KfK 2635
Mai 1978

Beschreibung und Auswertung diskreter dynamischer Systeme

F. Schumacher
Institut für Datenverarbeitung in der Technik
Projekt Wiederaufarbeitung und Abfallbehandlung

Kernforschungszentrum Karlsruhe

KERNFORSCHUNGSZENTRUM KARLSRUHE

Institut für Datenverarbeitung in der Technik

Projekt Wiederaufarbeitung und Abfallbehandlung

KfK 2635

Beschreibung und Auswertung diskreter dynamischer Systeme

F. Schumacher

Kernforschungszentrum Karlsruhe GmbH, Karlsruhe

Kurzfassung

Ein Beschreibungs- und Auswertungsverfahren wird entwickelt, das für diskrete dynamische Systeme und damit für Hardware- und Realzeitsoftwaresysteme und technische Prozesse geeignet ist.

Das Beschreibungsverfahren basiert auf einer modifizierten Form von Petri-Netzen. Die Auswertungsmethode geht von Verfahren der Graphentheorie und der rechnergestützten Simulation aus. Die Anwendung der Methodik wird anhand mehrerer Beispiele gezeigt.

Modelling and evaluation of discrete dynamic systems

Abstract

The paper presents a modelling and evaluation method for discrete dynamic systems which can be used for the description and evaluation of hardware and real-time software systems and technical processes.

The modelling method is based upon modified Petri nets. The evaluation is founded upon graph theory and computer aided simulation. The application of the method to several problems is shown.

Inhaltsverzeichnis

1. Einleitung und Problemstellung

Ziel dieses Berichts ist die Entwicklung einer Beschreibungs- und Auswertungsmethode für diskrete dynamische Systeme. Ein dynamisches System wird meistens definiert als eine Menge von in Beziehung zueinander stehenden Elementen, deren Zustände sich im Laufe der Zeit ändern. Geschehen diese Zustandsänderungen zu diskreten Zeitpunkten, nennt man ein System diskret. Beispiele solcher Systeme sind Computer, Job-Shopsysteme, Datenbanken, Realzeitsoftwaresysteme oder Transport- und Lagerhaltungssysteme.

Eine Analyse der Struktur und des Verhaltens diskreter dynamischer Systeme ist wegen deren Komplexität im allgemeinen eine schwierige Aufgabe. Dies liegt vor allem daran, daß es an Verfahren zur übersichtlichen und umfassenden Darstellung und Analyse des Zusammenspiels aller Systemkomponenten mangelt.

Ein Gebiet, in dem dieses Problem besonders hervortritt, ist die Prozeßdatenverarbeitung. Hier gibt es auf der einen Seite den zu steuernden technischen Prozeß und auf der anderen Seite das Steuer- und Regelungssystem, welches aus Hardwarekomponenten und einer meist umfangreichen Realzeitsoftware besteht. In praktischen Anwendungen besitzt jedes dieser drei Teilsysteme für sich gesehen schon ein komplexes Verhalten. Das dynamische Verhalten des Gesamtsystems ist dann kaum noch überschaubar.

Zur Zeit existieren zwar viele Verfahren zur Analyse des Verhaltens diskreter dynamischer Systeme, nur sind sie meistens auf die Analyse eines speziellen Aspekts ausgerichtet. So sind z.B. Netzplantechniken wie CPM oder PERT geeignet zur Berechnung des kritischen Pfades, mit ihnen läßt sich jedoch nicht die Auslastung von Betriebsmitteln berechnen. Mit Warteschlangenmodellen kann man den Durchsatz eines Systems analysieren, jedoch nicht den Nachweis der Deadlockfreiheit erbringen. In unserem Beispiel der Prozeßdatenverarbeitung

kann man Verfahren der Regelungstechnik, Flußdiagramme oder Hardwarebeschreibungsmethoden anwenden. Alle diese Verfahren sind jedoch immer nur für eine Komponente des Gesamtsystems geeignet.

Eine Beschreibungs- und Analysemethode für diskrete dynamische Systeme muß wegen der Komplexität der zugrundeliegenden Systeme eine hierarchische Darstellung eines Systems in verschiedenen Detaillierungsebenen gestatten. Die Übersichtlichkeit erfordert eine graphische Darstellungsform, die u.a. eine Festlegung von Spezifikationen ermöglicht und die Kommunikation zwischen allen beteiligten Personen unterstützt. Die Analyse sollte das Entdecken und Beseitigen von Entwurfsfehlern erleichtern, die Einhaltung der Spezifikationen prüfen, Entscheidungshilfen bei der Festlegung von Systemparametern liefern und das Verständnis über das System vertiefen.

In diesem Bericht wird eine Modellierungs- und Auswertungsmethode entwickelt, die darauf beruht, ein dynamisches System nicht so sehr als eine Menge von in Beziehung zueinander stehenden Elementen, sondern vielmehr als eine Kollektion paralleler Prozesse anzusehen. Diese Betrachtungsweise wurde in der Informatik vor einigen Jahren entwickelt /Horn 73, Baer 73/. Unsere Methode, die von obigen Anforderungen ausgeht, basiert auf einer modifizierten Form von Petri-Netzen und ist eine Fortführung der Arbeiten von Noe und Mitarbeitern /Noe 71-73/. Die Auswertungsmethode geht von Verfahren der Graphentheorie und der rechnergestützten Simulation aus.

Der Bericht enthält neben einer Literaturübersicht (Kap. 2) und Definitionen verschiedener Begriffe diskreter dynamischer Systeme (Kap. 3) eine Beschreibung der Modellierungsmethode (Kap. 4) und der Auswertungsverfahren (Kap. 5). In Kapitel 6 wird die Anwendung des Konzepts anhand mehrerer Beispiele gezeigt. Der Bericht ist in einigen Teilen eine Zusammenfassung der vom Autor auf diesem Gebiet durchgeführten Arbeiten /Schu 76-77A-77B/.

2. Verwandte Arbeiten

2.1 Übersicht

Das Problem der Beschreibung und Analyse von diskreten dynamischen Systemen ist u.a. Untersuchungsgegenstand der allgemeinen Systemtheorie /Kalm 69, Coug 74, Zade 69/, des Operations Research /Neu 75/ und der Informatik /Bau 71, AdCS 72/.

Während die allgemeine Systemtheorie bezüglich kontinuierlicher Systeme einen hohen Entwicklungsstand erreicht hat, der auch für praktische Anwendungen relevant ist, gilt dies nicht für diskrete Systeme. Die allgemeine Systemtheorie verbleibt bei diskreten dynamischen Systemen auf einem hohen abstrakten Niveau, das in einem geringen Bezug zur Praxis steht. So ist z.B. die Automatentheorie als Teilgebiet der allgemeinen Systemtheorie /Kalm 69/ für die Beschreibung und Analyse komplexer Systeme wegen des umfangreichen Zustandsraums solcher Systeme nicht geeignet.

Im Bereich des Operations Research existiert eine Vielzahl von Beschreibungs- und Analyseverfahren für diskrete dynamische Systeme wie Verfahren der Warteschlangentheorie, Graphentheorie, Netzplantechnik, Simulation, Zuverlässigkeitstheorie, um nur die für diesen Bericht wichtigsten zu nennen. Die Verfahren haben in der Praxis eine breite Anwendung gefunden und sind erprobt, wie z.B. die Netzplantechniken CPM, PERT und GERT /Neu 75/.

Der Nachteil der Verfahren liegt darin, daß jede Methode, wie in der Einleitung schon bemerkt, nur einen speziellen Aspekt eines diskreten Systems erfaßt und nur zur Analyse bestimmter Systemgrößen geeignet ist. Mit CPM /Neu 75/ kann man z.B. den kritischen Pfad, aber nicht die Betriebsmittelbelegung eines Systems untersuchen.

Eine Ausnahme bildet GERT /Whit 73/, das zwar zur Beschreibung und Analyse von Netzplänen entwickelt wurde, jedoch inzwischen auch in anderen Bereichen (Warteschlangensimulation, Zuverlässigkeitsanalyse, Betriebsmittelbelegung) Anwendung gefunden hat.

In der Informatik gibt es seit einigen Jahren zwei für diesen Bericht interessante Entwicklungen: Zur Lösung von Koordinationsproblemen heutiger größerer Rechnersysteme wurde die Theorie der parallelen Prozesse entwickelt /Dijk 68, Horn 73, Baer 73/. Ein diskretes System wird hierbei nicht mehr in seinem Gesamtzustand, sondern als Kollektion paralleler und sequentieller Prozesse betrachtet, die eine Folge von Teilzuständen und Teilzustandsänderungen darstellen. Mit diesem Ansatz wird der oben erwähnte Nachteil der Automatentheorie bei komplexen Systemen überwunden. Die Theorie der parallelen Prozesse, speziell die der Petri-Netze /Petr 73/, bildet eine Grundlage dieses Berichts.

Die zweite für uns interessante Vorgehensweise ist unter dem Begriff Software Engineering bekannt geworden /AdCS 72, IEEE 77/. Ausgangspunkt ist hierbei das Problem zuverlässiger Software /Boe 73/ bei komplexen Anwendungen von Rechnersystemen. Der Ansatz beruht auf einer stärkeren Unterstützung der Entwurfs- und Entwicklungsphase eines Softwaresystems. Es werden Verfahren entwickelt, mit denen Spezifikationen beschrieben und deren Einhaltung geprüft werden können. Ein wesentlicher Gesichtspunkt ist hierbei die hierarchische Modellierung. Die aus der Literatur bekannten Verfahren wie PSA/PSL, SREM, SADT, LOGOS /Proc 76, IEEE 77, Rose 72/ sind erst seit kurzem im praktischen Einsatz, so daß bisher wenige Erfahrungen veröffentlicht sind.

Im folgenden wird auf Petri-Netze ausführlicher eingegangen, da diese die Grundlage der nachfolgenden Kapitel bilden.

2.2 Petri-Netze

Die von Petri /Petr 62/ entwickelten und von Holt /Holt 68/ im Bereich der Informatik verbreiteten Netze werden meistens definiert als ein Quintupel

PN = (P, TR, Z, Q, M), wobei gilt:

P ist die endliche Menge der Plätze (Stellen),

TR ist die endliche Menge der Transitionen,

$Z \subseteq P \times TR$, $Q \subseteq TR \times P$, $P \cup TR = Feld(Z \cup Q)$,

p ε P heißt Eingangsplatz von tr ε TR
tr ε TR heißt Ausgangstransition von p ε P } wenn $(p, tr) \in Z$,

p ε P heißt Ausgangsplatz von tr ε TR
tr ε TR heißt Eingangstransition von p ε P } wenn $(tr, p) \in Q$

M: $P \to N \{0\}$ ist eine Markierung. M(p) ist die Anzahl der Marken auf p ε P. p ε P heißt markiert, wenn $M(p) \geq 1$. M_0 ist die Anfangsmarkierung.

Folgende Schreibweise wird eingeführt:

Seien tr ε TR und p ε P,

$\cdot tr := \{p \in P \mid (p, tr) \in Z\}$

$tr\cdot := \{p \in P \mid (tr, p) \in Q\}$

$\cdot p := \{tr \in TR \mid (tr, p) \in Q\}$

$p\cdot := \{tr \in TR \mid (p, tr) \in Z\}$

Eine Transition tr ε TR heißt aktiviert unter einer Markierung M, wenn für alle p ε ·tr $M(p) \geq 1$.

Eine Transition tr ε TR kann geschaltet (gefeuert) werden unter einer Markierung M, wenn tr aktiviert ist. Schalten (Feuern) einer Transition heißt, von jedem Platz p ε·tr eine Marke entfernen und zu jedem Platz p ε tr· eine Marke addieren:

$p \in {}^{\bullet}tr \quad M'(p) := M(p) - 1$

$p \in tr^{\bullet} \quad M'(p) := M(p) + 1$

mit der Markierung M', die aus dem Schalten von tr aus M hervorgeht:

$$M \overset{tr}{=>} M'.$$

Graphisch werden Plätze durch Kreise, Transitionen durch Rechtecke und Marken durch Punkte dargestellt. Bild 1 zeigt ein Beispiel eines Petri-Netzes.

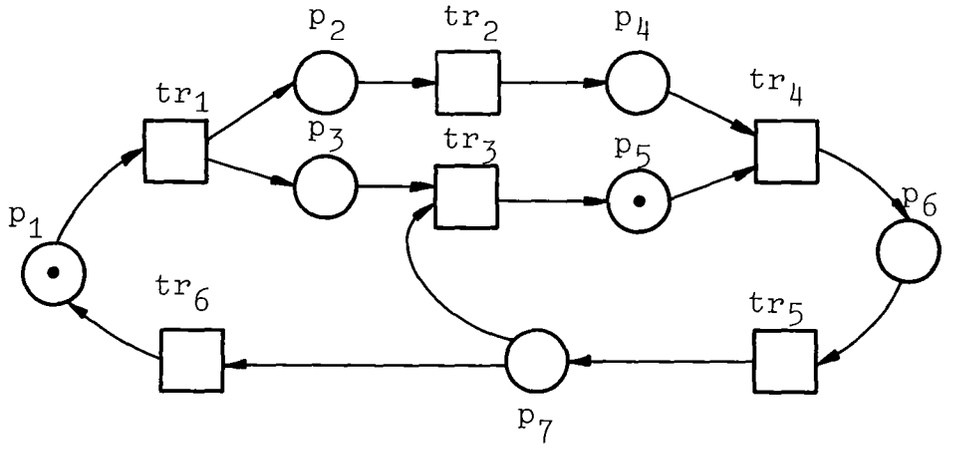


Bild 1: Beispiel eines Petri-Netzes

Folgende Definitionen beziehen sich auf Petri-Netze:

Ein Platz $p \in P$ heißt Nebenbedingung der Transition $tr \in TR$, wenn $(p, tr) \in Z$ und $(tr, p) \in Q$, d.h. p ist Ein- und Ausgangsplatz derselben Transition.

Zwei Transitionen $tr_1, tr_2 \in TR$ stehen in einem Konflikt miteinander, wenn

${}^{\bullet}tr_1 \cap {}^{\bullet}tr_2 \neq \emptyset$ (Vorwärtskonflikt),

$tr_1^{\bullet} \cap tr_2^{\bullet} \neq \emptyset$ (Rückwärtskonflikt).

Der Konflikt ist ein grundlegendes strukturelles Element von Petri-Netzen. Die Auflösung eines Konfliktfalls ist in Petri-Netzen nicht festgelegt.

M' heißt Folgemarkierung der Markierung M, wenn es eine Reihe von Transitionen $tr_1, tr_2, \ldots, tr_n$ und eine Reihe von Markierungen $M_1, M_2, \ldots, M_m$ gibt, so daß

$$M = M_1 \overset{tr_1}{\Rightarrow} M_2 \overset{tr_2}{\Rightarrow} \ldots \overset{tr_n}{\Rightarrow} M_m = M'.$$

Die Menge der Folgemarkierungen von M, einschließlich M, wird mit [M] bezeichnet.

Von zentralem Interesse sind bei Petri-Netzen die Begriffe Lebendigkeit und Beschränktheit:

Sei M eine Markierung und tr ε TR eine Transition,

tr heißt tot unter M, wenn tr durch M nicht aktivierbar ist.

M heißt tot, wenn alle Transitionen unter M tot sind.

Eine Markierung M heißt M-lebendig, wenn es keine tote Folgemarkierung von M gibt.

Eine Markierung M heißt tr-lebendig, wenn es keine tote Transitionen unter den Folgemarkierungen von M gibt.

Ein Platz p ε P heißt k-beschränkt unter einer Markierung M, wenn für alle M' ε [M]: $M'(p) \leq k$. Statt 1-beschränkt, sagt man auch sicher.

Lautenbach /Laut 75/ führte neben den beiden obigen in der Literatur gebräuchlichen Lebendigkeitsdefinitionen noch weitere ein. Albrecht /Alb 77/ entwickelte eine Systematik aller möglichen Lebendigkeitsbegriffe. Die Lebendigkeit ist eine grundlegende Eigenschaft bei der Deadlockerkennung.

Petri-Netze eignen sich zur Beschreibung und Analyse der Kontrollstruktur paralleler Prozesse. Sie stehen dabei in Konkurrenz zu anderen, in den letzten zehn Jahren entwickelten Verfahren, wie Semaphore /Dijk 68/, Parallele Programmschemata /Karp 69/, Vector-Addition-Systeme /Karp 69/, Vector-Replacement-Systeme /Kell 72/, UCLA-Graphenmodelle /Cerf 72/ usw,

auf die hier nicht näher eingegangen wird (s. etwa /Baer 73/). Peterson /Pet 74/ und Miller /Mill 73-74/ zeigten, daß Petri-Netze bezüglich bestimmter Eigenschaften zu diesen Verfahren - mit Ausnahme der Parallelen Programmschemata - äquivalent sind oder diese als Teilmenge enthalten.

Petri-Netze sind in ihrem Anwendungsbereich begrenzt. Kosaraju /Kosa 73/ zeigte ein Koordinierungsproblem, welches nicht mit Petri-Netzen modellierbar ist. Die Haupteinschränkung resultiert bei Petri-Netzen daraus, daß das Nichtvorhandensein von Marken auf einem Platz nicht getestet werden kann. Deswegen kann man mit einem Petri-Netz keine Turing-Maschine simulieren /Kell 72, Hack 75/.

Zur Beschreibung und Analyse des dynamischen Ablaufs eines Petri-Netzes wurden von Holt /Holt 68/ bzw. Crowley /Crow 76/ sogenannte O-Graphen bzw. R-Graphen eingeführt. Mit diesen gerichteten Graphen lassen sich Markierungsfolgen darstellen. In einem R-Graph z.B. entspricht jeder Knoten einer Markierung und jede Kante einer Transition, die eine Markierung (Anfangsknoten) in deren Folgemarkierung (Endknoten) überführt.

Bezüglich weiterer Begriffe und Eigenschaften von Petri-Netzen sei auf den ersten Übersichtsartikel über Petri-Netze /Pet 77/ hingewiesen.

In den vergangenen Jahren wurden Petri-Netze von verschiedenen Autoren erweitert oder modifiziert. Patil /Pat 70/ führte sogenannte Constraint Sets CS ein, mit CS ε 2^P und der Bedingung, daß verschiedene Plätze einer Constraint Set nicht gleichzeitig markiert sein dürfen. Mit dieser Erweiterung /Pet 73/ lassen sich komplexe Petri-Netzmodelle einfacher darstellen.

Mehrere Autoren /Kell 72, Hack 74, Pet 73, Lien 76/ modifizierten die Aktivierungs- und Schaltregel durch Einführung einer Pfeilbreite: Die Transition in Bild 2 ist aktiviert, wenn p_1 drei Marken besitzt. Beim Feuern der Transition werden drei Marken von p_1 abgezogen und zwei Marken auf p_2 gesetzt. Hack /Hack 74/ zeigte, daß ein Petri-Netz mit Pfeilbreiten durch Einführung geeigneter Transitionen und Plätze in ein gewöhnliches Petri-Netz umgeformt werden kann.

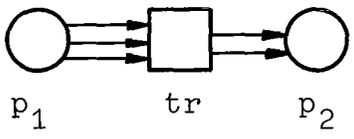


Bild 2: Modifizierung der Aktivierungs- und Schaltdefinition durch Pfeilbreiten

Noe und Nutt /Noe 71-75, Nutt 72/, Crowley /Crow 76/, Ramchandani /Ram 74/, Lockemann /Lock 74/ und Mayr /Mayr 76/ modifizierten Petri-Netze, um sie bei praktischen Anwendungen wie Modellierung von Rechnersystemen (Noe) oder Informationssystemen (Mayr, Lockemann) benutzen zu können. Neben anderem wurden eingeführt: Markenattribute; Transitionsprozeduren zur Änderung von Markenattributen; globale Variable, auf die alle Prozeduren Zugriff haben; Platzkapazitäten für maximale Markenanzahl eines Platzes; Transitionszeiten; Auswahltransitionen zur Modellierung von Entscheidungen; Schnittstellen, die die Passierfähigkeit von Marken einschränken. Die zur Zeit am meisten verbreitete Modifizierung von Petri-Netzen sind die Evaluation-Netze von Noe und Nutt /Nutt 72/.

Auf die Evaluation-Netze (E-Netze) wird im folgenden noch häufiger eingegangen. Da eine Beschreibung der E-Netze den Rahmen dieser Übersicht sprechen würde, wird auf die entsprechende Literatur verwiesen /Noe 72-73, Nutt 72/.

3. Diskrete dynamische Systeme

Ein System wird i.a. definiert als eine Menge von Elementen, zwischen denen Relationen bestehen. Ändern sich die Beziehungen der Elemente untereinander mit der Zeit, so nennt man ein System dynamisch. Geschehen diese Änderungen nur zu bestimmten Zeitpunkten, heißt ein System diskret.

Diese Definition eines diskreten dynamischen Systems (ddSystem) ist in der Literatur üblich /Kalm 69, Zade 69, Coug 74/. Weitere Begriffe, wie offenes, geschlossenes System, deterministisches, stochastisches System, Systemumwelt, Attribut, Werte von Attributen, Systemzustand, Zustandsänderung, werden in ähnlicher Weise definiert /Kalm 69, Coug 74/. Die Definitionen sind abstrakt und dienen hauptsächlich der Klassifizierung von Systemen. In diesem Bericht werden sie als bekannt vorausgesetzt.

Im folgenden werden, ausgehend von der Definition des Teilzustands und der Teilzustandsänderung, Begriffe und Eigenschaften von ddSystemen definiert, die direkt auf reale Systeme übertragbar sind und als Grundlage eines Beschreibungs- und Analyseverfahrens für solche Systeme dienen sollen. Die Definitionen lehnen sich an Arbeiten aus der Informatik über parallele Prozesse /Baer 73, Horn 73, Deus 74, Zima 76/ an.

Die Gesamtanzahl der Attribute eines Systems zum Zeitpunkt t aus der abzählbaren Zeitmenge T sei mit m(t), die Attribute seien mit $a_1(t)$, ..., $a_m(t)$ und deren Werte mit $w_1(t)$, ..., $w_m(t)$ bezeichnet.

Ein Teilzustand TZ(t) eines Systems S ist eine Zuordnung von Werten zu einer Teilmenge der Attribute des Systems S zum Zeitpunkt t:

$$TZ(t) : (a_1(t), \ldots, a_\ell(t)) \rightarrow (w_1(t), \ldots, w_\ell(t)) \text{ mit } \ell \leq m(t).$$

Ein Übergang U(t) ist eine Zustandsänderung eines Teilzustands $TZ_1(t_-)$ in einen Teilzustand $TZ_2(t_+)$ zum Zeitpunkt t. Der

Teilzustand $TZ_1(t_-)$ bzw. $TZ_2(t_+)$ heißt Eingangs- bzw. Ausgangszustand des Übergangs U(t). Die Menge der Attribute des Eingangs- bzw. Ausgangszustandes werden im folgenden mit Eingangsbereich TZE(t) und Ausgangsbereich TZA(t) bezeichnet.

Die Definition des Übergangs besagt: Befindet sich ein System in einem Zustand $Z(t_-)$, der einen Teilzustand $TZ_1(t_-)$ enthält, und ist $TZ_1(t_-)$ Eingangszustand eines Übergangs U(t), dann führt U(t) das System in einen anderen Zustand über, d.h. er schaltet. In der Literatur nennt man einen Übergang auch Ereignis.

Der Übergang benötigt keine Zeit. Die Änderung geschieht zu einem festen Zeitpunkt. Das System ist somit diskret.

Im Gegensatz zur Automatentheorie und zur Literatur über parallele Prozesse wird nicht differenziert in Input und Output oder Kontrollfluß und Datenfluß. Ein Übergang benötigt keine Anregung von außen, sondern schaltet, wenn der Eingangszustand erfüllt ist. Es gibt somit keine anderen agierenden Elemente wie bei Horning oder Zima /Horn 73, Zima 76/.

Nach obiger Definition können Übergänge gleichzeitig oder nacheinander schalten und sich gegenseitig beeinflussen:

Zwei Übergänge $U_1(t)$ und $U_2(t)$ heißen abhängig, wenn sie gleichzeitig schalten können und $(TZE_1(t) \cup TZA_1(t)) \cap (TZE_2(t) \cup TZA_2(t)) \neq 0$. Können sie nicht gleichzeitig schalten oder sind ihre Ein- und Ausgangsbereiche disjunkt, so heißen die beiden Übergänge unabhängig.

Zwei Übergänge können in folgenden Bereichen abhängig sein:

a) Überschneidung Eingangsbereich_1 - Eingangsbereich_2,

b) Überschneidung Ausgangsbereich_1 - Ausgangsbereich_2,

c) Überschneidung Eingangsbereich_1 - Ausgangsbereich_2, bzw. Ausgangsbereich_1 - Eingangsbereich_2,

d) Kombination aus a), b), c).

Der Fall b) - in der Literatur häufig als Konfliktfall bezeichnet - verursacht bei ddSystemen Schwierigkeiten, wie z.B. Race-Probleme bei integrierten Schaltungen.

Ein ddSystem heißt 1-indeterministisch, wenn es mindestens zwei nach Fall a) oder b) abhängige Übergänge enthält. Die 1-Indeterminiertheit entspricht dem Vorwärts- und Rückwärtskonflikt bei Petri-Netzen.

Man kann folgende Übergangsarten unterscheiden:

Zwei Übergänge $U_1(t)$ und $U_2(t)$ heißen parallel, wenn sie gleichzeitig schalten können und ihre Ein-/Ausgangsbereiche disjunkt sind.

Zwei Übergänge $U_1(t)$ und $U_2(t)$ heißen seriell, wenn von obigen Fällen c) gilt.

Ein Übergang U(t) heißt vereinigend, wenn der Eingangszustand von U(t) gleich der Vereinigung der Ausgangszustände mehrerer Übergänge U_1, ... U_k ($k > 1$) ist, deren Ausgangszustände nicht alle gleich sind.

Ein Übergang U(t) heißt verzweigend, wenn der Ausgangszustand von U(t) gleich der Vereinigung der Eingangszustände mehrere Übergänge U_1, ..., U_k ($k > 1$) ist, deren Eingangszustände nicht alle gleich sind.

Vereinigende und verzweigende Übergänge sind Grundtypen in ddSystemen und werden in der englischsprachigen Literatur als Join- und Fork-Übergänge /Baer 73/ bezeichnet.

Nach obiger Definition führt ein Übergang einen Teilzustand in einen anderen Teilzustand über. Wir wollen diese Definition auf mehrere Eingangs- und mehrere Ausgangszustände erweitern: Sei EZ(t) bzw. AZ(t) eine Menge von Eingangs- bzw. Ausgangszuständen, dann ist ein Übergang U(t) eine partielle Abbildung $U(t)$: $EZ(t) \rightarrow AZ(t)$.

In realen Systemen tritt oft der Fall auf, daß bei einem Übergang für einen Eingangszustand mehrere Ausgangszustände definiert sind und einer ausgewählt wird. Man spricht auch von einer Entscheidung, Auswahl oder Selektion.

Ein Übergang heißt Auswahlübergang, wenn für einen Eingangszustand mehrere Ausgangszustände definiert sind und einer ausgewählt wird. Die Auswahl kann zufällig oder deterministisch sein.

Ein ddSystem mit mindestens einem Auswahlübergang mit zufälliger Auswahl heißt 2-indeterministisch.

Die 2-Indeterminiertheit entspricht dem Vorwärtskonflikt bei Petri-Netzen und der Indeterminiertheit in der Automatentheorie.

Die Anzahl möglicher Ausgangszustände kann bei einem Auswahlübergang konstant oder zeitabhängig sein. Bei einem Mehrprozessorsystem mit einer Jobwarteschlange vor den Prozessoren besitzt die Auswahl eines Jobs für einen freien Prozessor eine zeitabhängige Anzahl an Ausgangszuständen, während die Auswahl eines Prozessors für einen Job eine konstante Anzahl besitzt.

Im folgenden werden zwei für ddSysteme wichtige Begriffe, Dauer und Prozeß, eingeführt:

Seien zwei Übergänge $U_1(t_1)$ und $U_2(t_2)$ gegeben und $t_2 \geq t_1$, dann ist eine Dauer $d(t_1, t_2)$ die Differenz zwischen t_2 und t_1:

$$d(t_1, t_2) := t_2 - t_1$$

Die Dauer kennzeichnet also den zeitlichen Abstand zwischen zwei Übergängen.

Im folgenden seien d_i eine bestimmte Dauer und U_j ein bestimmter Übergang. Mit den Begriffen Dauer und Übergang läßt sich der Begriff Prozeß definieren:

Seien U_i mit i=1(1)k Übergänge und d_j mit j=1(1)ℓ Dauern, dann ist ein Prozeß PR eine Folge von Übergängen U_i und Dauern d_j.

Ein Prozeß sei z.B. veranschaulicht durch

$$PR: \quad U_1 \rightarrow U_2 \rightarrow U_3 \rightarrow d_2 \rightarrow d_3 \rightarrow \ \ldots \ \rightarrow U_k \rightarrow d_\ell .$$

Wie aus diesem Beispiel ersichtlich, können in einem Prozeß Teilfolgen von Dauern ohne Übergänge und Teilfolgen von Übergängen ohne Dauern auftreten. Wenn diese Dauern bzw. Übergänge eines Prozesses keinen Kontakt zu Dauern bzw. Übergängen anderer Prozesse im System haben, lassen sie sich zu einer Dauer bzw. einem Übergang zusammenfassen. Ein Prozeß kann mit einer Dauer oder einem Übergang beginnen bzw. enden.

In der Literatur wird ein Prozeß als eine Folge von Zustandsänderungen definiert /Horn 73, Deus 74/. Auf eine Dauer oder ähnliches wird dabei nicht eingegangen, obwohl dieser Begriff für die Beschreibung und Auswertung realer ddSysteme eine wichtige Rolle spielt (s.u.).

Zwei Prozesse können, analog zu den Übergängen, seriell oder parallel verlaufen, in einen anderen Prozeß übergehen oder aus einem anderen Prozeß hervorgehen, usw. Welche Folge von Übergängen und Dauern welchem Prozeß entsprechen, ist nicht festgelegt, sondern eine Frage des Betrachters.

In der Literatur wird häufig ein Spezialfall des Zusammenwirkens von Prozessen, die Koordination, behandelt.

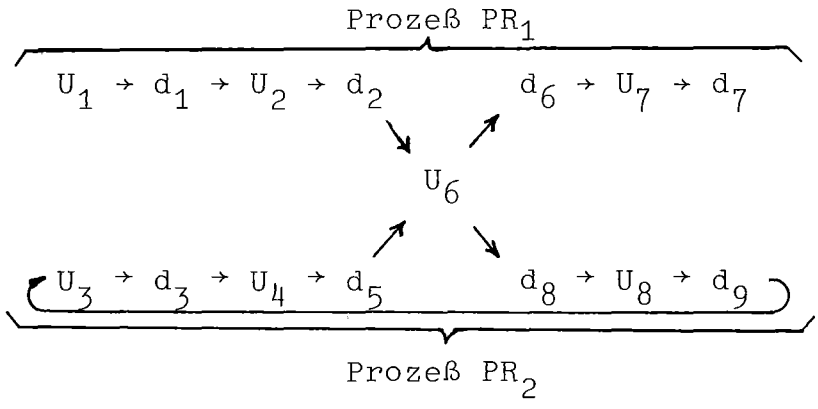


Bild 3: Koordination von Prozeß PR_1 und PR_2 durch Übergang U_6

In Bild 3 ist U_6 als koordinierender Übergang sowohl ein vereinigender als auch verzweigender Übergang. Der Prozeß PR_2 besitzt eine Kreisstruktur.

Ein Prozeß heißt zyklisch, wenn die Folge der Übergänge und Dauern einen Zyklus bildet, ansonsten heißt ein Prozeß linear.

Bisher war eine Dauer bestimmt durch den Anfangs- und Endzeitpunkt t_1 und t_2. Betrachtet man die Dauern in realen ddSystemen, so kann man zwei Fälle unterscheiden: Der Zeitpunkt t_2 ist zum Zeitpunkt t_1 innerhalb des Prozesses bekannt oder nicht. Ein Beispiel für den ersten Fall ist eine konstante Bedienungszeit eines Auftrags, während der zweite Fall bei Wartezeiten vorliegt. Häufig ist die Bedienungszeit eines Auftrags, etwa bei Batchsystemen, zu Beginn der Ausführung nicht bekannt und eine Zufallsgröße, womit wieder der zweite Fall vorliegt.

Die Wartezeit und die zufallsverteilte Bedienungszeit unterscheiden sich darin, daß bei der Wartezeit der Zeitpunkt t_2 von außerhalb des Prozesses befindlichen Übergängen, z.B. Ende der Bedienung des vorhergehenden Auftrags, bestimmt wird, während bei der zufallsverteilten Bedienungszeit kein äußerer Einfluß vorliegt.

Damit erhält man eine weitere Art der Indeterminiertheit:

Ein ddSystem mit mindestens einer zufallsverteilten Dauer heißt 3-indeterministisch. In der Literatur nennt man ein solches System auch häufig stochastisch.

Betrachtet man bei der Koordination von zwei Prozessen in Bild 3 die Dauern d_2 und d_5, so erkennt man eine Abhängigkeit im Endzeitpunkt der beiden Dauern. Schalten die beiden Übergänge U_2 und U_4 im gleichen Zeitpunkt t_i und sind die Dauern d_2 und d_5 zu diesem Zeitpunkt fest vorgegeben, aber ungleich, so schaltet U_6 im Zeitpunkt

$$t_{i+1} = t_i + \max(d_2, d_5).$$

Bei ddSystemen tritt häufig der Fall auf, daß z.B. d_5 zum Zeitpunkt t_i der Wert Null vorgegeben wird, dieser aber durch Prozeß PR_1 geändert wird. Diese Situation liegt z.B. bei einem Wartevorgang vor.

Damit kann man bei einem Prozeß vier verschiedene Phasen unterscheiden:

Ein Prozeß PR mit den Dauern $d_1, \ldots, d_\ell$ und den Übergängen $U_1, \ldots, U_k$ befindet sich zum Zeitpunkt t:

a) in einer aktiven Phase, wenn in diesem Zeitpunkt ein Übergang schaltet;

b) in einer passiven Phase, wenn zum Zeitpunkt t eine Dauer vorliegt;

b1) in einer wartenden Phase, wenn der Prozeß PR zum Zeitpunkt t in einer passiven Phase ist und t größer als die Summe aus dem Schaltzeitpunkt des vorhergehenden Übergangs und der bei diesem Übergang vorgegebenen Dauer ist;

b2) in einer suspendierten Phase, wenn der Prozeß PR zum Zeitpunkt t in einer passiven Phase ist und t kleiner gleich der Summe aus dem Schaltzeitpunkt des vorhergehenden Übergangs und der bei diesem Übergang vorgegebenen Dauer ist.

Bei einem Prozeß sind folgende Phasenübergänge möglich:

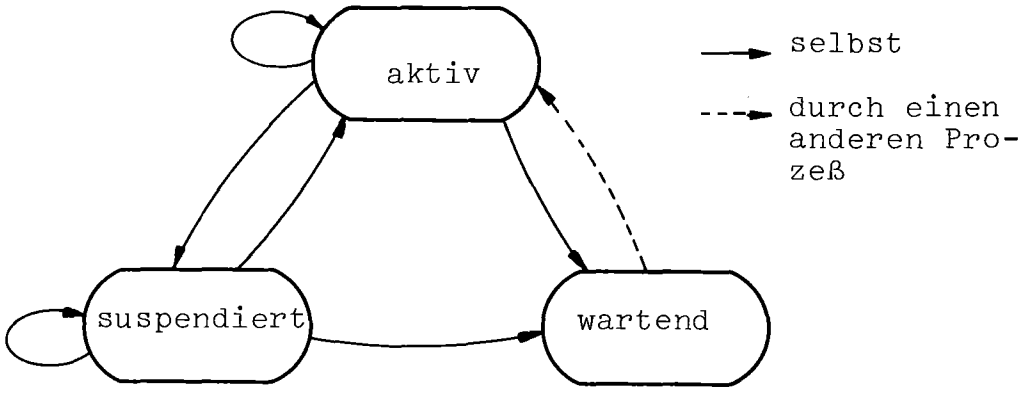


Bild 4: Phasenübergänge eines Prozesses

Obige Definitionen bilden eine Begriffsgrundlage für die nachfolgenden Kapitel. Sie gehen über die in der Literatur gebräuchlichen Begriffe hinaus, dadurch daß sie sich auf reale Systeme beziehen und einen Zeitparameter enthalten. Im folgenden wird ein Modellierungsverfahren für ddSysteme beschrieben.

4. Beschreibung diskreter dynamischer Systeme mit Simulationsnetzen

In diesem Kapitel wird ein auf Petri-Netzen basierendes Modellierungs- und Auswertungsverfahren, Simulationsnetz oder kurz S-Netz genannt, definiert. Es soll vor allem folgenden Anforderungen genügen:

- graphische Struktur,
- geeignet zur hierarchischen Modellierung,
- leicht verständlich und einfach benutzbar,
- geeignet für möglichst viele Anwendungen,
- möglichst geringer Modellierungszwang durch Beschreibungsmethode,
- einfache und umfassende Auswertungsmöglichkeiten.

Neben diesen sechs Anforderungen gibt es noch weitere wie z.B. Änderungsfreundlichkeit und Übersichtlichkeit. Diese sind jedoch entweder in obigen sechs Anforderungen enthalten oder von geringerer Priorität.

4.1 Informelle Einführung

S-Netze sind modifizierte Petri-Netze und enthalten mehrere Erweiterungen und Modifizierungen, die aus der Literatur bekannt sind (s. Kapitel 2). Vier Modifizierungen zeichnen S-Netze gegenüber Petri-Netzen und den bekannten Änderungen besonders aus:

a) Zuordnung von Attributen:

In S-Netzen können sowohl Marken als auch Plätze Attribute besitzen. Die Art der Attribute ist von der jeweiligen Anwendung abhängig.

Nach den Ausführungen in Kapitel 3 kann die Zuordnung der Attribute folgendermaßen interpretiert werden: Der Zustand eines ddSystems entspricht im S-Netz den Werten der Marken- und Platzattribute, wobei letztere sowohl die Markenanzahl pro Platz als auch die Verbindungsstruktur zu

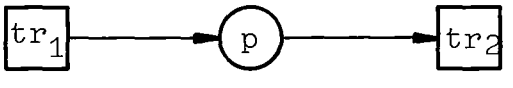
den Transitionen beinhalten. Eine Zustandsänderung wird durch das Schalten einer Transition dargestellt. Marken und Markenattribute sind temporäre Elemente, Plätze und Platzattribute permanente Elemente (s. SIMSCRIPT /Neu 75/).

Aus dieser Interpretation folgt, daß die Anzahl der Plätze und Transitionen sowie die Verbindungsstruktur in einem S-Netz während der gesamten Systemzeit unverändert bleiben. Nur Marken und Markenparameter können erzeugt oder vernichtet werden.

b) Markenverzögerung durch Platzzeit:

Petri-Netze beinhalten keine explizite Zeitstruktur und sind somit nur beschränkt zur Modellierung und Auswertung von ddSystemen geeignet.

Im Gegensatz zu den von Nutt /Nutt 72/ und Ramchandani /Ram 74/ eingeführten Transitionszeiten sind in S-Netzen den Plätzen Zeiten zugeordnet, d.h. hat ein Platz p die Platzzeit t_p, so wird eine Marke, die auf den Platz p gelangt, t_p Zeiteinheiten verzögert.



$t_p = 5$

In obigem Bild kann Transition tr_2 genau 5 Zeiteinheiten nach Feuern von tr_1 schalten.

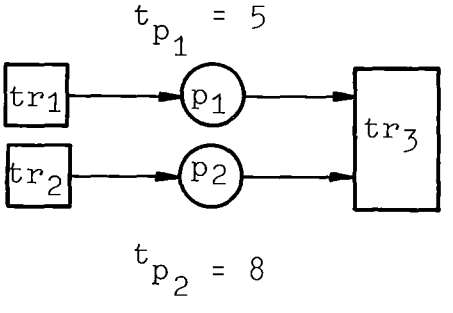
Der Grund für die Einführung von Platzzeiten liegt im wesentlichen in der Interpretation von S-Netzen: Eine Transition in S-Netzen entspricht einem Übergang in ddSystemen. Das Schalten einer Transition ist somit gleich einem Ereignis. Die Platzzeit wird interpretiert als Dauer (weitere Gründe für die Platzzeit s. Kap. 4.4).

In Verbindung mit der Platzzeit wird in S-Netzen die Definition der Aktivierung einer Transition in Petri-Netzen

modifiziert:

Gelangt eine Marke auf einen Platz p mit der Platzzeit t_p und ist der Platz p Eingangsplatz der Transition tr, dann wird nach t_p Zeiteinheiten geprüft, ob tr im Sinne der Petri-Netzdefinition aktiviert ist. Ist letzteres der Fall, so schaltet tr.

Diese Definition der Ablauffolge in S-Netzen sei durch ein Beispiel veranschaulicht:

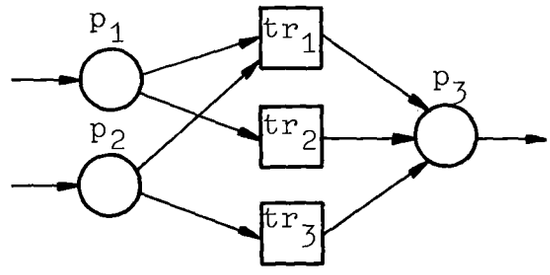


Ist zu einem Zeitpunkt t keine Marke auf Platz p_1 und p_2 und schalten tr_1 und tr_2 gleichzeitig zu diesem Zeitpunkt, dann schaltet tr_3 nach obiger Definition genau 5 Zeiteinheiten später, d.h. die Platzzeit der Marke auf Platz p_2 wird nicht beendet, sondern abgebrochen. Es ist also die Modellierung von Unterbrechungen möglich.

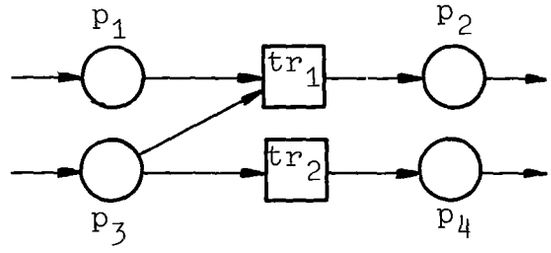
c) Auflösung von Konfliktfällen:

In S-Netzen wird die Einschränkung gegenüber Petri-Netzen eingeführt, daß Konflikte, ausgehend von einer Indizierung der Transitionen, nach aufsteigendem Transitionsindex aufgelöst werden, d.h. in dieser Reihenfolge wird versucht zu schalten. Wird keine explizite Reihenfolge gewünscht, so sind die entsprechenden Transitionen mit dem gleichen Index zu versehen und die Auflösung des Konfliktes erfolgt zufällig.

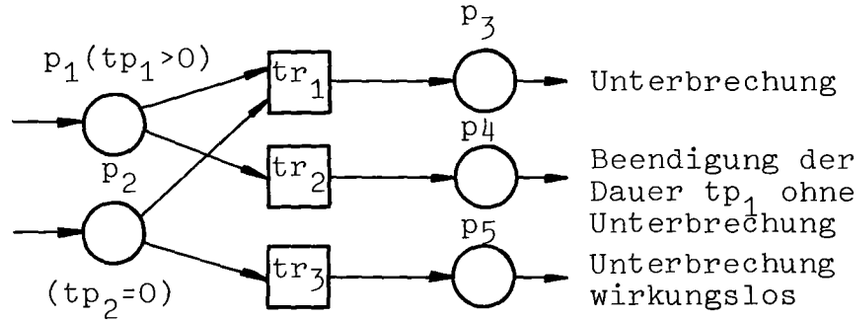
Die Einschränkung der Auflösung von Konfliktfällen wurde eingeführt, um spezielle logische Verknüpfungen modellieren zu können. Bild 5 a zeigt als Beispiel ein inklusives Oder und Bild 5 b ein logisches Nicht, wobei p_4 die Negation von p_1 ist und p_3 die Negation steuert. In Bild 5 c ist ein Unterbrechungsmodell dargestellt.



a) inklusives Oder



b) logisches Nicht

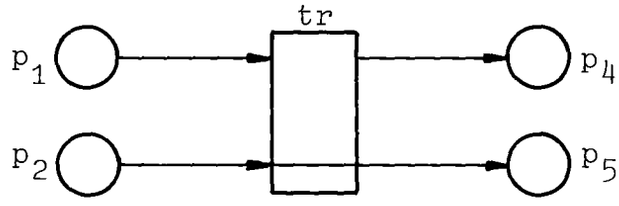


c) Unterbrechungsmodell

Bild 5: Beispiele für Konfliktauflösung

d) Erweiterung der Aktivierungs- und Schaltdefinition:

Beim Schalten einer Transition in Petri-Netzen werden Marken von den Eingangsplätzen abgezogen, d.h. zerstört, und Marken auf die Ausgangsplätze gesetzt, d.h. neu generiert. Diese Petri-Netzeigenschaft ist für die Modellierung realer ddSysteme nicht geeignet, wenn z.B. Marken als temporäre Elemente vorgegebene Wege im Netz durchlaufen sollen. Daher wurde in S-Netzen eine explizite Zuordnung der Eingangsplätze zu den Ausgangsplätzen einer Transition eingeführt. Der Weg einer Marke durch ein Netz kann damit festgelegt werden.



In obigem Bild wird beim Schalten von tr die Marke von p_2 auf p_5 gesetzt, die Marke von p_1 zerstört und eine neue Marke auf p_4 gesetzt.

In S-Netzen wird die Aktivierung in Analogie zur Pfeilbreite (s. Kapitel 2.2) allgemeiner formuliert:

Jedem Platz p werden zwei natürliche Zahlen A(p) und MI(p) mit $MI(p) \leq A(p)$ zugeordnet. Ist die Markenzahl auf jedem Eingangsplatz p einer Transition größer gleich A(p), so ist die Transition aktiviert. Beim Schalten werden dann MI(p) Marken von jedem Eingangsplatz p abgezogen.

Ist für einen Platz p beim Schalten die aktuelle Markenanzahl auf Platz p größer als MI(p), so müssen Marken ausgewählt werden. Dies kann nach einem vom S-Netzanwender festzulegenden Auswahlverfahren wie z.B. FIFO, LIFO oder Random erfolgen.

Eine der Markenauswahl entsprechende Platzauswahl ist in S-Netzen für die Ausgangsplätze einer Transition definiert: Bei einer Transition kann ein Eingangsplatz mit mehreren Ausgangsplätzen verbunden sein. Schaltet die Transition, so wird nach einem Auswahlverfahren, wie z.B. Random, zyklische Reihenfolge oder kleinste Platzpriorität, ein Ausgangsplatz bestimmt, der dann die Marken des Eingangsplatzes erhält. Diese Erweiterung wurde eingeführt, um Entscheidungen, wie z.B. if x then A else B, modellieren zu können.

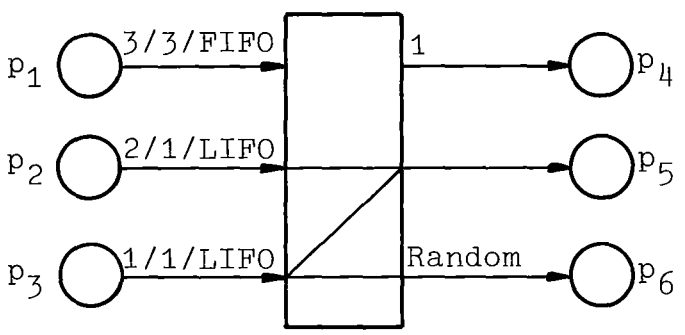


Bild 6: Beispiel einer S-Netztransition

In Bild 6 wird die Transition tr aktiviert, wenn $A(p_1) = 3$, $A(p_2) = 2$ und $A(p_3) = 1$. Beim Schalten werden vom Platz $p_1/p_2/p_3$ 3/1/1 Marken abgezogen. Die Markenauswahl erfolgt nach FIFO bzw. LIFO. Platz p_4 erhält eine neue Marke und Platz p_5 eine Marke von Platz p_2. Für die von p_3 abgezogene Marke wird p_5 oder p_6 zufällig ausgewählt. Die drei Marken von Platz p_1 verschwinden.

Im folgenden werden die Standardwerte A(p) = 1, MI(p) = 1, Markenauswahl FIFO, Platzauswahl Random nicht mehr in der graphischen Darstellung angegeben.

Weiterhin besitzen, wie bei Noe und Nutt /Noe 75, Nutt 72/, die Transitionen Transitionsprozeduren, mit denen Attribute von Marken und Plätzen, die jedoch direkt mit der Transition verbunden sein müssen, verändert werden können.

4.2 Definition eines Simulationsnetzes

Ein Simulationsnetz (S-Netz) SN ist ein 8-Tupel

$SN = (PN, T, ATTR, n_p, n_{ma}, t_p, A, S)$,

wobei gilt

$PN = (P, TR, Z, Q, M)$ ist ein Petri-Netz (siehe Kapitel 2.2), wobei TR indiziert sei.

$T \varepsilon \mathbb{R}^+$ ist eine geordnete Zeitmenge,

$t_N \varepsilon T$ ist die Netzzeit,

ATTR ist die Menge der Attribute,

$n_p \varepsilon \mathbb{N}$ Attribute sind jedem Platz $p \varepsilon P$ zugeordnet,

Notation: Die Menge der Marken in einem S-Netz zum Zeitpunkt $t \varepsilon T$ wird mit MA_t bezeichnet. $MA_t(p)$ ist die Menge der Marken auf dem Platz $p \varepsilon P$ zum Zeitpunkt t.

$n_{ma} \varepsilon \mathbb{N}$ Attribute sind jeder Marke $ma \varepsilon MA_t$ zugeordnet,

$t_p: P \times MA_t \to \mathbb{R}^+$ $t_p(p, ma)$ heißt die Platzzeit der Marke $ma \varepsilon MA_t(p)$ auf dem Platz $p \varepsilon P$.

$A: Z \to \mathbb{N}$ heißt Aktivierung. $A(p, tr)$ mit $(p, tr) \varepsilon Z$ ist die Aktivierungsanzahl des Platzes $p \varepsilon P$ bezüglich der Ausgangstransition $tr \varepsilon TR$,

Jedem $tr \varepsilon TR$ ist ein Schaltschema S zugeordnet mit:

$S = (IO, PIO, MI, MO, Proz)$,

$IO \subseteq \{{}^\cdot tr \cup \{0\}\} \times \{tr^\cdot \cup \{0\}\}$ mit $dom(IO) = {}^\cdot tr$ und $rg(IO) = tr^\cdot$ heißt Verknüpfung der Eingangs- mit den Ausgangsplätzen der Transition tr,

PIO : Faktormenge von IO mit
$\forall el \in PIO : \forall (p, p'), (p^*, p'') \in el : p = p^*$
und $\{p \mid \{(p,) \in el \in PIO\} = {}^\cdot tr$

Es gibt damit für jeden Eingangsplatz ein el ε PIO, das für diesen Eingangsplatz den Ausgangsplatz bzw. die Ausgangsplätze bestimmt.

MI : ${}^\cdot tr \to \mathbb{N} \cup \{0\}$. MI(p) ist die Anzahl der von Platz $p \in {}^\cdot tr$ abzuziehenden Marken mit $\forall\, p \in {}^\cdot tr : MI(p) \leq A(p, tr)$,

MO : PIO → ℕ ∪ {0}, mit el ε PIO ist MO(el) die Anzahl der auf die Plätze der Verknüpfungsteilmenge el zu setzenden Marken.

Proz: Proz heißt Transitionsprozedur und ist eine geordnete Menge von ALGOL 60 Assignment Statements ohne Procedure Identifier (Definition siehe ALGOL 60 Report Kapitel 4.2). Die Assignment Statements enthalten nur Attribute der Eingangsplätze von tr, Attribute der Marken auf den Eingangsplätzen und Attribute der gemäß MO zu erzeugenden Marken.

Anmerkung: ALGOL wurde nur aus Definitionsgründen gewählt. Es könnte auch FORTRAN, PASCAL oder eine andere Programmiersprache genommen werden.

Wird eine Marke ma zum Zeitpunkt $t' \in T$ auf den Platz $p \in P$ gesetzt, dann heißt die Platzzeit $t_p(ma, p)$ der Marke ma auf dem Platz p abgelaufen, wenn $t_N \geq t' + t_p(p, ma)$.

Sei $(p, tr) \in Z$ und $ma \in MA_t(p)$. Die Transition tr heißt aktiviert durch die Marke ma auf dem Platz p, wenn die Platzzeit $t_p(p, ma)$ abgelaufen ist und $\forall\, p \in {}^\cdot tr : MA_t(p) \geq A(p, tr)$.

Eine aktivierte Transition kann schalten. Das Schalten einer aktivierten Transition tr beinhaltet folgende untrennbaren Teilschritte:

1. Abziehen der Marken von den Eingangsplätzen:
   $\forall\, p \in {}^{\bullet}tr : MA_t(p) := MA_t(p) - MI(p)$

2. Erzeugen von m Marken für die Ausgangsplätze:

   $$m: \sum_{el=(O,p)} MO(el) + \sum_{el=(p,)\wedge MI(p) \neq MO(el)} MO(el)$$

3. Ausführen von Proz

4. Setzen der Marken auf die Ausgangsplätze:

   $\forall\, p \in tr^{\bullet} : MA_t(p) := MA_t(p) + \sum_{(,p)\in el \wedge |el|=1} MO(el)$ + Anzahl von Marken durch Platzauswahl

Ist bei Teilschritt 1 für einen Platz $p \in P$ $MA_t(p) > MI(p)$, so werden die $MI(p)$ abzuziehenden Marken der Reihe nach nach einem Auswahlverfahren bestimmt.

Ist bei Teilschritt 4 für ein $el \in PIO$ $|el| > 1$, so wird der Ausgangsplatz, der $MO(el)$ Marken erhält, nach einem Auswahlverfahren bestimmt.

Ein Auswahlverfahren ist für eine Menge Z und eine Bedingung b definiert durch: Bestimme $x \in Z$ mit x erfüllt Bedingung b.

Der dynamische Ablauf der Marken in einem Netz wird durch folgende untrennbaren Schritte bestimmt:

1. $t_N := O, \quad M := M_O$

2. Bestimme die Menge der Marken in M, deren Platzzeit am kleinsten ist, sowie deren Plätze:

   $$t_{p_{min}} := \min_{ma \in MA_t} t_p(p, ma)$$

   $$MA_{min} := \{ma \in MA_t \mid t_p(p, ma) = t_{p_{min}}\}$$

   $$P_{min} := \{p \in P \mid t_p(p, ma) = t_{p_{min}}\}$$

3. $t_N := t_N + t_{p_{min}}$

4. Bestimme für alle MA_{min} die aktivierten Transitionen

   $TR_{akt_{min}} := \{tr \in TR \mid \exists p \in P_{min} : p \in {}^{\cdot}tr \wedge tr$ aktiviert durch $p \wedge (\neg p \in {}^{\cdot}tr^* \wedge tr^*$ aktiviert durch $p \wedge Index(tr^*) < Index(tr))$.

5. $\forall ma \in MA_t : t_p(p,ma) := t_p(p, ma) - t_{p_{min}}$

6. Schalte alle Transitionen $tr \in TR_{akt_{min}}$

7. Fahre bei 2. fort.

Bezüglich der graphischen Darstellung von S-Netzen sowie den Standardwerten von S-Netzgrößen sei auf Kapitel 4.1 d) verwiesen.

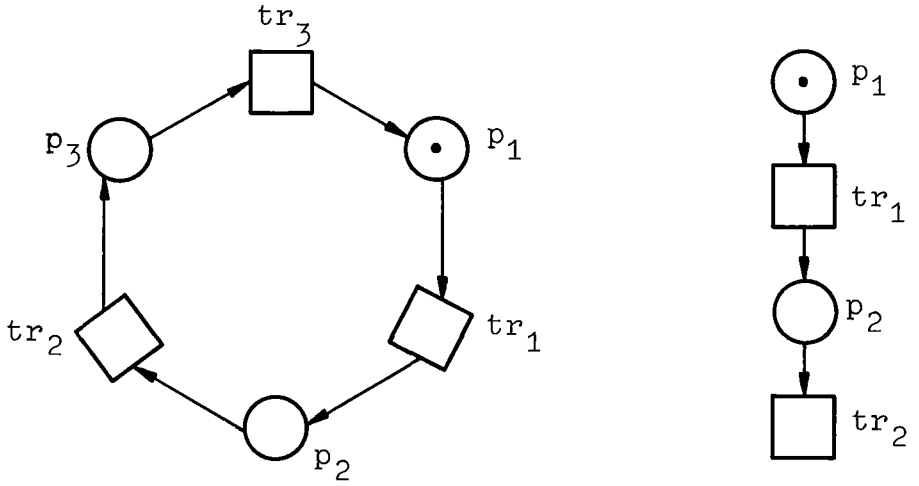
4.3 Modellierung diskreter dynamischer Systeme mit Simulationsnetzen

Wie man aus der Definition eines S-Netzes ersehen kann, bestehen zwischen Begriffen eines ddSystems und denen eines S-Netzes direkte Beziehungen:

ddSystem	S-Netz
Attribut	Attribut
permanentes Attribut	Platzattribut
temporäres Attribut	Markenattribut
Zustand	Netzstruktur und Zuordnung von Werten zu den Attributen
Übergang	Transition
Auswahlübergang	Platzauswahl
	Markenauswahl
Dauer	Platzzeit

Die Elemente eines ddSystems entsprechen in einem S-Netz den Marken und Plätzen. Der Eingangszustand eines Übergangs ist in einem S-Netz gleich der Markenanzahl auf allen Eingangsplätzen einer Transition und gleich den Werten der Marken- und Eingangsplatzattribute, während der Ausgangszustand eines Übergangs den Marken der Ein- und Ausgangsplätze und den Werten der Marken- und Platzattribute entspricht. Analog lassen sich die Begriffe (Un)Abhängigkeit eines Übergangs, 1/2/3-Indeterminiertheit, Vereinigung und Verzweigung von Übergängen, usw., auf S-Netze übertragen.

Ein Prozeß kann durch eine Folge von Transitionen und Plätzen dargestellt werden. Der Ablauf eines Prozesses entspricht dann dem Durchfluß einer Marke durch eine Transitions- und Platzfolge.



a) zyklischer Prozeß b) linearer Prozeß

Bild 7: Prozeßtypen

Bild 7 zeigt einen zyklischen und einen linearen Prozeß. Sind sämtliche Platzzeiten gleich Null, so bestehen die beiden Prozesse nur aus einer Folge von Ereignissen.

Nach obiger Interpretation wird durch jede Marke ein Prozeß dargestellt. Sind z.B. in Bild 7 zwei Plätze markiert, so liegen zwei Prozesse vor.

4.4 Vergleich mit anderen Modellierungsmethoden

Zuerst soll die Beziehung von S-Netzen zu Petri-Netzen diskutiert werden. Schränkt man ein S-Netz ein, indem man die Platzzeit t_p für alle Plätze gleich Null setzt, keine Attribute, keine Platzauswahl und keine Transitionsprozedur zuläßt, weiterhin die Verbindungsstruktur der Ein- und Ausgangsplätze einer Transition tr auf Elemente $({}^{\bullet}tr \times \{O\}) \cup (\{O\} \times tr^{\bullet})$ reduziert und A(p), MI(p) und MO(p) für alle Plätze und Transitionen gleich Eins setzt, so ist die Struktur eines S-Netzes gleich der eines Petri-Netzes.

Wenn man ferner bei dem dynamischen Ablauf für alle Markenauswahlen FIFO nimmt und generell keine Konfliktsituationen nach Transitionsindex auflöst, also alle Transitionen mit dem gleichen Index versieht, dann ist auch das dynamische Verhalten eines S-Netzes gleich dem eines Petri-Netzes.

Somit kann man für jedes Petri-Netz ein entsprechendes S-Netz konstruieren, welches für jede Anfangsmarkierung dieselbe Markierungsfolge liefert. Die Umkehrung der Aussage gilt nicht, wie durch Bild 5 b gezeigt wurde. Ein Testen auf Nichtvorhandensein einer Marke auf einem Platz (s. Kapitel 2.2) ist im übrigen auch im Rahmen eines Auswahlverfahrens möglich.

Im folgenden werden S-Netze mit anderen Modellierungsverfahren verglichen:

a) Zuordnung von Attributen:

Eine Zuordnung von Attributen oder Parametern ist in sämtlichen Petri-Netzerweiterungen enthalten, die einen Anspruch auf praktische Anwendbarkeit erheben. Die Grundidee der Attribute geht wahrscheinlich auf die Simulationssprache GPSS zurück, in der die Transactions und Facilities Attribute besitzen. Die Transactions entsprechen in S-Netzen den Marken und die Facilities den Stellen.

b) Einführung einer Zeitstruktur durch Platzzeiten:

In den bisherigen Petri-Netzmodifizierungen wie z.B. in den Evaluation-Netzen (E-Netze) wurden immer den Transitionen Zeiten zugeordnet /Ram 74, Noe 75, Nutt 72/. In S-Netzen werden dagegen die Marken durch Platzzeiten verzögert.

Auf den ersten Blick lassen sich beide Methoden ineinander überführen, wie Bild 8 zeigt.

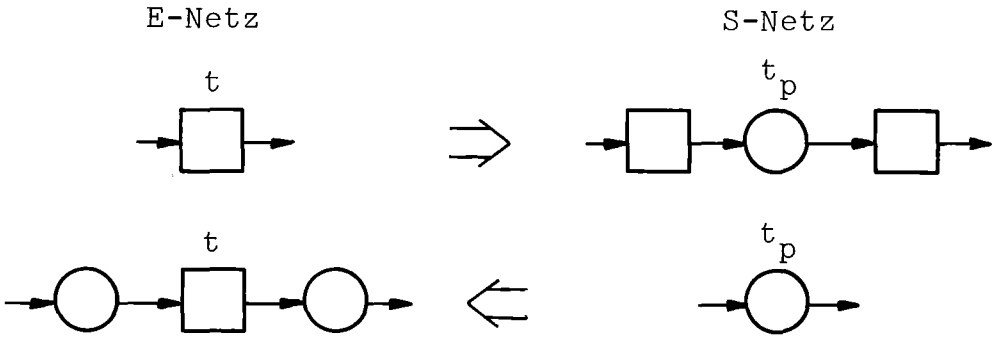


Bild 8: Analogie zwischen Transitionszeit und Platzzeit

Bei der Einführung von Zeiten ist jedoch noch das Verhalten der Marken zu spezifizieren. In E-Netzen sind die eine Transition aktivierenden Marken der Eingangsplätze beim Schalten der Transition in einem gesonderten Zustand, in dem nicht auf die Marken von außen zugegriffen werden kann.

Bei S-Netzen wurde die Aktivierungsdefinition dahingehend modifiziert, daß erst nach Ablauf der Platzzeit einer Marke eines Eingangsplatzes einer Transition letztere aktiviert werden kann. Die Modellierung einer Unterbrechung ist daher im Gegensatz zu S-Netzen in E-Netzen nicht möglich. Im übrigen beinhaltet die Transitionszeitmethode auch eine Platzzeit, die sich z.B. in einer Wartezeit einer Marke, die auf einem Eingangsplatz einer schaltenden Transition sitzt, äußert.

Bedingt durch die verschiedenen Zeitzuordnungen ergeben sich bei den S-Netzen im Gegensatz zu den Netzen von Ramchandani, Noe und seinen Mitarbeitern andere Interpretationsmöglichkeiten. Die Interpretation eines Platzes und einer Transition in einem Petri-Netz als Bedingung und als Ereignis bleibt in S-Netzen erhalten. In E-Netzen entspricht dagegen eine Transition zwei Ereignissen, wie z.B. Beginn und Ende einer Bedienung eines Jobs durch eine Zentraleinheit. Wegen der Darstellung zweier Ereignisse durch eine Transition haben E-Netze einen höheren Abstraktionsgrad als S-Netze, der jedoch durch die verschiedenartigen Transitionstypen teilweise wieder verloren geht.

c) Auflösung Konfliktfall:

In den Netzen von Noe und seinen Mitarbeitern ist der Konfliktfall ausgeschlossen. Die Modellierung logischer Strukturen wird durch Bereitstellung spezieller Transitionstypen ermöglicht. Das gleiche trifft auch auf GERT zu.

d) Modifizierung des Feuerns einer Transition:

Die Modifizierungen des Schaltens einer Transition in S-Netzen lassen sich aufteilen in

- definierte Verbindung zwischen Ein- und Ausgangsplätzen,
- Transitionsprozedur,
- Platzauswahl,
- Markenauswahl.

Die drei ersten Modifizierungen sind auch in E-Netzen enthalten, allerdings nur im Rahmen spezieller Transitionstypen im Gegensatz zur allgemeineren Vorgehensweise in S-Netzen.

e) Platzkapazität:

Eine Platzkapazität, wie sie z.B. von Noe und Mayr eingeführt wurde, ist in der S-Netzdefinition nicht enthalten, da sie zum einen durch einen gesonderten Platz darstellbar ist:

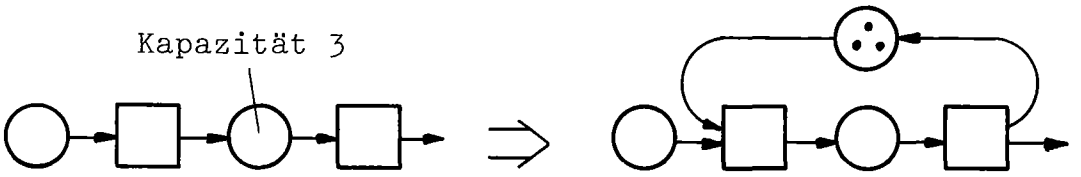


Bild 9: Modellierung von Platzkapazitäten

und zum anderen zu Schwierigkeiten bei der Definition von Platzauswahlverfahren und zu Rückwärtskonflikten führt. Obwohl mit Platzkapazitäten die Modellierung von ddSystemen kompakter ist, ist die Auswertung ohne diese Kapazitäten weniger aufwendig.

Abschließend wird noch auf diskrete Simulationssprachen, wie SIMULA, SIMSCRIPT und GPSS, eingegangen. Diskrete Simulationssprachen sind generell zur Beschreibung und Auswertung von realen ddSystemen geeignet, erfüllen aber nicht einige der zu Anfang dieses Kapitels aufgeführten Anforderungen, wie z.B. graphische Struktur (Ausnahme GPSS), leichte Erlernbarkeit und einfache Handhabung.

Mit S-Netzen könnte man einige dieser Nachteile beseitigen, wenn man sie, ähnlich wie Strukturdiagramme für algorithmische Sprachen, für die Beschreibung von Simulationsmodellen benutzt. Die Umsetzung eines Modells in ein Programm ist dann sehr einfach. Bei SIMSCRIPT müßte für jede Transition eine Subroutine geschrieben werden und bei SIMULA für jeden linear unabhängigen Zyklus mit mindestens einem Platz mit $t_p > 0$ eine Process Class.

5. Auswertung von Simulationsnetzen

In diesem Kapitel werden Meßgrößen für S-Netze eingeführt. Es wird gezeigt, wie diese Größen berechnet und die Ergebnisse interpretiert werden können.

Da der Ausgangspunkt dieses Berichts ddSysteme sind, ist die Liste möglicher Meßgrößen umfangreich. Aus diesem Grund wollen wir uns auf die wichtigsten Größen beschränken und diese mehr in einer tabellarischen Form auflisten.

Die Meßgrößen von S-Netzen lassen sich in strukturorientierte und verhaltensorientierte aufteilen. Erstere beziehen sich auf die Struktur eines S-Netzes, während verhaltensorientierte Meßgrößen direkt oder indirekt einen Zeitparameter enthalten und sich auf das dynamische Verhalten eines S-Netzes, d.h. auf die Folge der Aktivierungen und Schaltungen von Transitionen, beziehen.

Werte von Meßgrößen können analytisch oder experimentell berechnet werden. Unter einem Experiment soll bei S-Netzen die Simulation verstanden werden. Werte von verhaltensorientierten Meßgrößen werden i.a. experimentell ermittelt.

5.1 Strukturelle Auswertung

a) Syntaxanalyse:

Bevor man ein Modell auswertet, ist seine Übereinstimmung mit der Definition von S-Netzen zu überprüfen (Syntaxanalyse). Hierbei fallen die ersten strukturorientierten Meßgrößen an.

Syntaxfehler sind z.B. isolierte Plätze oder Transitionen, negative Platzzeiten, Zugriffe von Transitionsprozeduren auf nicht definierte Attribute, usw. Die Syntaxanalyse ist ähnlich der bei Programmiersprachen.

Im Rahmen der Syntaxanalyse lassen sich ferner einige der in Kapitel 2, 3 und 4 definierten Eigenschaften von Netzen bestimmen, wie Nebenbedingung, Konfliktfall, Indeterminiertheit und Plätze ohne Eingangs- bzw. Ausgangstransitionen, die z.B. bei offenen Systemen auftreten. Diese Eigenschaften können zur Klassifizierung von S-Netzen herangezogen werden und die weitere Auswertung festlegen.

b) Graphentheoretische Meßgrößen:

Jedes S-Netz besteht aus einem zweigeteilten gewichteten Graphen. Es ist naheliegend, Kenngrößen der Graphentheorie zur Auswertung von S-Netzen heranzuziehen.

Tabelle 1 enthält einige graphentheoretische Meßgrößen. Die Berechnung der Größen kann nach bekannten Algorithmen der Graphentheorie erfolgen /Neu 75, Berg 73/. Mögliche Interpretationen sind in der Tabelle angegeben.

Größe bzw. Eigenschaft	Interpretation
verbunden	wenn Netz nicht verbunden, liegt Syntaxfehler vor
streng verbunden	notwendige Bedingung für Lebendigkeit /Laut 75/
zyklomatische Zahl	Komplexitätsmaß, Faktor für Anzahl paralleler Prozesse
planar	Maß für übersichtliche Darstellung
Wurzeln Wald Gerüst Netzplan	Klassifizierung von Modellen
Weglänge	zeitlicher Abstand zwischen Ereignissen
kritischer Zyklus	Maß für Durchsatz /Ram 74/
kritischer Pfad	wie in der Netzplantechnik /Neu 75/

Tabelle 1: Graphentheoretische Meßgrößen für S-Netze

Eine der wichtigsten Größen in Tabelle 1 ist die zyklomatische Zahl, die inzwischen auch in der Graphentheorie eine zentrale Rolle spielt /Berg 73/. Die zyklomatische Zahl ist gleich der Anzahl der linear unabhängigen Zyklen in einem Graph und läßt sich bei einem S-Netz sehr einfach aus der Anzahl der Plätze und Transitionen sowie Pfeile berechnen:

$$\text{Zyklom.-Zahl} = 2 + |Z \cup Q| - |P| - |TR|$$

McCabe /McC 76/ zeigt die Anwendung der zyklomatischen Zahl als Komplexitätsmaß für Flußdiagramme. In S-Netzen ist die Bedeutung dieser Zahl umfassender. Als Klassifizierungsmaß bestimmt sie verschiedene Modelltypen. Weiterhin läßt sich mit ihr die Anzahl der zyklischen Prozesse berechnen, die sowohl für die Lebendigkeit als auch für die Berechnung der maximalen Markierung eine Eingangsgröße ist.

5.2 Auswertung des dynamischen Verhaltens

Bei der Einführung von Petri-Netzen in Kapitel 2 wurden schon einige für das Verhalten von Netzen charakteristische Größen wie Lebendigkeit, Schranken, O-Graphen usw. erwähnt.

In Tabelle 2 sind weitere Meßgrößen zum dynamischen Verhalten anhand der Netzelemente, Transitionen, Plätze und Markierungen, aufgelistet. Im unteren Teil der Tabelle sind noch die Meßgrößen Wege von Marken bzw. Zyklen und Weglängen bzw. Zyklenlängen aufgeführt. Mit letzteren ist der zeitliche Abstand zwischen Plätzen und Transitionen zu verstehen, die über einen Weg verbunden sind. Eine in der Praxis sehr wichtige Größe ist die maximale Weglänge bzw. der maximale Zyklus, auch kritischer Pfad bzw. kritischer Zyklus genannt.

Die Interpretationsmöglichkeiten der Größen in Tabelle 2 sind mannigfaltig und vom zugrundeliegenden System abhängig. Die durchschnittliche Markenplatzzeit kann z.B. in zyklischen ddSystemen als durchschnittliche Wartezeit eines Auftrags und in zyklenfreien ddSystemen, wie z.B. in

Platz:	Markenverweilzeit *
	Anzahl Marken auf Platz *
	Platzleerzeit *, d.h. Zeitphasen, in denen ein Platz nicht markiert ist
	Anzahl Marken, die Platz durchlaufen
	Zeitpunkt der ersten Markierung
	Zeitpunkt der letzten Markierung
Transition:	Anzahl Schaltungen
	Anzahl Aktivierungen
	Schaltzeitpunkte (erster, letzter, usw.)
	Aktivierungszeitpunkte (erster, letzter, usw.) für jeden Eingangsplatz
	zeitlicher Abstand zwischen Schaltungen *
	zeitlicher Abstand zwischen Aktivierungen *
	Markenbilanz, d.h. Anzahl der erzeugten minus vernichteten Marken
Markierung:	Anzahl Marken *
	Wege von Marken
	Wege (Zyklen)
	Weglängen * (Zyklenlängen*)
	Schnittkapazitäten, d.h. Anzahl Marken, die eine Platzmenge (Schnitt) durchlaufen

* Mittelwert, Standardabweichung, minimaler, maximaler Wert, Gesamtsumme

Tabelle 2: Dynamische Meßgrößen eines Simulationsnetzes

GERT-Netzplänen, als mittlere Vorgangsdauer interpretiert werden.

Die durchschnittliche Anzahl von Marken auf einem Platz kann bei Bedienungssystemen zum einen die durchschnittliche Warteschlangenlänge und zum anderen, wenn der Platz maximal eine Marke enthalten kann, die durchschnittliche Auslastung darstellen. Die ersten und letzten Schaltzeitpunkte einer Transition entsprechen bei Netzplänen den frühest oder spätest möglichen Anfangs- oder Endzeitpunkten eines Vorgangs.

Die Anzahl der Schaltungen einer Transition bzw. die maximale Anzahl der Marken auf einem Platz steht in direktem Zusammenhang mit der Lebendigkeit einer Transition bzw. eines Platzes. Die maximale Markenanzahl eines Platzes ist ein Maß für die Beschränktheit des Platzes.

Die Berechnung der in Tabelle 2 aufgeführten Meßgrößen erfolgt mit Hilfe der rechnergestützten Simulation /Köch 72, Neu 75/. Die Simulation ist bei komplexen Systemen wie z.B. größeren Warteschlangennetzen meistens die einzig mögliche Auswertungsmethode.

Abschließend sei noch auf eine spezielle Analysemethode bei der rechnergestützten Simulation, den Trace, hingewiesen. Ein Trace wird i.a. beim Austesten eines Programms bzw. Modells benötigt. Bei S-Netzen gibt es zwei Tracearten: Man kann den Modellablauf anhand der Ereignisse betrachten oder zu vorgegebenen Zeitpunkten, die meistens in einem äquidistanten Abstand liegen.

O-Graphen (s. Kapitel 2) sind z.B. graphische Darstellungen eines Trace. Bezieht man die Zeitstruktur in die graphische Darstellung ein, so erhält man Ganttdiagramme /Neu 75/.

Rauneker /Raun 77/ und Albrecht /Alb 77/ haben auf einem Kleinrechner mit 32 KW Hauptspeicher ein graphisches interaktives Modellierungs- und Auswertungssystem für S-Netze, auch GRIMMS genannt, entwickelt und in FORTRAN implementiert. Mit dem Programmsystem kann man ein S-Netzmodell auf einem Tektronix-Bildschirm interaktiv mit Hilfe eines Fadenkreuzes

entwerfen und anschließend per Simulation auswerten. Die Anwendung von GRIMMS wird in einem der nachfolgenden Beispiele gezeigt.

Die Auswertung baut bei GRIMMS auf einer früheren Arbeit von Ramöller /Ram 76/ auf, der ein SIMULA-Programm zur Auswertung von S-Netzen entwickelt hat. Die Auswertung kann jedoch wegen des SIMULA-Compilers nur im Stapelbetrieb auf einem Großrechner erfolgen.

5.3 Validation von Simulationsmodellen

Unter Validation eines Modells von einem System versteht man i.a. die Bestätigung, daß das Modell eine getreue, genauer homomorphe Abbildung eines meistens realen Systems darstellt. Die Validation - in unserem Falle von Simulationsmodellen - ist bis heute ein noch ungelöstes Problem. Man kann zwar meistens für bestimmte Datensätze eine homomorphe Abbildung nachweisen, jedoch nicht einen Nachweis für alle möglichen Parametersätze des Modells erbringen.

Peterson /Pet 77/ weist in seinem Übersichtsartikel auf einige Unentscheidbarkeitsprobleme bei Petri-Netzen hin: Gegeben seien zwei Petri-Netze PN_1 und PN_2 mit den Anfangsmarkierungen M_1 und M_2. Die Erreichbarkeitsmenge $E(M_1)$ ($E(M_2)$) der Anfangsmarkierung $M_1(M_2)$ des Netzes $PN_1(PN_2)$ ist dann definiert als die Menge aller möglichen Folgemarkierungen von $M_1(M_2)$. Rabin und später Hack /Hack 73-76/ zeigten, daß die Frage nach $E(M_1) \subseteq E(M_2)$ bzw. $E(M_1) = E(M_2)$ unentscheidbar ist, d.h. es existieren keine Algorithmen, die zeigen können, daß die Erreichbarkeitsmenge des Netzes PN_1 Teilmenge der Erreichbarkeitsmenge des Netzes PN_2 oder gleich dieser ist, unter der Voraussetzung einer beliebigen Anfangsmarkierung.

Bei festen Anfangsmarkierungen der Netze PN_1 und PN_2 kann man die Erreichbarkeitsmengen berechnen und dann vergleichen. Jedoch ist die Berechnung der Erreichbarkeitsmenge ein komplexes Problem, wie Peterson erwähnt, denn der Zeit- als auch Speicheraufwand ist in Abhängigkeit des Petri-Netzumfangs min-

destens exponentiell anwachsend.

In Kapitel 4.3 wurde gezeigt, daß mit Simulationsnetzen unter Elimination bestimmter Eigenschaften Petri-Netze dargestellt werden können. Interpretiert man nun obige Entscheidbarkeits- und Komplexitätsprobleme in der Art, daß das Netz PN_1 einem Modell und das Netz PN_2 dem zu modellierenden System entspricht, dann lassen sich obige Aussagen auf das Problem der Validation übertragen.

Diese mehr verbale Argumentation kann leider nicht formalisiert werden, da die Begriffe "Modell" und vor allem "reales System" nicht exakt definiert werden können.

6. Anwendungen

Die Anwendung von S-Netzen als Beschreibungs- und Auswertungsmethode soll anhand mehrerer Beispiele aus verschiedenen Fachgebieten gezeigt werden. Im ersten Beispiel wird ein zyklisches Transportsystem modelliert und für einen Datensatz ausgewertet. Bei diesem Beispiel wird auch die prinzipielle Vorgehensweise bei der hierarchischen Modellierung gezeigt.

Im zweiten Abschnitt wird ein Modell eines Computers entwickelt. Das Beispiel zeigt, wie ein schon relativ komplexes System, das nur in Spezialfällen mit analytischen Methoden untersucht werden kann, mit S-Netzen dargestellt und ausgewertet wird.

Im dritten und vierten Abschnitt wird die Anwendung von Simulationsnetzen in der Netzplan- bzw. Zuverlässigkeitstechnik gezeigt. Es folgt eine Beschreibung eines Realzeitprogramms. Schließlich werden noch einige grundlegende Bemerkungen zur Modellierung und Auswertung von diskreten dynamischen Systemen mit S-Netzen gemacht.

6.1 Modell eines Transportsystems

Es sollen Warteschlangendisziplinen in einem Transportsystem untersucht werden. Gegeben sei ein System, in dem mit Transportcontainern Güter von den Orten B_1 und B_2 zu einem Ort A transportiert werden:

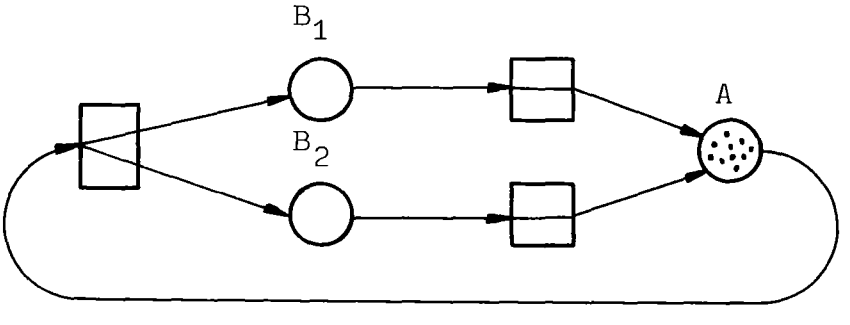


Bild 10 a: Modell eines Transportsystems

An den Orten B_1 und B_2 befinde sich jeweils eine Beladestation, vor denen Container, die eine besetzte Beladestation antreffen, warten sollen:

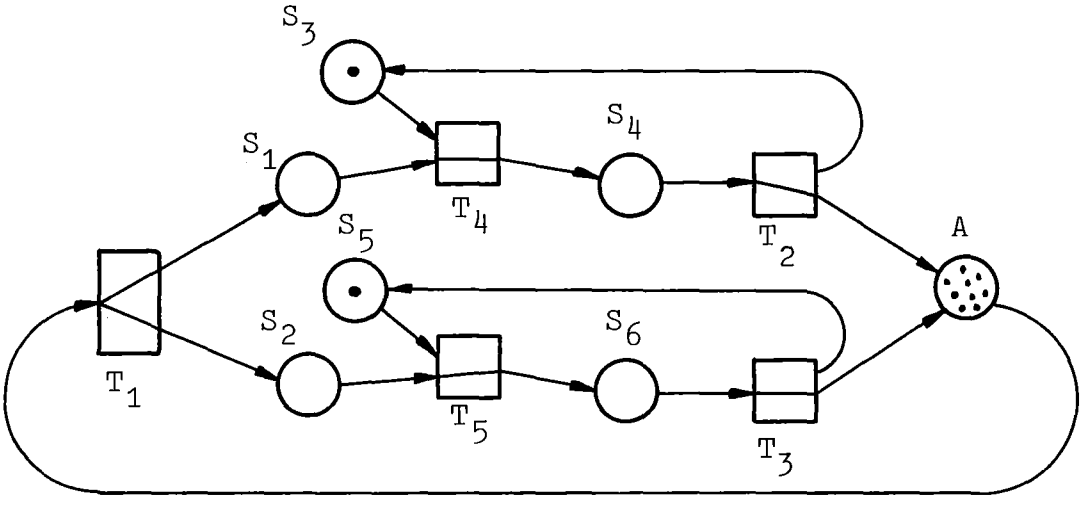


Bild 10 b: Bild 10 a mit einer Beladestation an dem Ort B_1 bzw. B_2

Bild 10 b entsteht aus Bild 10 a durch Verfeinerung eines Platzes und einer Transition. Die Plätze S_1 und S_2 repräsentieren Warteschlangen. Die Plätze S_3 und S_5 stellen die Bedingung dar, daß eine Beladestation nur immer einen Container, der als Marke dargestellt wird, beladen kann, d.h. die Plätze S_4 und S_6 die Kapazität eins haben.

Am Ort A seien drei parallele Entladestationen, die sich eine Warteschlange teilen. Jede Entladestation soll wie bei der Beladestation immer nur einen Container entladen können.

In Bild 10 c wurde der Platz A aus Bild 10 b verfeinert. Aus Gründen der Übersichtlichkeit wurde die Darstellung der Platzkapazitäten geändert.

Wir wollen auf dem Modellierungsniveau von Bild 10 c verbleiben und z.B. nicht mehr die Transporte von Ort A zu den Orten B_1 und B_2 darstellen, d.h. der Transport soll vernachlässigt werden.

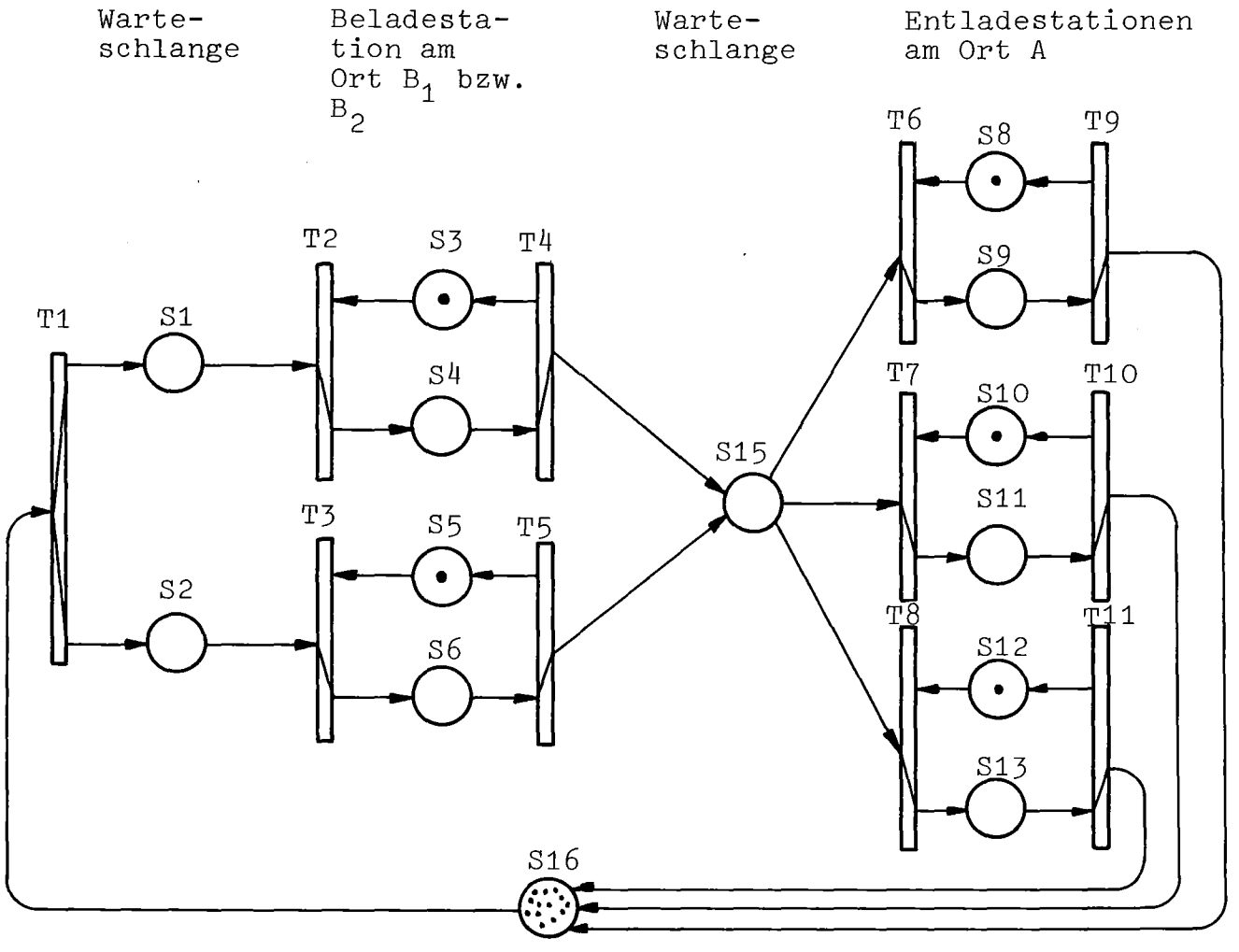


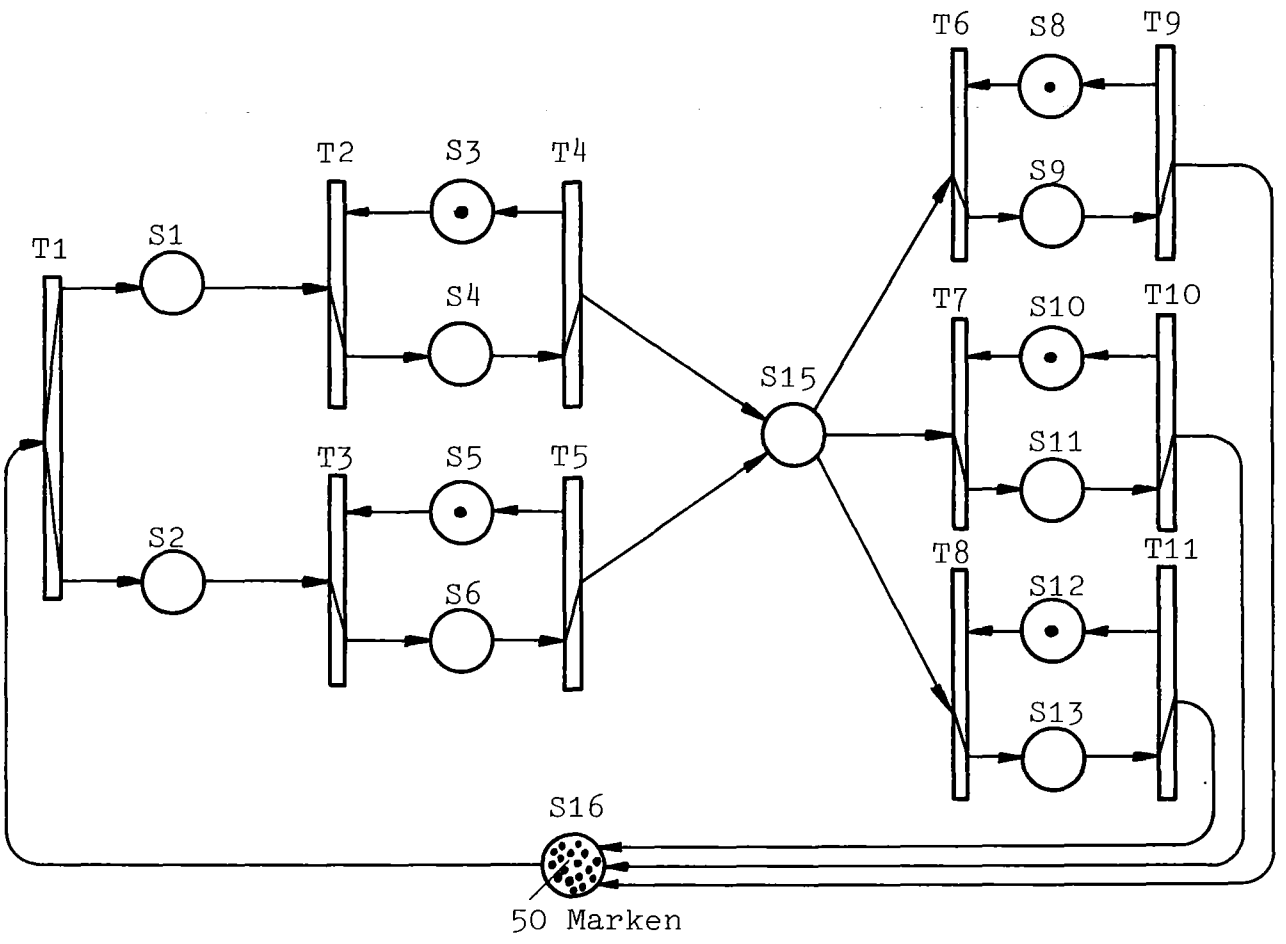
Bild 10 c: Bild 10 b, verfeinert um drei parallele Entladestationen

Das Modell in Bild 10 c beinhaltet nur die strukturellen Komponenten. Damit es ein auswertbares Modell wird, muß es noch weiter spezifiziert werden. Es wird z.B. angenommen, daß 50 Transportcontainer Güter von B_1 bzw. B_2 nach A transportieren. Die Beladezeit sei erlangverteilt mit den Parametern 0.2 und 3 bzw. 0.2 und 2 und die Entladezeit exponentialverteilt mit der Bedienungsrate 0.1. Die Warteschlangen sollen nach der FIFO-Regel abgearbeitet werden und die Entscheidung, wohin ein leerer Transport Container geschickt wird, soll nach der kürzesten Warteschlangenlänge getroffen werden.

Bild 10 d zeigt das endgültige Modell. Die Auswertung des Modells mit Hilfe der Simulation ist in Bild 10 e und 10 f zusammengestellt. Die Simulation wurde mit einem SIMULA-Programm, welches von Ramöller /Ram 76/ implementiert wurde, durchgeführt.

Die erste Tabelle in Bild 10 e enthält den Stichprobenumfang der Werte in den nachfolgenden Tabellen. Die Verweilzeittabelle zeigt Werte der Platzzeiten, die die Marken im Schnitt auf den einzelnen Plätzen verbracht haben. Die Verweilzeit kann verschiedenartig interpretiert werden. Bei den Plätzen P_1, P_2 und P_{15} ist sie gleich der Wartezeit, bei den Plätzen P_4, P_6, P_9 und P_{13} entspricht sie der Bedienungszeit. Die Werte in der Tabelle "Statistics of Tokens" repräsentieren z.B. Warteschlangenlänge (Platz P_1) oder Durchsatz einer Beladeeinheit (Platz P_4). Im unteren Teil von Bild 10 e ist die Auswertung zweier Pfade des Modells angegeben.

Warte-schlange | Belade-station am Ort B_1 bzw. B_2 | Warte-schlange | Entladestationen am Ort A



Platzzeiten: $t_{p1,\ 2,\ 3,\ 5,\ 8,\ 10,\ 12,\ 15,\ 16} = 0$

t_{p4} = erlang (0.2, 3)

t_{p6} = erlang (0.2, 2)

$t_{p9,\ 11,\ 13}$ = negexp (0.1)

Schaltschema:

	Transition T2-11	T1
Eingangsplatznr.	...	16
untere Ausgangsplatznr.	...	1
obere Ausgangsplatznr.	...	2
abzuziehende Markenanzahl	1	1
hinzuzufügende Markenanzahl	1	1
Markenauswahlverfahren	FIFO	FIFO
Platzauswahlverfahren		kürzeste Warteschl.

Bild 10 d: Simulationsnetzmodell eines Transportsystems

Bild 10 e: Auswertung des S-Netzes von Bild 10 d mit Hilfe der Simulation

RESULTS OF MEASUREMENT:
****

ABSOLUTE TIME: 11000.0000
SIMULATIONTIME : 10000.0000

STATISTIC OF PLACES
****

PLACENO	PLACENAME	TOTAL OUTPUT	CURRENT
1	QUEUE B	1788	0
2	QUEUE B	1195	0
3	LOADING.B1 (IDLE)	1788	0
4	LOADING.B1 (BUSY)	1788	1
5	LOADING.B2 (IDLE)	1195	0
6	LOADING.B2 (BUSY)	1195	1
8	UNLOADING.A (IDLE)	956	0
9	UNLOADING.A (BUSY)	956	1
10	UNLOADING.A (IDLE)	1002	0
11	UNLOADING.A (BUSY)	1002	1
12	UNLOADING.A (IDLE)	1024	0
13	UNLOADING.A (BUSY)	1024	1
15	QUEUE A	2982	45
16	CONNEXION A-B	2982	0

STATISTIC OF DWELLTIME
****

PLACENO	PLACENAME	AVERAGE	STAND.DEVIATION	MAXIMUM	MINIMUM
1	QUEUE B	5.604	5.862	33.186	0.00
2	QUEUE B	5.415	6.484	39.288	0.00
3	LOADING.B1 (IDLE)	0.688	2.094	19.246	0.00
4	LOADING.B1 (BUSY)	4.909	2.806	16.611	0.18
5	LOADING.B2 (IDLE)	3.211	8.061	77.695	0.00
6	LOADING.B2 (BUSY)	5.160	3.588	28.746	0.02
8	UNLOADING.A (IDLE)	0.000	0.000	0.000	0.00
9	UNLOADING.A (BUSY)	10.449	10.636	78.312	0.00
10	UNLOADING.A (IDLE)	0.000	0.000	0.000	0.00
11	UNLOADING.A (BUSY)	9.966	10.217	73.291	0.00
12	UNLOADING.A (IDLE)	0.000	0.000	0.000	0.00
13	UNLOADING.A (BUSY)	9.793	9.520	54.111	0.00
15	QUEUE A	147.257	22.578	219.500	95.44
16	CONNEXION A-B	0.000	0.000	0.000	0.00

STATISTIC OF TOKENS
****

PLACENO	PLACENAME	AVERAGE	STAND.DEVIATION	MAXIMUM	MINIMUM
1	QUEUE B	1.001	1.081	6.000	0.00
2	QUEUE B	0.647	0.979	6.000	0.00
3	LOADING.B1 (IDLE)	0.123	0.328	1.000	0.00
4	LOADING.B1 (BUSY)	0.877	0.328	1.000	0.00
5	LOADING.B2 (IDLE)	0.384	0.486	1.000	0.00
6	LOADING.B2 (BUSY)	0.616	0.486	1.000	0.00
8	UNLOADING.A (IDLE)	0.000	0.000	1.000	0.00
9	UNLOADING.A (BUSY)	1.000	0.001	1.000	0.00
10	UNLOADING.A (IDLE)	0.000	0.000	1.000	0.00
11	UNLOADING.A (BUSY)	1.000	0.001	1.000	0.00
12	UNLOADING.A (IDLE)	0.000	0.000	1.000	0.00
13	UNLOADING.A (BUSY)	1.000	0.001	1.000	0.00
15	QUEUE A	43.844	2.437	47.000	33.00
16	CONNEXION A-B	0.000	0.001	2.000	0.00

PATHS
*****

FROM PLACE	UNTIL PLACE	AVERAGE	STAND.DEVIATION	MAXIMUM	MINIMUM
16	15	10.535	6.812	43.350	0.304

PATHS
*****

FROM PLACE	UNTIL PLACE	AVERAGE	STAND.DEVIATION	MAXIMUM	MINIMUM
16	16	167.918	25.525	261.309	104.309

HISTOGRAM : FOR PLACE NO. 15

DWELLTIME

	FREQUENCY	
0.00E+00	17	******
0.00E+00	8	***
5.00E+00	2	*
1.00E+01	1	
1.50E+01	1	
2.00E+01	0	
2.50E+01	6	**
3.00E+01	10	***
3.50E+01	1	
4.00E+01	7	**
4.50E+01	5	**
5.00E+01	13	*****
5.50E+01	21	*******
6.00E+01	19	*******
6.50E+01	0	
7.00E+01	4	*
7.50E+01	11	****
8.00E+01	6	**
8.50E+01	14	*****
9.00E+01	20	*******
9.50E+01	23	********
1.00E+02	81	*****************************
1.05E+02	107	**************************************
1.10E+02	133	**
1.15E+02	161	***
1.20E+02	232	***
1.25E+02	214	**
1.30E+02	242	**
1.35E+02	266	**
1.40E+02	288	**
1.45E+02	273	***
1.50E+02	235	***
1.55E+02	241	**
1.60E+02	162	**
1.65E+02	111	**
1.70E+02	101	*************************************
1.75E+02	54	*******************
1.80E+02	57	********************
1.85E+02	41	**************
1.90E+02	48	*****************
1.95E+02	46	*****************

Bild 10 f: Histogramm der Markenverweilzeit auf Platz P15

6.2 Modell eines Rechnersystems

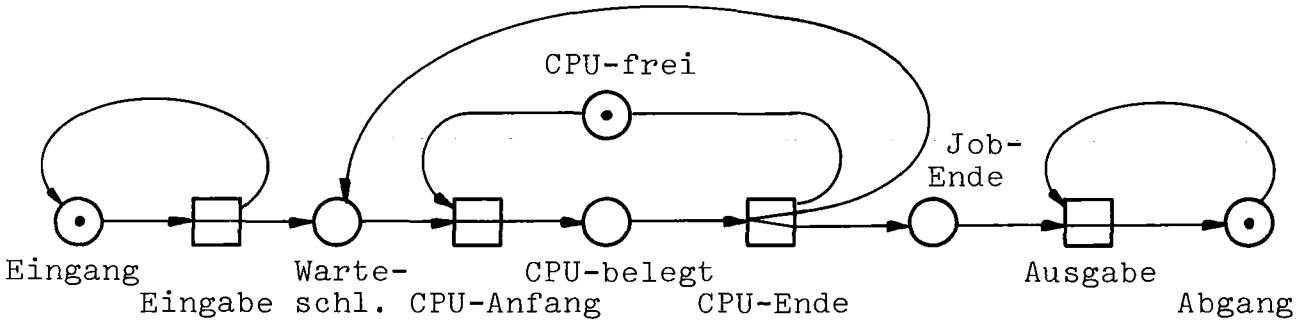
Gegeben sei ein Rechnersystem mit mehreren parallelen Prozessoren und Plattenspeichern. Der Auftragsankunftsstrom sei ein Poissonstrom mit der Ankunftsrate LA. Die Aufträge werden nach einem Zeitscheibenverfahren von den Prozessoren abgearbeitet. Die Bearbeitungszeiten eines Auftrags für einen Prozessor bzw. Plattenspeicher seien exponentialverteilt mit dem Parameter BED bzw. gleichverteilt im Bereich 1 bis PB. Nach NP Bearbeitungsphasen eines Auftrags erfolge jeweils ein Plattenspeicherzugriff. Nach vollständiger Bearbeitung verlasse ein Auftrag das Rechnersystem. Der Hauptspeicher sei begrenzt auf maximal HP Aufträge. Die Auslastung der einzelnen Bearbeitungseinheiten sowie die Wartezeiten der Aufträge sollen untersucht werden.

Eine hierarchische Modellierung dieses Systems kann dann folgendermaßen durchgeführt werden: Bild 11 a zeigt ein Basismodell ohne Plattenspeicher, ohne Hauptspeicherbegrenzung und mit einem Prozessor. Zur Modellierung werden dabei auch Transitionsprozeduren (s. Kap. 4.1) benutzt.

Die Erweiterung des Modells auf mehrere parallele Prozessoren erfolgt einfach durch Hinzufügen entsprechender Marken auf Platz CPU-FREI und eines Markenauswahlverfahrens in Transition CPU-ANFANG. Mit diesem Modell ist jedoch eine Vermessung der Auslastung der einzelnen Prozessoren nicht möglich.

Im Modell in Bild 11 b ist dieser Nachteil beseitigt sowie eine Hauptspeicherbegrenzung mit Aufspaltung der Warteschlange in zwei Warteschlangen eingeführt. Der Platz CPU-ENDE dient nur einer übersichtlichen Darstellung. Bild 11 c zeigt das endgültige Modell mit den Plattenspeichern, die dem Modell von Bild 11 b einfach hinzugefügt werden.

Ein Teil der möglichen Auswertung des Modells ist in Bild 11 d für einen Datensatz gezeigt. Die Berechnung der Größen erfolgte wieder mit Hilfe des von Ramöller /Ram 76/ implementierten Programms. Die Interpretation der Werte ist ähnlich der des obigen Beispiels eines Transportsystems.



$t_{pEingang}$:= negexp (LA);

$t_{pCPU\text{-}belegt}$:= wenn const > ma.attr (1), dann ma.attr (1)
sonst const;

Eingabe.Proz : ma.attr (1) := negexp (BED);

CPU-Ende. Proz : ma.attr (1) := ma.attr (1)- CPU-belegt.attr (1);

CPU-Ende.Platzauswahl: wenn ma.attr (1) ≤ 0, dann Job-Ende sonst
Warteschlange

Const ≙ Zeitscheibenlänge

Bild 11 a: Modell eines Rechnersystems mit Zeitscheiben-bearbeitung

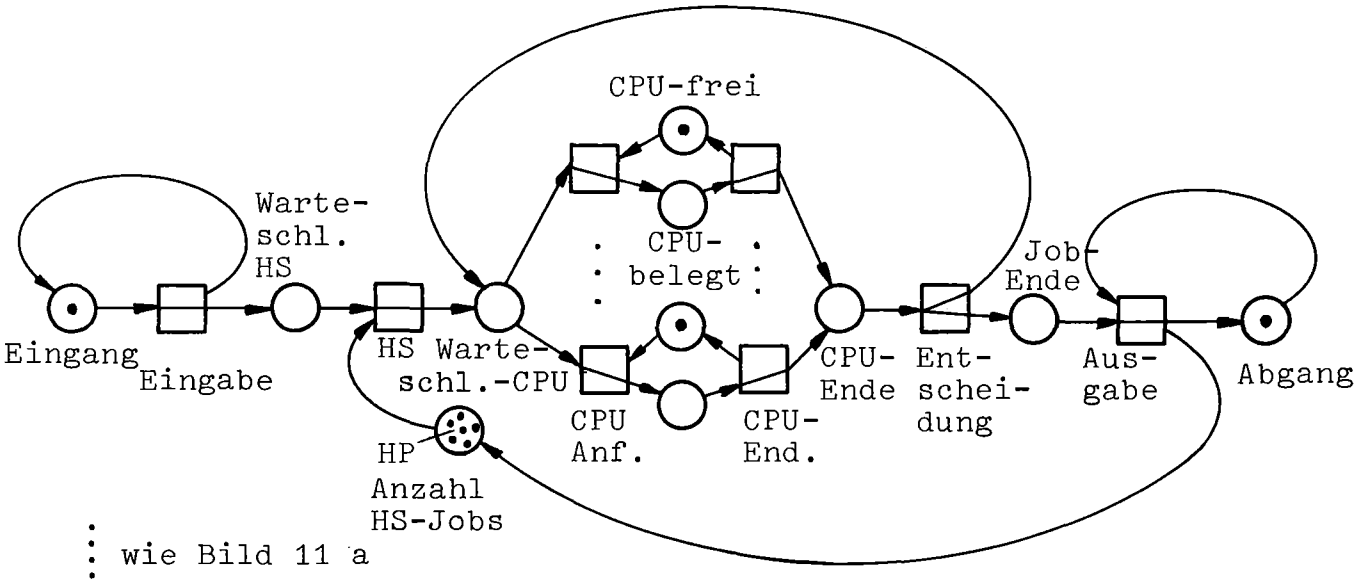
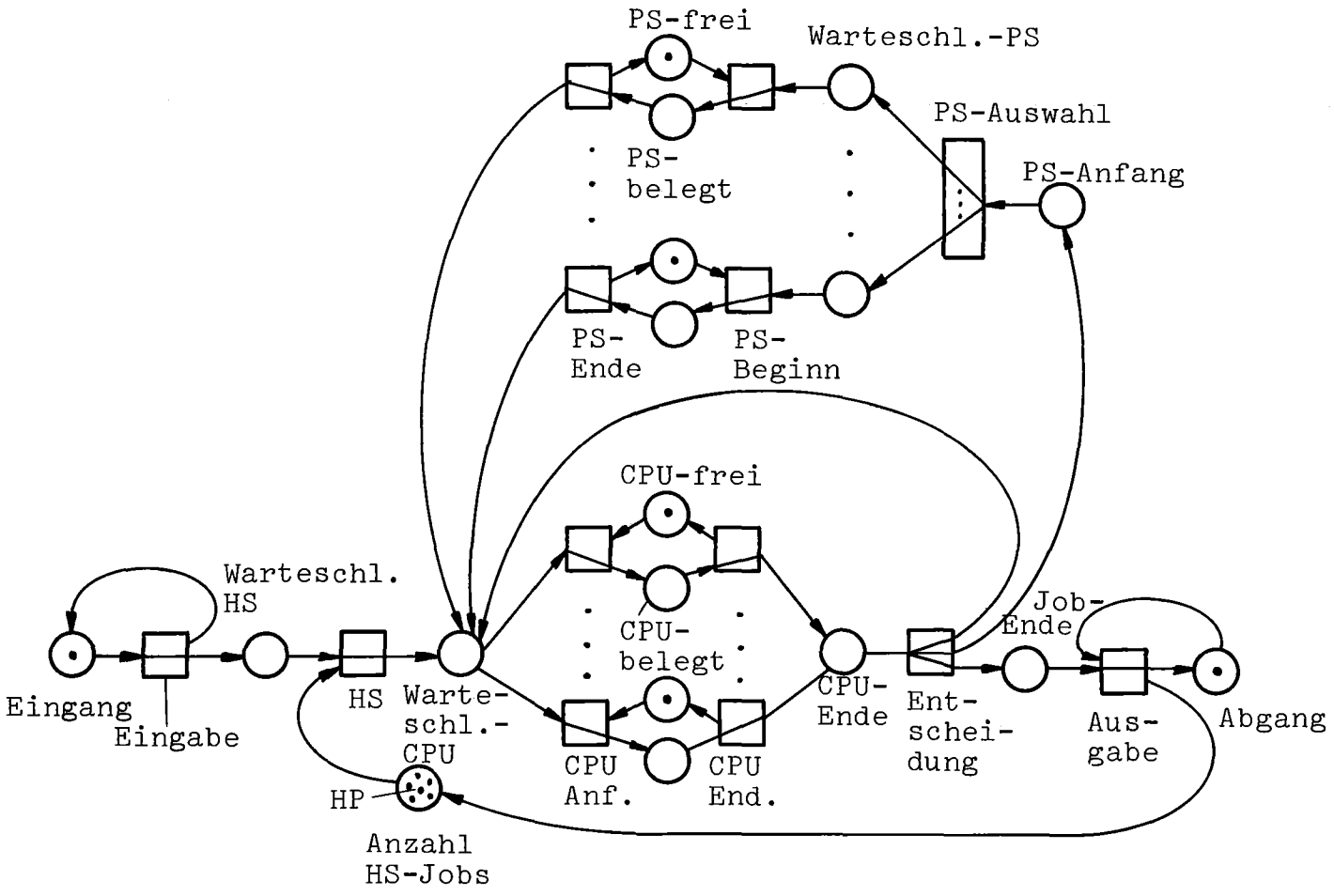


Bild 11 b: Modell nach Bild 11 a mit Hauptspeicherbegrenzung und mehreren parallelen CPU's



$t_{p_{Eingang}}$:= negexp (LA);

$t_{p_{CPU\text{-}belegt}}$:= wenn ma.attr (3) > ma.attr (2), dann ma.attr (2), sonst ma.attr (3)

$t_{p_{PS\text{-}belegt}}$:= randint (1, PB);

Eingabe.Proz: ma.attr (1) := negexp (Bed);
ma.attr (4) := randint (1, NP);
ma.attr (2) := ma.attr (1) / ma.attr (4);
ma.attr (3) := ma.attr (2);

Entscheidung.Proz: ma.attr (3) := ma.attr (3)- const;

PS-Auswahl.Proz: ma.attr (3) := ma.attr (2);
ma.attr (4) := ma.attr (4)-1;

Entscheidung.Platzauswahl: wenn ma.attr (3) > 0, dann Warte-schlange-CPU sonst, (wenn ma.attr (4) ≤ 0, dann Job-Ende, sonst PS-Anfang);

Bild 11 c: Modell nach Bild 11 b mit Plattenspeichern

Bild 11 d: Auswertung des S-Netzes von Bild 11 c

Datensatz:

LA: 0,005
BED: 0,002
PB: 100
NP: randint (1,10)
HP: 20
Zeitscheibenlänge: 100
Simulationszeit: 400000

STATISTIC OF PLACES

PLACENO	PLACENAME	TOTAL OUTPUT	CURRENT
1	EINGANG	2039	1
2	WARTESCHL-HS	2038	0
3	ANZAHL HS-JOBS	2038	7
5	WARTESCHL-CPU	18847	0
8	ABGANG	2032	1
11	CPU-FREI	5415	1
12	CPU-FREI	4796	1
13	CPU-FREI	3907	0
14	CPU-FREI	2844	0
15	CPU-FREI	1885	1
21	CPU-BELEGT	5416	0
22	CPU-BELEGT	4796	0
23	CPU-BELEGT	3907	1
24	CPU-BELEGT	2844	1
25	CPU-BELEGT	1986	0
31	WARTESCHL-PS	3946	3
32	WARTESCHL-PS	3800	3
33	WARTESCHL-PS	3480	2
41	PS-FREI	3946	0
42	PS-FREI	3800	0
43	PS-FREI	3480	0
51	PS-BELEGT	3946	1
52	PS-BELEGT	3800	1
53	PS-BELEGT	3479	1

STATISTIC OF DWELLTIME

PLACENO	PLACENAME	AVERAGE	STAND.DEVIATION	MAXIMUM	MINIMUM
1	EINGANG	196.244	193.116	1399.500	0.000
2	WARTESCHL-HS	79.691	250.547	1941.312	0.000
3	ANZAHL HS-JOBS	1414.564	1129.405	4387.312	0.000
5	WARTESCHL-CPU	2.468	9.171	125.500	0.000
8	ABGANG	196.527	199.515	1687.937	0.250
11	CPU-FREI	17.372	23.998	177.062	0.000
12	CPU-FREI	27.232	39.975	626.000	0.000
13	CPU-FREI	45.868	77.368	1055.312	0.000
14	CPU-FREI	84.638	164.895	2992.187	0.000
15	CPU-FREI	154.962	395.359	5274.000	0.000
21	CPU-BELEGT	56.486	36.659	100.000	0.000
22	CPU-BELEGT	56.169	36.124	100.000	0.000
23	CPU-BELEGT	56.497	36.601	100.000	0.000
24	CPU-BELEGT	56.030	36.188	100.000	0.000
25	CPU-BELEGT	57.113	36.548	100.000	0.000
31	WARTESCHL-PS	276.472	193.857	939.187	0.000
32	WARTESCHL-PS	253.580	181.479	810.875	0.000
33	WARTESCHL-PS	239.952	174.133	796.687	0.000
41	PS-FREI	0.956	6.686	169.812	0.000
42	PS-FREI	4.507	29.925	668.437	0.000
43	PS-FREI	15.515	116.852	3114.437	0.000
51	PS-BELEGT	100.397	58.087	200.000	1.000
52	PS-BELEGT	100.768	57.588	200.000	1.000
53	PS-BELEGT	99.440	57.790	200.000	1.000

STATISTIC OF TOKENS

PLACENO	PLACENAME	AVERAGE	STAND.DEVIATION	MAXIMUM	MINIMUM
1	EINGANG	1.000	0.031	1.000	1.000
2	WARTESCHL-HS	0.406	1.272	9.000	0.000
3	ANZAHL HS-JOBS	7.171	5.298	19.000	0.000
5	WARTESCHL-CPU	0.116	0.536	10.000	0.000
8	ABGANG	1.000	0.031	1.000	1.000
11	CPU-FREI	0.235	0.424	1.000	0.000
12	CPU-FREI	0.327	0.469	1.000	0.000
13	CPU-FREI	0.448	0.497	1.000	0.000
14	CPU-FREI	0.602	0.490	1.000	0.000
15	CPU-FREI	0.731	0.444	1.000	0.000
21	CPU-BELEGT	0.765	0.424	1.000	0.000
22	CPU-BELEGT	0.673	0.469	1.000	0.000
23	CPU-BELEGT	0.552	0.497	1.000	0.000
24	CPU-BELEGT	0.398	0.490	1.000	0.000
25	CPU-BELEGT	0.269	0.444	1.000	0.000
31	WARTESCHL-PS	2.728	1.707	6.000	0.000
32	WARTESCHL-PS	2.409	1.674	6.000	0.000
33	WARTESCHL-PS	2.089	1.661	5.000	0.000
41	PS-FREI	0.009	0.097	1.000	0.000
42	PS-FREI	0.043	0.202	1.000	0.000
43	PS-FREI	0.135	0.342	1.000	0.000
51	PS-BELEGT	0.991	0.097	1.000	0.000
52	PS-BELEGT	0.957	0.202	1.000	0.000
53	PS-BELEGT	0.865	0.342	1.000	0.000

Zwischen Rechnermodellen und Transportmodellen bestehen enge Beziehungen. Erstens ist die Problemstellung i.a. gleich, man möchte z.B. Warteschlangendisziplinen oder Zuordnungsalgorithmen auf ihre Effizienz hin untersuchen, zweitens sind die zu vermessenden Zielgrößen,wie Warteschlangenlänge, Durchsatz und Auslastung,gleich und drittens sind die Modelle von ähnlicher Struktur. Beide Beispiele stammen aus der Klasse der Job-Shopsysteme.

6.3 Anwendung in der Netzplantechnik

Simulationsnetze können zur Darstellung und Auswertung von Netzplänen herangezogen werden und sind für Vorgangspfeilnetze als auch Ereignisknotennetze geeignet.

Folgende Zuordnung von S-Netzkomponenten zu z.B. den Komponenten von Vorgangspfeilnetzen ermöglicht die Anwendung von S-Netzen:

Vorgangspfeilnetz		S-Netz
Knoten	$\Leftrightarrow$	Transition
Pfeil mit Vorgangsdauer	$\Leftrightarrow$	Platz mit Platzzeit

Ein einfaches Beispiel soll den Sachverhalt veranschaulichen. Neumann /Neu 75/ zeigt ein Vorgangspfeilnetz für den Bau einer Brücke. Das entsprechende S-Netzmodell ist in Bild 12 dargestellt.

Das S-Netzmodell ist unübersichtlicher als das Vorgangspfeilnetz in /Neu 75/. Dies resultiert im wesentlichen aus der unterschiedlichen Symbolik sowie aus der Einführung von Hilfsplätzen, die an den Stellen, an denen mehrere Plätze Eingangsplatz einer Transition sind, wegen der Aktivierungsdefinition (s. Kap. 4.1 b) eingeführt werden müssen. Die Unübersichtlichkeit könnte durch Benutzung von auf Netzpläne zugeschnittene Symbole für S-Netze beseitigt werden.

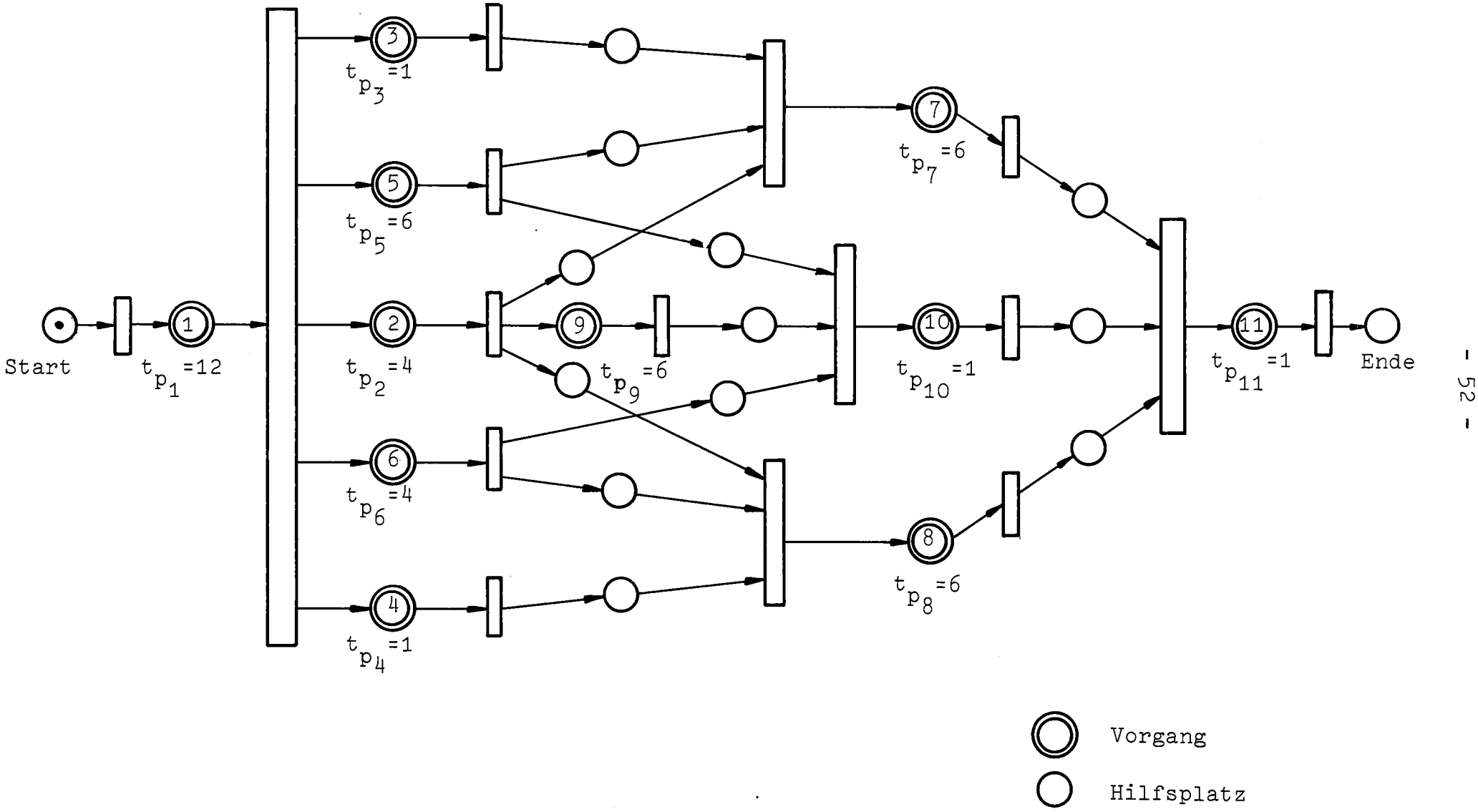


Bild 12: S-Netzmodell eines Vorgangspfeilnetzes aus /Neu 75/

Die Auswertung eines S-Netzmodells eines Netzplans ist einfach: man muß die Marke des Startplatzes einmal durch das Netz laufen lassen und die Schaltzeitpunkte der einzelnen Transitionen sowie den Pfad vom Startplatz zum Endeplatz aufzeichnen. Dies entspricht dem Lee-Algorithmus in der Netzplantechnik.

Sind die Vorgangsdauern nicht konstant sondern wie bei PERT zufallsverteilt, dann ist bei der Auswertung der Endeplatz mit dem Eingangsplatz zu verbinden und die Startmarke je nach gewünschtem Stichprobenumfang entsprechend oft im Kreis laufen zu lassen.

S-Netze haben im Gegensatz zu den Netzplantechniken wie CPM, PERT usw. den Vorteil, daß sie auch begrenzte Betriebsmittelanzahl und deren Verwaltung, wie in den beiden vorangegangenen Anwendungen in Kapitel 6.1 und 6.2 gezeigt, darstellen können.

6.4 Anwendung in der Zuverlässigkeitstheorie

Im Rahmen der Zuverlässigkeitstheorie werden diskrete dynamische Systeme untersucht, die sich in der Problemstellung zu Job-Shopsystemen unterscheiden. Zielgrößen sind z.B. neben der Zuverlässigkeit die Verfügbarkeit, die mittlere Zeit bis zum ersten Systemausfall oder die mittlere Zeit zwischen zwei Systemausfällen, wenn Komponenten repariert werden können, usw. Beim heutigen Stand der Zuverlässigkeitstheorie sind nur relativ einfache Systeme analytisch zugänglich, so daß sich die Simulation als Auswertungsmethode anbietet.

Die Anwendung von S-Netzen soll anhand zweier Beispiele gezeigt werden. Gegeben sei jeweils ein System, welches sich aus mehreren Komponenten zusammensetzt.

Eine Komponente soll stets in genau einem von zwei Zuständen sein, in Betrieb oder im Ausfall. Eine ausgefallene Komponente wird sofort repariert. Beide Zustände wechseln sich gegenseitig ab. Die Betriebsphasen und die Reparaturphasen seien zufallsverteilte Größen. Gesucht sei die Verfügbarkeit des Gesamtsystems.

Im ersten Beispiel wird ein System, das aus zwei parallelen Einheiten, einer Grundeinheit und einer Reserveeinheit, besteht, behandelt. Die Reserve erfolge unbelastet, d.h. die Reserveeinheit ist nicht in Betrieb, wenn die Grundeinheit läuft, und umgekehrt. Der Übergang von einer ausgefallenen Einheit auf eine Ersatzeinheit soll eine gewisse Anlaufzeit erfordern, die auch zufallsverteilt sei.

Das entsprechende S-Netzmodell ist in Bild 13 a dargestellt. Das Modell wurde mit dem interaktiven Modellierungs- und Auswertungssystem GRIMMS (s. Kap. 5.2) erstellt. Die graphischen Symbole sind bei GRIMMS aus technischen Gründen abgeändert (□-Platz, ||-Transition). Die Größe der Symbole kann vom Benutzer festgesetzt werden. Die Abkürzungen bedeuten REP - Reparatur, WART - Warten, ANL - Anlaufen und BETR - Betrieb. Der Platz P006 dient der Vermessung der Systemverfügbarkeit. Die weiteren Spezifikationen der Plätze sind in Tabelle 3 gezeigt. Sie werden bei GRIMMS alphanumerisch definiert.

Die Auswertung des Modells ist in Bild 13 b zusammengestellt. Die erste Tabelle der Platzverweilzeit enthält Werte für die Reparaturzeit (P1), die Zeit, die eine Anlage bereitstellt (P2), die Anlaufzeit (P3), die Betriebszeit (P5) und die Systemausfallzeit (P6).

Die zweite Tabelle enthält Werte für die Zeit, in der ein Platz keine Marken enthält. Die Verfügbarkeit des Systems ist gleich dem Wert von Platz P6, der zum Wert von Platz P5 komplementär ist. Die letzte Tabelle zeigt Werte der durchschnittlichen Markenanzahl pro Platz.

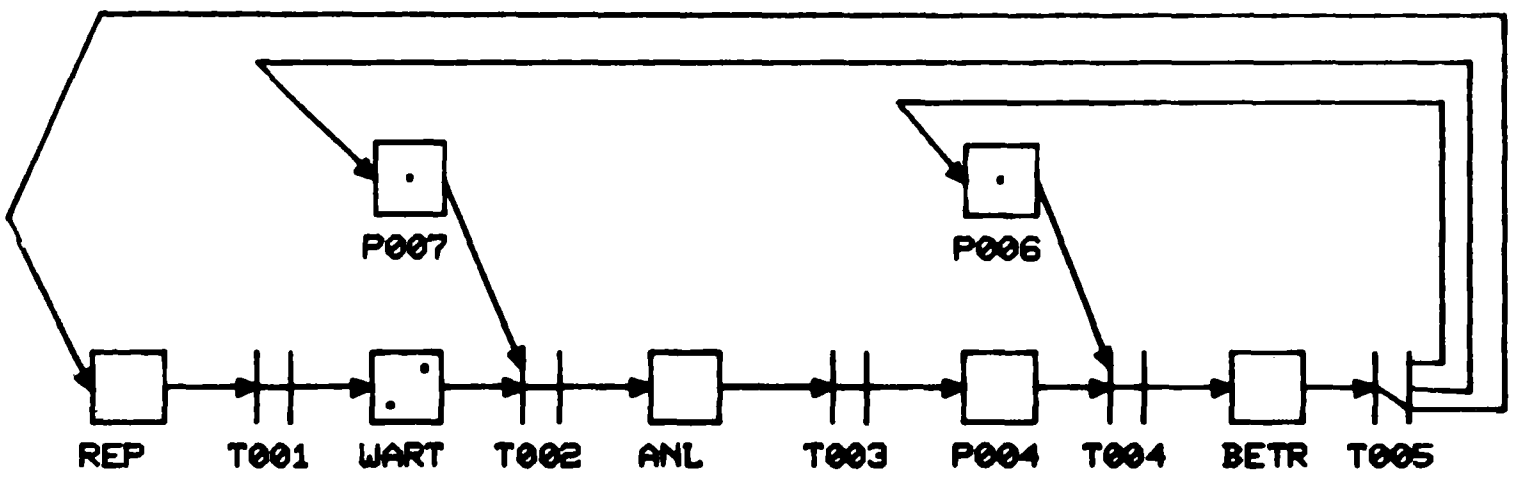


Bild 13 a: Modell eines Reservesystems mit Umschaltzeit

PLACES :
****

NO	PLACENAME	↑ DELAYTIME ATTRIBUTES ↑ MODE	DISPAR.1	DISPAR.2	↑ PLACESELECTION ATTRIBUTES ↑ PRIORITY	DEBIT	↑ PATH CONSTITUENCY ↑ SOURC	DEST	↑ MEASUREMENT ↑ SPEC EVTS	MAIN STAT	DWT HIST	EMT HIST	↑ OUTPUTTRANSITIONS ↑
1	REP	EXP	10.000	0.000	0.000	0.000			*	*			1
2	WART	ZERO	0.000	0.000	0.000	0.000			*	*			2
3	ANL	EXP	5.000	0.000	0.000	0.000			*	*			3
4	P004	ZERO	0.000	0.000	0.000	0.000			*	*			4
5	BETR	EXP	5.000	0.000	0.000	0.000			*	*			5
6	P006	ZERO	0.000	0.000	0.000	0.000			*	*			4
7	P007	ZERO	0.000	0.000	0.000	0.000			*	*			2

Tabelle 3: Spezifikation der Plätze des Modells nach Bild 13 a

1) STATISTICS OF D W E L L T I M E :
****

PLACENR	PLACENAME	AVERAGE	STAND.DEV.	MAXIMUM	MINIMUM	TOTAL OUTPUT
1	REP	9.9199	9.7361	58.3286	.0098	820
2	WART	4.4309	6.4896	40.9609	0.0000	822
3	ANL	4.9873	4.9014	29.1641	.0039	822
4	P004	0.0000	0.0000	0.0000	0.0000	822
5	BETR	4.9311	4.8663	29.1641	.0049	822
6	P006	7.2026	6.6096	48.9648	.0430	822
7	P007	2.2153	4.2593	30.8135	0.0000	822

2)STATISTICS OF E M P T Y T I M E:
****

PLACENR	PLACENAME	AVERAGE	STAND.DEV.	MAXIMUM	MINIMUM
1	REP	.3642	2.2223	40.9609	.0518
2	WART	.6358	2.8633	36.3564	.0098
3	ANL	.5900	2.5851	35.0156	0.0000
4	P004	1.0000	4.0534	57.6836	.8262
5	BETR	.5947	2.7506	48.9648	.0430
6	P006	.4053	1.9439	29.1641	.0049
7	P007	.8153	3.3813	41.4922	.1484

3)STATISTICS OF M A R K I N G S :
****

PLACENR	PLACENAME	AVERAGE	STAND.DEV.	MAXIMUM	MINIMUM
1	REP	.8205	3.3928	2	0
2	WART	.3642	2.2223	2	0
3	ANL	.4100	1.9619	1	0
4	P004	0.0000	0.0000	1	0
5	BETR	.4053	1.9439	1	0
6	P006	.5947	2.7506	1	0
7	P007	.1847	1.3882	1	0

Bild 13 b: Auswertung des Modells nach Bild 13 a nach einer Simulationszeit von 10000 Zeiteinheiten

Das Modell von Bild 13 ist mit Ausnahme von vereinfachenden Annahmen über die einzelnen Verteilungen nicht mehr analytisch auswertbar. Im übrigen kann das Modell von zwei parallelen Einheiten leicht auf N parallele Einheiten erweitert werden, indem am Anfang der Simulation N Marken auf den Platz WART gesetzt werden. Setzt man mehr als eine Marke auf die Plätze P6 und P7, so erhält man Modelle von m aus n-Zuverlässigkeitsstrukturen.

Bild 14 zeigt ein S-Netzmodell eines Systems mit zwei parallelen und einer seriellen Einheit. Bei seriell verknüpften Einheiten müssen immer alle Einheiten in Betrieb sein. Fällt eine Einheit aus, werden alle anderen Einheiten stillgelegt, solange bis die ausgefallene Einheit repariert ist.

Im oberen Teil von Bild 14 sind die beiden parallelen Anlagen modelliert (ohne Anlaufzeit). Im unteren Teil ist die dritte serielle Einheit dargestellt. Die weitere Spezifikation des Modells und die Auswertung geschehen analog zu obigem Beispiel und sollen hier aus Platzgründen nicht gezeigt werden.

In den letzten Jahren ist im Bereich der booleschen Zuverlässigkeitstheorie eine für die Praxis wichtige Technik zur Analyse der Zuverlässigkeit komplexer Systeme, die Methode der Fehlerbäume, entwickelt worden. Fehlerbäume bestehen aus NICHT, UND und ODER - Logikverknüpfungen (s. auch /Web 74/).

Mit S-Netzen kann man Fehlerbäume modellieren und auswerten. Die UND-Logik ist Grundelement von Petri-Netzen. Auf die Darstellung von NICHT und ODER-Verknüpfungen mit S-Netzen wurde in Kapitel 4.1 hingewiesen. Die Entwicklung und Auswertung von S-Netzmodellen von Fehlerbäumen geschieht dann ähnlich wie bei den Netzplänen in Kapitel 6.3.

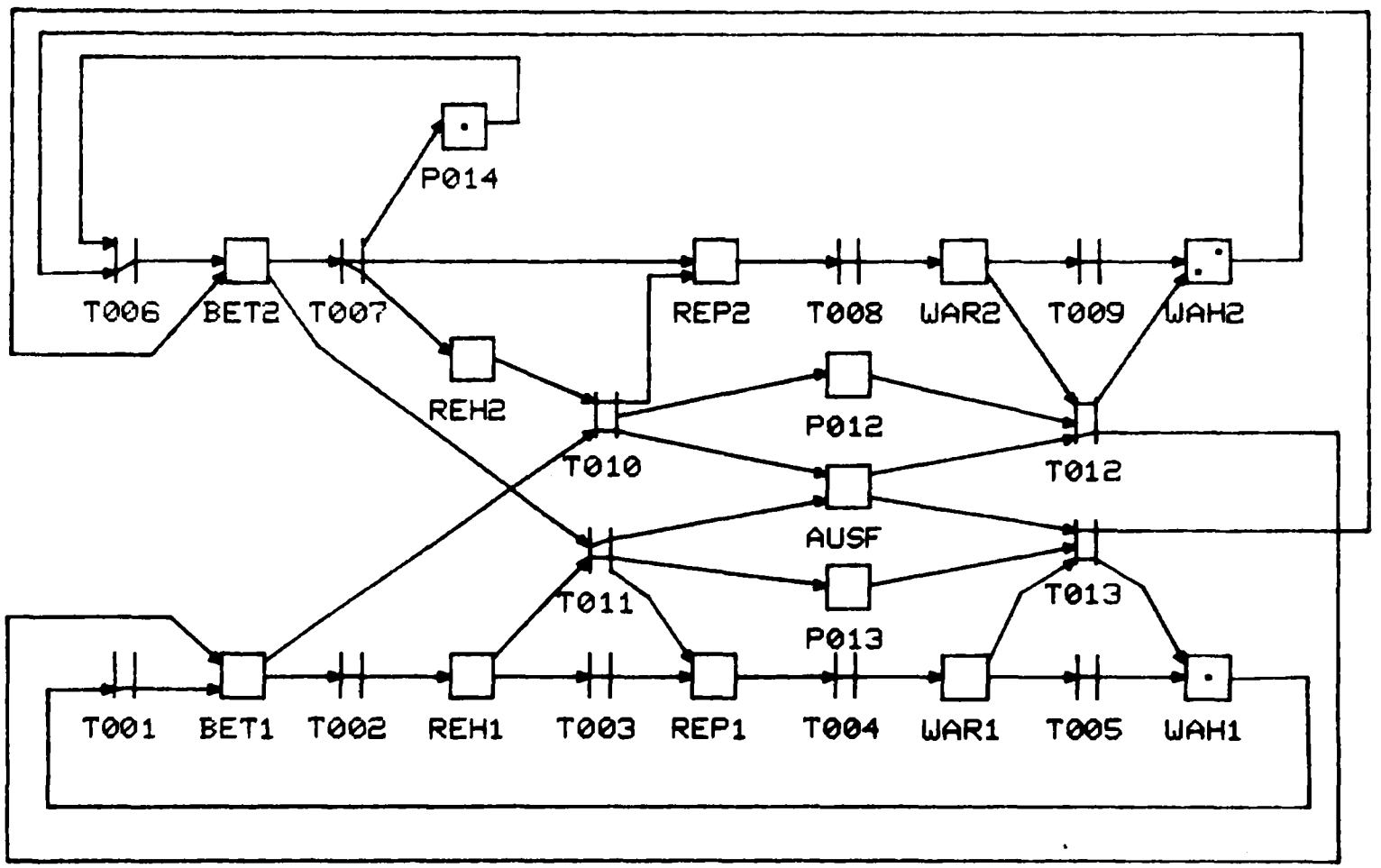


Bild 14: Modell eines Reservesystems mit zwei parallelen Einheiten und einer seriellen Einheit

6.5 Beschreibung eines Realzeitprogramms

Ein einfaches Beispiel aus der Haushaltstechnik, das dem Leser vertraut ist, soll die Beschreibung von Realzeitsystemen mit Simulationsnetzen veranschaulichen. Die Steuerung einer Waschmaschine mit einem Mikrocomputer soll modelliert werden.

Bild 15 a zeigt den Ablauf einer Wäsche. Das S-Netzmodell besteht, von Start- und Endeplätzen abgesehen, nur aus Plätzen mit $t_p > 0$. Die Startmarke soll dabei folgende Parameter haben:

ma(1) : { 1 Hauptwäsche
2 Vorwäsche und Hauptwäsche

ma(2) : Temperatur Vorwäsche

ma(3) : Temperatur Hauptwäsche

ma(4) : Anzahl Spülvorgänge Vorwäsche

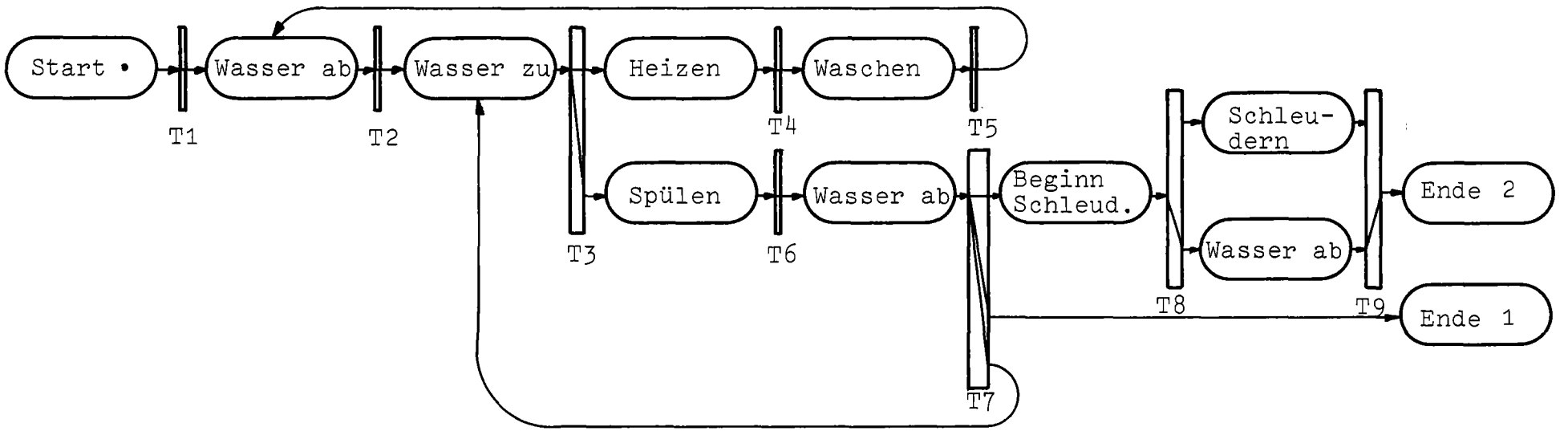
ma(5) : Anzahl Spülvorgänge Hauptwäsche

ma(6) : { 0 kein Schleudern
1 Schleudern

ma(7) : Kennzahl für vorhergehenden Vorgang

Die Werte der Parameter werden vom Waschmaschinenbediener oder intern am Anfang vorbesetzt, d.h. den Parametern der Startmarke werden zu Beginn diese Werte zugeordnet. Das S-Netzmodell zeigt nur den groben Ablauf des Prozesses, dabei werden auch Transitionsprozeduren benutzt, die in mehr verbaler Form beschrieben sind.

In Bild 15 b ist ein grobes Ablaufdiagramm des Realzeitprogramms des Mikrocomputers in S-Netzform dargestellt. Die Markenparameter besitzen dieselbe Interpretation wie oben. Das Programm besteht im wesentlichen aus fünf Unterprogrammen, die jeweils bestimmte Steuerungsaufgaben übernehmen. Jedes Unterprogramm ist durch zwei Plätze und zwei Transitionen repräsentiert. Der erste Platz entspricht dabei der Einsprungmarke des Unterprogramms. Der zweite Platz ist



T5. Proz : ma(1) := ma(1) - 1.

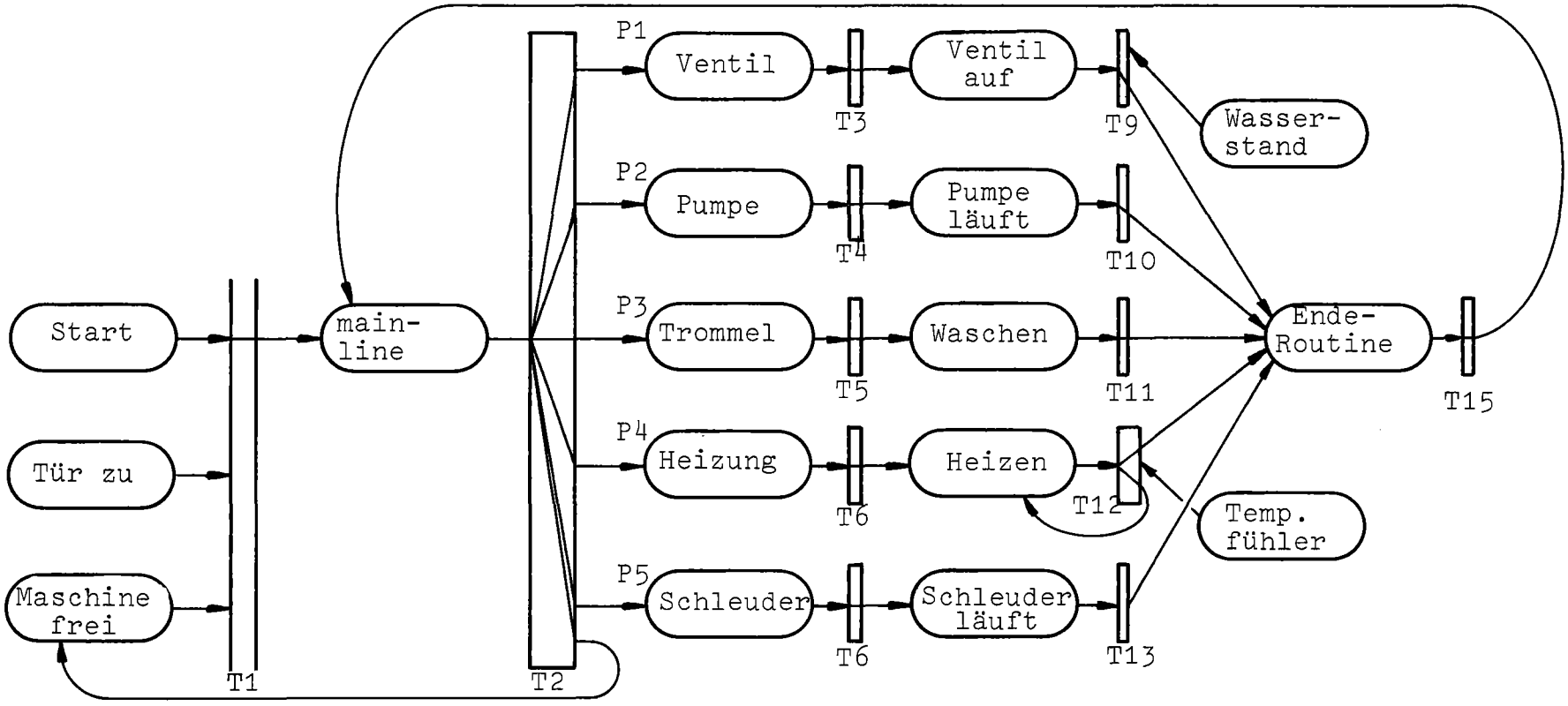
T6. Proz : wenn ma(1) = 2, dann ma(4) := ma(4) - 1, sonst ma(5) := ma(5) - 1.

T3. Platzauswahl : wenn ma(1) = 2 oder (ma(1) = 1 und ma(4) = 0), dann Heizen, sonst Spülen.

T7. Platzauswahl : wenn ma(5) > 0, dann Wasser zu, sonst wenn ma(6) = 1, dann Beginn Schleudern, sonst Ende 1.

außer Start, Beginn Schleudern und Ende 1/2 alles Zeitplätze

Bild 15 a: Simulationsnetzmodell des Ablaufs in einem Waschautomaten (Modell des technischen Prozesses)



T2. Platzauswahl : Springe nach (ma(7)) :

1: wenn ma(1) = 2 oder (ma(1) = 1 und ma(4) = 0), dann P4, sonst P3.
2: wenn ma(4) > 0 oder ma(5) > 0, dann P1, sonst, wenn ma(6) = 1, dann P5, sonst P6.
3: P2.
4: P3.
5: P6.

T3. Proz : Setze ma(7) auf Ventil.
Setze Delay-Parameter auf unendlich.
Öffne Ventil.

T4/5/6/7. Proz entsprechend

T9. Proz : Schließe Ventil.

T10/11/12/13. Proz entsprechend

T12. Platzauswahl : wenn Temperatur ≥ Wert Temp.fühler, dann Heizen, sonst Ende Routine.

Bild 15 b: Simulationsnetzmodell des Realzeitprogramms

ein Platz mit $t_p > 0$ und entspricht einem Vorgang.

Während Bild 15 a ein Modell des zu steuernden Prozesses und Bild 15 b ein grobes Modell der Realzeitsoftware zeigen, ist das Zusammenspiel dieser beiden Teilsysteme durch ein S-Netzmodell in Bild 15 c dargestellt.

Zur Beschreibung des Realzeitsystems wurde aus Platzgründen ein relativ hohes, d.h. grobes Modellierungsniveau gewählt. Durch Verfeinerung bestimmter Plätze und Transitionen läßt sich das Modell jedoch leicht auf die unterste Systemebene ableiten.

Die Auswertung des Modells erfolgt in völlig analoger Weise wie bei den anderen Beispielen und wird hier deswegen nicht gezeigt. Man kann dann anhand der Werte der Verweilzeiten der Marken, der Schalthäufigkeit der Transitionen, der Anzahl der Marken, die einen Platz durchlaufen, usw. den Entwurf überprüfen und gegebenenfalls über Variation einzelner Parameter auch optimieren. Eine Optimierung ist vor allem bei zeitkritischen Steuerungsaufgaben, die in unserem Beispiel jedoch nicht vorliegen, ein wichtiger Aspekt.

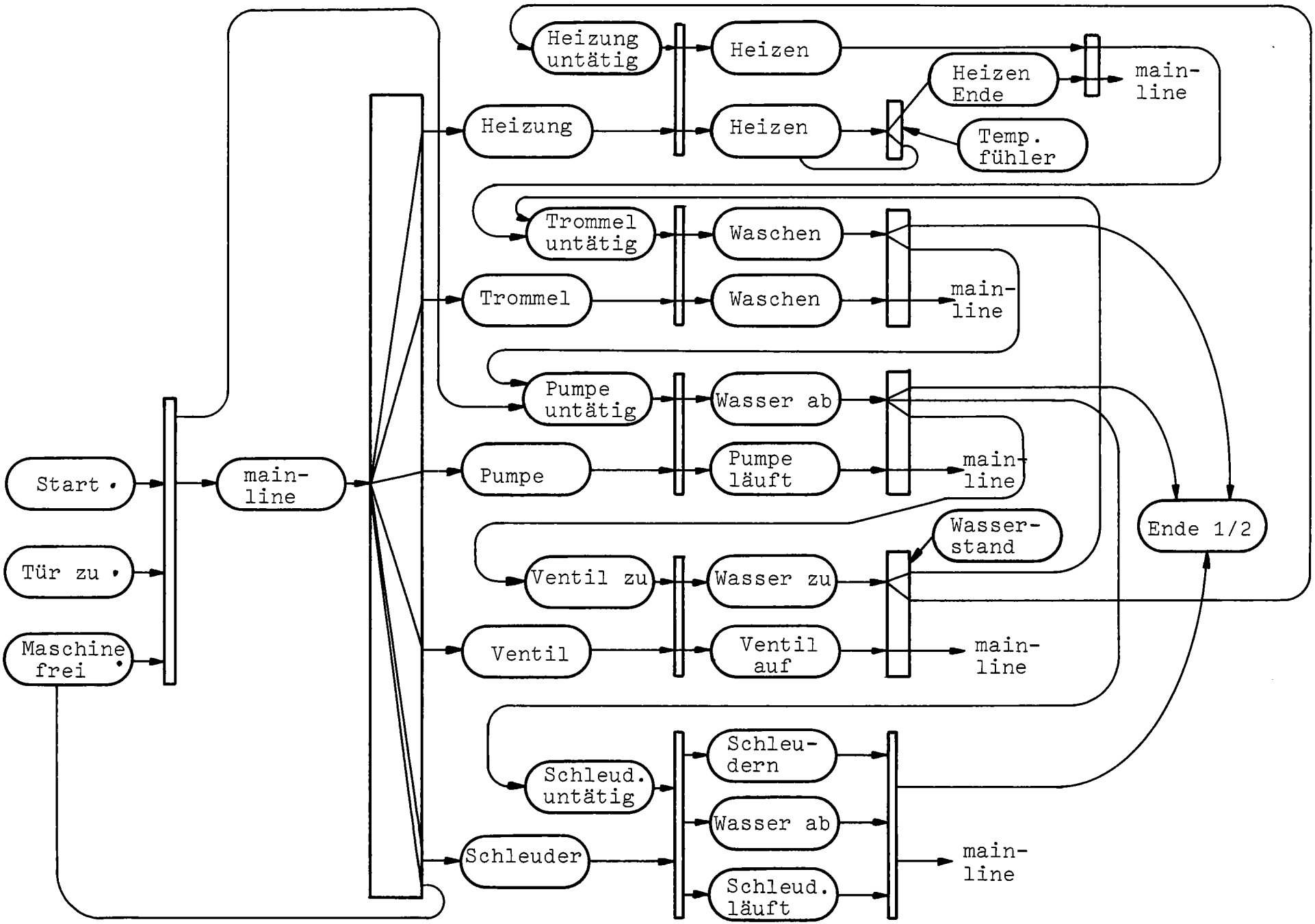


Bild 15 c: Simulationsnetzmodell des Gesamtsystems

6.6 Zusammenfassung

Obige fünf Beispiele geben einen Einblick in die Anwendungsmöglichkeiten von Simulationsnetzen. Um den Rahmen dieses Berichts nicht zu sprengen und das Verständnis des Lesers nicht unnötig zu belasten, wurden die Beispiele einfach gehalten.

Folgende Aussagen können anhand der Ausführungen in Kapitel 4 und der Beispiele getroffen werden:

- Simulationsnetze sind geeignet zur hierarchischen Modellierung. Wegen dieser Eigenschaft kann ein Modellierungsprozeß nach der Top-Down- oder Bottom-Up-Methode durchgeführt werden.
- Spezifikationen können mit S-Netzen festgelegt und nachgeprüft werden. Dieser Sachverhalt beruht im wesentlichen auf der Interpretation eines Simulationsnetzes als Condition-Event-Netz (s. Kap. 4.4 b). Ein Platz entspricht in Verbindung mit seiner Markierung einer Bedingung, die zutrifft, wenn der Platz die entsprechende Markenanzahl enthält. Im ersten Modell in Kapitel 6.1 wurden z.B. die Be/Entladestationen mit der Bedienungskapazität eins spezifiziert. Die Kapazität wurde dann durch einen entsprechenden Platz mit einer Marke dargestellt. Wäre etwa die Kapazität zwei vorgegeben worden, so hätte der Platz zwei Marken erhalten. Ähnlich lassen sich bei S-Netzen dynamische Spezifikationen wie Durchsatz, Antwortzeit usw. festlegen und mittels Simulation nachprüfen.
- Ein gegebenes System läßt sich verschiedenartig modellieren, d.h. es können verschiedene S-Netzmodelle des Systems entwickelt werden, die das gleiche Verhalten besitzen. Diese Aussage ist im Grunde trivial. Jeder Programmierer z.B. weiß, daß man einen Algorithmus verschiedenartig programmieren, sprich modellieren, kann. Nach Erfahrungen des Autors ist dieser Punkt jedoch bei der Anwendung von S-Netzen sehr wichtig. Die Modelle in Bild 11 c bzw. 13 a sehen einfach aus. Sie sind jedoch Resultat einer längeren Modellentwicklung mit mehreren unübersichtlicheren Vorentwürfen. So

ist das Modell in Bild 14 aller Voraussicht nach noch nicht das einfachste und übersichtlichste des zugrundeliegenden Systems. Insgesamt ist nach Meinung des Autors die Modellierung mit Netzen nur durch Erfahrung erlernbar.

- Marken- bzw. Platzauswahlverfahren und Transitionsprozeduren ermöglichen die Darstellung von strukturellen S-Netzkomponenten und umgekehrt. Dieser Sachverhalt soll anhand eines Beispiels veranschaulicht werden: Das folgende Bild 16 zeigt zwei Darstellungsmöglichkeiten der Platzauswahl "zyklische Reihenfolge", die zum einen semantisch und zum anderen strukturell mittels Netzkomponenten

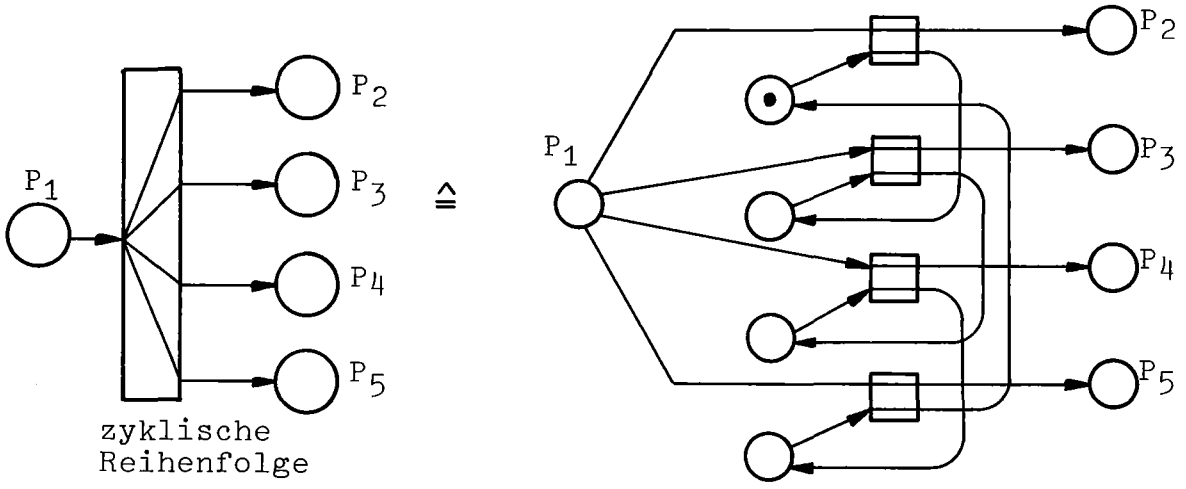


Bild 16: Modellierung des Platzauswahlverfahrens "zyklische Reihenfolge"

modelliert wird. Ähnlich kann man auch andere Marken- oder Platzauswahlverfahren in struktureller Form modellieren. Umgekehrt lassen sich in vielen Anwendungen komplizierte und unübersichtliche Teilstrukturen eines S-Netzmodells durch Einführung von Auswahlverfahren oder auch Transitionsprozeduren vereinfachen. Diese Beziehung zwischen Semantik und Syntax ist in der praktischen Anwendung gleich

der Wahl zwischen graphischer oder alphanumerischer Darstellung, da Auswahlverfahren und Transitionsprozeduren im Rahmen einer höheren Programmiersprache wie ALGOL oder FORTRAN implementiert werden müssen (s. Kap. 4.2).

- Die Grenze der Anwendbarkeit von S-Netzen wird durch die Rechenzeit für die Modellauswertung bestimmt. Die Simulation von S-Netzen benötigt eine Ausführungszeit von etwa 2 msec pro Schalten einer Transition beim SIMULA-Programm von Ramöller /Ram 76/ bzw. ca. 0.5 msec pro Transitionsschaltung beim FORTRAN-Programm von Albrecht /Alb 77/ auf einer IBM 370/168. Die Durchschnittswerte ergaben sich bei mehreren S-Netzmodellen. Mit diesen Werten erhält man einen weiteren Vorteil bei der Anwendung von Simulationsnetzen: Der Modellierer kann schon anhand des Modells eine grobe Abschätzung der voraussichtlichen Ausführungszeit der Modellauswertung erhalten. Unter diesem Aspekt läßt sich damit auch die Frage nach der optimalen Modellierungstiefe beantworten.

Obige Aussagen zeigen einige Gesichtspunkte bei der Anwendung von Simulationsnetzen auf. Ein Teil der behandelten Punkte muß in Zukunft noch intensiver untersucht werden.

7. Schlußbemerkungen

Es wurde ein Verfahren beschrieben, das zur Modellierung und Auswertung allgemeiner diskreter dynamischer Systeme geeignet ist. Der Bericht enthält neben anderem:

- eine Definition von Begriffen und Eigenschaften diskreter dynamischer Systeme (ddSysteme) unter Einbeziehung des Zeitparameters, wie 1/2/3-Indeterminiertheit, Dauer, Prozeßphasen, usw.
- eine Beschreibungs- und Modellierungsmethode, die auf Petri-Netzen und der rechnergestützten Simulation aufbaut und vor allem folgende Vorteile bietet:
  - geeignet für allgemeine ddSysteme und damit nicht beschränkt auf spezielle Anwendungen (Universalität),
  - graphische Struktur und damit übersichtliche Darstellung der Systemelemente und -restriktionen,
  - leicht verständlich und einfach benutzbar, da die Definition des Verfahrens auf wenigen Seiten möglich ist (kein umfangreiches Handbuch),
  - geeignet zur hierarchischen Modellierung und damit bei komplexen ddSystemen anwendbar.
- eine Zusammenstellung von Meßgrößen, die eine umfassende Analyse von ddSystemen ermöglichen. Die Meßgrößen können mit Verfahren der Graphentheorie, Netztheorie und der rechnergestützten Simulation berechnet werden.
- eine Übertragung von Ergebnissen der Petri-Netztheorie.
- Anwendungen der Methode zur Beschreibung und Auswertung von
  - Simulationsmodellen,
  - Realzeitprogrammen,
  - diskreten technischen Prozessen.

Insgesamt können die Ergebnisse dieser Arbeit als Grundlage für ein rechnergestütztes Planungssystem für DV-Anwendungen betrachtet werden. Sie bilden eine Verbindung zwischen den Arbeiten auf dem Gebiet der Theorie der parallelen Prozesse und praktischen Anforderungen aus der Informatik und benachbarten

Fachgebieten. In Zukunft gilt es, diese Verbindung, vor allem zwischen der Theorie der Petri-Netze und den Simulationsnetzen, weiter zu vertiefen.

8. Literaturverzeichnis

/AdCS 72/ Advanced Course on Software Engineering
Technische Universität München, 21. Feb.-4. März 1972

/Alb 77/ H. Albrecht:
Berechnung von Auswertungsgrößen modifizierter Petrinetze.
Diplomarbeit, Universität Karlsruhe, 1977

/Baer 73/ J.L. Baer:
A Survey of Some Theoretical Aspects of Multiprocessing.
Computing Surveys, Vol. 5, No. 1, März 1973, S.31-80

/Bau 71/ F.L. Bauer, G. Goos:
Informatik.
Springer-Verlag, Berlin, 1971

/Berg 73/ C. Berge:
Graphs and Hypergraphs.
North-Holland Publishing Comp., London, 1973

/Boe 73/ B.W. Boehm:
Software and its Impact: A Quantitative Assessment.
Datamation, Mai 1973, S.48-59

/Cerf 72/ V. Cerf:
Multiprocessors, Semaphores and a Graph Model of Computation.
UCLA Technical Report No. UCLA-Eng-7723,
University of California, Los Angeles, 1972

/Coug 74/ J.D. Couger, R.W. Knapp:
System Analysis Techniques.
J. Wiley, London, 1974

/Crow 76/ C.P. Crowley:
Analysis and Simulation of Graph Models.
Ph.D. Thesis, University of Washington, Seattle, Washington, Technical Report No. 76-03-04, 1976

/Deus 74/ P. Deussen:
Description of Processes.
in A. Günther (Hrsg.): International Computing Symposium 1973, North Holland Publishing Comp., 1974, S.11-17

/Dijk 68/ E.W. Dijkstra:
Co-operating Sequential Processes.
in F. Genuys (Hrsg.): Programming Languages,
Academic Press, New York, 1968

/Hack 73/ M. Hack:
A Petri Net Version of Robin's Undecidability Proof for Vector Addition Systems.
Computation Structures Group Memo 94, Project MAC, M.I.T., Cambridge, Massachusetts, Dez. 1973

/Hack 75/ M.H. Hack:
Petri Net Languages.
Computation Structures Group Memo 124, Project MAC, M.I.T., Cambridge, Massachusetts, Juni 1975

/Hack 76/ M. Hack:
The Equality Problem for Vector Addition Systems is Undecidable.
Computation Structures Group Memo 121, Project MAC, M.I.T., Cambridge, Massachusetts, Juni 1976

/Holt 68/ A.W. Holt, u.a.:
Information System Theory Project.
Technical Report No. RADC-TR-68-305, Applied Data Research Inc., AD 676972, Sept. 1968

/Horn 73/ J.J. Horning, B. Randell:
Process Structuring.
Computing Surveys, Vol. 5, No. 1, März 1973, S.5-30

/IEEE 77/ IEEE Transactions on Software Engineering
Vol. SE-3, No. 1, Jan. 1977

/Kalm 69/ R.E. Kalman, P.L. Falb, M.A. Arbib:
Topics in Mathematical System Theory.
McGraw-Hill Comp., London, 1969

/Karp 69/ R.M. Karp, R.E. Miller:
Parallel Program Schemata.
Journal of Computer and Systems Sciences: 3(1969), S.147-195

/Kosa 73/ S.R. Kosaraju:
Limitations of Dijkstra's Semaphore Primitives and Petri Nets.
Operating Systems Review, Vol. 7, No. 4, Okt. 1973, S.122-126

/Kell 72/ R.M. Keller:
Vector Replacement Systems: A Formalism for Modeling Asynchronous Systems.
Technical Report 117, Princeton University, Princeton, N.J., Dez. 1972

/Köch 72/ D. Köcher, u.a.:
Einführung in die Simulationstechnik.
Deutsche Gesellschaft für Operations Research, DGOR-Schrift, Nr. 5, 1972

/Laut 75/ K. Lautenbach:
Lebendigkeit in Petri-Netzen.
unveröffentlicht, 1975

/Lock 74/ P.C. Lockemann:
Informationssysteme.
Vorlesungsmanuskript, Universität Karlsruhe, Sept.1974

/Mayr 76/ H.C. Mayr, P.C. Lockemann:
Formal Modelling of Discrete Dynamic Systems.
Proc. 2nd International Workshop on Modelling and Performance Evaluation of Computer Systems, North-Holland Publ. Comp., 1976

/McC 76/ Th.J. McCabe:
A Complexity Measure.
IEEE Transactions on Software Engineering, Vol. SE-2, No. 4, Dez. 1976, S.308-320

/Mill 73/ R.E. Miller:
A Comparison of Some Theoretical Model of Parallel Computation.
IBM Research Report RC 4230, Febr. 1973

/Mill 74/ R.E. Miller:
Some Relationships between various Models of Parallelism and Synchronization.
IBM Research Report RC 5074, Okt. 1974

/Neu 75/ K. Neumann:
Operations Research Verfahren, Band I, II, III.
Carl Hanser Verlag, München, 1975

/Noe 71/ J.D. Noe:
A Petri Net Model of the CDC 6400.
Proceedings of ACM Workshop on Systems Performance Evaluation, Harvard University, 1971, S.362-378

/Noe 72/ J.D. Noe, G.J. Nutt:
Validation of a Trace-driven CDC 6400 Simulation.
Proceedings of the Spring Joint Computer Conference, Vol. 40, 1972, S.749-757

/Noe 73/ J.D. Noe, G.J. Nutt:
Macro E-Nets for Representation of Parallel Systems.
IEEE Transactions on Computers, Vol. C-22, No. 8, Aug. 1973, S.718-727

/Noe 75/ J.D. Noe:
Pro-Nets: For Modeling Processes and Processors.
University of Washington, Seattle, Washington, Technical Report No. 75-07-15, Juli 1975

/Nutt 72/ G.J. Nutt:
The Formulation and Application of Evaluation Nets.
Ph. D. Thesis, University of Washington, Seattle, Washington, Technical Report No. 72-07-02, 1972

/Pat 70/ S.S. Patil:
Coordination of Asynchronous Events.
Technical Report No. TR-72, M.I.T., Project MAC, Cambridge, Massachusetts, Juni 1970

/Pet 73/ J.L. Peterson:
Modelling of Parallel Systems.
Ph. D. Thesis, Stanford University, Stanford, California, Dez. 1973

/Pet 74/ J.L. Peterson, Th.H. Bredt:
A Comparison of Models of Parallel Computation.
in J. Rosenfeld (Hrsg.): Information Processing 74, North-Holland Publishing Comp., 1974, S.466-470

/Pet 77/ J.L. Peterson:
Petri Nets.
Computing Surveys, Vol. 9, No. 3, Sept. 1977, S.223-252

/Petr 62/ C.A. Petri:
Kommunikation mit Automaten.
Doktorarbeit, Universität Bonn, 1972

/Petr 73/ C.A. Petri:
Concepts of Net Theory.
Proceedings of the Symposium and Summer School on Mathematical Foundations of Computer Science, High Tatras, Sept. 1973, Math. Inst. Slovak Academy of Science, 1973, S.137-146

/Proc 76/ Proceedings of the 2nd International Software Engineering Conf., San Franzisco, Okt. 1976

/Ram 74/ Ch. Ramchandani:
Analysis of Asynchronous Concurrent Systems by Petri Nets.
Technical Report No. TR-120, M.I.T., Project MAC-TR-120, Cambridge, Massachusetts, Febr. 1974

/Ram 76/ R. Ramöller:
Implementierung eines erweiterten Petrinetzmodells zur Simulation diskreter Systeme.
Diplomarbeit, Universität Karlsruhe, 1976

/Raun 77/ W.G. Rauneker:
Interaktives graphisches Modellierungssystem zur Simulation modifizierter Petrinetze auf einem Kleinrechner.
Diplomarbeit, Universität Karlsruhe, 1977

/Rose 72/ C.W. Rose:
LOGOS and the Software Engineer.
Proceedings of the Fall Joint Computer Conf.,
Vol. 41, 1972, S.311-323

/Schu 76/ F. Schumacher:
Vereinfachung von Simulationsstudien:
Definition eines Simulationsnetzes.
unveröffentlicht, Aug. 1976

/Schu 77 A/ F. Schumacher
Modeling and Simulation of Computer Systems.
Third Winterschool on the Theory of Operating
Systems, Visegrad, Ungarn, Jan. 1977

/Schu 77 B/ F. Schumacher:
Modelling and Evaluation of Discrete Dynamic
Systems with Extended Petri Nets.
XXIII International Meeting of the Institute of
Management Science, Athens, Juli 1977

/Web 74/ G. Weber:
Einführung in Methoden und Probleme der Zuverlässigkeit.
KFK 1811, Kernforschungszentrum Karlsruhe, Jan. 1974

/Whit 73/ G.E. Whitehouse:
Systems Analysis and Design Using Network Techniques.
Prentice-Hall, New Jersey, 1973

/Zima 76/ H. Zima:
Betriebssysteme: Parallele Prozesse.
Bibliographisches Institut, Mannheim, 1976

/Zade 69/ L.A. Zadeh, E. Polak:
System Theory.
McGraw-Hill Comp., London, 1969