Beschreibung paralleler Abläufe mit Petri-Netzen

Grundlagen und Beispiele für den Unterricht

Petri-Netze sind ein graphisches Mittel zur Beschreibung, Modellierung, Analyse und Simulation von dynamischen Systemen, die eine feste Grundstruktur besitzen. Beispiele hierfür sind Rechenanlagen, Büroabläufe oder Herstellungsverfahren. Petri-Netze sind auf der einen Seite anschaulich und können daher auch von Nicht-Fachleuten verwendet werden, auf der anderen Seite sind sie mathematisch exakt definiert und ermöglichen daher präzise Untersuchungen.
Petri-Netze wurden 1962 von dem deutschen Wissenschaftler C.A. Petri vorgeschlagen.

Ein Petri-Netz (in seiner einfachsten Form) ist ein gerichteter Graph, der aus zwei verschiedenen Sorten von Knoten besteht, Stellen und Transitionen. Eine *Stelle* wird durch einen Kreis O dargestellt und symbolisiert eine Ablage für Objekte oder Daten. *Transitionen* beschreiben die Verarbeitung von Objekten und werden durch Balken repräsentiert. Gerichtete Kanten (also Pfeile) dürfen nur von Knoten der einen Sorte zu Knoten der anderen Sorte führen. Alle Stellen, von denen Kanten zu einer Transition t führen, heißen *Eingabestellen* von t, alle Stellen, zu denen von t aus Kanten führen, heißen *Ausgabestellen* von t. Unten werden wir definieren, wie Transitionen Objekte aus den Eingabestellen herausnehmen, verarbeiten und die verarbeiteten Objekte in Ausgabestellen ablegen.

Beispiel: Eine Bierflaschenabfüllanlage.

Wir wollen eine kleine (primitive) Bierflaschenabfüllanlage durch ein Petri-Netz darstellen. Es handelt sich hierbei um eine Form des klassischen Erzeuger-Verbraucher-Problems. Die Fabrik besteht aus zwei Maschinen, der Abfüllmaschine und der Verschlußmaschine, mit der die Flaschen verkorkt werden. Zwischen beiden Maschinen befindet sich ein kleines Zwischenlager, das von zwei Gabelstaplern benutzt wird. Der erste Gabelstapler transportiert jeweils einen Kasten mit gefüllten Flaschen von der Maschine ins Lager, der zweite Gabelstapler transportiert jeweils einen Kasten aus dem Lager an die Verschlußmaschine. Der erste Gabelstapler soll nur in Aktion treten, wenn die Maschine einen Kasten mit Flaschen abgefüllt hat, der zweite darf nur dann transportieren, wenn die Verschlußmaschine bereit ist (also nicht gerade einen Kasten verkorkt) und das Lager nicht leer ist. Hat der erste Gabelstapler einen Kasten aus der Abfüllmaschine entnommen, so kann die Maschine den nächsten Kasten abfüllen. Die Zulieferung der noch nicht gefüllten Flaschen bzw. der Abtransport der bereits verkorkten Flaschen erfolgt über ein Fließband. Abb. 1 zeigt die Situation anschaulich.
Die genannten Bedingungen sollen nun durch ein Petri-Netz korrekt dargestellt werden. Abb. 2 zeigt ein zugehöriges Petri-Netz mit der Knotenmenge

$$\{s_0, s_1, s_2, s_3, s_4, s_5, s_6, t_1, t_2, t_3, t_4\}.$$

$s_0,...,s_6$ sind Stellen, $t_1,...,t_4$ sind Transitionen. Zum Beispiel ist s_1 eine Eingabestelle von t_2, s_2 und s_3 sind die Ausgabestellen von t_2. Man beachte: Eine Stelle kann gleichermaßen Eingabe- und Ausgabestelle einer Transition sein. Welcher Knoten des Petri-Netzes welches Element innerhalb der Fabrik repräsentiert, ist jeweils verzeichnet.

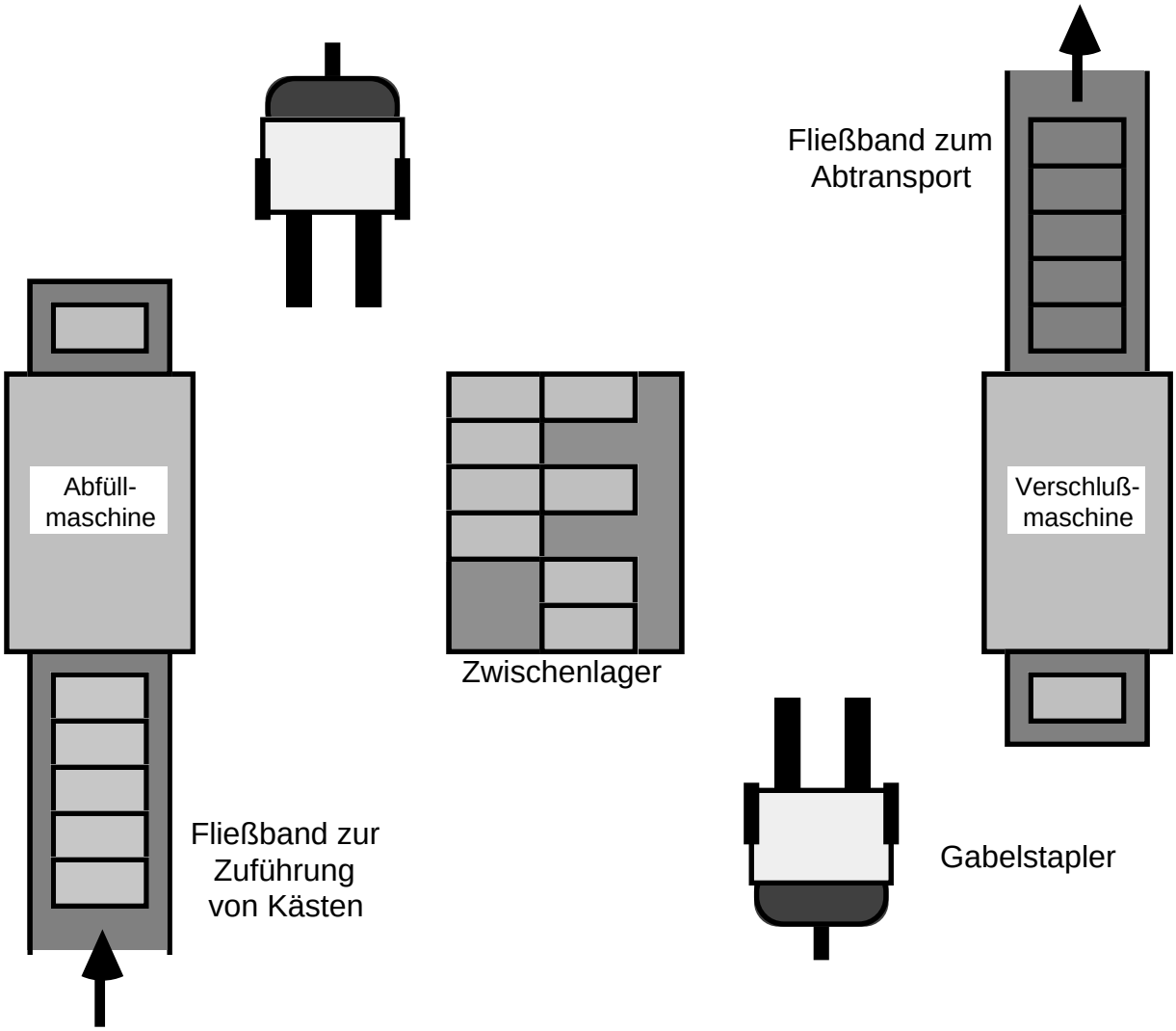


Abb. 1: Situation in der Fabrik

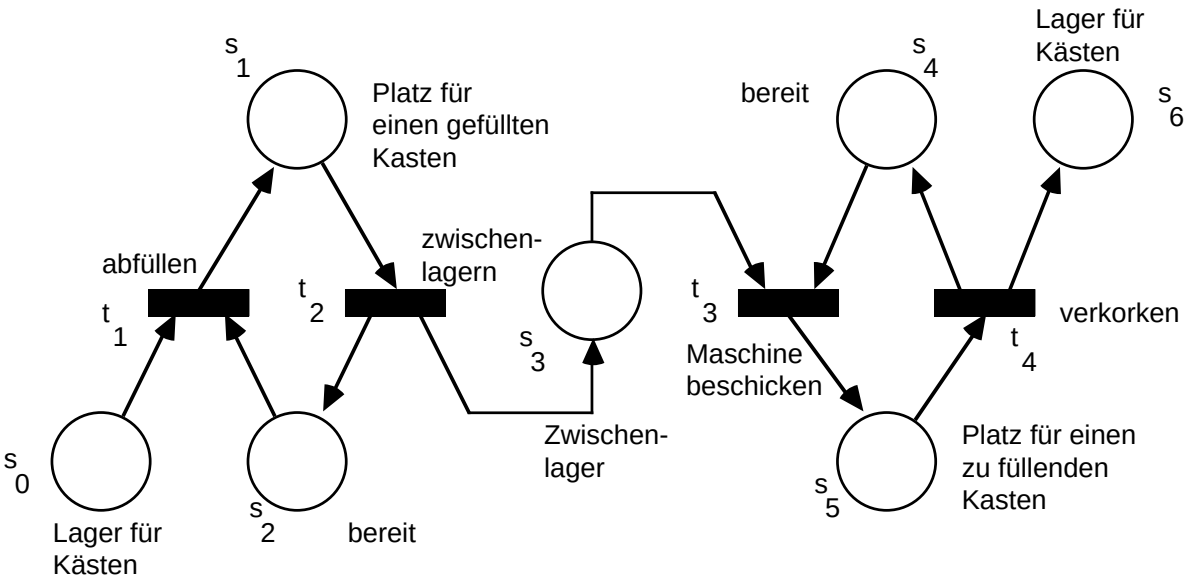
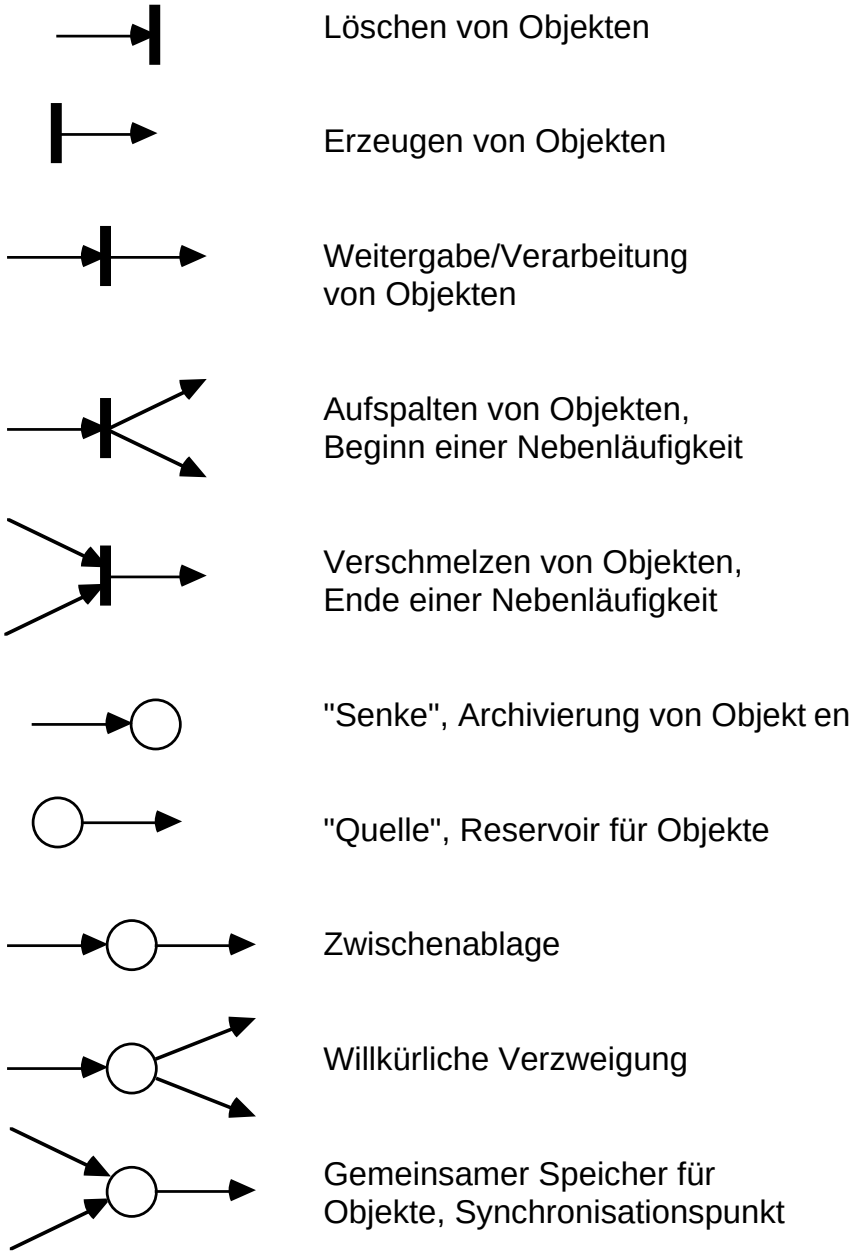


Abb. 2: Petri-Netz

Die einzelnen Elemente, aus denen ein Petri-Netz zusammengesetzt ist, kann man sich folgendermaßen als Abbild der Wirklichkeit vorstellen:



Petri-Netze über natürlichen Zahlen.

Das Petri-Netz in Abb. 2 spiegelt nur die statische Struktur eines Ablaufs wider, anschaulich die einzelnen Stationen, die ein zu verarbeitendes Objekt durchläuft. Um dynamische Vorgänge zu beschreiben, werden die Stellen mit Objekten belegt, die über die Transitionen von Stelle zu Stelle weitergegeben werden. Allgemein kann man beliebige Objekte zulassen (z.B. Bleistifte, Autoteile, Akten, bei uns Flaschenkästen). Möchte man Petri-Netze formal darstellen, so beschränkt man sich meist auf Objekte einer vorgegebenen Menge oder eines vorgegebenen Datentyps. Wir wollen uns im folgenden zunächst auf die Menge IN_0 beschränken, weil wir nur mit einer Sorte von Objekten (Bierkästen) zu tun

haben und uns nur die Anzahlen der Kästen in den einzelnen Verarbeitungsstadien (auf den Stellen) interessieren.

Jede Stelle kann ein Objekt der Menge IN_0, also eine natürliche Zahl aufnehmen. Um anschaulich darzustellen, daß eine Stelle eine natürliche Zahl $n \in IN_0$ enthält, und um die Arbeitsweise eines Petri-Netzes besser von Hand nachvollziehen zu können, zeichnet man n Punkte (sog. *Marken*) in die Stelle hinein.

Beispiel: Abb. 3 zeigt das um eine Reihe von Marken ergänzte Petri-Netz aus Abb. 2. In Abb. 3 befinden sich also fünf Marken (d.h. die Zahl 5) in der Stelle s_0, eine Marke (d.h. die Zahl 1) in der Stelle s_1, drei Marken (d.h. die Zahl 3) in s_3 und eine Marke (d.h. die Zahl 1) in s_5. Folglich befinden sich fünf ungefüllte Bierkästen im Lager, ein Kasten ist fertig abgefüllt, drei Kästen stehen im Zwischenlager usw.

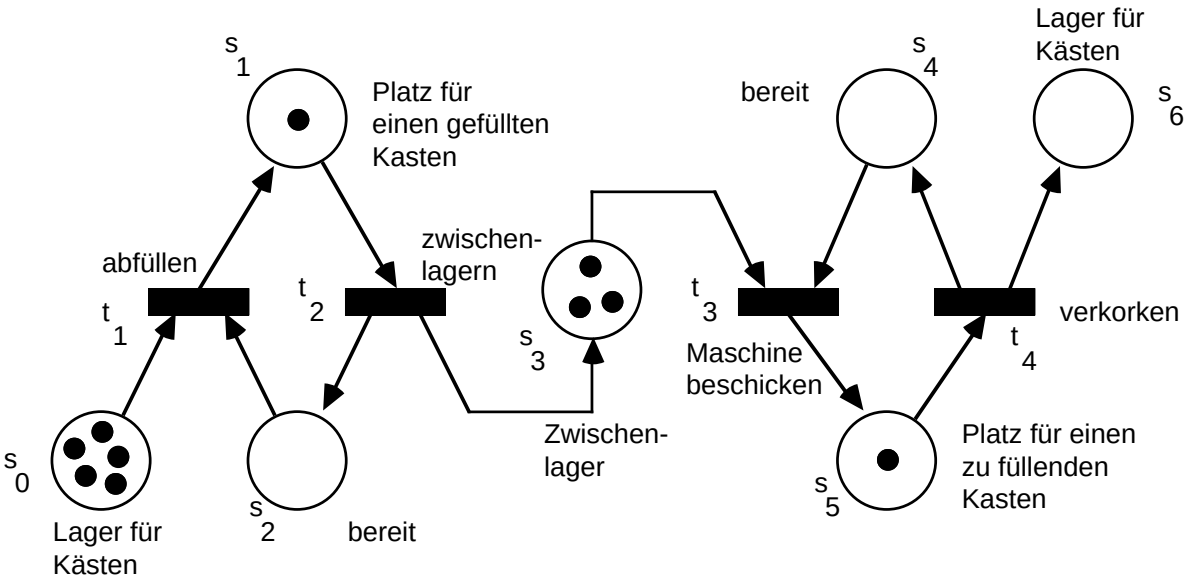


Abb. 3: Petri-Netz mit Marken

Wie schalten Petri-Netze?

Soweit die Beschreibung der statischen Anteile eines Systems mit Petri-Netzen. Kommen wir nun zum dynamischen Aspekt. Der Bewegungsablauf der Marken im Petri-Netz wird durch folgende *Schaltregel* für Transitionen festgelegt:

a) Eine Transition t kann *schalten* (oder *zünden* oder *feuern*), wenn jede Eingabestelle mit einer Marke belegt ist. Besitzt t keine Eingabestelle, so kann t immer feuern.
b) Können mehrere Transitionen schalten, so schaltet willkürlich (*nichtdeterministisch*) eine dieser Transitionen.
c) Schaltet eine Transition, so wird aus jeder Eingabestelle eine Marke entfernt und zu jeder Ausgabestelle eine Marke hinzugefügt.

Beispiel: In Abb. 3 können die Transitionen t_2 und t_4 schalten. t_1 und t_3 können nicht schalten, da jeweils nicht alle ihre Eingabestellen mit Marken belegt sind. Nehmen wir an, es schaltet t_2, so zeigt Abb. 4 die Folgesituation. Aus der Eingabestelle s_1 von t_2 wird eine Marke entfernt, zu den Ausgabestellen s_2 und s_3 wird je eine Marke hinzugefügt. Anschließend können nur t_1 und t_4 schalten. Schaltet t_4, so ergibt sich Abb. 5, schaltet danach t_3, so erhält man die Situation in Abb. 6.
Bezogen auf die Situation in der Flaschenabfüllfabrik, die ja durch unser Petri-Netz modelliert werden soll, bedeutet die Schaltfolge t_2, t_4, t_3 folgendes: Der erste Gabelstapler transportiert eine abgefüllten Kasten Bier in das Lager und macht die Abfüllmaschine bereit (Schalten von t_2), anschließend verkorkt die zweite Maschine einen Kasten, liefert ihn in das Endlager und macht den Gabelstapler bereit (Schalten von t_4). Der beschickt daraufhin die Verschlußmaschine mit einem weiteren Kasten Bier (Schalten von t_3).

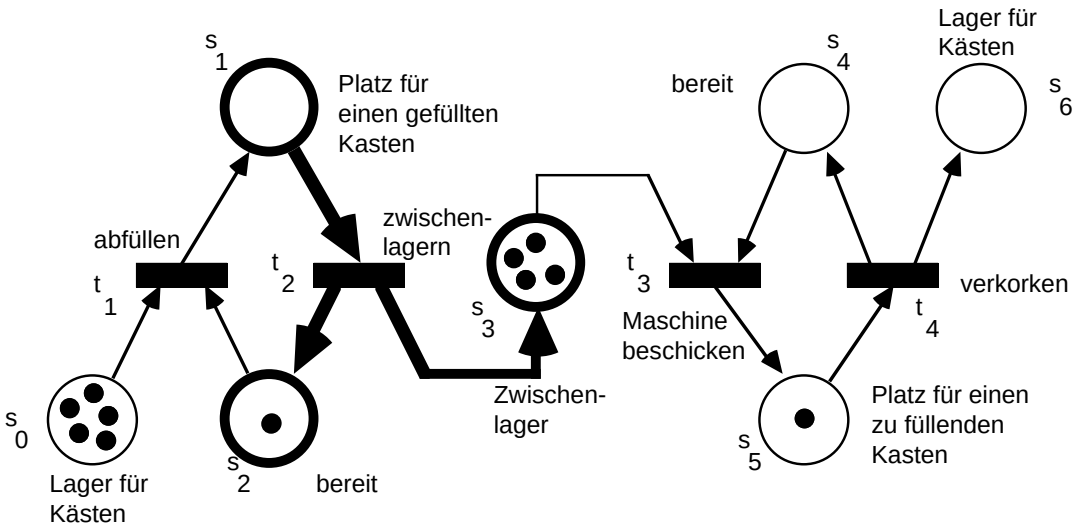


Abb. 4: Petri-Netz, nachdem t_2 geschaltet hat

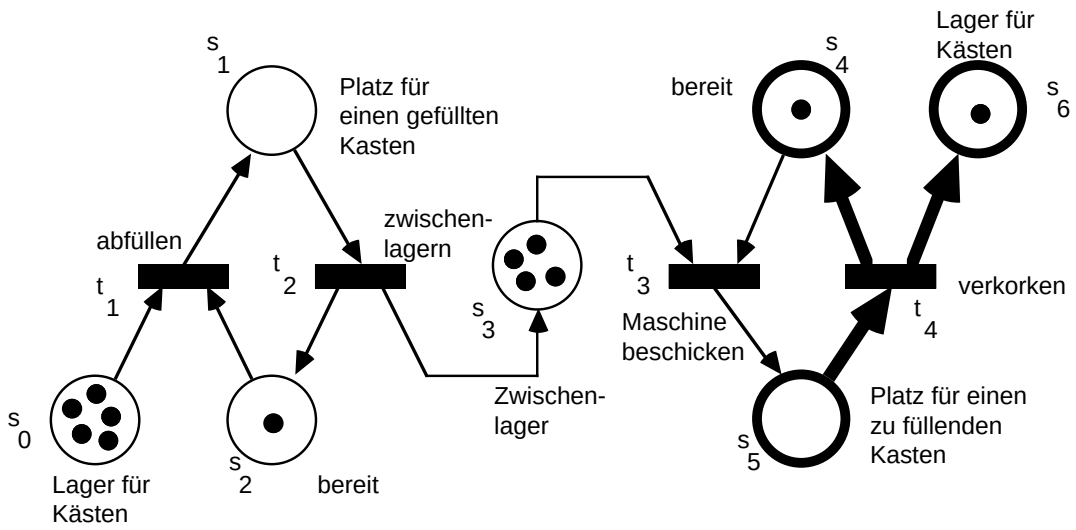


Abb. 5: Petri-Netz, nachdem t_4 geschaltet hat

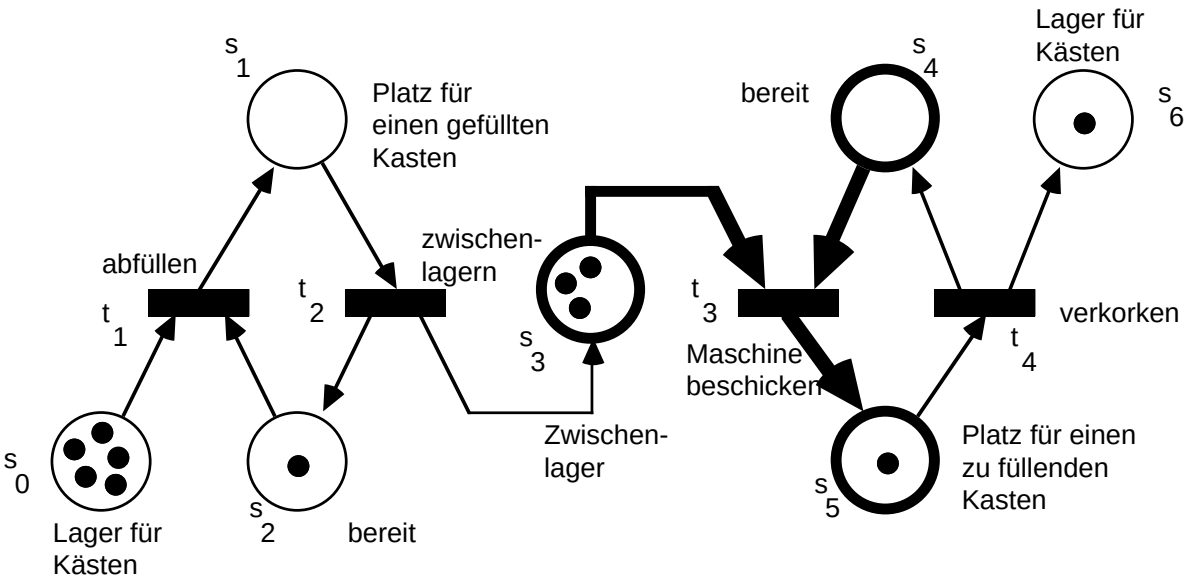


Abb. 6: Petri-Netz, nachdem t_3 geschaltet hat

Aufgabe 1:
In Abb. 2 ist das Zwischenlager so modelliert, daß es eine beliebige Kapazität besitzt, d.h. die Abfüllmaschine kann beliebig lange weiterarbeiten, auch wenn die Verschlußmaschine defekt ist. Wie muß das Petri-Netz geändert werden, wenn das Zwischenlager maximal fünf Kästen fassen kann. Die Abfüllmaschine muß also warten, wenn das Zwischenlager gefüllt ist, und sie kann erst wieder anlaufen, wenn Plätze im Zwischenlager frei sind.

Formale Definition von Petri-Netzen.

Die bisherigen anschaulichen Erläuterungen wollen wir nun in eine formale Definition fassen. Zunächst das Petri-Netz:

Definition A:
Ein *Petri-Netz* ist ein 4-Tupel P=(S,T,A,E), wobei gilt:
- S ist eine nichtleere endliche Menge von *Stellen*,
- T ist eine nichtleere endliche Menge von *Transitionen*,
- S und T sind disjunkt, also $S \cap T = \emptyset$,
- $A \subseteq S \times T$ ist eine endliche Menge von gerichteten Kanten, die von Stellen ausgehen und zu Transitionen führen,
- $E \subseteq T \times S$ ist eine endliche Menge von gerichteten Kanten, die von Transitionen ausgehen und bei Stellen enden.

Eine Abbildung $M: S \rightarrow IN_0$ heißt *Markierung* von P und gibt an, wieviele Marken sich in jeder Stelle befinden.

Beispiel: Das Petri-Netz aus Abb. 2 ist gemäß Definition A folgendermaßen definiert:

$P=(S,T,A,E)$ mit

$S=\{s_0,...,s_6\}$, $T=\{t_1,...,t_4\}$,

$A=\{(s_0,t_1), (s_1,t_2), (s_2,t_1), (s_3,t_3), (s_4,t_3), (s_5,t_4)\}$,

$E=\{(t_1,s_1), (t_2,s_2), (t_2,s_3), (t_3,s_5), (t_4,s_4), (t_4,s_6)\}$.

Die Markierung in Abb. 3 lautet $M: S\rightarrow IN_0$ mit

$M(s_0)=5$, $M(s_1)=1$, $M(s_3)=3$, $M(s_5)=1$,

$M(s_2)=M(s_4)=M(s_6)=0$.

Die folgenden beiden Definitionen präzisieren das dynamische Verhalten von Petri-Netzen. Die erste beschreibt diejenigen Transitionen, die in der Folge schalten können, die zweite definiert einen Schaltvorgang. Eine Transition heißt *aktiviert*, wenn jede ihre Eingabestellen mit mindestens einer Marke belegt ist. In Abb. 3 sind z.B. die Transitionen t_1, t_2 und t_4 aktiviert.

Definition B:

Sei $P=(S,T,A,E)$ ein Petri-Netz.

a) Für jede Transition $t\in T$ ist

$e(t)=t^-=\{s\in S \mid (s,t)\in A\}$

die Menge der *Eingabestellen* und

$a(t)=t^+=\{s\in S \mid (t,s)\in E\}$

die Menge der *Ausgabestellen* von t.

b) Sei M eine Markierung von P. Eine Transition $t\in T$ ist *aktiviert*, wenn $e(t)=\emptyset$ oder $M(s)>0$ für alle $s\in e(t)$ gilt.

Die folgende Definition präzisiert die Schaltregel, also die Änderung der Markierung eines Petri-Netzes beim Schalten einer Transition.

Definition C:

Sei $P=(S,T,A,E)$ ein Petri-Netz und M eine Markierung von P. Eine aktivierte Transition $t\in T$ *schaltet* von der Markierung M in die Markierung M': $S\rightarrow IN_0$, wenn gilt:

$$M'(s)=\begin{cases} M(s)+1, & \text{falls } s\in a(t), \text{ aber } s\notin e(t),\\ M(s)-1, & \text{falls } s\in e(t), \text{ aber } s\notin a(t),\\ M(s), & \text{in allen übrigen Fällen.}\end{cases}$$

Beispiel: Modellierung des Verkehrs über eine Brücke.

Man betrachte einen Ausschnitt aus einem Stadtplan (Abb. 7). Eine Brücke geringer Tragfähigkeit, die zu einem Zeitpunkt nur von einem Fahrzeug befahren werden darf, kann aus zwei Richtungen A und B erreicht werden. Fahrzeuge, die aus Richtung A kommen, möch-

ten stets in Richtung C weiterfahren. Fahrzeuge aus Richtung B fahren in Richtung D weiter. Gesucht ist ein Petri-Netz, das diese Situation simuliert.

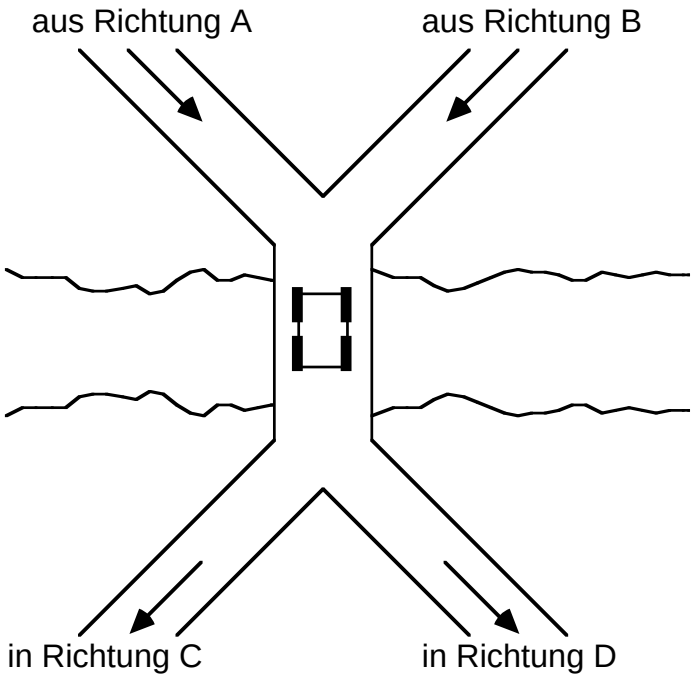


Abb. 7: Ausschnitt aus einem Stadtplan

Abb. 8 zeigt ein mögliches Petri-Netz. Die aus Richtung A und B anfahrenden Fahrzeuge werden in s_1 und s_2 gesammelt. s_5 sorgt dafür, daß stets nur ein Fahrzeug die Brücke befahren kann. In s_4 und s_6 wird - gewissermaßen als Laufkarte - die Information mitgeführt, aus welcher Richtung das Fahrzeug kam, um es nach Überfahren der Brücke in die richtige Richtung weiterzuleiten.
Die Brücke ist hier ein *kritischer Abschnitt*. Die Stelle s_5 besitzt die Funktion eines *Semaphors*, der den kritischen Abschnitt „Brücke“ kontrolliert.

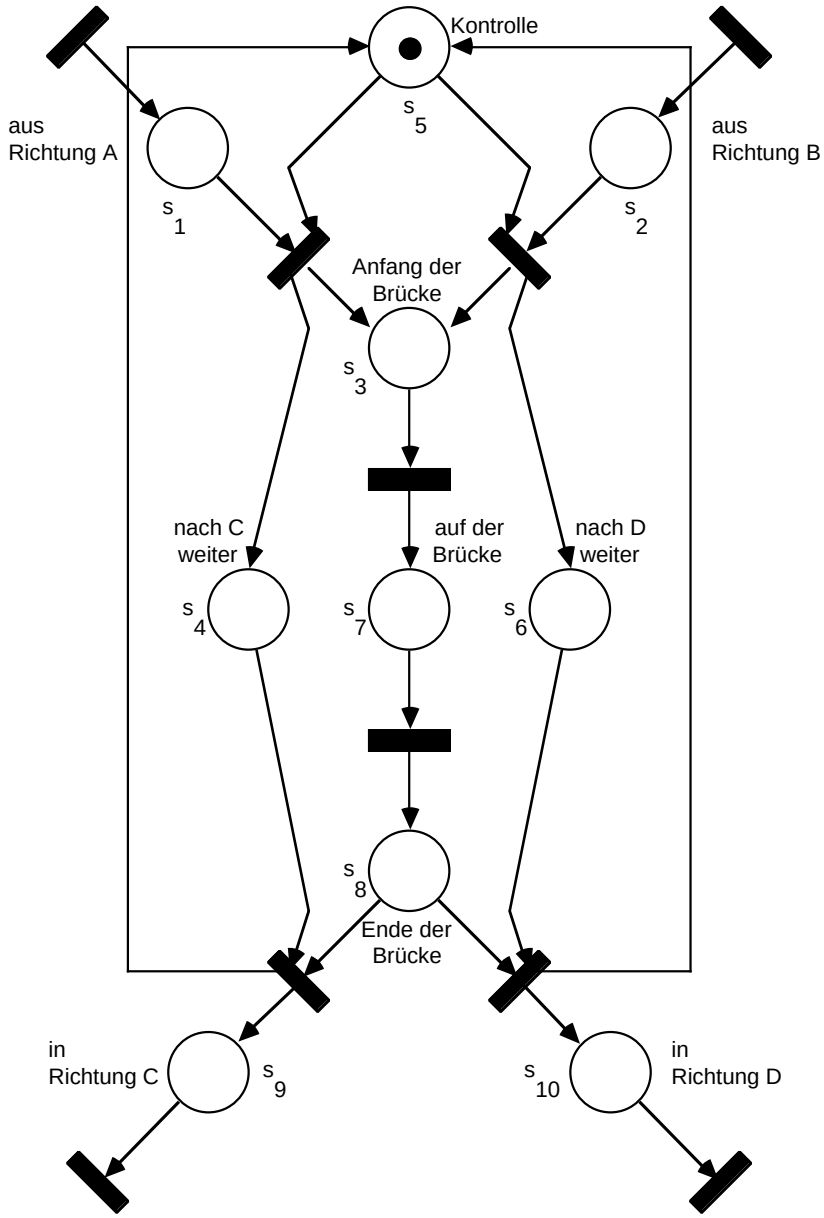


Abb. 8: Simulation der Brückenüberquerung

Aufgabe 2:

a) Definieren Sie das Petri-Netz aus Abb. 8 formal.

b) Führen Sie einige Schaltvorgänge durch und überzeugen Sie sich von der Korrektheit des Netzes.

c) Die Brücke wurde verstärkt: Nun können maximal zwei Fahrzeuge gleichzeitig die Brücke passieren. Ändern Sie das Petri-Netz aus Abb. 8 geringfügig ab, um die neue Situation zu modellieren.

d) Wir wollen nun annehmen, daß die beiden Wege in Richtung C und D auch gesperrt werden können. In diesem Falle sollen die Fahrzeuge in die noch freie Richtung gelei-

tet werden. Ergänzen Sie das Petri-Netz so um Stellen und Transitionen, daß in der Lösung (neben evtl. weiteren Stellen) vier ausgezeichnete Stellen C_{frei}, D_{frei}, $C_{gesperrt}$, $D_{gesperrt}$ existieren, über die man die Wege kontrollieren kann. Eine Marke in einer dieser Stellen symbolisiert, daß der entsprechende Weg frei oder gesperrt ist. Befindet sich also in Ihrer Lösung eine Marke in C_{frei} und eine Marke in D_{frei}, so verhält sich Ihre Lösung genauso wie das Petri-Netz aus Abb. 8.

Hat man ein System durch ein Petri-Netz dargestellt, so kann man sein Verhalten analysieren und daraus Erkenntnisse für das System herleiten, z.B. über Fehler, Engpässe und Verbesserungsmöglichkeiten. Typische Fragestellungen auf Petri-Netzen sind:

- *Terminiert* das Petri-Netz? D.h., kann man ausgehend von einer Startsituation stets nur endlich oft Transitionen schalten?
- Ist jede Transition *lebendig*? D.h., kann man ausgehend von einer Startsituation die Transitionen stets so schalten, daß eine vorgegebene Transition im weiteren Verlauf noch mindestens einmal schalten kann?
- Treten *Verklemmungen* auf? D.h., gibt es Situationen, in denen keine Transition schalten kann, die aber bei anderer Schaltreihenfolge hätten vermieden werden können?
- *Erreichbarkeitsproblem*. Gegeben seien zwei Markierungen M und M'. Gibt es eine Schaltfolge, mit der man ausgehend von der Markierung M die Markierung M' erreicht?

Beispiel: Ein großer Rechnerhersteller hat einmal Teile seines Betriebssystems mit Petri-Netzen analysieren lassen und dabei mögliche Verklemmungssituationen entdeckt, die beim Entwurf des Systems übersehen worden sind.

Aufgabe 3:
Modellieren Sie das Philosophenproblem (s. den gleichnamigen Beitrag von A. Schwill in diesem Heft) durch ein Petri-Netz.

Erweiterte Petri-Netze.

Unsere Petri-Netze haben eine Reihe von Nachteilen und sind daher für die Praxis (also für die Modellierung realer Systeme) noch recht wenig brauchbar, denn:

- Die durch Stellen dargestellten Plätze (bei der Abfüllstation z.B. das Zwischenlager) sind selten beliebig groß. Vielmehr haben sie meist nur eine begrenzte Aufnahmefähigkeit (s. Aufgabe 1). Man benötigt also für jede Stelle s eine Kapazität K(s), die angibt, wieviele Objekte höchstens in s liegen dürfen.
- Die Transportwege (bei der Abfüllstation z.B. die Gabelstapler), also die Kanten zwischen Stellen und Transitionen, können häufig mehrere Objekte gleichzeitig weiterleiten. Man benötigt also für jede Kante $k=(s,t)\in A$ eine Gewichtsfunktion g(k), die angibt, daß beim Schalten der Transition t genau g(k) Objekte aus der Stelle s abgezogen werden.
- Größtes Manko unserer Petri-Netze ist aber die Beschränkung auf natürliche Zahlen als Objekte. Allgemein hat man es mit beliebigen Objekten zu tun, die in Stellen gela-

gert und durch Transitionen in beliebiger Weise verarbeitet werden müssen.

Wie man Petri-Netze hinsichtlich des letzten Punktes erweitert, wollen wir im folgenden anschaulich erläutern. Man gruppiert dazu die möglichen Objekte zu Typen (vergleichbar zu Datentypen). Sei $D_1,...,D_n$ eine beliebige Menge von Objekttypen.

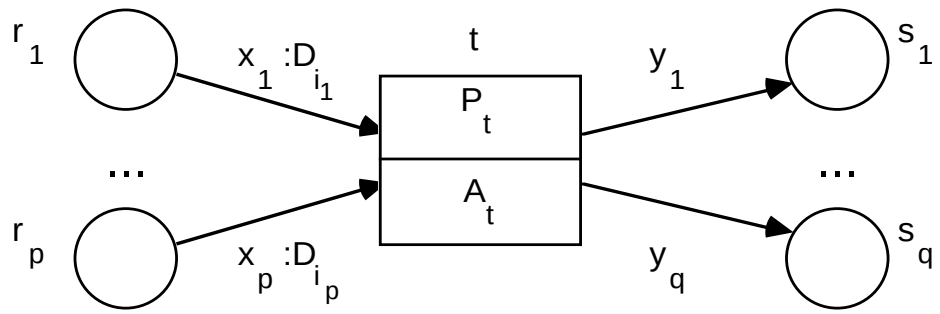


Abb. 9: Transition im erweiterten Petri-Netz

1) Damit Stellen beliebige Objekte aufnehmen können, ersetzt man die Markierung M: $S \rightarrow IN_0$ durch eine Abbildung M von S in die Menge aller Objekte über den Typen $D_1,...,D_n$.
2) Kanten von Stellen zu Transitionen sind mit Objekttypen und Bezeichnern der Form „x:D“ markiert, um darzustellen, daß nur Objekte des angegebenen Typs D die Kante durchlaufen dürfen. Das Objekt vom Typ D, das die Kante beim Schalten der Transition passiert, erhält für den Moment den Bezeichner x. Dieser Bezeichner ist nötig, um die Objekte innerhalb von P_t und A_t (s. Punkt 3)) ansprechen zu können (Abb. 9).
   Kanten von Transitionen zu Stellen sind nur mit Bezeichnern markiert. Die durch A_t berechneten Resultate werden entsprechend der Bezeichner weitergeleitet.
   Die Bezeichner an den einlaufenden und auslaufenden Kanten einer Transition müssen verschieden gewählt sein.
3) Jede Transition t (gem. Abb. 9) ist in zwei Teile unterteilt und besteht aus einem Prädikat P_t und einer Aktion A_t. Transitionen stellen wir nun nicht mehr durch Balken sondern durch zweigeteilte Kästen dar.
4) Eine Transition t mit den Eingabestellen $r_1,...,r_p$ ist aktiviert, wenn jedes r_i ein Objekt des Typs enthält, mit dem die Kante (r_i,t) markiert ist, so daß das Prädikat P_t für diese Objekte wahr ist.
5) Wenn t schaltet, so ändert sich die Markierung M folgendermaßen zu M’:
   - Aus jeder Eingabestelle r_i wird ein (bezgl. P_t passendes) Objekt abgezogen und mit x_i bezeichnet.
   - Auf die Objekte $x_1,...,x_p$ wird die Operation A_t angewendet. A_t möge die Ergebnisse $y_1,...,y_q$ liefern.
   - Die y_j werden gemäß der Bezeichner auf den Ausgangskanten auf die Ausgabestellen $s_1,...,s_q$ verteilt.

Beispiel: Modellierung einer Bibliothek.

Wir modellieren das Leihverfahren in einer Bibliothek. Die Bibliothek besteht aus zwei zentralen Organisationseinheiten, dem Magazin und einer Kartei der entliehenen Bücher. Zu jedem Buch gehört eine natürliche Zahl als Signatur und eine Karteikarte, auf der die Signatur verzeichnet ist. Befindet sich ein Buch im Magazin, so steckt die zugehörige Karteikarte vorne im Buch. Ist ein Buch ausgeliehen, so bewahrt man die Karteikarte im Karteikasten auf.

Mit der Bibliothek kommunizieren die Benutzer über drei Theken,

- die Bestelltheke, an der man einen Bestellzettel mit der Signatur des gewünschten Buches ausfüllt und abgibt,
- die Abholtheke, an der man das gewünschte Buch in Empfang nimmt oder, falls es ausgeliehen ist, den Bestellzettel mit dem Vermerk „entliehen“ zurückerhält,
- der Rückgabetheke, an der man entliehene Bücher zurückgeben kann. Zurückgegebene Bücher werden wieder mit der zugehörigen Karteikarte zusammengeführt und in das Bücherlager zurückgebracht.

Welche Objekttypen kommen in der Bibliothek vor? Es gibt Bücher und Karteikarten, die jeweils durch eine natürliche Zahl identifiziert werden. Ferner gibt es Bestellzettel, auf die man natürliche Zahlen oder den Vermerk „entliehen“ einträgt. Dann gibt es noch die Objektpaare (Buch, Karteikarte) und (Buch, Bestellzettel).

Bezeichnen wir mit $\mathrm{nr}(...)$ eine Funktion, die die Signatur eines beliebigen Objektes „Buch“, „Bestellzettel“ oder „Karteikarte“ liefert; und für ein beliebiges Paar (x,y) seien π_1 und π_2 die Projektionen, also $\pi_1(x,y)=x$ und $\pi_2(x,y)=y$. Abb. 10 zeigt ein Petri-Netz, das die Bibliothek modelliert.

Zur Arbeitsweise: Angenommen, jemand gibt an der Bestelltheke einen Bestellzettel mit der Zahl 5 ab, d.h. er möchte das Buch mit der Signatur 5 ausleihen. Dann kann in Abb. 10 die Transition „ausleihen“ schalten, denn das Prädikat ist für diesen Bestellzettel x und das Paar y:(Buch,Karteikarte) erfüllt, weil die Nummern von Bestellzettel (=x), Buch ($=\pi_1(y)$) und Karteikarte ($=\pi_2(y)$) übereinstimmen. Das Schalten von „ausleihen“ bewirkt nun, daß y in Buch u und Karteikarte v aufgespalten wird, wobei u zur Abholtheke und v zur Kartei der entliehenen Bücher wandert.

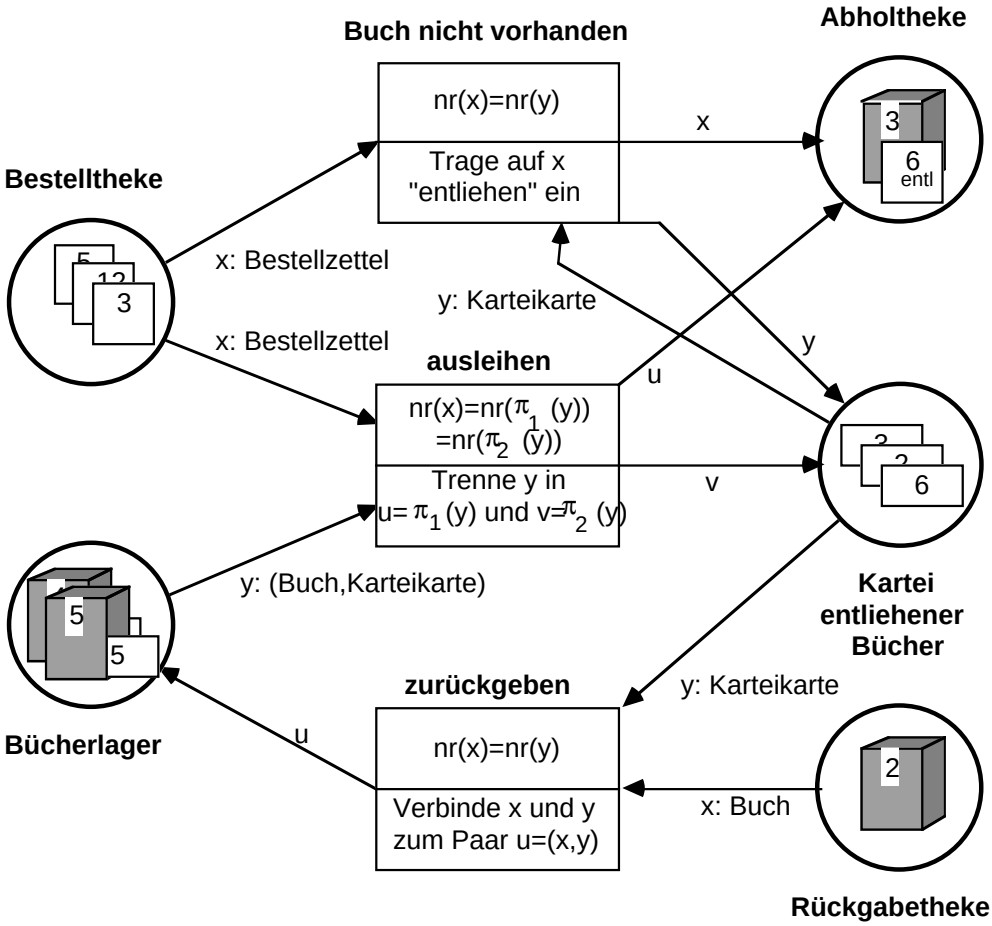


Abb. 10: Modell der Bücherei

Aufgabe 4:

Wir betrachten eine Frankiermaschine. Es sollen Briefe des Gewichts 10g, 40g und 80g ins Inland und Ausland verschickt werden. Gebührentabelle hierfür:

Gewicht	Gebühr/Inland	Gebühr/Ausland
10	0,80	1,20
40	1,30	1,80
80	1,90	2,30

Zur Verfügung stehen Briefmarken zu 50, 60 und 80 Pfennigen. Eine unbekannte Anzahl von In- und Auslandsbriefen wird der Maschine übergeben und soll korrekt frankiert werden. Modellieren Sie die Maschine durch ein erweitertes Petri-Netz.

Petri-Netze haben in den letzten Jahren einen Boom erfahren. Langfristig ist zu erwarten, daß sehr große Systeme auf hoher Ebene (nämlich in Form erweiterter Petri-Netze) entworfen und analysiert werden: Man erhält schnell lauffähige Prototypen, die entweder in maschinennahe Sprachen übersetzt werden oder die man per Hand in eine effiziente Version überträgt. Als Beispiele seien genannt: (auf mehrere Rechner verteilte) Betriebssysteme, Kommunikationssysteme, Netzpläne, Rechnerarchitekturen, Bürosysteme usw.

Didaktische Anmerkungen.

Wie die obigen Beispiele ansatzweise gezeigt haben, lassen sich mit Petri-Netzen alle wichtigen Begriffe und Phänomene der parallelen Programmierung, wie exklusiver Ausschluß, Verklemmung, Semaphor, Nichtdeterminismus, kritischer/unkritischer Abschnitt anschaulich studieren und beschreiben.

Petri-Netze können folglich im Unterricht in zweierlei Weise eingesetzt werden:

1) Als Ergänzung zu programmiersprachlichen Darstellungen unterstützen sie das Verständnis für parallele Konzepte, indem sie der mehr formalen und relativ maschinennahen Beschreibung durch eine Programmiersprache eine mehr informelle und für Schüler einprägsamere Darstellung auf höherem Niveau gegenüberstellen. Alle Problemlösungen sollten dann zunächst auf Petri-Netz-Basis entwickelt und erst anschließend in parallele Programme umgesetzt werden.
2) Auf der anderen Seite - und dies wird die zur Zeit in der Schule typische Situation sein - steht keine Programmiersprache mit parallelen Konzepten zur Verfügung. Übungen am Rechner sind ausgeschlossen, und der Unterricht wird zu einem „Trockenkurs“ mit den bekannten negativen Effekten auf das Verständnis und die Behaltensfähigkeit der Schüler. Mit Petri-Netzen kann man diese Begleiterscheinungen teilweise abmildern und den Unterricht dennoch attraktiv und lebendig gestalten. Reale Situationen können problemorientiert mit Petri-Netzen modelliert und auf dem Schreibtisch simuliert werden, wobei der dynamische Aspekt, anders als bei Programmen, die man „trocken“ nachvollzieht, deutlich sichtbar bleibt. Als Marken verwendet man bei Simulationen von Petri-Netzen über natürlichen Zahlen am besten Heftzwecken, die man umdreht und an der Nadel anfassend verschiebt, oder Geldstücke. Nichtdeterminismus realisiert man durch einen Würfel, mit dem man diejenige Transition auswürfelt, die unter mehreren gleichzeitig aktivierten im nächsten Schritt schalten soll.

   Später können Petri-Netze im Rahmen von Projekten in einer sequentiellen Programmiersprache auf dem Rechner implementiert werden. Hier bietet sich die Entwicklung von Programmen an, um Petri-Netze benutzerfreundlich eingeben, auf dem Bildschirm darstellen und manipulieren oder sie nach Eingabe einer Anfangsbelegung mit Marken vom Computer simulieren zu können.

Literaturhinweise

Baumgarten, B., Petri-Netze : Grundlagen und Anwendungen, BI-Wissenschaftsverlag 1990
Reisig, W., Systementwurf mit Netzen, Springer 1985
Rosenstengel, B. ; Winand, U.: Petri-Netze : eine anwendungsorientierte Einf}hrung, Vieweg 1991

Dr. Andreas Schwill
Fachbereich Mathematik/Informatik
Universität - Gesamthochschule Paderborn
Postfach 1621
D-33046 Paderborn