Prozesse, Simulation und Eigenschaften netzmodellierter Systeme

Jörg Desel, Thomas Freytag, Universität Karlsruhe, Institut AIFB
jde|tfr@aifb.uni-karlsruhe.de

Kurzfassung

Dieser Beitrag berichtet aus einem DFG-Projekt, dessen Ziel die Validierung und Überprüfung dynamischer Eigenschaften von mit höheren Petrinetzen modellierten Systemen ist. Das Grundkonzept basiert nicht auf Sequenzen nacheinander stattfindender Systemereignisse, sondern auf der Erzeugung halbgeordneter Abläufe. Auf dieser Basis können auch temporallogische Anfragen an das System ausgewertet werden.

Schlüsselworte

Petrinetz, Simulation, Halbordnung, Prozeß.

1 Einleitung

Petrinetze modellieren dynamische verteilte Systeme mit diskreten Aktionen. Sie werden innerhalb der Automatisierungstechnik in verschiedenen Bereichen angewendet (siehe z.B. [Sc95] und frühere Tagungen).

Bei der Beschreibung des Verhaltens eines Petrinetzes gibt es traditionell zwei unterschiedliche Sichtweisen: Abläufe können entweder als Sequenzen zeitlich nacheinander geschehender Ereignisse bzw. Ereignismengen oder als kausal halbgeordnete Mengen von Ereignissen dargestellt werden. Bei der erstgenannten Sichtweise ist die *sequentielle Semantik* eines Netzmodells durch die Menge seiner *Schalt-* oder *Schrittfolgen* gegeben. Bei der zweiten Sichtweise definiert die Menge der *Prozesse*, d.h. der passend beschrifteten *Kausalnetze*, die *kausale Semantik*. Während schon in frühen Arbeiten der Netztheorie – insbesondere durch ihren Begründer Carl Adam Petri – kausale Semantik untersucht und propagiert wurde, hat sich im Bereich der Anwendungen fast ausschließlich die sequentielle Semantik durchsetzen können. Dies ist im wesentlichen dadurch zu begründen, daß die Definition der sequentiellen Semantik wesentlich einfacher ist und sie die Intuition von verteilten Systemen als spezielle sequentielle Systeme unterstützt. So kann eine Schaltfolge als Sequenz globaler Zustände und Transformationen zwischen diesen Zuständen interpretiert werden, ganz in Analogie zu Modellen sequentieller Systeme wie sprachengenerierende endliche Automaten.

In jüngeren Arbeiten wurde die kausale Semantik als anwendungsrelevant "wiederentdeckt" (siehe z.B. [Esp94]), allerdings geht es in diesen Arbeiten ausschließlich um die Ausnutzung der Halbordnungen zur Effizienzsteigerungen von Analyse- und Verifikationsalgorithmen, insbesondere für Software. Andere Arbeiten verwenden kausale Semantik für die Repräsentation und Verifikation verteilter Algorithmen [Rei95, WVV96].

Die überwiegende Zahl der Anwendungen von Petrinetz-Technologie im Bereich der Automatisierungstechnik, verwendet das Modell Petrinetz nicht ausschließlich als effiziente inter-

ne Struktur eines Verifikationsprogramms, sondern ausdrücklich als ausführbares Modell. Dieses Modell wird zur Spezifikation, Dokumentation und Kommunikation geplanter oder existierender Systeme verwendet. Die unmittelbare Ausführbarkeit erlaubt die Darstellung möglichen Systemverhaltens durch Visualisierung des Markenspiels und die automatische Simulation größerer Systemausführungen. Das Ziel von Visualisierung und Simulation ist i.a. nicht der Beweis der Korrektheit bezüglich einer formal angegebenen Eigenschaft, sondern eine möglichst weitgehende Validierung des Modells bezüglich der individuellen Vorstellung des geplanten Verhaltens bzw. einiger relevanter Eigenschaften (vergleichbar mit Testläufen von Programmversionen).

Gegenstand dieser Arbeit ist die Verwendung von kausaler Semantik zur Visualisierung und Simulation. Einige Vorteile dieses Ansatzes werden geschildert. Diese Vorteile sind zum Teil qualitativer Natur, d.h. nur mit Hilfe kausaler Semantik können gewisse wichtige Eigenschaften unmittelbar ausgedrückt und untersucht werden. Darüberhinaus führt die Verwendung von kausaler Semantik zu quantitativen Vorteilen: Im Gegensatz zur Simulation sequentieller Abläufe wird i.a. auf sehr viel effizientere Weise ein relevanter Ausschnitt des möglichen Systemverhaltens dargestellt. Das von der DFG geförderte Projekt *Verifikation von Informationssystemen durch Auswertung halbgeordneter Petrinetz-Abläufe (VIP)* [DO95, Fre96] beschäftigt sich mit dem genannten Themenbereich und verwandten Fragestellungen. Der vorliegende Aufsatz stellt insofern einen Bericht über dieses laufende Projekt dar.

Die Arbeit ist wie folgt gegliedert:

Im *zweiten Kapitel* wird die Verwendung von Prozessen zur Beschreibung kausaler Semantik im Rahmen der Simulation netzmodellierter Systeme genauer angegeben. Insbesondere werden die prinzipiellen Vorteile eines derartigen Visualisierungs- bzw. Simulationskonzeptes im Gegensatz zu herkömmlichen, auf Sequenzen basierenden Ansätzen herausgestellt. Das *dritte Kapitel* enthält einige notwendige formale Definitionen. Im *vierten Kapitel* werden einige Algorithmen zur Überprüfung von Eigenschaften in halbgeordneten Abläufen skizziert. Das *fünfte Kapitel* enthält neben grundsätzlichen übelegungen einen konkreten Algorithmus zur Erzeugung einer halbgeordneten Simulation. Schließlich gibt das *sechste Kapitel* einen kurzen überblick über das Software-Werkzeug ***VIPtool***, in dem die theoretischen Konzepte prototypisch implementiert sind.

2 Prinzipien halbordnungsbasierter Simulation

Das Verhalten eines diskreten verteilten Systems konstituiert sich aus seinen *Abläufen*. Jeder Ablauf besteht aus einer Menge von (evtl. teilweise ununterscheidbaren) *Systemereignissen*. Die Ereignisse sind nur unter gewissen Voraussetzungen möglich, d.h. sie haben *Vorbedingungen*. Durch das Eintreten eines Ereignisses werden andere Bedingungen erfüllt; das sind die *Nachbedingungen* des Ereignisses. Zeitpunkte, zu denen Ereignisse geschehen, haben keinen unmittelbaren Einfluß auf das Verhalten, wenn dieser Einfluß nicht ausdrücklich z.B. in Form von Abhängigkeiten von Uhren modelliert wurde. Die Kombination von Ereignissen mit ihren Vor- und Nachbedingungen erzeugt die kausale Struktur eines Ablaufs. Diese Struktur läßt sich z.B. in Form eines *Kausalnetzes* beschreiben. Die Nachbedingung eines Ereignisses kann zugleich Vorbedingung eines anderen Ereignisses sein; die Ereignisse sind dann kausal geordnet. Beim Fehlen einer Kette derartiger unmittelbarer Abhängigkeitsbeziehungen geschehen zwei Ereignisse *nebenläufig*. Will man nebenläufige

Ereignisse auf eine Zeitachse projizieren, können sie zugleich oder in beliebiger Reihenfolge stattfinden. Im Gegensatz zur sequentiellen Semantik soll nicht mithilfe derartiger Projektionen eine Vielzahl von Abläufen konstruiert werden, sondern *ein* Kausalnetz soll *einen* Systemablauf darstellen. Kausalnetze, deren Elemente mittels geeigneter Abbildungen zu den Elementen eines Systemnetzes in Beziehung stehen, heißen Prozesse des Systemnetzes. Die kausale Semantik eines Petrinetzes ist durch die Menge seiner Prozesse gegeben.

Die Simulation von Systemverhalten unter Verwendung kausaler Semantik hat den offensichtlichen Nachteil, daß Abläufe nicht durch Sequenzen modelliert bzw. durch eine sequentielle Darstellung von Schaltvorgängen visualisiert werden können, sondern durch komplexere kausale Strukturen repräsentiert werden. Wir wollen diesem Nachteil eine Reihe von Vorteilen entgegensetzen, die in diesem Kapitel kurz skizziert werden.

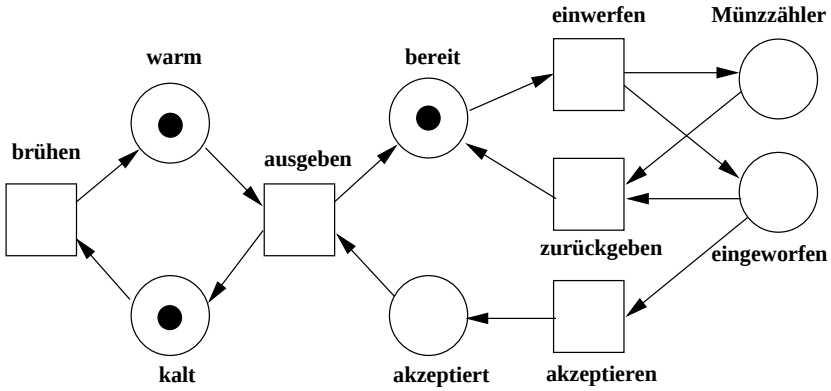

Abbildung 1: Ein Getränkeautomat

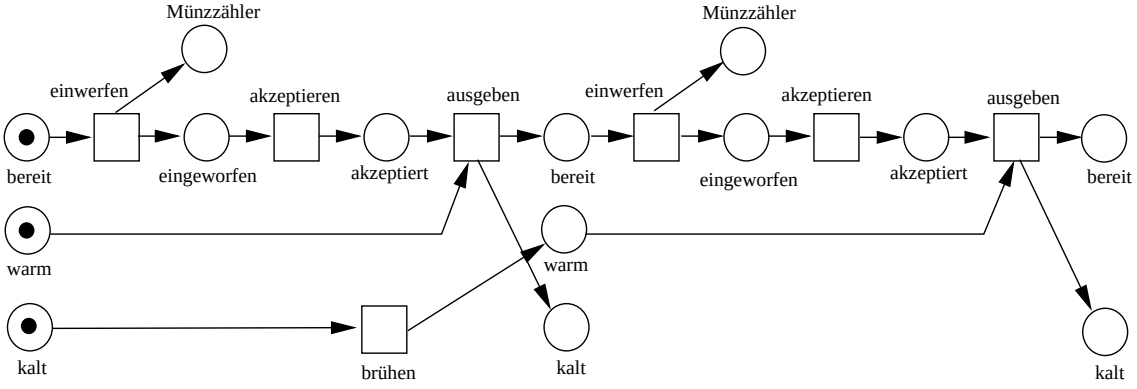

Abbildung 2: Ein Ablauf für den Getränkeautomat aus Abbildung 1

Zur Illustration wird das in Abbildung 1 dargestellte Stellen/Transitions-Petrinetz verwendet. Ein Ablauf in Form eines Prozesses ist in Abbildung 2 angegeben. Wir verzichten hier auf formale Definitionen und verweisen stattdessen auf die Definitionen höherer Petrinetze in Kapitel 3. Formal lassen sich Stellen/Transitions-Netze als spezielle höhere Petrinetze interpretieren, wobei alle Stellen einen einelementigen Wertebereich besitzen und zu jeder Kante eine (nicht dargestellte) Variable x gehört.

2.1 Repräsentation kausaler Abhängigkeiten

Prozesse repräsentieren die kausalen Abhängigkeiten zwischen Ereignissen eines Systemablaufs. Im Ablauf aus Abbildung 2 werden zwei Münzen eingeworfen, akzeptiert, und es wird jeweils ein Getränk ausgegeben. Ein weiteres Getränk wird gebrüht; dieses wird das nächste ausgegebene Getränk. Eine mögliche Schaltfolge für denselben Ablauf ist:

$$ein, akz, bru, aus, ein, akz, aus$$

Durch die Einordnung des Ereignisses *bru* ("brühen") geht die Unterscheidung zwischen kausaler Abhängigkeit und zufälliger Nacheinanderausführung auf Ausführungsebene verloren. So ist bei der angegebenen Schaltfolge unklar, ob zunächst das bereits anfangs warme Getränk ausgegeben wird. Gerade Fragestellungen wie

- *Wird jemals das "kältere" Getränk ausgegeben?*
- *Bleibt ein gebrühtes Getränk für immer ungenutzt?*

sind also mittels sequentieller Semantik nicht zu beantworten, da diese Information in Schaltfolgen nicht zur Verfügung steht.

Dieses Beispiel soll auch demonstrieren, daß der Fluß von Waren und Informationen durch Pfade in den Kausalnetzen dargestellt wird. Dadurch wird das Verhalten des Systems transparenter repräsentiert. Ein Anwender, der sich mit der Notation vertraut gemacht hat, kann an Prozeßen ablesen, ob er das richtige Modell konstruiert hat.

Für die automatische Erzeugung von Kausalnetzen werden im Rahmen des Projekts *VIP* heuristische Plazierungsalgorithmen entwickelt. Abhängig vom Anwendungsbereich ist zudem eine Darstellung mittels spezieller Symbole denkbar, die die Akzeptanz und Lesbarkeit von Prozessen erhöht.

2.2 Effizienz des Simulationskonzeptes

Der in Abbildung 2 dargestellte Ablauf beinhaltet verhältnismäßig wenig Nebenläufigkeit. Nur das Ereignis *bru* ist nebenläufig zu fast allen anderen Ereignissen. Insofern ist die Darstellung des Prozesses nicht sehr unterschiedlich von entsprechenden sequentiellen Darstellungen. Neben der oben angegebenen Schaltfolge existieren allerdings fünf weitere Schaltfolgen zu diesem Prozeß: Das Ereignis *bru* läßt sich an sechs verschiedene Stellen zwischen bzw. vor die anderen Ereignisse einordnen. Formal kann die Zuordnung zwischen kausaler und sequentieller Semantik durch das Markenspiel auf Kausalnetzen beschrieben werden: Jeder Schaltfolge, die von der angegebenen *trivialen* Anfangsmarkierung des Kausalnetzes aktiviert wird, entspricht auf kanonische Weise eine zu dem Prozeß gehörige Schaltfolge des Systemnetzes. Umgekehrt läßt sich jede Schaltfolge des Systemnetzes durch eine Schaltfolge wenigstens eines Prozesses erzeugen.

In diesem Sinne werden in unserem Beispiel durch *einen* Prozeß *sechs* Schaltfolgen repräsentiert. Dieses Verhältnis nimmt bei zunehmendem Grad an Nebenläufigkeit noch drastisch zu. So kann bei terminierenden verteilten Systemen das vollständige Verhalten meist durch eine sehr viel kleinere Anzahl von Prozessen als bei einer Verwendung von Schaltfolgen dargestellt werden. Auch wenn – wie bei einer Simulation üblich – nur ein kleiner Ausschnitt

des gesamten Verhaltens betrachtet wird, ist dieser relative Ausschnitt bei der Konstruktion von Prozessen i.a. ungleich größer als der relative Ausschnitt bei derselben Zahl von Schaltfolgen. Eine entsprechende Aussage gilt, falls das betrachtete System unendlich viele und unendlich große Prozesse und Schaltfolgen besitzt.

Diese Effizienz der Verhaltensrepräsentation durch Prozesse hat unmittelbare Vorteile bei der Validierung des Systemverhaltens. Darüberhinaus führt sie in Zusammenhang mit den in den folgenden Abschnitten genannten Analyseverfahren zu einem Effizienzvorteil gegenüber Verfahren, die auf sequentieller Semantik basieren.

2.3 Analyse von invarianten Eigenschaften

Oftmals ist man an Eigenschaften von Systemen interessiert, die für alle erreichbaren Systemzustände erfüllt sein sollen. Der wechselseitige Ausschluß zweier kritischer Bereiche ist ein bekanntes Beispiel. Diese Eigenschaften betreffen Markierungen einer oder einiger Stellen des Systems. Die Überprüfung einer Eigenschaft mittels sequentieller Semantik erfordert die Überprüfung aller durch Schaltfolgen erreichbarer Markierungen auf die Gültigkeit der Eigenschaft. Bei der Verwendung von Prozessen können wir – über die meist geringere Anzahl von Prozessen hinaus – effizientere Verfahren entwickeln. So muß bei dem Beispiel des wechselseitigen Ausschlusses nur gewährleistet werden, daß zwei entsprechende Stellen niemals zusammen markiert werden können. Das ist genau dann der Fall, wenn in keinem Prozeß zwei Bedingungen existieren, die jeweils der Markierung dieser Stellen entsprechen, und die nebenläufig, also nicht kausal geordnet sind. Anstatt alle möglichen erreichbaren globalen Zustände zu konstruieren, reicht also die Konzentration auf entsprechend beschriftete lokale Bedingungen in Kausalnetzen aus. Es ist leicht einzusehen, daß dadurch bei zunehmender Nebenläufigkeit des Systems große Effizienzvorteile für die kausale Semantik impliziert werden. Bei zunehmender Anzahl von Stellen in der Invarianten nimmt diese Vorteil i.d.R. ab.

Wir können hier die folgenden theoretischen Zusammenhänge ausnutzen: Zu jeder Menge paarweise ungeordneter Bedingungen eines Prozesses existiert eine von der trivialen Anfangsmarkierung (die nur die minimalen Bedingungen markiert) erreichbare Markierung (ein *Schnitt*), der wenigstens alle Bedingungen der betrachteten Menge enthält. Dieser Schnitt korrespondiert zu einer durch dieselbe Schaltfolge im Systemnetz erreichbaren Markierung. Umgekehrt findet man zu jeder erreichbaren Markierung eine passende maximale Menge ungeordneter Bedingungen in einem Prozeß. Um das Vorkommen von Marken auf einer Stellenmenge in einer globalen Markierung zu überprüfen, kommt man also mit der Analyse von Ordnungsbeziehungen zwischen lokalen Bedingungen aus.

2.4 Analyse von Kausalketten

Während invariante Eigenschaften statische Fehlersituationen ausschließen, lassen sich dynamische Fehlersituationen auf diesem Weg nicht identifizieren. Es gibt Ansätze, z.B. Sequenzen erreichbarer Zustände als fehlerhaft auszuzeichnen [Sa94]. Das eigentlich gemeinte Fehlerkriterium hängt meist nur ab von lokalen Bedingungen oder Ereignissen, nicht aber von unabhängigen anderen Systemkomponenten. So mag in unserem Beispielnetz abgefragt werden, ob dreimal hintereinander eine Münze zurückgegeben wurde. Diese Sequenz ist in einer Schaltfolge nicht unmittelbar zu finden, denn zwischen den Ereignissen *ein* und *zur*

kann jeweils noch das Ereignis *bru* stattfinden. I.a. können unabhängige Teilstücke, die das Wiederfinden des gesuchten Musters erschweren, sehr viel komplexer sein. Im Prozeß dagegen wird die gesuchte Sequenz durch ein Teilnetz von sechs aufeinanderfolgenden Ereignissen dargestellt, die abwechselnd mit *ein* und *zur* beschriftet sind. Umgekehrt lassen sich ähnliche Aussagen mit Hilfe sequentieller Semantik nicht überprüfen, wenn sie auf kausalen Zusammenhängen beruhen. Als Beispiel betrachte man die Eigenschaft, daß kein Getränk zweimal gebrüht wird, bevor es ausgegeben wird. Diese Eigenschaft ist im Systemmodell erfüllt, aber es gibt Schaltfolgen, die zwei aufeinanderfolgende Ereignisse *bru* enthalten.

2.5 Analyse von Zielen

Systeme sollen i.a. gewisse Aufgaben schließlich erfüllen. Daher ist es sinnvoll, von einem System nicht nur die Vermeidung von Fehlersituationen zu fordern, sondern auch das Erreichen von *Zielen*. Wir wollen hier eine spezielle Klassen von Zielen betrachten:

- *Wann immer eine Stelle s markiert ist, wird später eine Stelle s' markiert sein*

In der Sprache der Prozesse läßt sich dasselbe wie folgt formulieren:

- *Für jede zu s gehörende Bedingung existiert eine kausal spätere zu s' gehörende Bedingung*

In unserem Beispiel könnte eine entsprechende Forderung sein:

- *Wann immer eine Münze akzeptiert wurde (akz markiert ist), wird ein Getränk ausgegeben und danach „ber“ markiert}*

Die Interpretation von Zielen über Prozesse stellt sicher, daß Nachbedingungen nebenläufiger und in Schaltfolgen zufällig hintereinandergeordneter Ereignisse nicht als gültiges Erreichen eines Ziels betrachtet werden – es gibt in diesem Fall stets auch die umgekehrte Reihenfolge, bei der das Ziel nicht erreicht wird. In diesem Sinn repräsentiert ein Prozeß die Information des *worst case* aller seiner zugeordneten Schaltfolgen.

Ziele werden nur erreicht, wenn aktivierte Transitionen auch jemals schalten, und das System nicht z.B. unnötig in seinem Anfangszustand verharrt. Um das Erreichen von Zielen zu analysieren, müssen also gewisse *Progreßannahmen* an die Transitionen gestellt werden. Diese Annahmen sind für Schaltfolgen als etwas komplizierte *Fairneßannahmen* zu formulieren, während man bei Prozessen mit der Forderung nach relativer Maximalität auskommt: Die Menge der maximalen Bedingungen eines Prozesses (Bedingungen mit leerem Nachbereich) läßt sich nicht durch ein Ereignis fortsetzen (vgl. [Rei95]).

Wir werden unterscheiden, ob eine Transition allein zum modellierten System gehört oder als Schnittstelle zur Außenwelt zu betrachten ist. Im zweiten Fall ist die Forderung nach Progreß nicht immer sinnvoll. In unserem Beispiel fordern wir z.B. nicht daß die Transition *ein* schaltet, wenn *ber* eine Marke trägt. Das modellierte System kann nicht garantieren, daß ein Kunde tatsächlich eine Münze einwirft.

Diese sogenannten *externen Transitionen* lassen sich bei der Konstruktion von Prozessen zur Analyse von Zielen gewinnbringend einsetzen, wie in Kapitel 5 ausführlicher beschrieben wird.

3 Formale Grundlagen

3.1 Netzklasse

Für die Modellierung komplexer Systeme haben sich höhere Petrinetze wegen ihrer kompakten Darstellung und mächtigen Ausdrucksmöglichkeiten als geeignet erwiesen. Wir betrachten in diesem Zusammenhang eine auf Definitionen u.a. von ([WVV96, Jen92, Rei86]) basierende eingeschränkte Form von Prädikat/Transitions-Netzen:

- Endlicher Netzgraph
- Keine Transition mit leerem Vor- oder Nachbereich
- Endliche Anfangsmarkierung
- Endliche Wertebereiche für alle Stellen und Variablen
- Kantenanschriften enthalten nur Multimengen von Variablen (keine Konstanten oder Operatoren)

Ansonsten seien im folgenden die üblichen Begriffe und Notationen wie ${}^\bullet s, s^\bullet$ für Vor- und Nachbereich usw. als bekannt vorausgesetzt.

3.2 Multimenge, Variablenbelegung

Die grundlegende Datenstruktur bei Prädikat/Transitions-Netzen ist die *Multimenge*. Unter einer Multimenge über einer Menge A verstehen wir eine Abbildung $m : A \to \mathbb{N}_0$. Dabei legt m fest, wie oft ein Element $a \in A$ in der Multimenge vorkommt (Schreibweise: $m[a]$). Die Menge aller Multimengen über A bezeichnen wir mit $\mathbb{N}^A$. Die Addition $\oplus$ von zwei Multimengen m_1 und m_2 ist komponentenweise definiert: $(m_1 \oplus m_2)[a] = m_1[a] + m_2[a]$; ebenso die kanonische Halbordnung $\preceq$: $m_1 \preceq m_2 \iff m_1[a] \leq m_2[a]$ für alle $a \in A$. Falls $m_2 \preceq m_1$, dann ist auch die Multimengen-Subtraktion komponentenweise definiert: $(m_1 \ominus m_2)[a] = m_1[a] - m_2[a]$.

Ein Paar $\mathcal{X} = (X, D)$, wobei X eine Menge von Bezeichnern ist und $D : X \to \mathcal{P}(A)$ eine Abbildung, die jedem Element $x \in X$ eine Teilmenge von A zuordnet, heißt *D-typisierte Variablenmenge* über A. A heißt dann *Wertebereich* von $\mathcal{X}$. Eine Abbildung $\beta : X \to A$ mit $\beta(x) \in D(x)$ für jedes $x \in X$ heißt *Variablenbelegung* für $\mathcal{X}$ in A. Die Menge aller Variablenbelegungen für $\mathcal{X}$ in A bezeichnen wir mit $\Phi_A^{\mathcal{X}}$.

Eine Variablenbelegung β für $\mathcal{X}$ in A kann erweitert werden zu einer Multimengen-Belegung $\overline{\beta}$ für eine Multimenge m über X:

$$\overline{\beta}(m)[a] = \sum_{x \in X : \beta(x) = a} m[a]$$

3.3 Prädikat/Transitions-Netz

Wir definieren ein *Prädikat/Transition-Netz* (kurz: *Pr/T-Netz*) als ein 9-Tupel $N = (S, T, F, A, d, \mathcal{X}, W, g, M_0)$:

- (S, T, F) ist ein *Petrinetz*
- A ist eine endliche Menge von *Konstanten*
- $d : S \to \mathcal{P}(A)$ ordnet jeder Stelle eine Teilmenge von A als *Wertebereich* zu
- $\mathcal{X} = (X, D)$ ist eine endliche *D-typisierte Variablenmenge* über A
- $W : F \to \mathbb{N}^X$ ordnet jeder Kante $(s,t) \in F$ bzw. $(t,s) \in F$ eine Multimenge von Variablen des zur Stelle s passenden Wertebereichs als *Kantengewicht* zu
- $g : T \to \mathcal{P}(\Phi_A^{\mathcal{X}})$ ordnet jeder Transition eine Menge von zulässigen Variablenbelegungen als *Schaltbedingung* zu (diese kann syntaktisch durch Boolesche Terme repräsentiert sein)
- $M_0 : S \to \mathbb{N}^A$ ordnet jeder Stelle eine Multimenge von Konstanten ihres Wertebereichs als *Anfangsmarkierung* zu

Für eine Transition t und eine Variablenbelegung β heißt die Einschränkung von β auf die in Kanten um t vorkommenden Variablen *Schaltmodus* von t. Wie oben benutzen wir für die Erweiterung eines Schaltmodus auf Multimengen die Notation $\overline{\beta}$. Eine Abbildung $M : S \to \mathbb{N}^A$ heißt *Markierung* eines Pr/T-Netzes N. Wir sagen, daß eine Transition t für einem Schaltmodus $\beta \in g(t)$ unter einer Markierung M *aktiviert* ist (Schreibweise $M[(t,\beta)\rangle$), wenn für alle $s \in {}^\bullet t$ gilt: $\overline{\beta}(W(s,t)) \preceq M(s)$.

Eine für einen Schaltmodus β unter einer Markierung M aktivierte Transition t kann schalten und erzeugt *die Folgemarkierung* M' (Schreibweise $M[(t,\beta)\rangle M'$ oder $M' = M[(t,\beta)\rangle$):

$$M'(s) = \begin{cases} M(s) \ominus \overline{\beta}(W(s,t)) & \text{falls } s \in {}^\bullet t \setminus t^\bullet \\ M(s) \oplus \overline{\beta}(W(t,s)) & \text{falls } s \in t^\bullet \setminus {}^\bullet t \\ M(s) \oplus \overline{\beta}(W(t,s)) \ominus \overline{\beta}(W(s,t)) & \text{falls } s \in t^\bullet \cap {}^\bullet t \\ M(s) & \text{sonst} \end{cases}$$

3.4 Beispiel Pr/T-Netz

Das Pr/T-Netz in Abbildung 3 modelliert eine Aufzugsteuerung:

- Es gibt 5 Stockwerke (1 bis 5) und zwei Aufzüge: Aufzug a befindet sich derzeit in Etage 1, Aufzug b in Etage 5 (Markierung auf Stelle *Steht*). In jeder Etage befindet sich ein Knopf, um einen Aufzug anzufordern (Stelle *Etagen* enthält alle ungedrückten Knöpfe)
- Die (schraffiert dargestellte) Transition *Knopf* stellt eine Schnittstelle zur nicht näher modellierten Außenwelt dar (Benutzer drückt Knopf in einer Etage) Derzeit sind in den Stockwerken 3 und 4 Anforderungen vorhanden (Markierung auf Stelle *Anford*)

- Konstantenmenge $A = \{1, \ldots 5\} \cup (\{a, b\} \times \{1, \ldots, 5\})$, $d(Etagen) = d(Anford) = \{a, b\}$, $d(Steht) = d(Fährt) = \{1, \ldots, 5\}$, Variablenmenge $X = \{f, e, z\}$ mit $D(f) = \{a, b\}$ und $D(e) = D(z) = \{1, \ldots, 5\}$, Schaltbedingungen $g(Start) = \{\beta \in \Phi_A^X \mid \beta(z) \neq \beta(e)\}$, $g(Stop) = g(Knopf) = \Phi_A^X$

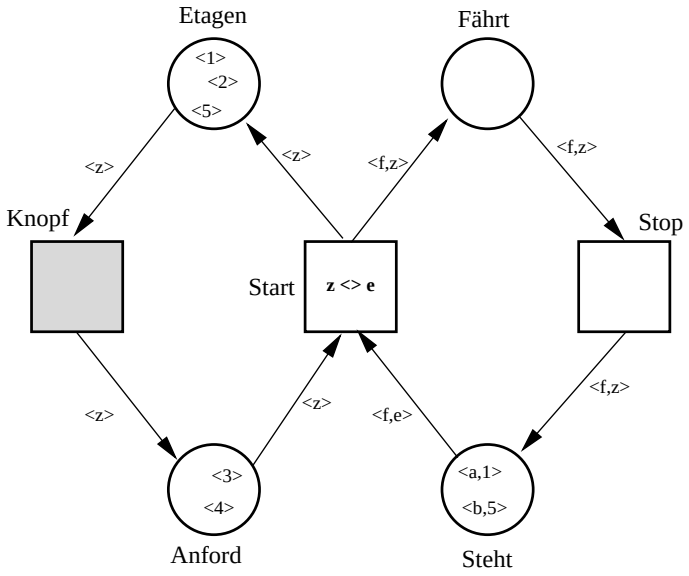

Abbildung 3: Ein Pr/T-Netz (Aufzugsteuerung)

3.5 Kausalnetze

Als ein *Kausalnetz* bezeichnen wir ein Petrinetz $K = (B, E, F_K)$ mit folgenden Eigenschaften:

- B ist eine endliche Menge von vorwärts und rückwärts unverzweigten *Bedingungen*
- E ist eine endliche Menge von *Ereignissen* mit nichtleeren Vor- und Nachbereichen
- Die Flußrelation F_K ist zyklenfrei, d.h. die transitive Hülle (bezeichnet mit $\leq$) ist eine *Halbordnung*
- Die Mengen ${}^\circ K \subseteq B$ und $K^\circ \subseteq B$ bezeichnen jeweils die *minimalen* bzw. *maximalen* Elemente von K bzgl. $\leq$

3.6 Ablauf, Prozeß, Simulation

Ein Ablauf (Prozeß) eines Pr/T-Netzes kann als ein Kausalnetz aufgefaßt werden:

- Jede Bedingung entspricht dem Vorhandensein eines Markierungselements auf einer Stelle des Pr/T-Netzes – dies wird formalisiert durch die Abbildung $\sigma : B \to (S \times A)$
- Jedes Ereignis entspricht dem Schalten einer Transition des Pr/T-Netzes für einen bestimmten Schaltmodus – dies wird formalisiert durch die Abbildung $\tau : E \to (T \times \Phi_A^X)$

- Wir stellen die Abbildungen σ und τ dar durch *Beschriftungen* der Netzknoten des Kausalnetzes mit Netzknoten und Markierungen bzw. Schaltmodi des Pr/T-Netzes

Für ein Kausalnetz K und ein Pr/T-Netz N bezeichnen wir das Tripel $\pi = (K, \sigma, \tau)$ als *Prozeß* von N, wenn gilt:

- $\overline{\sigma}(^\circ K) = M_0$ (wobei $\overline{\sigma}$ als Erweiterung von σ auf Multimengen analog zu $\overline{\beta}$ und β definiert ist
- Für jedes Ereignis $e \in E$ mit $\tau(e) = (t, \beta)$ gilt

$$\sum_{b \in {}^\bullet e} \sigma(b) = \sum_{s \in {}^\bullet t} \overline{\beta}(W(s,t)) \text{ und } \sum_{b \in e^\bullet} \sigma(b) = \sum_{s \in t^\bullet} \overline{\beta}(W(t,s))$$

Die *Menge aller Prozesse* eines Pr/T-Netzes N bezeichnen wir mit Π_N, eine Teilmenge $\Sigma \subseteq \Pi_N$ als *Simulation* von N.

3.7 Beispiel Prozeß

Folgende Notation wird im Beispielnetz verwendet:

- Jede Bedingungen wird mit dem Namen der zugeordneten Stelle, sowie in Klammern mit dem zugeordneten Markierungselement beschriftet (Beispiel: *Steht(a,3)* an der Bedingung b bedeutet, daß $\sigma(b) =$ *(Steht, <a,3>))*
- Jedes Ereignis wird mit dem Namen der zugeordneten Transition, sowie in Klammern mit dem zugeordneten Schaltmodus beschriftet (Beispiel: *Stop(f=a,z=4)* am Ereignis e bedeutet, daß $\tau(e) =$ *(Stop, β)* mit $\beta(f) = a$ und $\beta(z) = 4$)

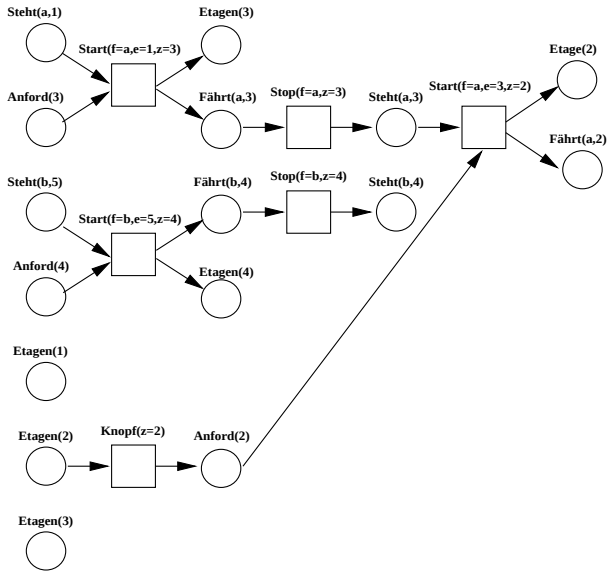

Abbildung 4: Ein Prozeß für das Pr/T-Netz aus Abbildung 3

4 Eigenschaften

4.1 Invariante Eigenschaften

Sehr oft ist man an der Überprüfung von invarianten Eigenschaften interessiert, die in jedem erreichbaren Zustand erfüllt sin. Eine sehr anwenderfreundliche Notation für Formulierung solcher Eigenschaften ist das Konzept der *Fakten (*[GTM76]).

Auf Netzebene sind Fakten Transitionen, die unter keiner erreichbaren Markierung und unter keinem Schaltmodus aktiviert sind. Dargestellt wird ein Fakt im Netzgraph als Transition mit eingeschriebenem stilisiertem 'F', sowie (gestrichelten) Kanten zu seinen Vorbedingungen. Kantenanschriften und Schaltbedingung sind analog zu Transitionen definiert. Wir beschränken uns hier auf Fakten ohne Nachbereich, d.h. die Aktivierung hängt nur von Marken im Vorbereich ab.

Abbildung 5 zeigt eine graphische Fakt-Anfrage: Der Fakt *Steht und fährt* modelliert die invariante Eigenschaft, daß niemals derselbe Aufzug sowohl steht als auch fährt.

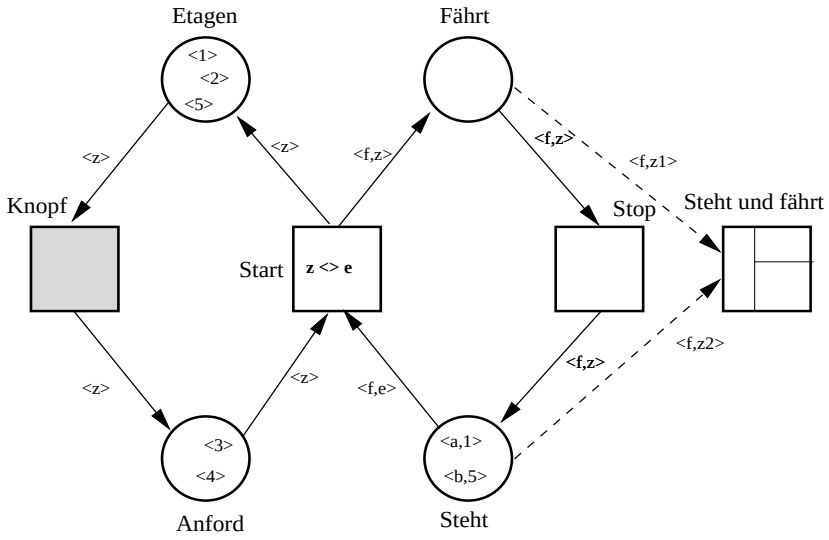

Abbildung 5: Graphische Fakt-Anfrage an das Beispielnetz aus Abbildung 3

Zur Widerlegung der Fakt-Eigenschaft reicht es aus, einen Prozeß in der Simulation zu finden, der alle den Fakt aktivierenden Bedingungen mit passenden Beschriftungeen (im Beispiel (*Steht,* $<x,\ldots>$) und (*Fährt* $<x,\ldots>$) für irgendein $x \in \{a,b\}$) kausal ungeordnet enthält (Abbildung 6). Dabei nutzen wir den Sachverhalt aus, daß jede Menge von ungeordneten Bedingungen eines Prozesses einer Teilmenge einer erreichbaren Markierung (hier der Vorbedingungen des Fakts) entspricht.

Der Test läuft im allgemeinen Fall nach folgendem Algorithmus ab:

INPUT Simulation Σ, Fakt ϕ (aufgefaßt als Transition)
INIT fakt_gilt:= TRUE
FOR Alle Prozesse $\pi = (K, \sigma, \tau)$ von Σ **DO**
  IF Es existiert eine Menge $C \subseteq B$ von ungeordneten Bedingungen in K,
    so daß $\forall s \in^\bullet \phi \exists b \in C : \sigma(b) = (s, \overline{\beta}(W(s, \phi)))$ für ein $\beta \in \Phi_A^{\mathcal{X}}$
  THEN fakt_gilt:= FALSE; Ende
  FI
OD
OUTPUT fakt_gilt

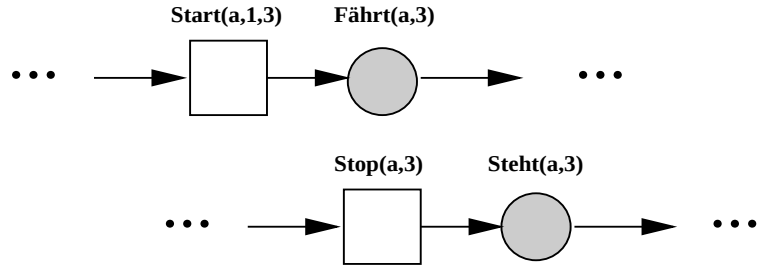

Abbildung 6: (Fiktiver) Prozeß, der die Fakt-Eigenschaft aus Abbildung 5 widerlegen würde

4.2 Kausalketten-Eigenschaften

Als Kausalkette bezeichnen wir die Eigenschaft, daß zwei gegebene Ereignisse unmittelbar kausal nacheinander eintreten, d.h. nur durch eine Bedingung getrennt sind. In Anlehnung an die im vorigen Abschnitt vorgestelle Fakt-Notation wollen wir auch Kausalketten durch ein graphisches, unmittelbar in die Netzgraphik integriertes Symbol darstellen. Wir wählen hierzu eine Transition mit eingeschriebenem, stilisiertem 'K', einer natürlichen (geraden) Zahl für die gesuchte Länge der Kausalkette, sowie je einer eingehenden (gestrichelten) Kante für die erste Bedingung und einer ausgehenden Kante für die zweite Bedingung der Kausalkette.

Zur Überprüfung einer Kausalketten-Eigenschaft reicht es aus, einen Prozeß in der Simulation zu finden, der einen Pfad der gegebenen Länge enthält, dessen erste Bedingung eine passende Beschriftung für die Vorbedingung enthält und dessen letzte Bedingung eine passende Beschriftung für die Nachbedingung enthält.

Abbildung 7 zeigt eine graphische Kausalketten-Anfrage: Die Kausalkette *Verlust* modelliert die Fragestellung, ob es irgendeinen Ablauf gibt, bei dem ein Dokument erzeugt und unmittelbar danach wieder vernichtet wird (Kausalkettenlänge 4). Abbildung 8 skizziert die Überprüfung dieser Eigenschaft auf Prozeßebene.

Mit sequentieller Simulation und der Auswertung von Schaltfolgen wäre diese Eigenschaft nicht überprüfbar, weil hier keine Informationen über kausale Abhängigkeit von Ereignissen mehr ableitbar sind.

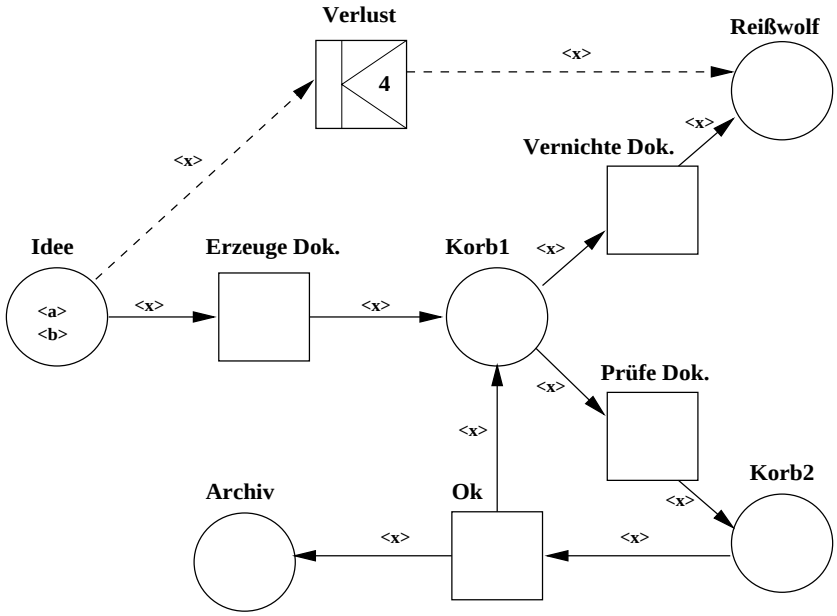

Abbildung 7: Beispiel für eine Graphische Kausalketten-Anfrage

Der Test läuft im allgemeinen Fall nach folgendem Algorithmus ab:

INPUT Simulation Σ, Kausalkette $\kappa = (s_1, k, n, s_2)$
INIT kausalkette_existiert:= FALSE
FOR Alle Prozesse $\pi = (K, \sigma, \tau)$ von Σ **DO**
  FOR Alle Bedingungen $b_1 \in B$ **DO**
    FOR Alle $\beta \in \Phi_A^{\mathcal{X}}$ mit $\sigma(b_1) = (s_1, \overline{\beta}(W(s_1, k)))$ **DO**
      FOR Alle Nachfolger b_2 von b_1 mit Abstand n **DO**
        IF $\sigma(b_2) = (s_2, \overline{\beta}(W(k, s_2)))$
        THEN kausalkette_existiert:= TRUE; Ende
        FI
      OD
    OD
  OD
OD
OUTPUT kausalkette_existiert

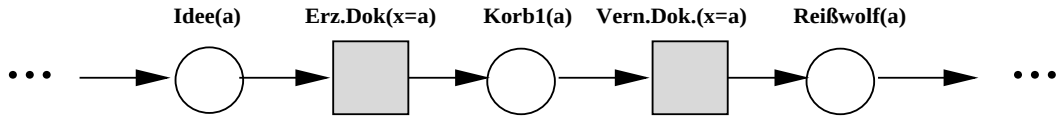

Abbildung 8: Prozeß, der die Kausalkette aus Abbildung 7 identifiziert

Eine denkbare - und im Rahmen des Projekts *VIP* auch geplante - Erweiterung ist die Spezifikation von Kausalketten mit mehreren Vor- und Nachbedingungen.

4.3 Ziel-Eigenschaften

Als Ziel bezeichnen wir die Eigenschaft, daß aus der Gültigkeit einer Bedingung in einem Ablauf zwingend folgt, daß irgendwann (kausal später) eine zweite Bedingung erfüllt sein wird.

Wie schon die Kausalketten wollen wir Ziel-Eigenschaften in Anlehnung an die Fakt-Notation durch ein graphisches, unmittelbar in die Netzgraphik integriertes Symbol darstellen. Wir wählen hierzu eine Transition mit eingeschriebenem, stilisiertem 'Z' und einer eingehenden (gestrichelten) Kante für die Vorbedingung und einer ausgehenden Kante für die Nachbedingung.

Ein sehr nützliches Hilfsmittel bei der Überprüfung von Ziel-Eigenschaften ist das u.a. in [WVV96] eingeführte Konzept der *externen Transitionen*. Es wird eine Teilmenge $T_e \subseteq T$ von Transitionen des Systemnetzes festgelegt, die als Schnittstellen zur nicht modellierten Außenwelt fungieren und beim Schalten insofern eine Sonderstellung haben, als sie zunächst wie alle anderen Transitionen behandelt werden, dann aber ab einem gewissen Zeitpunkt der Simulation nicht mehr schalten. Intuitiv steht diese Sichtweise für das Verhalten realer Systeme, die zwar intern terminierende Abläufe enthalten, aber durch ihre permanente Interaktion mit der Umwelt zyklisch (nichtterminierend) sind. Externe Transitionen können also ein sehr realitätsnahes Konzept für die simulative Terminierung eines zyklischen Systems sein. Insbesondere können wir bei der Analyse von Zielen die Eigenschaft ausnutzen, daß das Erreichen eines Zieles weder vom Schaltverhalten externer Transitionen abhängt noch von diesem behindert wird (vgl. [De96]).

Überprüft wird eine Ziel-Eigenschaft, indem nachgewiesen wird, ob es in jedem Prozeß, der eine entsprechend der Vorbedingung des Zieles beschriftete Bedingung als Knoten enthält, einen Pfad zu einer entsprechend der Nachbedingung des Zieles beschrifteten Bedingung gibt. Abbildung 9 zeigt eine graphische Ziel-Anfrage: Das Ziel *Aufzug kommt* modelliert die Forderung, daß eine Aufzuganforderung in einem Stockwerk (Marke auf der Stelle *Anford*), in jedem Ablauf irgendwann von einem Aufzug befriedigt wird (Marke auf der Stelle *Steht*). Abbildung 10 deutet, an wie diese Ziel-Anfrage auf Prozeßebene überprüft wird.

Die Überprüfung läuft im allgemeinen Fall nach folgendem Algorithmus ab:

INPUT Simulation Σ, Ziel-Eigenschaft $\zeta = (s_1, z, s_2)$
INIT Ziel_erfüllt:= TRUE
FOR Alle Prozesse $\pi = (K, \sigma, \tau)$ von Σ **DO**

  FOR Alle Bedingungen $b_1 \in B$ mit $\sigma(b_1) = (s_1, \overline{\beta}(W(s_1, z)))$ für ein $\beta \in \Phi_A^{\mathcal{X}}$ **DO**

    Suche das letzte Vorkommen von b_1 in K
    IF Es gibt keinen Pfad von $b_1$nach b_2 , so daß $\sigma(b_2) = (s_2, \overline{\beta}(W(z, s_2)))$
  THEN Ziel_erfüllt:= FALSE; Ende
  FI

OD
OUTPUT Ziel_erfüllt

Eine denkbare - und im Rahmen des Projekts *VIP* auch geplante - Erweiterung ist das Zulassen von Zielen mit mehreren Vor- und Nachbedingungen.

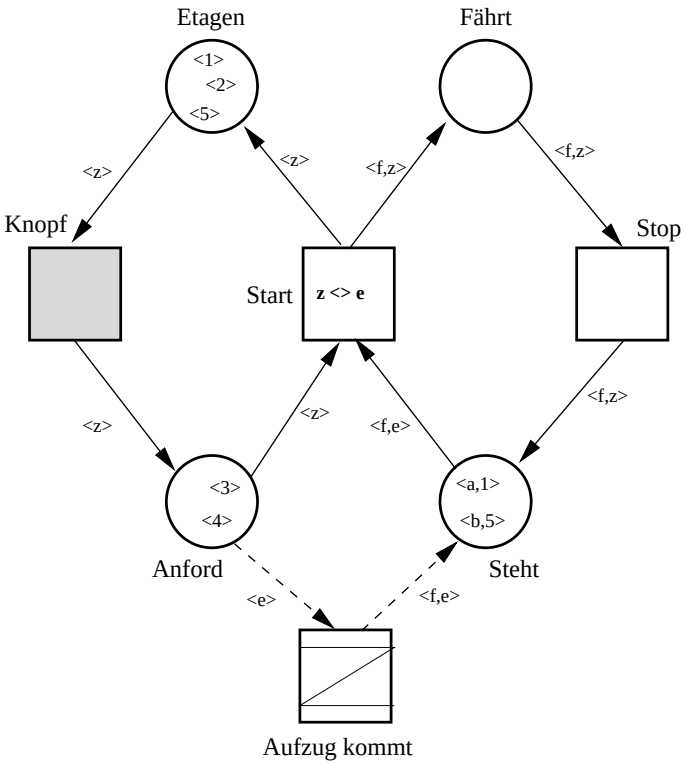

Abbildung 9: Graphische Ziel-Anfrage an das Beispielnetz aus Abbildung 3

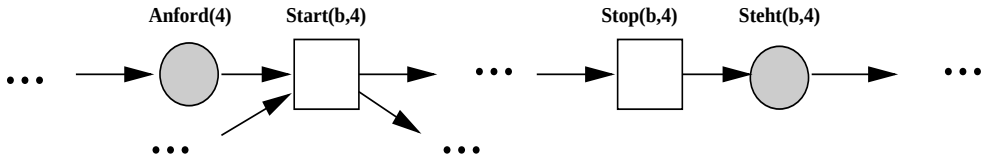

Abbildung 10: Ein Prozeß, in dem die Ziel-Eigenschaft aus Abbildung 9 gilt

5 Prozeßerzeugung

5.1 Vorgehensweise

Unter einer Simulation verstehen wir eine Teilmenge der Prozeßmenge eines Netzes. Eine Möglichkeit der Erzeugung von Prozessen für ein gegebenes Netz ist die auf dem Prinzip der *optimalen Simulation* basierende und in [DOZ96] ausführlich beschriebene Erzeugung der vollständigen Prozeßmenge. Da diese Menge bei komplexeren Systemnetzen enorm groß werden kann, ist diese Vorgehensweise in der Praxis meist nicht anwendbar. Es empfiehlt sich die Erzeugung einer - durch adäquate Rahmenbedingungen determinierten - *echten Teilmenge der Prozeßmenge.* Diese Rahmenbedingungen können sein:

- Eine *Abbruchbedingung*, die festlegt, ob mit der laufenden Erzeugung fortgefahren werden soll - diese kann lokal (der aktuelle Prozeß wird beendet) oder global sein (die Erzeugung als Ganzes wird beendet)

- Eine *Schaltauswahl*, die unter den aktivierten Transitionen eine auswählt, um einen Prozeß fortzusetzen
- Das Konzept der *externen Transitionen*, das es ermöglicht, die Simulation durch Einschränkung der Schaltauswahl auf die übrigen Transitionen unter realistischen Bedingungen terminieren zu lassen

5.2 Abbruchbedingungen

Als Abbruchbedingungen für die Prozeßerzeugung können folgende Kriterien herangezogen werden:

- *Globale Abbruchbedingungen:* Entweder es ist die gesamte Prozeßmenge erzeugt ($\Sigma = \Pi_N$) oder eine obere oder untere Schranke für die Anzahl des Eintretens eines bestimmten Ereignisses oder einer bestimmten Transition (bezogen auf **alle Prozesse**) ist erreicht
- *Lokale Abbruchbedingungen:* Es ist eine Transition mehr aktiviert, eine obere Schranke für die Prozeßketten-Länge oder die Anzahl des Vorkommens einer bestimmten Transition/Stelle (bezogen auf **einen Prozeß**) ist erreicht
- *Dynamische Abbruchbedingungen:* Abhängig von der Art der angefragten Eigenschaft wird eine (globale oder lokale) Abbruchbedingung dynamisch festgelegt

5.3 Schaltauswahl

Bei der Auswahl der zu schaltenden Transition und ihres Schaltmodus sind folgende Kriterien möglich:

- *Benutzerdialog:* Der Benutzer wählt unter den aktivierbaren Transitionen und Schaltmodi eine im Dialog aus
- *Zufallsgenerator:* Es wird ein Kandidat unter den aktivierten Transitionen und Schaltmodi zufällig ausgewählt
- *Gewichteter Zufallsgenerator:* Wie oben, es kann jedoch für jede Transition und jeden Schaltmodus a priori eine Trefferwahrscheinlichkeit festgelegt werden
- *Dynamische Auswahl:* Abhängig von der Art der angefragten Eigenschaft wird eine bestimmte Auswahl begünstigt oder vernachlässigt

5.4 Externe Transitionen

Ein wichtiges Hilfsmittel bei der Prozeßerzeugung sind *externe Transitionen*, die ab einem bestimmten Zeitpunkt der Simulation von der Schaltauswahl ausgeschlossen werden und so die Terminierung eines zyklischen Systems unter realen Bedingungen zulassen. Folgendes Verhalten ist möglich:

- Der *Benutzer* legt zu einem bestimmte Zeitpunkt fest, daß ab jetzt keine externen Transitionen mehr schalten dürfen
- Es wird - ausgehend von einer Voreinstellung - bei jedem Schalten einer externen Transition *automatisch* ermittelt, ob ein weiteres Schalten ermöglicht sein soll oder nicht

5.5 Algorithmus

Die Erzeugung von Prozessen zu einem gegebenen Pr/T-Netz geschieht nach folgendem Algorithmus:

INPUT Pr/T-Netz N
INIT $\Sigma := \oslash$
WHILE Globale Abbruchbedingung nicht erfüllt **DO**
  IF Deaktivierungsbedingung für externe Transitionen erfüllt
  THEN Setze externe Transitionen auf inaktiv
  FI
  Erzeuge Kausalnetz $K = (B, E, F_K)$ mit

$$\bigcup_{b \in B} \sigma(b) = M_0; E := \emptyset; F_K := \emptyset$$

  $M := M_0$
  WHILE Lokale Abbruchbedingung nicht erfüllt **DO**
    Prüfe, ob externe Transitionen auswählbar sind
    Wähle ein $t \in T$ und ein $\beta \in \Phi_A^{\mathcal{X}}$, so daß $M[(t,\beta)\rangle$
    $E := E \cup e$ so daß $\tau(e) = (t, \beta)$
    $F_K := F_K \cup (b, e)$ für alle $b \in B$ mit $\sigma(b) = (s, \overline{\beta}(W(s,t))$ für ein $s \in^\bullet t$
    $B := B \cup B'$
      wobei B' je eine mit $(s, \overline{\beta}(W(t,s)))$ beschriftete Bedingung für alle $s \in t^\bullet$ enthält
    $F_K := F_K \cup (e, b)$ für alle $b \in B'$
    $M := M[(t, \beta)\rangle$
  OD
  $\Sigma := \Sigma \cup (K, \sigma, \tau)$
OD
OUTPUT Σ

6 Das Softwarewerkzeug *VIPtool*

Im Rahmen des oben erwähnten *VIP*-Projekts ist auch die prototypische Implementierung der Konzepte in einem Software-Werkzeug names ***VIPtool*** vorgesehen, das folgende Funktionalität anbietet:

- Graphischer Editor zum Bearbeiten und/oder Importieren von Pr/T-Netzen sowie graphische Anfrageschnittstelle für Systemeigenschaften

- Benutzergesteuerte, automatische oder Eigenschafts-gesteuerte Simulation mit hoher Konfigurierbarkeit
- Abspeicherung der Simulationsdaten als halbgeordnete Abläufe (Prozesse) in einer Simulationsdatenbank und Auswertung der Anfragen auf der Basis der halbgeordneten Abläufe (Verifikator)
- Intelligente Visualisierung der Abläufe (Prozeßbrowser) unter Ausnutzung von Plazierungskonzepten aus dem Bereich der genetischen Algorithmen
- Integration in die am Institut AIFB der Universität Karlsruhe entworfene evolutionäre Software-Entwicklungsumgebung *INCOME* ([OSS94])

Das Werkzeug wird in den Programmiersprachen *PYTHON (*Editor, Prozeßbrowser) und *C/C++* (Simulator, Verifikator) implementiert und soll im wesentlichen auf dem an der HU Berlin entwickelten *Petrinetz-Kern* ([DK96]) – einer in PYTHON geschriebenen universellen Klassenbibliothek – aufbauen. Abbildung 11 zeigt den schematischen Aufbau von ***VIPtool.***

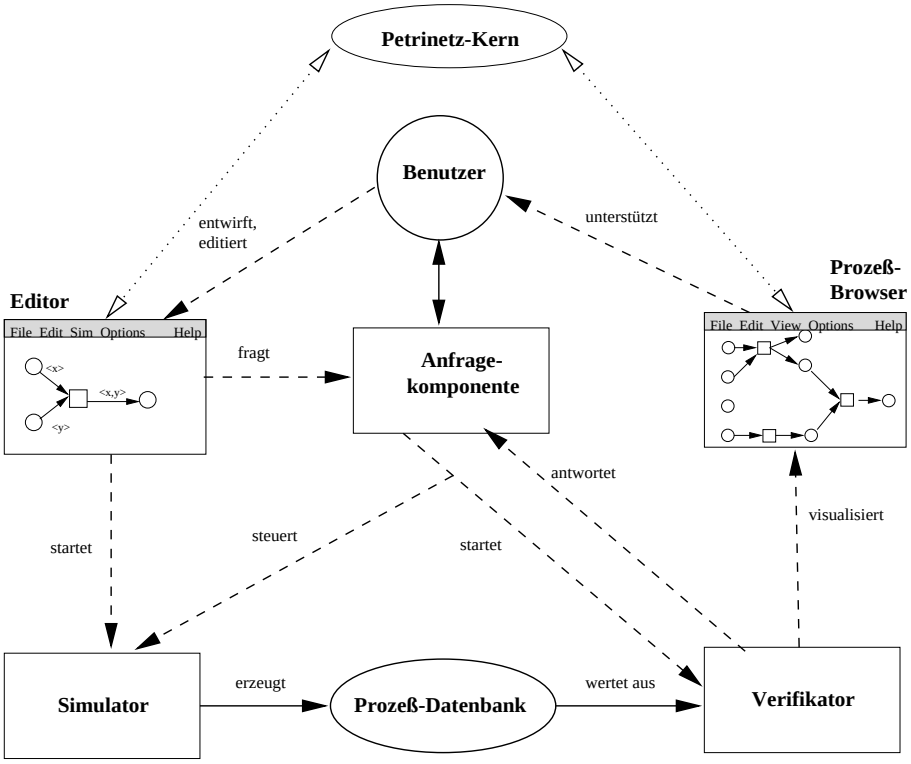

Abbildung 11: Der Aufbau des Werkzeugs **VIPtool**

Literatur

[De96] J. Desel.
Über den Beweis von Zielen mit linear-algebraischen Techniken.
Seiten 8–13. Forschungsbericht 341, AIFB, Uni Karlsruhe, 1996.

[DK96] J. Desel und E. Kindler.
Der Traum von einem universellen Petrinetz-Werkzeug - der Petrinetz-Kern.
Seiten 27–32. Forschungsbericht 341, AIFB, Uni Karlsruhe, 1996.

[DO95] J. Desel und A. Oberweis.
Verifikation von Informationssystemen durch Auswertung halbgeordneter Petrinetz-Abläufe.
Forschungsbericht 324, AIFB, Uni Karlsruhe, 1995.

[DOZ96] J. Desel, A. Oberweis und T. Zimmer.
Simulation-based Analysis of Distributed Information System Behaviour.
Proceedings des 8th European Simulaiton Symposium ESS96, Genua 1996.

[Esp94] J. Esparza.
Model checking using net unfoldings.
Science of Computer Programming, (23):151–195, 1994.

[Fre96] T. Freytag.
Simulation halbgeordneter Petrinetz-Abläufe
Seiten 14–20. Forschungsbericht 341, AIFB, Uni Karlsruhe, 1996.

[GTM76] H. Genrich und G. Thieler-Mevissen.
The Calculus of Facts.
In A. Mazurkiewicz, Hrsg., *Mathematical Foundations of Computer Science*, Seiten 588–595. Springer-Verlag, 1976.

[Jen92] K. Jensen.
Coloured Petri Nets, Vol.1: Basic Concepts.
Springer-Verlag, Berlin, 1992.

[OSS94] A. Oberweis, G. Scherrer (jetzt: Zimmermann) und W. Stucky.
INCOME/STAR - Methodology and tools for the development of distributed information systems.
Information Systems, 19(8):641–658, 1994.

[Rei86] W. Reisig.
Petrinetze - eine Einführung.
Springer-Verlag, Berlin, 2. Auflage, 1986.

[Rei95] W. Reisig.
Petri Net Models of Distributed Algorithms.
ComputerScience Today, LNCS 1000, S. 441–454.
Springer-Verlag, Berlin, 1995.

[Sa94] V. Sänger.
Eine graphische Anfragesprache für temporale Datenbanken.
Dissertation, Universität Karlsruhe, 1995.

[Sc95] E. Schnieder (Hrsg.).
Proceedings der 4. Fachtagung.
Institut für Regelungs- und Automatisierungstechnik, TU Braunschweig, 1995.

[WVV96] R. Walter, H. Völzer, T. Vesper, W. Reisig, E. Kindler, J. Freiheit und J. Desel
Memorandum: Petrinetzmodelle für verteilte Algorithmen.
Forschungsbericht 67, Humboldt-Universität Berlin, 1996.