M_1, M_2 \cdots M_k$ von Markierungen durch die folgende Definition.

Definition 7.12

$$M_0(p) = \begin{cases} M(pro(p)) & für\ p \in BEG(\sigma) \\ 0 & für\ p \notin BEG(\sigma) \end{cases}$$

$$M_{i+1}(p) = \begin{cases} M_i(p) & wenn\ M_i(p) \neq 0 \\ M_i'(pro(p)) & wenn\ es\ im\ Netz\ N(\sigma)\ ein\ Schritt\ \bar{\mu}\ gibt,\ so\ daß\ \bar{\mu} = (M_i, \mu, M_i') \end{cases}$$

Diese Folge ist endlich, weil das Netz $N(\sigma)$ endlich ist. Das letzte Element dieser Folge, d.h. die Markierung M_k ist die gesuchte Markierung des Netzes $N(\sigma)$.

Beispiel 7.18

Betrachten wir das Netz $N(\sigma)$ aus dem obigen Beispiel. Wir haben jetzt:

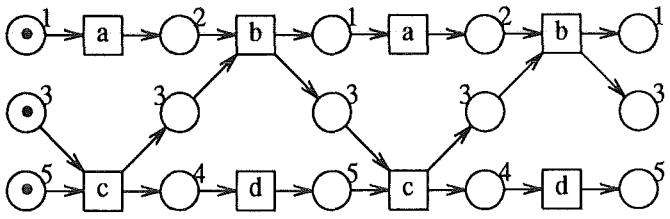


Abb. 7.1–5 : Die Markierung M_0

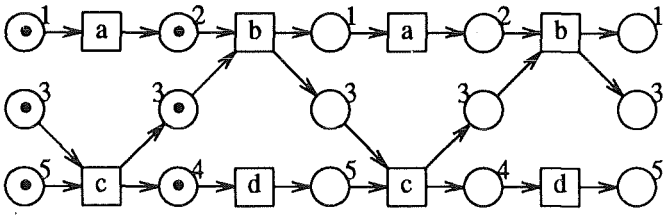


Abb. 7.1–6 : Die Markierung M_1

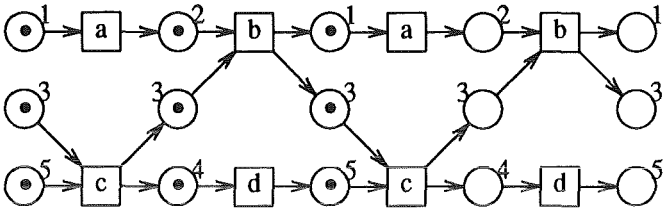


Abb. 7.1–7 : Die Markierung M_2

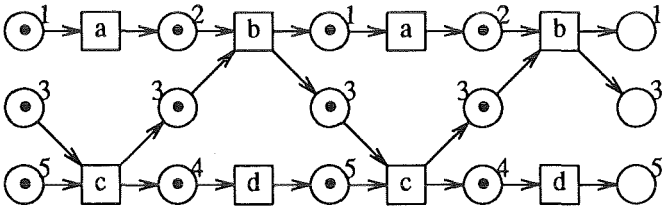


Abb. 7.1–8 : Die Markierung M_3

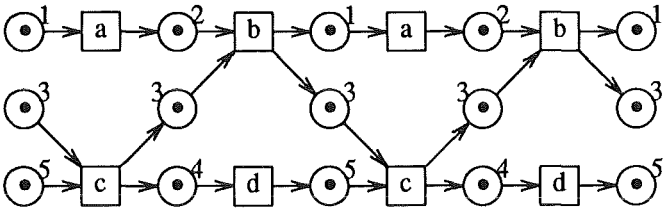


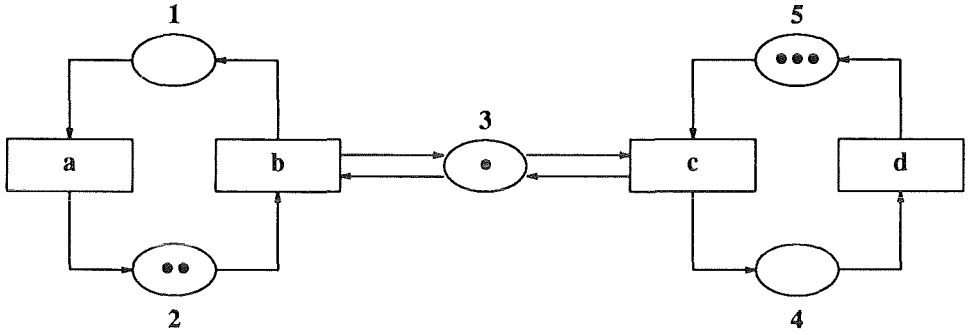
Abb. 7.1–9 : Die Markierung M_4

b) Event Structure

Eine andere Methode zur Beschreibung des Verhaltens eines Systems wurde von G. Winskel vorgeschlagen. Ähnlich zur Spurentheorie werden hier Vorkommen von Transitionen (Ereignisse) in einem System registriert, aber die Anordnung von diesen Ereignissen wird auf ganz andere Weise bestimmt. Bei dieser Methode werden keine einzelnen Ereignisse sondern Multimengen von diesen Ereignissen registriert. Jede solche Multimenge "beschreibt, was bis zu einem gewissen Moment" im System passiert ist.

Beispiel 7.19

Wir betrachten das Netz aus Beispiel 7.5, das wir zur Erinnerung noch einmal angeben wollen.

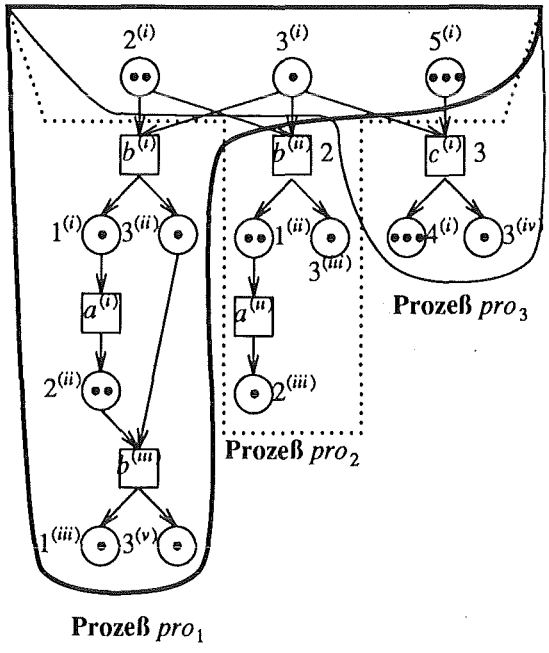


Eine mögliche Schaltfolge in dem Netz ist die Folge *b, a, b, c*. Die Familie der Multimengen, die dieses Ereignis beschreibt sind die Multimengen M_1={b}, M_2={b,a}, M_3={2b,a} und M_4={2b,a,c}.

Eine Spur beschreibt einen Prozeß in einem System (PrT-Netz) in dem Sinne, daß sie registriert was in einem System wirklich geschehen ist. Sie kann also als eine "ex post"-Beschreibung betrachet werden. Manchmal ist es aber wichtig Informationen über verschiedene mögliche Abläufe zu haben, d.h. von einer gewissen Markierung aus wollen wir wissen, was für Prozesse möglich sind.

Beispiel 7.20

Wir zeigen in dem folgenden Bild ein Netz, das eine Menge **P** von möglichen Prozessen pro_i darstellt. Die Prozesse pro_i sind gerade Prozesse des Netzes aus Beispiel 7.19.

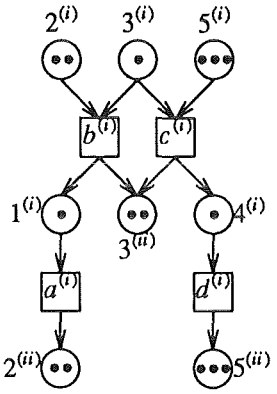


Die Menge **P** besteht also aus den Prozessen pro_1, pro_2 und pro_3.

Die Menge **P** kann man durch ein Netz charakterisieren indem man die entsprechenden *Vorkommen* von Stellen (Prädikaten) und Transitionen zusammenklebt. Die Flußrelation F in dem so erhaltenen Netz hat die Eigenschaft, daß die reflexive und transitive Hülle von F, d.h. die Relation F^* eine Halbordnung ist. Eine sehr natürliche Eigenschaft für das Netz wäre, daß jede kontaktfreie P-Schicht von dem Netz eindeutig die vorangehende T-Schicht bestimmt, d.h. daß wenn man von einer P-Schicht zurückschaut, immer einen Prozeß zu sehen ist. Grob kann man sagen, daß man immer weiß woher man kommt, aber nicht wohin man geht.

Beispiel 7.21

Ein Netz mit dieser Eigenschaft haben wir schon im vorhergehenden Beispiel gesehen. Das folgende Bild zeigt ein Netz ohne diese Eigenschaft. Das betrachtete Netz stellt allerdings keine möglichen Ereignistrukturen dar.



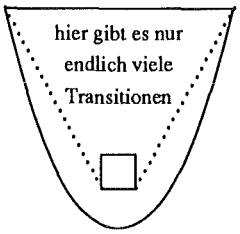
Bei diesem Netz kann man von der Stelle $3^{(ii)}$ nicht entscheiden woher man kam. Sowohl die Möglichkeit über Transition b als auch über Transition c ist gegeben. Wenn man von dieser Stelle aus zurückschaut, so sieht man keinen Prozeß.

Die obige Eigenschaft kann man sehr einfach durch eine Eigenschaft der Stellen des Netzes beschreiben. Jede dieser Stellen soll ein Nachfolger von höchstens einer Transition sein, d.h. das Netz soll die folgenden Bedingungen erfüllen:

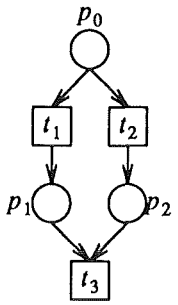
(i) Für jede Stelle $p \in P$ ist die Menge ${}^\bullet p = \{t \in T(N): tFp\}$ leer oder einelementig.
Diese Bedingung gewährleistet, daß jede P-Schicht des Netzes genau eine vorangehende T-Schicht bestimmt. Die folgende Abbildung verdeutlicht diese Bedingung.

Zwei Kanten als Input einer Stelle
sind nicht möglich!

(ii) Für jedes $t \in T(N)$ sollte die Menge $\{t' \in T(N): t'F^*t\}$ aller Vorgänger, die der Menge $T(N)$ angehören endlich sein.
Diese Bedingung garantiert, daß ein Vorkommen einer Transition nur vom Vorkommen von vorherigen Transitionen abhängt. Es ist also ausgeschlossen, daß eine Transition t erst nach unendlich vielen Vorkommen von anderen Transitionen schalten kann. Die folgende Abbildung verdeutlicht diese Bedingung.



(iii) Keine Transition des Netzes N kann von zwei anderen in einem Konflikt stehenden Transitionen abhängen, d.h. wenn Transitionen t_1 und t_2 eine gemeinsame Inputstelle p_0 haben (das ist wegen Bedingung (i) die einzig mögliche Art von Konflikt), dann dürfen sie keinen gemeinsamen Nachfolger haben. Man kann erreichen, daß die Konfliktrelation vererbbar und irreflexiv ist. Die folgende Abbildung verdeutlicht diese Bedingung.



Zwei in Konflikt stehende Transitionen
dürfen keine Transition
als gemeinsamen Nachfolger haben!

Diese Bedingung formuliert man durch Einführung von einer zusätzlichen Relation $\# \subseteq X(N) \times X(N)$, die eine Art von Konflikt zwischen den Elementen des Netzes N beschreibt. Zuerst definiert man die Relation des unmittelbaren Konfliktes: Transitionen t_1 und t_2 sind in unmittelbarem Konflikt, dann und nur dann wenn sie eine gemeinsame Inputstelle haben.

$$t_1 \#_m t_2 \Leftrightarrow {}^\bullet t_1 \cap {}^\bullet t_2 = \emptyset$$

Dann definiert man die Relation des Konfliktes durch die Vererbung beziehungsweise die Ordnungsrelation F^* d.h. Elemente $x, x' \in X(N)$ stehen in Konflikt, dann und nur dann wenn sie Vorgänger t und t' haben, die in unmittelbarem Konflikt stehen:

$$x \# x' \Leftrightarrow \exists_{t,t' \in T(N)} \; t \#_m t' \wedge t \, F^* x \wedge t' F^* x'$$

Jetzt kann man einfach die Bedingung (iii) durch die Irreflexivität der Relation # charakterisieren.

Die Netze mit den Eigenschaften (i) - (iii) wurden von G. Winskel, (vielleicht nicht besonders glücklich) "Occurence Netze" genannt. Wir werden sie *Occurence Netze im Sinne von Winskel* nennen damit sie von den Occurence Netzen der Definition 7.1 unterscheidbar sind. Ein wichtiges Merkmal von diesen Netzen ist, daß man sie durch eine spezielle Mengenfamilie und eine Relation charakterisieren kann. Diese Mengenfamilie besteht aus allen Mengen der Gestalt $\bar{p} \cap T(N)$, wobei $\bar{p}$ eine Untermenge der Menge X(N') aller Elemente eines Netzes N' ist, welches ein Occurence Netz im klassischen Sinne und ein Subnetz des Netzes N ist. Grob kann man sagen, daß die Elemente dieser Mengenfamilie Teile von den im Netz möglichen Prozessen sind. Diese Mengen kann man durch einfache (Pre-) Klassen der Relation $\bar{\#}$, d.h. der Relation $T \times T - \#$ beschreiben.

$$con(N) = \{X \subseteq T(N) \mid \forall_{t,t' \in X} \; \neg(t \# t') \wedge X \text{ ist endlich } \}$$

Die Relation, die zusammen mit der Familie $con(N)$ das Netz N charakterisiert ist einfach die Relation F^* beschränkt auf die Menge $T(N)$. Man kann zeigen, daß das Tupel $(T, con(N), F^* \mid_T)$ die folgenden Bedingungen erfüllt.

(i) alle Elemente der Familie $con(N)$ sind endliche Mengen.

(ii) die Familie $con(N)$ ist nach unten gerichtet, d.h. für beliebige Mengen X,Y $\subseteq$ T gilt:

$$X \in con(N) \wedge Y \subseteq X \Rightarrow Y \in con(N)$$

(iii) für jedes t$\in$T ist die Menge aller Vorgänger {t'$\in$T | t'F^*t} endlich.

(iv) für jedes t$\in$T gehört die einelmentige Menge {t} zu der Familie $con(N)$.

(v) für jede Menge X und Elemente $t \in T(N)$ gilt:

$$x \in con(N) \wedge \exists_{t' \in X} \; t F^* t' \Rightarrow X \cup \{t\} \in con(N)$$

Unter einer *primären Ereignisstruktur* (Event Structure) versteht man jedes Tupel der Gestalt $(T, con(N), \leq)$, das die folgenden, zu den obigen Bedingungen (i) - (v) gleichartigen, Bedingungen erfüllt.

(i') $con(N) \subseteq POW(T)$ ist eine Familie der endlichen Teilmengen der Menge T.

(ii') die Familie $con(N)$ ist nach unten gerichtet, d.h. für beliebige Mengen $X,Y \subseteq T$ gilt:

$$X \in con(N) \wedge Y \subseteq X \Rightarrow Y \in con(N)$$

(iii') $\leq \subseteq T \times T$ ist eine Halbordnung in T mit der Eigenschaft, daß jede Menge der Gestalt $(\leftarrow t] = \{x \in T \mid x \leq t\}$ endlich ist.

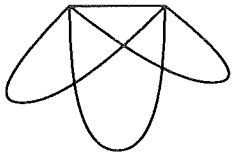
(iv') für jedes $t \in T$ gehört die einelmentige Menge $\{t\}$ zu der Familie $con(N)$.

(v') für jede Menge $X \subseteq T$ und Elemente $t \in T(N)$ gilt:

$$x \in con(N) \wedge \exists_{t' \in X} \; tF^* t' \Rightarrow X \cup \{t\} \in con(N)$$

Die Bedingung (i') besagt, daß es keine unendliche Prozesse geben kann. Durch Bedingung (iii') wird ausgeschlossen, daß eine Transition erst nach unendlich vielen Vorkommen von anderen Transitionen schalten kann. Das mindestens einmalige Vorkommen einer Transition wird durch Bedingung (iv') ausgedrückt.

Nun beschreiben wir den Zusammenhang zwischen Ereignisstrukturen und Spuren. Grob kann man sagen, daß eine Ereignisstruktur eine Vereinigung von Mengen von Spuren ist. Das ist ähnlich zu der Situation wo ein Occurence Netz im Sinne von Winskel eine Vereinigung von einer Menge von Occurence Netzen im klassischen Sinne ist.



Dieses Bild zeigt eine grobe Darstellung einer Ereignisstruktur als eine Menge von Spuren.

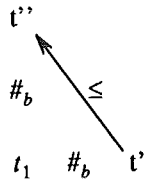
Es sei $(T, con(N), \leq)$ eine Ereignisstruktur. Wir definieren die binäre Konfliktrelation in folgendermaßen:

$$\text{für } t,t' \in T \text{ gilt: } t \#_b t' \Leftrightarrow \{t,t'\} \notin con(N)$$

Es ist leicht zu sehen, daß diese Relation symmetrisch und irreflexiv ist. Sie ist auch vererbbar (im Sinne der Relation "$\leq$"), d.h. für $t, t', t'' \in T$ haben wir

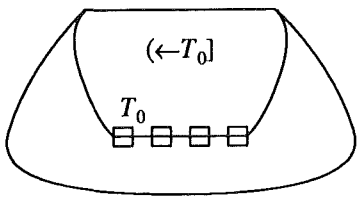
$$t \#_b t' \wedge t' \leq t'' \Rightarrow t \#_b t''$$

Das folgende Bild veranschaulicht diese Vererbung.



Diese Relation charakterisiert auch die Familie $con(N)$. Wir haben nämlich $con(N)=\{x \subseteq T \mid \forall_{t,t' \in X} \; \neg(t \#_b t')\}$. So kann man anstatt des Tripels $(T, con(N), \leq)$ auch das Tripel $(T, \#_b, \leq)$ betrachten, was manchmal günstiger ist.

Es sei $T_0 \subseteq T$ eine Klasse der Relation $\overline{\#_b \cup \leq}$, d.h. eine maximale (im Sinne der Mengeninklusions-Relation) Menge, von der zwei beliebige Elemente weder in der Relation "$\#_b$" noch in der Relation "$\leq$" stehen. Sie ist natürlich eine Antikette (üblicherweise keine maximale Antikette) der Ordnung $(T, \leq)$. Betrachten wir einmal die Menge $(\leftarrow T_0] = \{x \in T \mid \exists_{t \in T_0} x \leq t\}$



Das Paar $((\leftarrow T_0], \bar{\#}_b |_{(\leftarrow T_0] \times (\leftarrow T_0]})$ ist natürlich ein Ähnlichkeitsraum. Jede Klasse dieses Raumes, d.h. jede maximale Menge $X \subseteq (\leftarrow T_0]$, die die Bedingung $\forall_{x,x' \in X} \neg(x \#_b x')$ erfüllt, ist eine Spur. So kann man jede Ereignisstruktur mit Spuren überdecken, d.h. jede Ereignisstruktur kann als eine Vereinigung von Spuren betrachtet werden.

7.2 Hinweise zur Literatur

Prozesse in Petri Netzen werden in vielen Arbeiten beschrieben. Wir beschränken uns hier auf einige der unserer Überzeugung nach wichtigsten Auffassungen dieses Begriffes. Die Definition von Prozessen als Abbildungen wird analog zu der Arbeit [Gen80a] beschrieben. Die von uns beschriebene Algebra von Prozessen wurde von J. Winkowski eingeführt. Eine Beschreibung der Algebra kann man in seinen Arbeiten (vor allem in [Win82a]) finden.

Die Spurentheorie wurde von A. Mazurkiewicz und seinen Schülern entwickelt. Eine gute Übersicht über dieses Theorie mit Literaturverzeichnis kann man in [Maz86a] finden. Weitere interessante Arbeiten zur Spurentheorie sind [Roz83a, Roz83b] und [Roz86a].

Ähnlich zu der Spurentheorie ist die Theorie der partiellen Sprachen ([Gra81a, Sta81a]), die unter einem ganz anderen Gesichtspunkt entwickelt wurde. In der Arbeit von Starke [Sta81a] kann man die Beschreibung einer Algebra dieser Sprache finden.

Die Theorie der Ereignisstrukturen wurde vor allem in den Arbeiten von G. Winskel entwickelt. Eine gute Einführung in diese Theorie findet man in [Win86a].

8 Erreichbarkeitsanalyse

8.1 Grundlegende Definitonen

Im Folgenden sei (N,A_N,M_0) ein PrT-System und $v = (v_s : V_S(s) \rightarrow A_S(s))_{s\in S}$ bezeichne irgendeine zugehörige Auswertung. Bevor wir mit den grundlegenden Definitionen dieses Kapitels beginnen, wollen wir die hierfür aus dem Kapitel Petri Netze benötigten Definitionen noch einmal in etwas modifizierter Form aufführen. Diese sind:

- Die Menge aller Markierungen des Netzes N nennen wir im folgenden $\mathcal{M}_N$.
- Sei $\mu = \sum_{t\in T} a_t t$ eine Multisumme von Transitionen. μ heißt *Schritt*, wenn eine Markierung $M \in \mathcal{M}_N$ existiert, sodaß μ unter M aktiviert ist. Die Menge aller Schritte bezeichnen wir mit S_N.
- Ein *Schaltvorgang* (*das Schalten eines Schrittes*) ist ein Tripel $(M,\mu,M') \in \mathcal{M}_N \cup S_N \cup \mathcal{M}_N$, für das gilt:
  (a) μ ist ein Schritt, der unter der Markierung M aktiviert ist und
  (b) es gibt eine Auswertung v, sodaß μ bzgl. der Auswertung α unter M aktiviert ist und gilt:

$$\forall s\in S: M'(s) = M(s) - \sum_{t\in T} a_t A_F(s,t)(v) + \sum_{t\in T} a_t A_F(t,s)(v).$$

  Wenn wir betonen wollen, mit welcher Auswertung ein Schaltvorgang stattfindet, schreiben wir auch $(M,\mu{:}v,M')$ und sprechen dann von einem Schalten des Schrittes μ im *Mode* v unter der Markierung M. Die Menge aller Schaltvorgänge bezeichnen wir im folgenden mit S_N. Die Menge aller Schaltvorgänge in speziellen Modes (d.h.:, die Menge der Tripel der Gestalt $(M,\mu{:}v,M')$) bezeichnen wir im folgenden als $S\mathcal{M}_n$.

  Weiter schreiben wir einen Schaltvorgang (M,μ,M') auch in der Form $M[\mu>M'$ (oder $M[\mu{:}v>M'$) um suggestiv darzustellen, daß das System durch Schalten des Schrittes μ von der Markierung M in eine Markierung M' übergeht.
- Eine *Schrittfolge* ist ein Folge von Schaltvorgängen $(M_i,\mu_i,M_i')_{i\in\{1,\ldots,n\}}$ mit $M_i'=M_{i+1}$ für $i=1,2,\ldots,n-1$. Eine Schrittfolge schreiben wir suggestiv auch in der Form $M_1[\mu_1>M_2[\mu_2>M_3 \ldots\ldots M_n[\mu_n M_{n+1}=M_n'$.

Schaltvorgänge definieren eine Relation zwischen Markierungen eines PrT-Netzes in der folgenden Weise: M' ist von M aus erreichbar (im Zeichen $M\rightarrow M'$) genau dann, wenn es einen Schaltvorgang (M,μ,M') gibt. Wenn wir den reflexiven und transitiven Abschluß dieser Relation betrachten, erhalten wir eine weitere Relation, die man Erreichbarkeitsrelation des PrT-Netzes nennt.

Definition 8.1 : (Erreichbarkeitsrelation)

Es sei N ein PrT-Netz. Die Erreichbarkeitsrelation $\Rightarrow \subseteq \mathcal{M}_N \times \mathcal{M}_N$ ist wie folgt definiert: Eine Markierung M' ist von einer Markierung M aus *erreichbar*, wenn es eine Schrittfolge $(M_i,\mu_i,M_i')_{i\in\{1,\ldots,n\}}$ gibt, für die $M=M_1$ und $M'=M_n'$ ist. Dies läßt sich auch in der Form $M=M_1[\mu_1>M_2 \cdots M_n[\mu_n>M_n=M'$ schreiben.

$\mathcal{M}_N$ zerfällt bzgl. der Erreichbarkeitsrelation in der Regel nicht disjunkte Mengen $[M>$ von M aus erreichbaren Markierungen, wobei M beliebig ist. Eine Sonderrolle nimmt dabei die Menge $[M_0>$ ein, wobei M_0 die Anfangsmarkierung des Netzes ist.

Definition 8.2 : (Erreichbarkeitsmenge)

Sei N ein PrT-Netz mit Anfangsmarkierung M_0. $[M_0>$ heißt *Erreichbarkeitsmenge* von N.

Die Erreichbarkeitsmenge eines Netzes enthält alle Folgemarkierungen, die iterativ aus der Anfangsmarkierung durch Schalten von Schritten entstehen, die bzgl. einer bereits ermittelten Folgemarkierung von M_0 aktiviert sind. Das durch die Anfangsmarkierung definierte dynamische Verhalten eines PrT-Netzes wird nun aber gerade durch die von M_0 aus erreichbaren Markierungen und allen Schaltvorgänge, die von solchen Markierungen ausgehen können, bestimmt. Wir führen hierzu die folgenden Bezeichnungen ein.

Notation 8.1

Die Menge aller Schaltvorgänge, die von Markierungen $M \in [M_0>$ ausgehen können, bezeichnen wir im folgenden mit S_{N,M_0}. Entsprechend bezeichnen wir die Einschränkung von $\mathcal{SM}_N$ auf Schaltvorgänge, die nur von Markierungen $M \in [M_0>$ ausgehen können, mit $\mathcal{SM}_{N,M_0}$.

Als Beschreibungsmittel für das dynamische Verhalten eines PrT-Netzes mit der Anfangsmarkierung M_0 bietet sich nun ein Graph an, der die von M_0 erreichbaren Markierungen $[M_0>$ als Knoten und die Schaltvorgänge $(M,\mu{:}v,M') \in \mathcal{SM}_{N,M_0}$ als Kanten besitzt, wobei wir bewußt Schaltvorgänge nach verschiedenen Modi unterscheiden. Der betrachtete Graph ist dann ein Graph mit im allgemeinen mehreren gerichteten Kanten zwischen je zwei Knoten. Eine mögliche Definition für einen solchen Graph mit Mehrfachkanten ist:

Definition 8.3 : (gerichteter Graph)

Ein 4-Tupel $G=(N,A,d,e)$ heißt *gerichteter Graph*, wenn gilt:

(a) N und A sind disjunkte Mengen, die wir Mengen der *Knoten* und Menge der *Kanten* des Graphes nennen,

(b) $d{:}A \to N$ und $e{:}A \to N$ sind Abbildungen, die jeder Kante $a \in A$ ihren *Anfangspunkt* $d(a) \in N$ und ihren *Endpunkt* $e(a) \in N$ zuordnen.

Wir wollen zu dieser Definition ein Beispiel betrachten.

Beispiel 8.1

Es sei $N=\{1,2,3\}$ und $A=\{a,b,c,d\}$. Die Abbildungen $e,d{:}A \to N$ seien durch $d(a)=1$, $d(b)=1$, $d(c)=2$, $d(d)=2$ und $e(a)=2$, $e(b)=3$, $e(c)=1$, $e(d)=1$ gegeben. Den zugehörigen Graph können wir dann bildlich wie folgt darstellen:

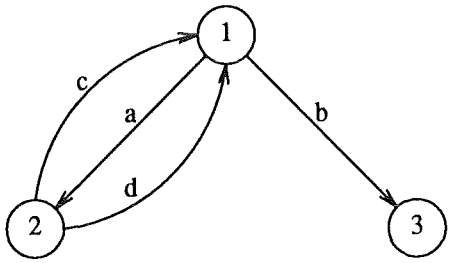


Abb. 8.1–1 : Bildliche Darstellung des Graphen aus Beispiel 8.1

Mit der obigen Definition eines Graphes können wir nun den Schrittgraph eines PrT-Netzes wie folgt definieren:

Definition 8.4 : (Schrittgraph)

Sei N ein PrT-Netz. Der *allgemeine Schrittgraph* $\mathcal{RGM}_N$ von N ist definiert als der Graph $\mathcal{RGM}_N=([M_0>,\mathcal{SM}_{N,M_0},d,e)$, für den gilt:

(a) $d((M,\mu{:}v,M'))=M$ für alle $(M,\mu{:}v,M') \in \mathcal{SM}_{N,M_0}$,

(b) $e((M,\mu{:}v,M'))=M'$ für alle $(M,\mu{:}v,M') \in \mathcal{SM}_{N,M_0}$.

Wenn auch die in 8.3 dargelegte Definition eines Graphes sehr anschaulich ist und sich damit besonders bei Problemstellungen, die bildlich dargestellt werden können, eignet, so läßt sie sich weniger gut für eine Algebraisierung von Problemen verwenden. Wir geben daher noch eine alternative, "mehr algebraische" Definition des Schrittgraphes an, die auf der folgenden Definition eines Graphes basiert.

Definition 8.5 : (alternative Definition eines Graphs)

Ein 3-Tupel (V,dom,cod) heißt (gerichteter) Graph, wenn gilt:

(a) V ist eine Menge, die wir Menge der Knoten und Kanten des Graphs nennen wollen.

(b) $dom,cod:V\to V$ sind Operationen auf V.

(c) Es gelten die Gleichungen

(i) $dom(dom(x))=cod(dom(x))=dom(x)$ für alle $x\in V$.

(ii) $dom(cod(x))=cod(cod(x))=cod(x)$ für alle $x\in V$.

Wie können wir nun diese Definition als Graph im Sinne unserer ursprünglichen Definition sehen? Nun, dazu müssen wir die Menge V zunächst in die Menge der Knoten und Kanten zerlegen. Ausgangspunkt hierfür ist, daß wir die Menge der Knoten als die Menge der Fixpunkte der beiden Operationen *dom* und *cod* definieren:

$$N=\{x\in V \mid dom(x)=cod(x)=x\}.$$

Die Menge der Kanten ist dann die Differenz $A=V\backslash N$. Die Abbildungen d und e ergeben sich dann als Einschränkung von *dom* und *cod* auf die Menge der Kanten, d.h.: $d=dom\mid_A$ und $e=cod\mid_A$. Da für alle $a\in A$ aus Gleichung (i) von 8.4 $dom(y=dom(a))=dom(a)=y$ und $cod(y=dom(a))=dom(a)=y$ folgt, ist die oben definierte Einschränkung d offensichtlich eine Abbildung $d:A\to N$, wie dies gewünscht ist. Aus (ii) von 8.4 folgt analog, daß die als e definierte Abbildung wie gewünscht eine Abbildung $e:A\to N$ ist. Damit ist das 4-Tupel (N,A,d,e) wie oben definiert ein Graph in unserem ursprünglichen Sinne.

Wenn wir umgekehrt von einem Graphen (N,A,d,e) ausgehen, so läßt sich hieraus in einfacher Weise ein Graph in unserer alternativen Form definieren. Hierzu definieren wir $V=N\cup A$ und die Abbildungen $dom:V\to V$ und $cod:V\to V$ durch $dom(x)=d(x)$, $cod(x)=e(x)$ für alle $x\in A$ und $dom(x)=cod(x)=x$ für alle $x\in N$. Es ist dann leicht zu zeigen, daß (V,dom,cod) ein Graph gemäß unserer alternativen Definition ist. Man kann sich *dom* und *cod* ebenfalls als Funktionen vorstellen, die Elementen von V ihren Anfangs- bzw. Endpunkt zuordnen, wobei ein Knoten sich selbst als Anfangs- und Endpunkt besitzt.

Unsere alternative Definition des Schrittgraphes ergibt sich damit zu:

Definition 8.6 : (Alternative Definition eines Schrittgraphs)

Der Schrittgraph eines PrT-Netzes N ist der Graph $\mathcal{RGM}_N=(V,dom,cod)$, für den gilt:

(a) $V=[M_0>\cup\mathcal{SM}_0$.

(b) $dom:V\to V$ ist definiert durch $dom(x)=M$ für alle $x=(M,\mu:v,M')\in\mathcal{SM}_0$ und $dom(x)=x$ für alle $x\in[M_0>$.

(c) $cod:V\to V$ ist definiert durch $cod(x)=M'$ für alle $x=(M,\mu:v,M')\in\mathcal{SM}_0$ und $cod(x)=x$ für alle $x\in[M_0>$.

Wir werden je nach Situation im Folgenden einmal die eine oder die andere dieser äquivalenten Definitionen benutzen.

Schrittfolgen im Netz kann man im zugehörigen Schrittgraph als "Wege" interpretieren. Dazu müssen wir zunächst definieren, was ein Weg in einem Graph ist.

Definition 8.7 : (Weg in einem Graphen)

Sei $G=(N,A,d,e)$ ein Graph. Ein Wort $w\in A^*$ heißt *Weg in G*, wenn gilt:
Falls $w=a_1\cdots a_n$ ist, so ist $d(a_i)=e(a_{i-1})$ für alle $i=2,...,n$.
Ein *Weg* $w=a_1\cdots a_n$ heißt *geschlossen*, wenn zusätzlich $d(a_1)=e(a_n)$ ist.

Es sollte klar sein, daß auf Grund unserer Definition der Erreichbarkeitsrelation eine Markierung M' von der Anfangsmarkierung M_0 des Netzes genau dann erreichbar ist, wenn es einen Weg im Schrittgraph von der Wurzel M_0 des Graphen zum Knoten M gibt. Wir haben also:

Bemerkung 8.1

Eine Markierung M ist von der Anfangsmarkierung M_0 genau dann erreichbar, wenn M ein Knoten des Schrittgraphes $\mathcal{RGM}_N$ ist und wenn es einen Weg w im Schrittgraph von der Wurzel M_0 nach M gibt.

Im Schrittgraph ist jeder Knoten, da er per Definition eine von der Anfangsmarkierung M_0 aus erreichbare Markierung ist, über einen Weg vom Knoten M_0 aus erreichbar. Einen Graphen, der einen Knoten K besitzt, sodaß alle Knoten von diesem aus über Wege erreichbar sind, nennt man auch Wurzelgraph mit Wurzel K. Es gilt daher:

Bemerkung 8.2

Der Schrittgraph ist ein Wurzelgraph mit M_0 als Wurzel.

Die Erreichbarkeit einer Markierung von einer anderen Markierung eines Netzes aus ist eine der vielen Eigenschaften von Netzen, die von den Markierungen des Netzes und dem Änderungverhalten der Markierungen nicht aber davon, wie diese Änderung zustande kommt, abhängen. Insbesondere ist der Mode, mit dem ein Schritt schaltet, für viele Eigenschaften von Netzen nicht relevant. Wir werden in solchen Fällen nur einen (reduzierten) Schrittgraphen betrachten, der aus dem allgemeinen Schrittgraphen dadurch entsteht, daß wir Schritte nicht mehr nach der Art der Substitution unterscheiden.

Definition 8.8 : (reduzierter Schrittgraph)

Sei N ein PrT-Netz. Der *(reduzierte) Schrittgraph* $\mathcal{RG}_N$ von N ist definiert als der Graph $\mathcal{RG}_N=([M_0>,S_{N,M_0},d,e)$, für den gilt:

(a) $d((M,\mu,M'))=M$ für alle $(M,\mu,M')\in S_{N,M_0}$,

(b) $e((M,\mu,M'))=M'$ für alle $(M,\mu,M')\in S_{N,M_0}$.

Unsere Bemerkung 8.1 können wir jetzt so formulieren:

Bemerkung 8.3

Eine Markierung M ist von der Anfangsmarkierung M_0 genau dann erreichbar, wenn M ein Knoten des reduzierten Schrittgraphes $\mathcal{RG}_N$ ist und wenn es einen Weg w im reduzierten Schrittgraph von der Wurzel M_0 nach M gibt.

Der reduzierte Schrittgraph entsteht aus dem allgemeinen Schrittgraph durch Identifizieren von Kanten. Ein weiterer für die Analyse wichtiger Graph entsteht aus dem Schrittgraph durch Identifizierung von Knoten und Kanten. Hierzu wollen wir die Erreichbarkeitsrelation zunächst zu einer Äquivalenzrelation zwischen Markierungen ausdehnen, die uns die starken Zusammenhangskomponenten der Markierungen eines Netzes liefert.

Definition 8.9 : (starke Zusammenhangskomponenten)

Sei N ein PrT-Netz und M_0 seine Anfangsmarkierung. Die Äquivalenzrelation $\Leftrightarrow\subseteq[M_0>\cup[M_0>$ sei wie folgt definiert:

$$M\Leftrightarrow M' \text{ genau dann, wenn gilt: } M\Rightarrow M' \text{ und } M'\Rightarrow M.$$

Die Äquivalenzklassen von $\Leftrightarrow$ in $[M_0>$ nennt man *starke Zusammenhangskomponenten* von N.

Es ist offensichtlich, daß $\Leftrightarrow$ eine Äquivalenzrelation ist. Man kann daher den Faktorgraph $\mathcal{RG}_n/\Leftrightarrow$ bilden, der dadurch entsteht, daß wir zunächst alle Knoten des reduzierten Schrittgraphes identifizieren, die innerhalb derselben starken Zusammenhangskomponente liegen und dann Kanten nur noch zwischen den Zusammenhangskomponenten einführen, für die ein Weg im reduzierten Schrittgraph von einem Repräsentant der einen Zusammenhangskomponente zu einem Repräsentant der anderen Zusammenhangskomponente geht.

Definition 8.10 : (Graph der starken Zusammenhangskomponenten)

Sei N ein PrT-Netz und $\mathcal{RG}_N$ der zugehörige reduzierte Schrittgraph. Der *Graph der starken Zusammenhangskomponenten von N* (im Zeichen SCG_N) ist der Graph $SCG_N=(SCC_N,A,d,e)$, für den gilt:

(a) $SCC_N=[M_0>/\Leftrightarrow$,

(b) $A=\{(scc_1,scc_2)\in SCC_N\times SCC_N \mid \exists M\in scc_1, M'\in scc_2 \; mit: M\rightarrow M'\}$,

(c) für $a=(scc_1,scc_2)\in A$ ist $d(a)=scc_1$ und $e(a)=scc_2$).

Wie der Schrittgraph ist auch der Graph der starken Zusammenhangskomponenten ein Wurzelgraph. Auf Grund seiner Konstruktionsweise ist er weiter azyklisch.

Bemerkung 8.4

Sei N ein PrT-Netz und M_0 die zugehörige Anfangsmarkierung. SCG_N ist ein azyklischer Wurzelgraph, dessen Wurzel diejenige Zusammenhangskomponente ist, die die Anfangsmarkierung M_0 enthält.

Wie bei der Definition der Erreichbarkeit kann man auch zwischen den starken Zusammenhangskomponenten eines Netzes eine Erreichbarkeitsrelation einführen.

Definition 8.11 : (Nachfolgerrelation zwischen starken Zusammenhangskomponenten)

Sei N ein PrT-Netz und SCC_N die Menge der starken Zusammenhangskomponenten von N.

(a) $scc_2 \in SCC_N$ heißt *direkter Nachfolger* von $scc_1 \in SCC_N$, wenn (scc_1, scc_2) eine Kante des Graphs der starken Zusammenhangskomponenten SCG_n ist. Wir schreiben diese Relation im Zeichen mit $ssc_1 \rightarrow scc_2$.

(b) Den reflexiven und transitiven Abschluß der unter (a) eingeführten Relation bezeichnen wir mit $\Rightarrow$. Wir nennen diese Relation *Nachfolgerrelation* und sagen für $scc_1 \Rightarrow scc_2$ auch, daß scc_2 ein *Nachfolger* von scc_1 ist.

(c) Elemente aus SCC_N, die keine Nachfolger besitzen, nennt man *Blätter* des Graphen SCG_n.

8.2 Lebendigkeits-Eigenschaften von PrT-Netzen

Jede Markierung, die nicht der Anfangsmarkierung eines Netzes entspricht und unter der keine Schritte aktivierbar sind, nennt man eine (statische) Verklemmung.

Definition 8.12 : (statische Verklemmung)

Sei N ein PrT-Netz mit der Anfangsmarkierung M_0 und $M \in [M_0\rangle$ mit $M \neq M_0$.

(a) M heißt *(statische) Verklemmung des Netzes N* genau dann, wenn kein Schritt existiert, der unter M aktivierbar ist.

(b) N heißt *verklemmungsfrei* genau dann, wenn für alle Markierungen $M \in [M_0\rangle$ gilt: M ist keine statische Verklemmung von N.

Der Begriff der (statischen) Verklemmung beschreibt somit Markierungen eines Netzes, in denen der dynamische Ablauf des Tokenspiels sein Ende findet, da unter der betreffenden Markierung keine weiteren Schritte mehr aktivierbar sind. Im Gegensatz hierzu beschreiben dynamische Verklemmungen Markierungen (bzw. Mengen von Markierungen) unter denen zwar noch Schritte aktivierbar sind, die aus den Schritten resultierenden Folgemarkierungen jedoch der Ausgangsmarkierung (einem Element der Ausgangsmarkierungsmenge) entsprechen.

Definition 8.13 : (dynamische Verklemmung)

Sei N ein PrT-Netz mit der Anfangsmarkierung M_0 und $M \in [M_0\rangle$ mit $M \neq M_0$. M heißt *triviale dynamische Verklemmung* genau dann, wenn gilt:

(1) M ist keine statische Verklemmung.

(2) Für jeden Schaltvorgang (M, μ, M') gilt: $M' = M$.

Bei einer trivialen dynamischen Verklemmung kann als Folgemarkierung also nur noch die Markierung selbst auftreten. Der Zustand des Systems (beschrieben durch die Markierung) ist also ab einer trivialen dynamischen Verklemmung konstant.

Verklemmungen lassen sich sehr einfach bzgl. dem Graphen der starken Zusammenhangskomponenten interpretieren. Die Bedingung, daß keine Folgemarkierung oder höchstens die Markierung selbst als Folgemarkierung auftritt, bedeutet, daß eine Verklemmung ein Blatt von SCG_N sein muß, das genau ein Element enthält. Bei einer dynamischen Verklemmung muß zusätzlich ein Schritt unter der besagten Markierung aktivierbar sein.

Bemerkung 8.5

$M \in [M_0>$ ist eine triviale dynamische Verklemmung genau dann, wenn ein Blatt $scc \in SCC_N$ des Graphen $\mathcal{SCG}_N$ existiert, für das gilt:

(1) $scc=\{M\}$,

(2) $M \neq M_0$,

(3) es gibt einen Schritt, der unter M aktivierbar ist.

Wenn (1) und (2) nicht aber (3) erfüllt sind, handelt es sich offensichtlich um eine statische Verklemmung.

Die obige Bemerkung legt es nahe, daß man die Definition der dynamischen Verklemmung auf Blätter des Graphen der starken Zusammenhangskomponenten ausdehnt, die nicht nur aus einem Element bestehen.

Definition 8.14 : (dynamische Verklemmung)

Sei N ein PrT-Netz mit der Anfangsmarkierung M_0. Eine Teilmenge $\mathcal{M} \subset [M_0>$ heißt *dynamische Verklemmung* genau dann, wenn gilt:

(1) Es gibt ein Blatt $scc \in SCC_N$ von $\mathcal{SCG}_N$ mit: $scc=\mathcal{M}$.

(2) $M_0 \notin \mathcal{M}$ und $|\mathcal{M}|>1$.

Es sollte nun klar sein, daß gilt:

Bemerkung 8.6

(a) Statische Verklemmungen, triviale Verklemmungen und dynamische Verklemmungen sind wechselseitig disjunkte Blätter von $\mathcal{SCG}_N$.

(b) Blätter von $\mathcal{SCG}_N$, die nicht die Anfangsmarkierung enthalten, sind entweder dynamische, trivial dynamische oder statische Verklemmungen.

Wir wollen uns nun einzelnen Transitionen zuwenden und Eigenschaften dieser bzgl. ihrer Aktivierbarkeit untersuchen. Wir beginnen mit der Definition:

Defnition 8.15 : (tote Transitionen und schwache Lebendigkeit)

Sei N ein PrT-Netz und M_0 dessen Anfangsmarkierung.

(a) Eine Transition $t \in T$ heißt *tot*, wenn t unter keiner Markierung $M \in [M_0>$ aktivierbar ist. Andernfalls nennen wir *t schwach lebendig*.

(b) Das PrT-Netz N heißt *schwach lebendig*, wenn keine seiner Transitionen tot ist.

(c) Eine Transition $t \in T$ heißt *stark lebendig*, wenn zu jeder Markierung $M \in [M_0>$ eine Folgemarkierung $M' \in [M_0>$ existiert, sodaß t unter M' aktiviert ist.

(d) Ein PrT-Netz N heißt *stark lebendig*, wenn alle seine Transitionen stark lebendig sind.

Damit eine Transition $t \in T$ schwach lebendig ist, reicht es offensichtlich aus, wenn im zum Netz N gehörigen reduzierten Schrittgraph $\mathcal{RG}_N$ eine Kante der Gestalt (M,t,M') existiert. Damit eine Transition stark lebendig ist, müssen in jeder starken Zusammenhangskomponente $scc \in SCC_N$ Markierungen $M,M' \in scc$ existieren, sodaß (M,t,M') ein Schaltvorgang von N ist. Weiter ist jede stark lebendige Transition sicher schwach lebendig und bei nur einer starken Zusammenhangskomponente ist jede schwach lebendige Transition schon stark lebendig. Wir haben also:

Bemerkung 8.7

Sei N ein PrT-Netz und M_0 dessen Anfangsmarkierung.

(a) Eine Transition $t \in T$ ist genau dann stark lebendig, wenn innerhalb jeder starken Zusammenhangskomponente $scc \in SCC_N$ Markierungen $M,M' \in scc$ existieren, sodaß die Relation $M \rightarrow M'$ gilt.

(b) Jede stark lebendige Transition ist trivialer Weise schwach lebendig.

(c) Besitzt das Netz nur eine starke Zusammenhangskomponente, so ist jede schwach lebendige Transition bereits stark lebendig.

(d) Ein Netz, daß stark lebendig ist, ist verklemmungsfrei.

Wenn eine Transition lebendig ist, so stellt sich weiter die Frage, wie oft diese Transition im dynamischen Ablauf des Netzes schalten kann. Hier interessiert vor allem, ob die Transition nur endlich oft oder unendlich oft aktivierbar ist.

Definition 8.16 : (Beschränkte Aktivierbarkeit)

Sei N ein PrT-Netz und M_0 die Anfangsmarkierung. Eine Transition $t \in T$ heißt *beschränkt aktivierbar* per Definition genau dann, wenn sie nicht tot ist, aber eine Konstante $n \in N$ existiert, sodaß $t \in T$ höchstens n-mal in einer beliebigen Schaltfolge $(M_i, t_i, M_i')_{i \in I}$ auftreten kann.

Da wir es mit endlichen Erreichbarkeitsgraphen zu tun haben, kann eine Transition nur dann unbeschränkt aktivierbar sein, wenn sie in einem geschlossenen Weg im Erreichbarkeitsgraph vorkommt. Solche geschlossenen Wege werden aber durch die starke Zusammenhangsrelation gerade identifiziert. Daher gilt:

Bemerkung 8.8

Sei N ein PrT-Netz und M_0 die Anfangsmarkierung. Eine Transition $t \in T$ ist genau dann beschränkt aktivierbar, wenn sie nicht tot ist und nur in Übergängen zwischen starken Zusammenhangskomponenten schalten kann.

Wenn alle Transitionen nur beschränkt aktivierbar sind, kann der Erreichbarkeitsgraph keine geschlossenen Wege enthalten. Es gilt daher:

Bemerkung 8.9

Wenn alle Transitionen eines PrT-Netzes N beschränkt aktivierbar sind, gibt es im Netz keine dynamischen Verklemmungen.

Neben dem Verhalten von Transitionen sind auch Fragen interessant, die die Reproduzierbarkeit von Markierungen betreffen. Hierzu führen wir die folgende Definition ein:

Definition 8.17 : (Schwache Reproduzierbarkeit von Markierungen)

Sei N ein PrT-Netz und M_0 die Anfangsmarkierung von N. Eine Markierungsteilmenge $\mathcal{M} \subset [M_0>$ heißt *schwach reproduzierbar* genau dann, wenn jede Markierung $M \in \mathcal{M}$ nach ihrem Verlassen wieder erreicht werden kann.

Markierungen innerhalb einer starken Zusammenhangskomponente, die mehr als ein Element enthält, sind offensichtlich nach Verlassen der Markierung wieder erreichbar. Wir haben also.

Bemerkung 8.10

Jede starke Zusammenhangkomponente $SCC \in SCC_N$, die mehr als ein Element enthält, ist schwach reproduzierbar.

Eine stärkere Version der schwachen Reproduzierbarkeit ist die starke Reproduzierbarkeit.

Definition 8.18 : (Starke Reproduzierbarkeit von Markierungen)

Sei N ein PrT-Netz und M_0 die Anfangsmarkierung von N. Eine Markierungsteilmenge $\mathcal{M} \subset [M_0>$ heißt *stark reproduzierbar*, wenn jede Markierung $M \in \mathcal{M}$ von jeder Folgemarkierung $M' \in [M>$ mit $M' \neq M$ aus erreichbar ist.

Mit der Beobachtung, daß starke Zusammenhangskomponenten, die Blätter des Graphs der starken Zusammenhangskomponenten sind, nur noch Markierungen enthalten, von denen aus Schaltvorgänge die Zusammenhangskomponente nicht mehr verlassen können, können wir folgern:

Bemerkung 8.11

Starke Zusammenhangskomponenten, die Blätter des Graphs der starken Zusammenhangskomponenten sind und mehr als ein Element enthalten, sind stark reproduzierbar.

8.3 Hinweise zur Literatur

Eine Einführung in die Theorie der Petri-Netze, die auch die Erreichbarkeitsanalyse behandelt, findet man in [Rei86a].

Eine weiterführende Arbeit, die speziell die Erreichbarkeitsanalyse behandelt, ist [Hub86a].

In [May80a] und [Raz85a] findet man Algorithmen, die sich für eine computergestützte Erreichbarkeitsanalyse von Netzen eignen.

9 Liste der häufig verwendeten Zeichen

A^*	Menge aller Wörter über dem Alphabet A
A^+	Menge aller Wörter über dem Alphabet A mit l(w)≥1
A^w	$A(s_1)\times A(s_2)\times..\times A(s_k)$ (mit $w=s_1s_2,\ldots,s_k$)
A_N	Beschriftung des Netzes N
AL	Symbole des Alphabets $ALPH^0$
$ALG=(A_S,A_\Omega)$	Algebra
$ALG(LSIG)$	logische Matrix
$ALPH^0$	Alphabet einer formalen Sprache der Ordnung 0
$ALPH^1$	Alphabet erster Ordnung
$ALS=(A_S, A_\Omega, A_\Pi)$	algebraisches System
$BEH(N)$	Verhalten des Netzes N
$FORM(SIG,V_S)$	Sprache der elementaren Formeln der Signatur SIG
$FORM(ALPH^0)$	Menge der Aussagenformeln über dem Alphabet $ALPH^0$
$FORM(LSIG,SIG,V_S,Q)$	Formeln einer Sprache erster Ordnung
$FSM(SIG,V)$	Menge aller formalen Summen über $TERM(SIG,V)$
G	gerichteter Graph
L	Sprache der Ordnung 0
L^1	Sprache erster Ordnung
$M : P \rightarrow FSM(SIG,V_S)$	Markierung des PrT-Netzes (N,A_N)
(M, μ, M') bzw. $(M[\mu>M')$	Schaltvorgang
$(M,\mu:v,M')$ bzw. $(M[\mu:v>M')$	Schalten des Schrittes μ im *Mode* v unter der Markierung M
$\mathcal{M}_N$	Menge aller Markierungen des Netzes N
$[M_0>$	Erreichbarkeitsmenge eines Netzes
$MULT(A)$	Menge aller Multimengen über der Menge A
$N = (P,T,F,K)$	Petri Netz
(N,A_N)	PrT-Netz
(N,A_N,M_0)	PrT-System mit Anfangsmarkierung M_0
N^{oc}	T-Multinetz
(N^{oc},M)	markiertes T-Multinetz
$\mathbf{R}$	Realisation einer Sprache erster Ordnung
$\mathbf{R}_T$	Realisation von Termen
$\mathbf{R}_{EF}$	Realisation der elementaren Formeln
$\mathbf{R}_F$	Realisation der Formeln
$\mathbf{R}_{FSM}$	Realisation einer formalen Summe
$POL(ALS)$	Menge der Polynome eines algebraischen Systems ALS
$REL(ALS)$	Menge der einfachen Relationen des Systems ALS
$\mathcal{RGM}_N$	allgemeiner Schrittgraph
SCC_N	starke Zusammenhangskomponente von N
SCG_N	Graph der starken Zusammenhangskomponenten von N
$SCHRITT(N,A_N)$	Menge aller Schritte
$SIG=(S,\Omega,\Pi,typ_\Omega,typ_\Pi)$	Signatur
$\mathcal{SM}_{N,M_0}$	Menge aller Schaltvorgänge, die nur von Markierungen $M\in[M_0>$ ausgehen können
S_{N,M_0}	Menge aller Schaltvorgänge, die von Markierungen $M\in[M_0>$ ausgehen können
$TERM(SIG,V_S)$	Sprache der Terme
$TERM_s(SIG,V)$	Menge aller Terme vom Typ s
(V,dom,cod)	gerichteter Graph
$card(\omega)$	Kardinalität von ω
$dom(w)$	Urbildmenge des Wortes w
ε	leeres Wort
$l(w)$	Länge des Wortes w

pro	Prozeß in einem PrT-Netz N'
spro	Semiprozeß in dem Netz N'
typ(t)	Typ eines Terms t
v	Auswertung von Variablen
$\bar{v}_T$	Auswertung der Terme $t \in TERM(SIG, V_S)$
$\bar{v}_F$	Auswertung von elementaren Formeln
$/w/$	Menge der Elemente der Folge w

LITERATUR

[ACPN86] W. Brauer, W. Reisig, and G. Rozenberg, (ed.), *Advanced Course on Petri Nets (Springer-Verlag, LNCS-254/-255)*, Bad Honnef, September 1986.

[Coh65a] P.M. Cohn, *Universal Algebra*, Reidel Publication, Dortrecht, 1981, 1965.

[Der79a] N. Dershowitz and Z. Manna, "Proving termination with multisets orderings in Automata, Languages and Programming," in *Lecture Notes in Computer Science*, vol. 71, Springer Verlag, Berlin, Heidelberg, New York, 1979.

[Gen79a] H.J. Genrich and K. Lautenbach, "The Analysis of Distributed Systems by Means of Predicate/Transition-Nets," in *Lecture Notes in Computer Science Vol. 70: Semantics of Concurrent Computation*, ed. Gilles Kahn, pp. 123-146, Springer-Verlag, 1979.

[Gen80a] H.J. Genrich, K. Lautenbach, and P.S. Thiagarajan, "Substitution Systems - A Family of Systems Models based on Concurrency," in *Lecture Notes in Computer Science*, ed. J. Hartmanis, vol. 88, pp. 698-723, Springer-Verlag, Berlin, Heidelberg, New York, 1980.

[Gen81a] H. J. Genrich and K. Lautenbach, "System Modeling with high-level Nets," *THEOR. COMPUTER SCIENCE*, no. 13, pp. 109-136, North-Holland, 1981.

[Gen86a] H. J. Genrich, "Predicate/Transition Nets," in *[ACPN86]*, pp. Part I,207-247, 1986.

[Gra81a] J. Grabowski, "On Partial Languages," *Fund. Inform.*, vol. 4, no. 1, pp. 427-498, 1981.

[Gr83a] G. Grätzer, *Universal Algebra*, Springer Verlag, Berlin, Heidelberg, New York, 1979, 1983.

[Her76a] H. Hermes, in *Einführung in die mathematische Logik*, B.G. Teubner Stuttgart, 1976.

[Hic80a] J.L. Hickman, "A note on the concept of multiset," *Bulletin Of The Australian Mathematical Society*, vol. 27, 1980.

[Hub86a] P. Huber, et al, "Reachability Trees for High-level Nets," *Theor. Computer Science*, vol. 45, pp. 261-292, 1986.

[Jen81a] K. Jensen, "Coloured Petri Nets and the Invariant Method," *THEOR. COMPUTER SCIENCE*, vol. 14, pp. 317-336, North-Holland, 1981.

[Jen81b] K. Jensen, "How to Find Invariants for Coloured Petri Nets," in *Lecture Notes in Computer Science Vol. 118: Mathematical Foundations of Computer Science*, ed. A. Mazurkiewicz, pp. 327-338, Springer-Verlag, New York, 1981.

[Kor80a] W. Korczynski, "Eine axiomatische Charakterisierung einer Algebra der Prozesse in nebenläufigen Systemen," IPI PAN Report Nr. 400, 1980. (Polnisch)

[Kor88a] W. Korczynski, "An algebraic characterization of Concurrent Systems," *Fundamenta Informaticae, North-Holland*, no. 11, pp. 171-194, 1988.

[Kra85a] B. Kraemer, "Stepwise Construction of Non-sequential Software Systems using a Net-based Specification Language," in *[ATPN85]*, 1985.

[Lug76a] H. Lugowski, *Grundzüge der Universellen Algebra*, Teubner Verlag, Leibzig, 1976.

[Mal76a] A.I. Malcev, in *The metamathematics of algebraic systems*, North Holland, 1976.

[Man76a] E.G. Manes, *Algebraic Theories*, Springer Verlag, Berlin, Heidelberg, New York, 1976.

[May80a] Ernst W. Mayr, "Ein Algorithmus für das allgemeine Erreichbarkeitsproblem bei Petrinetzen und damit zusammenhängende Probleme," in *Dissertation*, ed. Fakultät für Mathematik der TU München, München, 1980.

[Maz86a] A. Mazurkiewicz, "Trace Theory," in *[ACPN86]*, pp. Part II,279-324, 1986.

[Mem87a] G. Memmi and J. Vautherin, "Analysing Nets by the Invariant Method," in *Petri Nets: Central Models and Their Properties, Advances in Petri Nets 1986, Part I*, ed. G. Rozenberg, pp. 300-336, Springer Verlag, Heidelberg, 1987.

[Ras68a] H. Rasiowa and R. Sikorski, *The Mathematics of Metamethematics*, P.W.N., Warschau, 1968.

[Ras74a] H. Rasiowa, *An algebraic approach to nonclassical Logic,* North Holland, Amsterdam, 1974.

[Raz85a] R. R. Razouk and D. S. Hirschberg, "Tools for Efficient Analysis of Concurrent Software Systems," *Proc. SOFTAIR II (Software Development Tools, Techniques and Alternatives)*, San Francisco, Dezember 1985.

[Rei89a] Wolfgang Reisig, "Petri Nets and Abstract Data Types," Vorabdruck, 1989.

[Rei85a] W. Reisig, "Petri Nets with Individual Tokens," *THEOR. COMPUTER SCIENCE*, vol. 41, pp. 185-213, North-Holland, 1985.

[Rei86a] W. Reisig, *Petrinetze -- Eine Einfuehrung (2., ueberarb. u. erw. Aufl.)*, Springer-Verlag, 1986.

[Roz83a] G. Rozenberg and R. Verraedt, "Subset Languages of Petri Nets Part I: The Relationship to String Languages and Normal Forms," *Theoretical Computer Science*, vol. 26, pp. 301-326, 1983.

[Roz83b] G. Rozenberg and R. Verraedt, "Subset Languages of Petri Nets Part II: Closer Properties," *Theoretical Computer Science*, vol. 27, pp. 85-108, 1983.

[ATPN85] G. Rozenberg, (ed.), "4th and 5th European Workshop on Application and Theory of Petri Nets," *LNCS 188*, Springer-Verlag, 1985.

[Roz86a] G. Rozenberg, "Behaviour of Elementary Net Systems," in *[ACPN86]*, pp. Part I,60-94, 1986.

[Sal73a] A. Saloma, *Formal Languages,* Academic Press, New York, 1973.

[Sal70a] A. Salwicki, "Formalized algorithmic Languages," *Bull. Acad. Polon. Sci. Ser. Sci. Math. Astrono, Phys.*, vol. 18, Warschau, 1970.

[Sta81a] P.H. Starke, "Processes in Petri Nets," Lecture Notes in Computer Science Vol. 117, Springer-Verlag, Berlin, 1981.

[Sta85a] P.H. Starke, *Petri Netze,* Akademie Verlag, Berlin, 1985.

[Vau87a] J. Vautherin, "Parallel Systems Specifications with Coloured Petri Nets and Algebraic Specifications," in *Lecture Notes in Computer Science Vol 222: Advances in Petri Nets 1987*, ed. G. Rozenberg, pp. 293-308, Springer Verlag, Heidelberg, 1987.

[Win82a] J. Winkowski, "An Algebraic Description of System Behaviours," *Theor. Computer Science*, vol. 21, pp. 315-340, 1982.

[Win86a] G. Winskel, "Event Structures," in *[ACPN86]*, pp. Part II,325-392, 1986.