UNIVERSITÄT BERN INFORMATIK	KURS	TYP	BLATT	AUSGABE
	PN	V		1994

PETRI-NETZE

Script zur Vorlesung
Petri-Netze,
gehalten von Dr. P. Kropf
im Wintersemester 1993
Editor ist Nissim Buchs

Die Texte wurden
geschrieben von:

Blasenbrei Andreas
Frank Fredi
Guggisberg Karl
Held Michael
Kühni Daniel
Lörincze Georg
Matthey Thierry
Meier Markus
Mössinger Peter
Nottaris Jean-Luc

Bern, den 29. Juni 1994

Credits

Ich danke allen Studenten, die mitgeholfen haben, dieses Skript zu schreiben. Die meisten haben mir einfach dadurch sehr die Arbeit erleichtert, dass Sie mir die Texte im ASCII-Format per Diskette oder Mail zur verfügung gestellt haben. Andere haben extra für dieses Skript LaTeX gelernt, und mir so die Übungen schon in LaTeX abgegeben. Wieder andere haben defür gesorgt, dass ich die verwendeten Bilder im Postscript-Format erhalte. Und als letztes haben sich ein paar geopfert und das Skript nach Fehlern durchsucht.

All diesen Leuten danke ich viel mals...

bj

UNIVERSITÄT BERN	KURS	TYP	BLATT	AUSGABE
INFORMATIK	PN	V		1994

Inhaltsverzeichnis

1 Einführung

1.1 Allgemeines

1962 publizierte C.A.Petri die Petri-Netze. Petri-Netze sind begriffliche, theoretische Grundlagen für:

- Informationsübertragung
- Informationsverarbeitung
- Einheitliche und exakte Beschreibung

Petri-Netze sind ein Systemmodell für:

- Abläufe, Vorgänge, Zustände, Ereignisse
- Organisation
- Geräte, bei denen geregelte Informationsflüsse eine Rolle spielen

1.2 Charakteristik von Petri-Netzen

1. Kausalität von Ereignissen
   - Parallelität, Concurrency
   - Synchronisation, Kommunikation
   - Auswahl, Nichtdeterminismus
2. Beschreibung vom Aufbau, Arbeitsweisen und Eigenschaften von Systemen, die nicht unbedingt sequentiell ablaufen
   - Datenbanken
   - Steuerungen von Produktionssystemen
   - eingebettete Systeme
   - endlose Abläufe
3. Systeme auf verschiedenen Abstraktionsebenen, uniform und mit derselben Sprache
   - Bitmanipulationen
   - Benutzeroberflächen
   - Umgebung

   $\Rightarrow$ Strukturiertes Vorgehen
   - hierarchische Gliederung
   - Dekomposition (Aufteilung in Module)
   - Top-Down und Bottom-Up können verwendet werden

4. Abstraktion von Eigenschaften
   - Ereignisse
   - Zustände
   - paarweise Synchronisation
   - Variablen, Werte
   - Zuweisungen

   ⇒ passive Systembeschreibung (Struktur)
   - aktive Systembeschreibung (Dynamik)
5. Formale Analyse: Netztheorie
   - strukturelle Eigenschaften (Konflikte)
   - dynamische Eigenschaften
     - Sicherheit
     - Lebendigkeit
     - Deadlock
     - Trap

     ⇒ Korrektheitsbeweise

1.3 Petri-Netze als SE-Tool

Design (siehe Tab. 1)

- Analyse des Problems
- Systemdesign (erste Strukturen)
- Programmdesign (programmierbare Systemteile/Module)
- Implementation (Codierung)
- Test

Ziel Aus einer informellen Problemstellung will man ein Informationsverarbeitungssystem konstruieren, welches dann vollständig und präzis ist.

1.3.1 Anforderungen an eine Designmethode

- Uebersichtlichkeit, 'easy to use'
- Prototypen, Simulatoren
- iterierbare, schrittweise Verfeinerung
- Modularität, Problemzerlegung (decomposition)
- formale, präzise Beschreibung → möglichst beweisbar
- Computerumgebung soll erfasst werden können und äussere Umgebung auch

	Pseudo Code	Strukturiete Programmierung	Michael Jackson	Petri-Netze
Analyse			X	X
Systemdesign	x		X	X
Programmdesign	X	x	x	X
Implementation	X	X		x
Test	X	X		X

Tabelle 1: Vergleich verschiedener SE-Tools

1.3.2 Petri-Netze als Designmethode

1. Ermöglichen eine wichtige Unterscheidung zwischen passiven und aktiven Komponenten des Systems.
2. Stellen Beziehungen her zwischen einzelnen Komponenten des Systems.
3. Schrittweise Verfeinerung.
4. Statische und dynamische Beschreibung.
5. Formale Analyse $\rightarrow$ Korrektheitsbeweise.

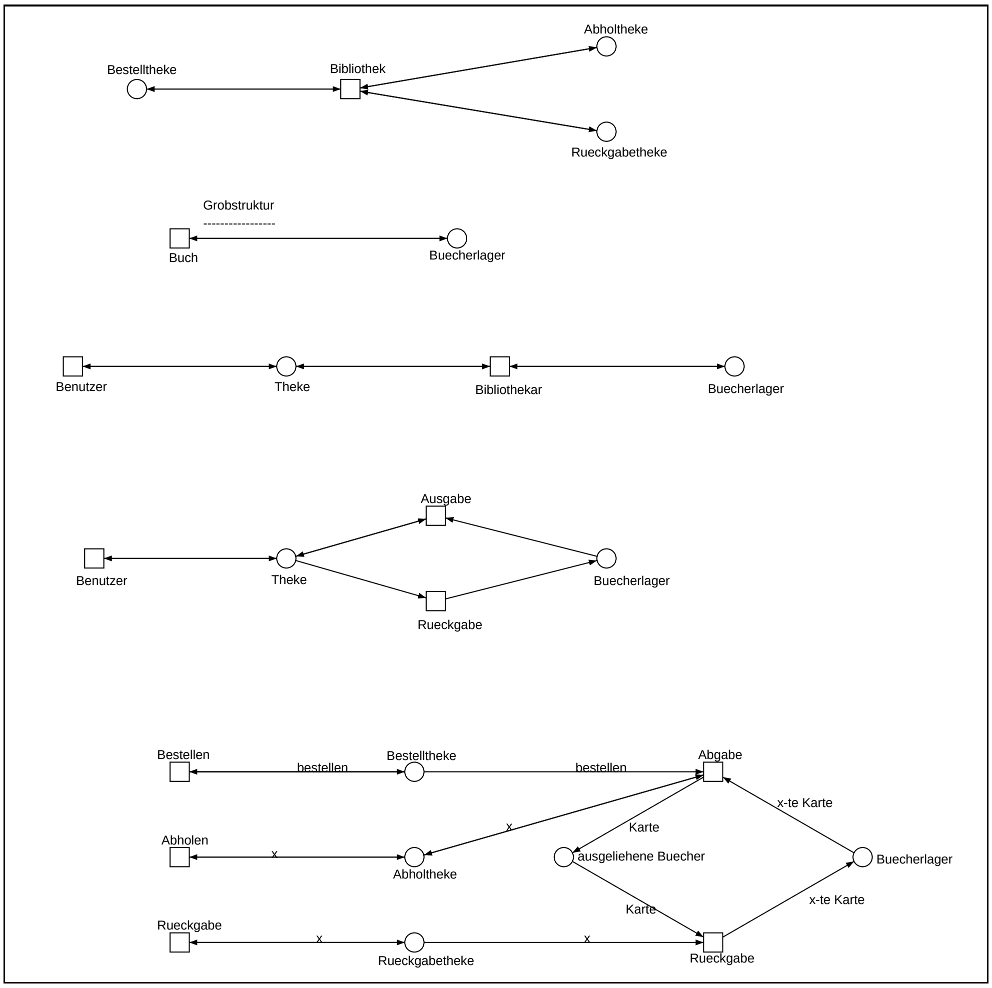
$\Rightarrow$ Lineare Algebra für Beweise Bsp. $Ax = b$

UNIVERSITÄT BERN	KURS	TYP	BLATT	AUSGABE
INFORMATIK	PN	V	4	1994

Beispiel: Organisation der Ausleihe und Rückgabe von Büchern in einer Bibliothek.

Sichtweisen:

- Benutzer
- Buch
- Verwaltung
- Bibliothekar
- Lieferant



Frage: Ist es möglich gleichzeitig ein Buch zu bestellen sowie zurückzugeben?
Antwort: Ja, ähh es ist möglich.

UNIVERSITÄT BERN	KURS	TYP	BLATT	AUSGABE
INFORMATIK	PN	V	5	1994

2 Elemente der Petri-Netz-Theorie

Petri-Netze sind äquivalent zu Turing-Maschinen (Church'sche These: TM ist adäquate Formalisierung des intuitiven Algorithmus-Begriffs). Mit Petri-Netzen ist also jede Berechnung darstellbar, modellierbar (d.h. Petri-Netze sind gleichmächtig wie TM). Mit Markierungen (Tokens) und Schaltregeln lässt sich das dynamische Verhalten eines Netzes untersuchen.
Es gibt mehrere Markierungsvarianten:
Ein-Marken-Netze, Mehr-Marken-Netze, individuelle Marken (Marken mit zusätzlichen Informationen).

Beispiel: Problem der Andenbahn Zwei Züge, einer von Bolivien nach Peru, der zweite von Peru nach Bolivien (siehe Abb. 1). In der Mitte der Strecke hat es einen einspurigen Tunnel. Man muss also die Züge synchronisieren, damit sie ohne crash aneinander vorbei kommen.

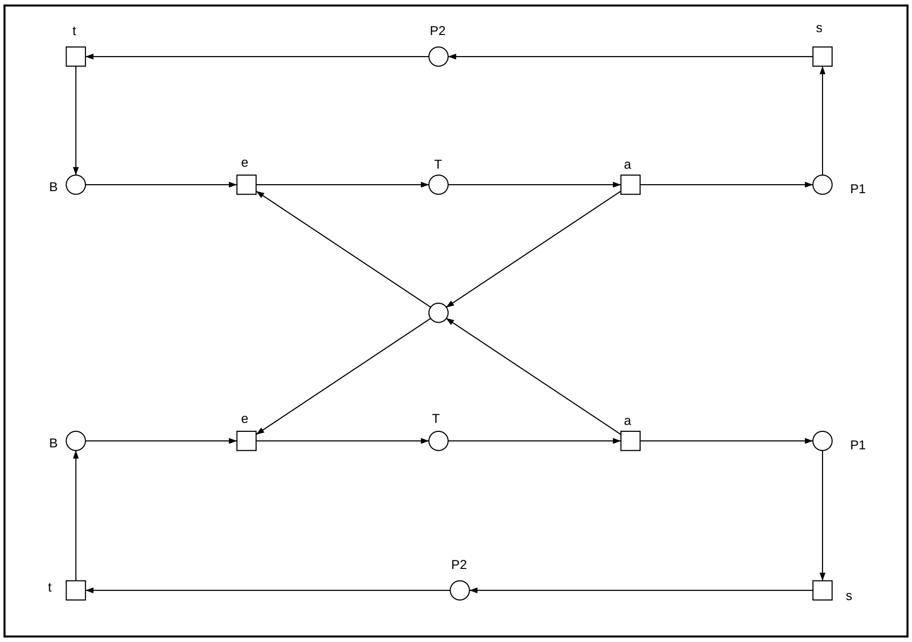


Abbildung 1: Beispiel Andenbahn

Zustände:

Px : Zug auf 1. und 2. Teilstück
T : Zug in Tunnel
B : Zug bereit für Tunneleinfahrt

Ereignisse

s : Bahnhof wird erreicht
t : Tunneleinfahrt erreicht
e,a : in Tunnel ein und ausfahren

2.1 Beschreibung von Petri-Netzen

Ein Petri-Netz kann als Graph dargestellt werden mit besonderen Eigenschaften:

Bedingung	Ereignis	Pfeil	Marken
Stelle	Transition	Datenfluss	Token
S-Element	T-Element		
passiv	aktiv	kausal-logische Beziehung	

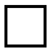
Tabelle 2: Elemente eines Petri-Netzes

1. Ein Petri-Netz ist ein gerichteter Graph.

2. Es gibt zwei Arten von Knoten:

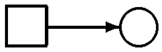
Plätze (Zustände)

Transitionen (Ereignisse)

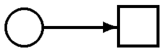


3. Kanten:

Transitionen → Plätze

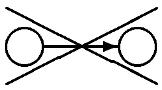


Plätze → Transitionen

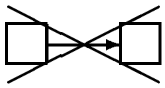


4. Nicht erlaubt ist:

Plätze → Plätze



Transitionen → Transitionen



5. Es gibt keine isolierten Knoten oder Mehrfachkanten!

2.2 Statische Elemente

Das Petri-Netz beschreibt den strukturellen Zusammenhang (welche Objekte sind beteiligt, wie hängen sie zusammen). Mit einem Netz sind alle formalen Möglichkeiten dargestellt.

2.3 Dynamische Elemente

1. Marken beschreiben das Verhalten
2. Marken sind Spielsteine, die verschoben werden können

2.3.1 Ein-Marken-Netze

In Ein-Marken-Netzen (Siehe Abb. 2) kommt nur eine Marke pro S-Element vor.

Schaltregeln

1. Ein Ereignis ist aktiviert, falls alle Vorbedingungen erfüllt und alle Nachbedingungen unerfüllt sind.
2. Jedes aktivierte Ereignis kann schalten (Marken verschieben, feuern). Es sei betont, dass das Element feuern kann, aber nicht zu einem bestimmten Zeitpunkt feuern muss (kein zeitlicher Takt)
3. Beim Schalten werden Vorbedingungen unerfüllt, Nachbedingungen werden erfüllt.

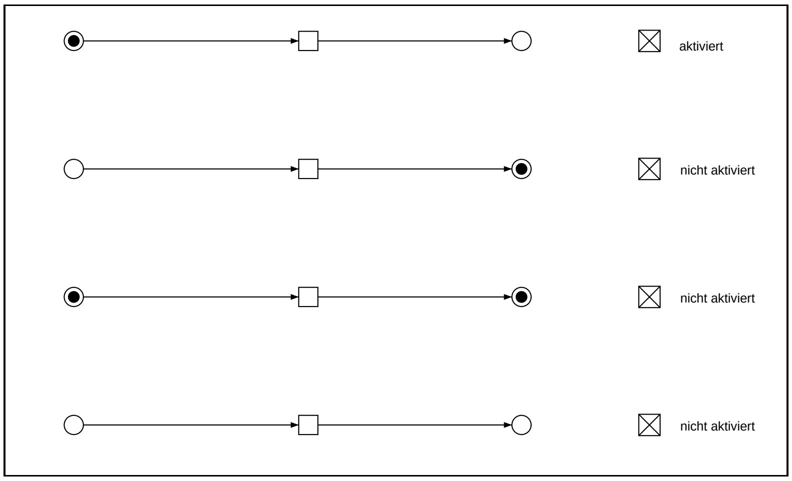


Abbildung 2: Ein-Marken-Netz

2.3.2 Mehr-Marken-Netze

In Mehr-Marken-Netzen (Siehe Abb. 3) können S-Elemente mehrere Marken enthalten, T-Elemente können auch mehrere Marken transportieren.

Schaltregeln

1. Ein Ereignis ist aktiviert, falls alle Vorbedingungen erfüllt, d.h. falls im Vorbereich genügend Marken vorhanden sind, und alle Nachbedingungen unerfüllt sind, d.h. genügend Platz im Nachbereich.
2. analog Schaltregeln Ein-Marken-Netze

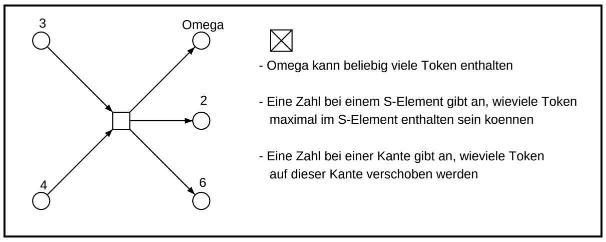
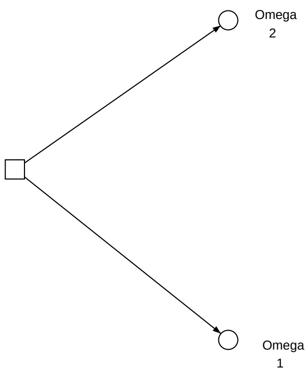


Abbildung 3: Mehr-Marken-Netz

2.3.3 Besondere Schaltsituationen

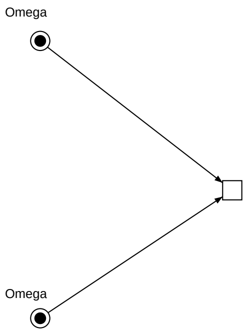
- Generator

  Transition ohne Vorbereich, ist nur von den Nachbedingungen abhängig. Die Transition ist immer aktiv.



- Senke (Absorber)

  Transition ohne Nachbereich, ist nur aktiv, wenn die Vorbedingungen erfüllt sind.



- Auswahl, Nichtdeterminismus
- Generell: Schalten wird als zeitlos angesehen
- Schlaufen:

  Ein-Marken-Netze ausgeschlossen
  Mehr-Marken-Netze möglich, aber nicht erlaubt

Beispiel: Copy Corner (siehe Abb. 4) mit folgenden Elementen:

- Eingabewarteschlange
- Verarbeitung
- Ausgangswarteschlange

Folgendes lässt sich beobachten:

- Parallele Abläufe
- Koordination
- Asynchroner Ablauf
- Nichtdeterminismus

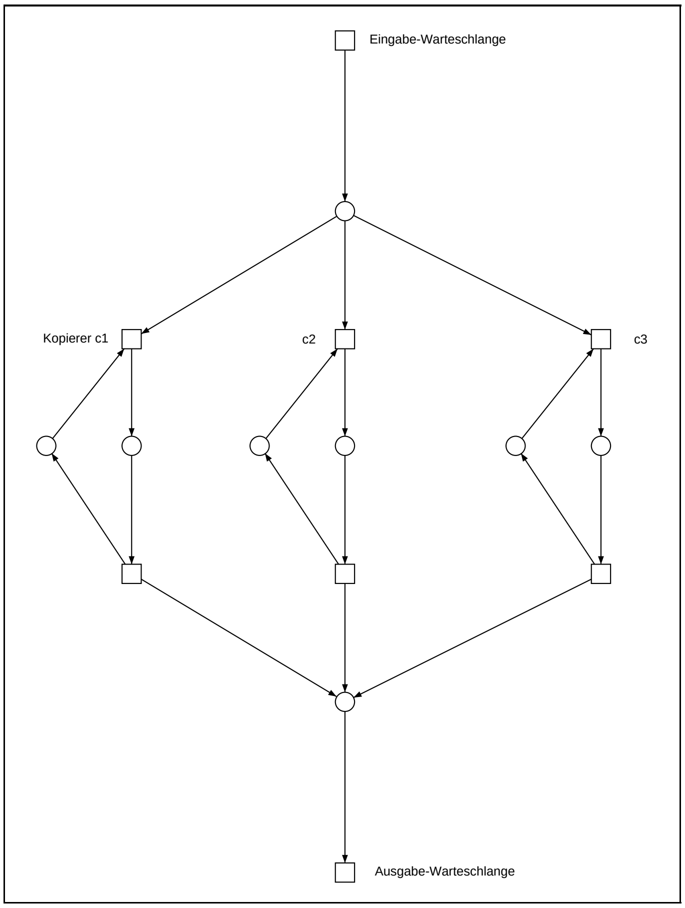


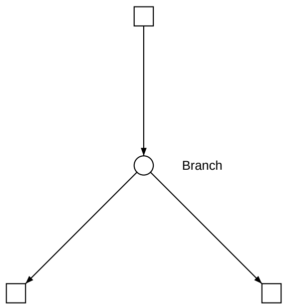
Abbildung 4: Copy Corner

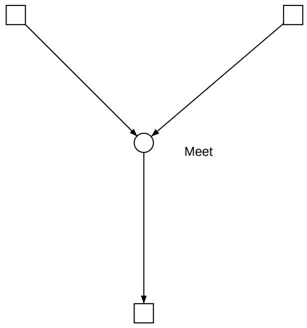
2.4 Strukturelle und dynamische Situationen

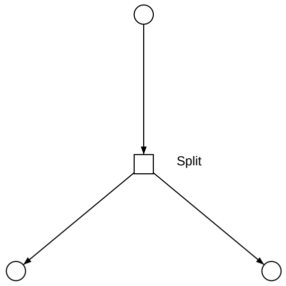
2.4.1 Begriffe des dynamischen Verhaltens

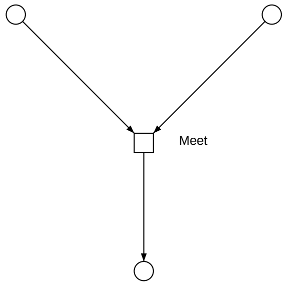
Fall	Markierung eines Netzes: Jede maximale Menge von Zuständen, die gleichzeitig realisiert sind.
Schritt	Uebergang von einem Fall zu einem Folgefall: Schalten von aktivierten Transitionen.
System	Netz mit allen zu betrachtenden Fällen: Fallmenge

2.4.2 Elementare Verknüpfungen





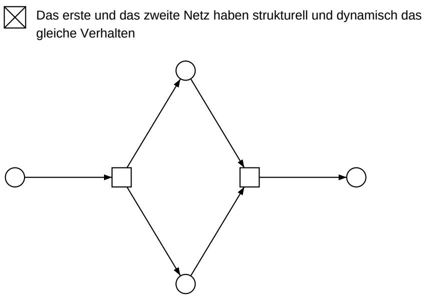


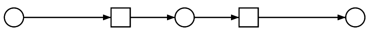


2.4.3 Charakterisierung

Reines Netz: ohne Schlaufen

Schlichtes Netz: ohne Redundanz

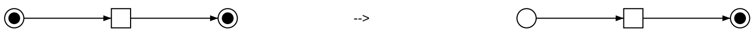




2.4.4 Markierungssituationen

Kontakt

Ein Platz im Nachbereich, welcher schon eine Marke hat. In diesem Fall ist Schalten nicht möglich, das heisst die folgende Transition ist nicht erlaubt:



Wie das Beispiel in Abb. 5 zeigt, wären je nach Schaltfolge zwei verschiedene Folgefälle möglich.

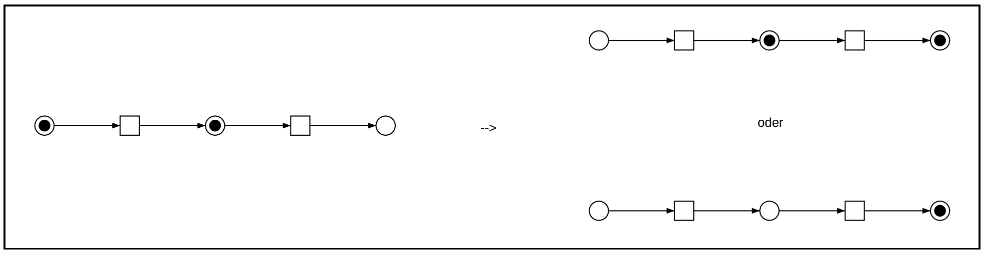
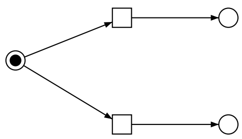


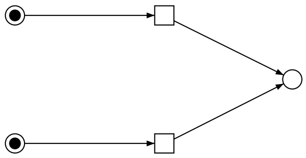
Abbildung 5: unbestimmte Schaltfolgen

Konflikt, Entscheidungsprobleme

- Verzweigungskonflikt, Markenmangel:

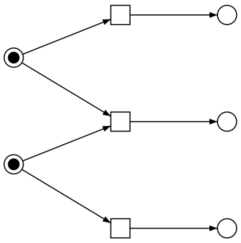


- Wettbewerbskonflikt, Markenüberangebot:

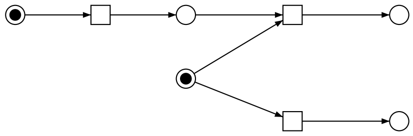


- Konfusion, verflochtene Konfliktsituation: ein Konflikt wird durch das Schalten einer dritten Transition gelöst.

symmetrisch



asymmetrisch



2.5 Dynamische Eigenschaften

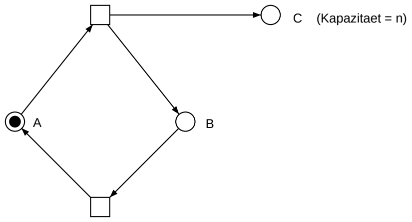
Die Petrinetztheorie erlaubt, analytische Aussagen über die Netzstruktur in Hinblick auf eine gegebene Markierung zu machen.

Lebendigkeit
Sicherheit
Deadlock-Freiheit
...

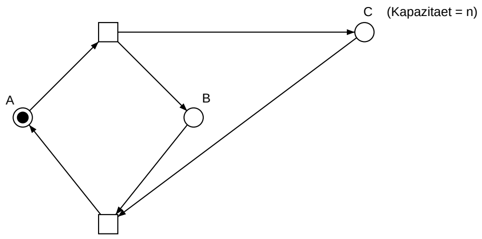
2.5.1 Sicherheit

Sicherheit in einem System bedeutet, dass in keinem Ablauf etwas 'Schlechtes' geschehen kann. Sicherheitsverletzungen treten typischerweise bei endlichen, beschränkten Abläufen

auf. Ein Netz heisst sicher bezüglich einer Anfangsmarkierung, falls durch das Schalten keine Stelle, mehr als eine maximale Anzahl Marken, belegt werden kann.
Das folgende Netz ist unsicher in C, da die Kapazität des Platzes C beschränkt ist:

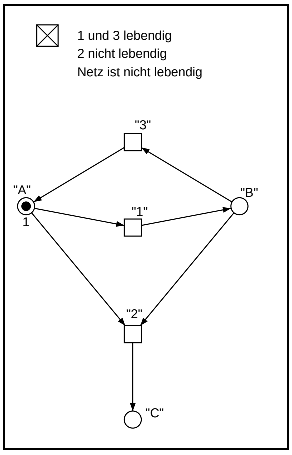


Hier wurde die Unsicherheit durch eine neue Verbindung behoben:



2.5.2 Lebendigkeit

Ein lebendiges System ist ein 'gutes' System. Ein Ereignis heisst lebendig, falls es nach endlich vielen Schaltvorgängen aktiviert wird. Ein Netz heisst lebendig, wenn alle Transitionen lebendig sind.

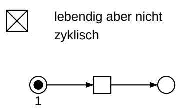


Lebendigkeit

2.5.3 Zyklische Netze

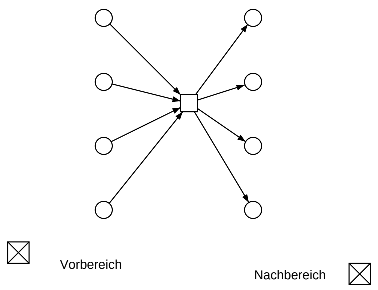
Jeder Fall lässt sich nach endlich vielen Schritten wiederholen.

Satz 2.1 *Satz: Jedes zyklische System ist lebendig. (Umkehrung gilt nicht!)*

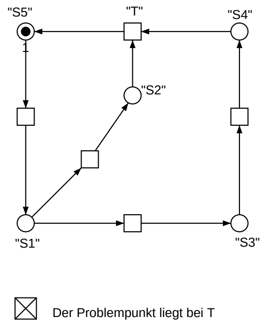


2.5.4 Deadlock

Stellenmenge S, einmal unmarkiert, dann kann sie nie mehr markiert werden. Dies ist genau dann der Fall, wenn keine Transition mehr schalten kann, die in ihrem Nachbereich eine Stelle aus S hat. V(S) Teilmenge von N(S) mit V=Vorbereich, N=Nachbereich



Beispiel:



Nach maximal 3 Schaltvorgängen kommen wir in einen Deadlock:
Stellenmenge $S_A = \{S_1, S_5\}$ oben durch.
Stellenmenge $S_B = \{S_1, S_3, S_5\}$ unten durch.

2.5.5 Trap

Einmal markierte Stellenmenge S, die nie mehr Marken verlieren kann. Dies ist der Fall, wenn jede Transition, die Marken aus S entnimmt auch wieder mindestens eine zurückgibt. N(S) Teilmenge von V(S). Trap entspricht einer 'Endlosschlaufe'.

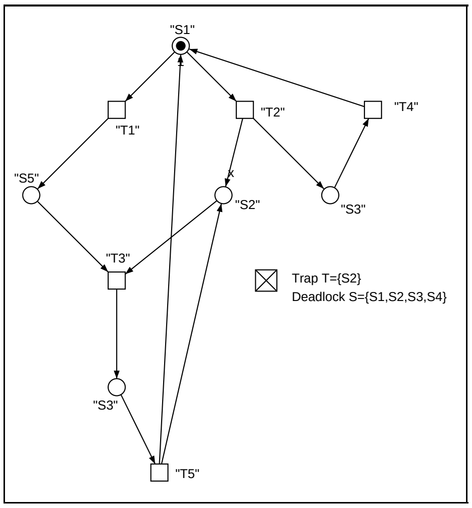


Abbildung 6: Trap und Deadlock

2.6 Eigenschaften von Petri Netzen

- Frage: Wie können die strukturellen (statischen) und die dynamischen Eigenschaften bestimmt werden ?
- Theorie:
  - formale Grundlagen, Definition
  - Theoreme
- Analyse: Qualitative Charakteristiken:

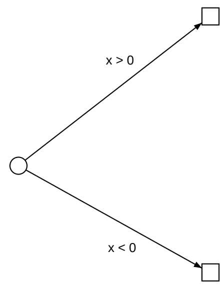
  - Lebendigkeit, Sicherheit
  - Beschränktheit
  - Erreichbarkeit

- Erreichbarkeit: alle möglichen Systemzustände (bezüglich Anfangsmarkierung); Erreichbarkeitsraum
- Erreichbarkeitsanalyse: Aufwand mit exponentiellem Wachstum
- Invariantenanalyse: Stelleninvariante, die keine Marken mehr hat, ist ein Deadlock. Berechenbar mit linearer Algebra → sehr grosse Gleichungssysteme.
- Quantitative Charakteristiken:
  - Mittlere Anzahl Feuerungen pro Zeiteinheit
  - Erwartete Anzahl Tokens pro Stelle (In Mehrmarkennetzen)
  - stochastische Prozesse → Markov Ketten Analyse
  - analytische Methoden stossen an Grenzen wegen Komplexität von Netzen.
    → Zerlegung in kleinere Teile
    → Simulation und die Analyse der Simulationsläufe
    → Parallele Simulation (Transputer Netzwerke), und dafür sind ja die Petri-Netze besonders geeignet.

2.7 Analyse von Petri-Netzen

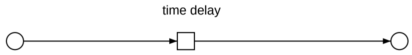
2.7.1 Zusätzliche Analysen von Petri-Netzen

Prädikatenanalyse



hier tritt ein Problem im Falle von $x = 0$ auf.

'frequency-analyse' für timed Petri-Netze



d.h. kann nur für eine vorbestimmte Zeit feuern ('firing delay').

3 Fundamentale Konzepte

3.1 Einführung

Petri-Netze sind ein Systemmodell und sind für die Spezifikation von eingebetteten Systemen (reaktive Systeme, sie reagieren auf die Umwelt) geeignet. Z.B. Betriebssyteme oder auch Kontrollprogramme / Steuerungen. Petri-Netze unterscheiden sich sehr stark von funktionalen Sprachen, wie PASCAL, die Problemlösungen nach dem 'Input → Computation → Ouput'-Schema (funktionelles System) verlangen.

Für die Spezifikation der Parallelität sind Petri-Netze sehr gut geeignet. Bei eingebetteten Systemen steht die inhärente Parallelität (vom Problem gegeben) im Vordergrund und nicht die Leistungssteigerung bei funktionalen Problemen.

3.1.1 Modelle von parallelen (concurrent) Systemen

In den 60er Jahren hat man das Multiprocessing eingeführt für die Lösung des Multiuserproblems. Weiter kamen 'shared variable' (Problem des gemeisamen Zugriffes) und 'distributed memory' (Kommunikation zwischen Prozessen) dazu und führten zu den folgenden wichtigen Modellen : CSP, CCS, I-structures, actors etc.

3.1.2 Ausführung von Systemen

Definition 3.1 (Prozess) *Prozess ist die Ausführung, das Verhalten einer Ausführung; worüber im Voraus nichts bekannt ist.*

Was hier neu betrachtet werden muss, ist die Parallelität (concurrency):

- 'interleaved run', die Reihenfolge der Ereignisse spielt keine Rolle.
- 'causale run', es wird eine kausale, partielle Ordung eingeführt. D.h. zwei Ereignisse können nicht immer zeitlich geordnet werden.

Definition 3.2 *Eine partielle Ordnung ist eine reflexive, transitive und antisymmetrische Relation.*

3.2 Einführungsbeispiel

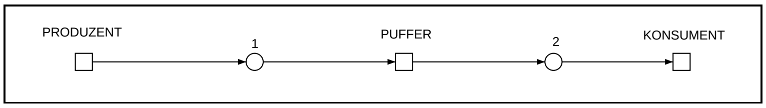


Abbildung 7: Produzent-Kunsument

Aufgabe (siehe Abb. 7): Produzent (generiert) → Puffer (Kapazität 1) → Konsument (absorbiert)
Ziel: gute, sinnvolle Beschreibung des Programmes und des Verhaltens
Lösungvorschläge:

1) PASCAL-ähnliche Sprachen mit dem zusätzlichen Paralleloperator $\|$. z.B mit concurrent PASCAL $P = (P1 \| P2)$ (siehe Uebungsserie 1 Petri-Netze theoretisch)

2) OCCAM-ähnlich

```
P1(CHAN p.c)      ;Produzent
  WHILE TRUE
  produce
  p.c ! data

P2(CHAN p.c)      ;Kosument
  WHILE TRUE
  p.c ? data
  consume

CHAN p.c
PAR
  P1(p.c)
  P2(p.c)
```

In diesem Lösungsbeispiel wir der Puffer vom System bereitgestellt. Ueber das Verhalten der möglichen Lösung wurde nichts gesagt. Für die Aussage über das Verhalten gibt es zwei Arten:

denotationell: Repräsentation + Semantik (z.B ADA).

operationell: Beschreibung des Verhaltens (z.B Petri-Netze).

3) Zustands-basierte Darstellung

Produzent:

rd (ready to deliver)
$\downarrow \qquad \uparrow$
rp (ready to produce)
$\uparrow$
init

Puffer:

fd (filled buffer)
$\downarrow \qquad \uparrow$
em (emty buffer)
$\uparrow$
init

Konsument:

rc (ready to consume)
$\downarrow \qquad \uparrow$
rg (ready to gather)
$\uparrow$
init

So kann das System auch nicht ganz vollständig beschrieben werden, denn Ueber gänge rd → rp, em → fd oder rg → rc, fd → em dürfen hier simultan erfolgen.

4) Ereignis-basierte Darstellung

Definitionen:

p	: produce
d	: deliver
c	: consume
q	: gather
$\bar{x}$	: Komplement von x

Darstellung:

consumer	= q.c.consumer
producer	= p.d.producer
buffer	= $\bar{d}.\bar{q}$.buffer

Hier treten Probleme der Paralleliät auf, da x und $\bar{x}$ simultan auftreten müssen.

5) Petri-Netz-Darstellung

Bis jetzt wurde das Problem immer dekomponiert:

p : producer
c : consumer
q : gather
d : deliver
rd : ready to deliver
rp : ready to produce
rc : ready to consume
rg : ready to gather
fd : filled buffer
em: empty buffer

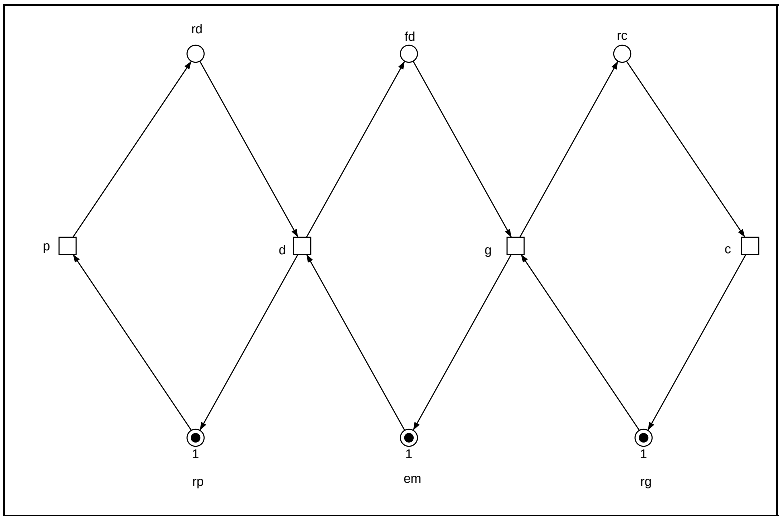


Abbildung 8: Beispiel Puffer

- Die Petri-Netz-Darstellung ist prinzipiel unstrukturiert.
- Petri-Netze sind Strukturen der Problemlösung.
- Aus dem Verhalten der Feuerregeln geht die Dynamik hervor.

3.3 Allgemeine Definition von Netzen

Definition 3.3 (Netz)
P, T disjunkte, endliche Mengen.
$F \subseteq (P \times T) \cup (T \times P)$ ist Relation.
$N = (P, T, F)$ ist ein Netz.
P sind die Plätze und T die Transitionen.

Notation:

P_N, T_N, F_N gehören dem Netz N an.
F ist Relation.
F^{-1} ist das Inverse von F.
F^+ ist die transitive Hülle von F.
F^* ist die reflexive, transitive Hülle von $F (F^* = F^{-1} \cup F^+)$.
$a \in N$ bedeutet $F(a)$ ist die Menge der Elemente in Relation zu a.
${}^\bullet a$ ist $F^{-1}(a)$ und ${}^\bullet A = \bigcup_{a \in A} {}^\bullet a$ ist Vorbereich (Preset).
$a^\bullet$ ist $F(a)$ und $A^\bullet = \bigcup_{a \in A} a^\bullet$ ist Nachbereich (Postset).
$x, y \in N$, dann gilt $x \in {}^\bullet Y$, falls $y \in X^\bullet$.

Definition 3.4 (Isomorphie) *Isomorphie zwischen zwei Netzen Σ und Σ', falls es eine Bijektion $\beta : N \to N'$ gibt.*

Definition 3.5

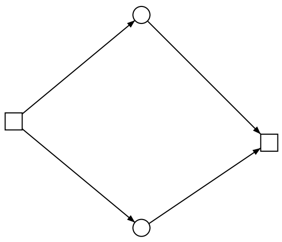
i) *$x \in N$, isoliert, falls der Vor- und Nachbereich leer sind $({}^\bullet x \cup x^\bullet = \emptyset)$.*

ii) *$x, y \in N$ beschreiben eine Schlaufe, falls $x(F \cap F^{-1})y$*

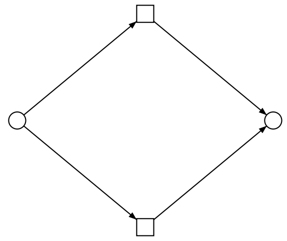
iii) *$x, y \in N$, detached (unverbunden), falls $({}^\bullet x \cup x^\bullet) \cap ({}^\bullet y \cup y^\bullet) = \emptyset$*

iv) *$A \subseteq N$, N ist A-einfach, falls $\forall x, y \in A$ gilt ${}^\bullet x = {}^\bullet y$ und $x^\bullet = y^\bullet \Rightarrow x = y$*

v) *N ist einfach, falls N N-einfach.*



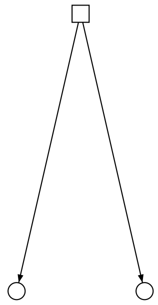
nicht P-einfach



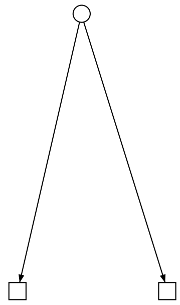
nicht T-einfach

vi) *N ist connected (zusammenhängend), falls $\forall x, y \in N$ gilt $x(F \cup F^{-1})^* y$*

vii) *N ist streng connected (zusammenhängend), falls $\forall x, y \in N$ gilt $x(F^*)y$*



nicht streng
zusammenhaengend/
connected



streng
zusammenhaengend/
connected

Beispiel: Petri-Netz (siehe Abb. 8)

- keine isolierten Elemente
- keine Schlaufen
- ein einfaches Netz
- zusammenhängend
- detached (überprüfbar mit $x = p$ und $y = g$)

Definition 3.6 (concurrency) *Nur detached Elemente können sich concurrent (parallel) verhalten.*

Satz 3.7 *Ist N ein einfaches Netz, so sind die P_N, T_N-einfachen Netze eindeutig durch Vor- (Pre-) und Nachbereich (Postset) bestimmt (vgl. Satz 3.2.7.4.11b), Analysis 6).*

Definition 3.8 (Subnetz) *N ist ein Subnetz (Unterstruktur) von N', $N \subseteq N'$, falls $P_N \subseteq P_{N'}, T_N \subseteq T_{N'}, F_N \subseteq F_{N'}$ gilt. D.h. im Subnetz gibt es keine weiteren Relationen, die nicht schon im Netz vorhanden sind.*

3.4 Elementare Netze

Definition 3.9 (es-Netz) *N sei ein Netz , $a \in P_N$*
Dann ist $\Sigma = (N, a)$ ein elementares Netz (es-Netz)
d.h. $\Sigma = (P_\Sigma, T_\Sigma, F_\Sigma, a_\Sigma)$

Definition 3.10 *Σ sei ein es-Netz*

i) $c \subseteq P_\Sigma$ heisst Zustand von Σ

ii) $t \in T_\Sigma$ ist aktiviert im Zustand c, falls $t \subseteq c$ und $(t^\bullet \setminus {}^\bullet t) \cap c = \emptyset$

iii) t sei aktiviert ($d \subseteq P_\Sigma$).
Dann ist $d = (c \setminus {}^\bullet t) \cup t^\bullet$ der Folgezustand von c mit t
(c, t, d) ist ein Schritt in Σ.
oder $c \overset{t}{\to} d$

Beispiel: Buffer (vgl. Abb. 8)
$\{rp, em, rg\} \overset{p}{\to} \{rd, em, rg\}$
$\{rd, em, rg\} \overset{d}{\to} \{rp, fd, rg\}$

Zustände c, d und Transitionen t sind gekoppelt $\Rightarrow$ backtrace, zurückschliessen
$x = l(y) \qquad y, d \not\to x$

Lemma 3.11 (Symmetrie) *Sei Σ ein es-Netz, $\quad c, d \in P_\Sigma, \quad t \in T_\Sigma$*

i) falls $c \overset{t}{\to} d$, dann $c = (d \setminus t^\bullet) \cup {}^\bullet t$

ii) falls $c \overset{t}{\to} d$, dann $c \setminus t^\bullet = d \setminus t^\bullet$

iii) $c \xrightarrow{t} d$, *falls* $c \setminus d = {}^\bullet t \setminus t^\bullet$ *und*
$d \setminus c = t^\bullet \setminus {}^\bullet t$ *und*
${}^\bullet t \subseteq c$

Beweis : Übung (Kann man Bedingung ${}^\bullet t \subseteq c$ weglassen?)
Das Eintreten eines Ereignisses t verursacht denselben Effekt in jedem Kontext.

Korollar 3.12 *Σ sei ein es-Netz,* $c, d, c^\bullet, d^\bullet \in P$, $t, u \in T_\Sigma$

i) *falls* $c \xrightarrow{t} d$, $c^\bullet \xrightarrow{t} d^\bullet$, *dann*
$c \setminus d = c^\bullet \setminus d^\bullet$
$d \setminus c = d^\bullet \setminus c^\bullet$

ii) *falls* $c \xrightarrow{t} d$, $c \xrightarrow{u} d$, *dann*
${}^\bullet t \setminus t^\bullet = {}^\bullet u \setminus u^\bullet$
$t^\bullet \setminus {}^\bullet t = u^\bullet \setminus {}^\bullet u$

iii) *sei* $c \xrightarrow{t} d$, $p \in P_\Sigma$, *dann gilt:*
$p \in {}^\bullet t \setminus t^\bullet \Rightarrow p \in c \setminus d$
$p \in t^\bullet \setminus {}^\bullet t \Rightarrow p \in d \setminus c$
$p \in {}^\bullet t \cap t^\bullet \Rightarrow p \in c \cap d$
falls $p \notin {}^\bullet t \cup t^\bullet$, *dann entweder* $p \in c \cap d$ *oder* $p \notin c \cup d$

iv) *falls* $c \xrightarrow{t} d$, $c \xrightarrow{t} d^\bullet$, *dann* $d = d^\bullet$

Korollar 3.13 *Σ sei ein es-Netz ohne Schlaufen,* $t, u \in T_\Sigma$, $c, d \in P_\Sigma$

i) $c \xrightarrow{t} d$, *falls* ${}^\bullet t \subseteq c, t^\bullet \cap c = \emptyset, d = (c \setminus t^\bullet) \cup t^\bullet$

ii) $c \xrightarrow{t} d$, *falls* $c \setminus d = {}^\bullet t$ *und* $d \setminus c = t^\bullet$

iii) *falls* $c \xrightarrow{t} d$, $c \xrightarrow{u} d$, *dann* ${}^\bullet t = {}^\bullet u, t^\bullet = u^\bullet$

iv) *sei* $c \xrightarrow{t} d$
falls $p \in {}^\bullet t$, *dann* $p \notin d$
falls $p \in t^\bullet$, *dann* $p \notin c$

v) $t \in {}^\bullet p$, *falls* $\forall a \xrightarrow{t} b$: $p \in b \setminus a$
$t \in p^\bullet$, *falls* $\forall a \xrightarrow{t} b$: $p \in a \setminus b$

3.5 RUNS

Ein RUN eines Netzes ist eine Menge von auftretenden Ereignissen, geordnet nach der Struktur des Netzes.
p,g unabhängig aktiviert :

- in beliebiger Reihenfolge
- gleichzeitig
- interleaving : Reihenfolge unabhängig (siehe Kapitel 3.6)
- concurrent / Kausale RUN's (siehe Kapitel **??**)

3.6 Interleaved Runs

Das Verhalten wird in bezug auf einen Beobachter definiert.

Interleaved RUN eines es-Netzes Σ :
$c_0 \overset{t_1}{\rightarrow} c_1 \overset{t_2}{\rightarrow} \dots \overset{t_n}{\rightarrow} c_n \quad c_0 = c_\Sigma$
$\rightarrow$ unabhängige Ereignisse werden willkürlich geordnet
$\rightarrow$ unendliche Mengen solcher Sequenzen

3.6.1 Transitions System

Ein endliches es-Netz hat ein endliches Transitions System

Definition 3.14 (Transitions-System) *Seien K, T disjunkte Mengen und $G \subseteq K \times T \times A$*
Dann ist $\mathcal{G} = (G, k)$ ein Transitions System

K	*: Knoten*
T	*: Kanten Nummern*
A	*: Kanten*
$k \in K$	*: Initialknoten von $\mathcal{G}$*
Kanten $(k, l, k^\bullet)$	*:* $k \overset{l}{\rightarrow} k^\bullet$
Initialknoten	*:* $\overset{t_1}{\rightarrow} k$

Definition 3.15 *Sei $\mathcal{G} = (G, k)$ ein Transitions System*

$k_0 \overset{t_1}{\rightarrow} k_1 \dots k_n$	*: Weg von $\mathcal{G}$*
$k_0, \dots, k_n$	*: erreichbare Knoten*
$k_{i-1} \overset{t_i}{\rightarrow} k_i$	*: Menge der erreichbaren Kanten*
$w = t_1, \dots, t_n \in T^*$	*: Wort in $\mathcal{G}$, k_n akzeptierter Zustand von w*
$\mathcal{L}(\mathcal{G})$	*: Sprache von $\mathcal{G}$*

Σ sei ein es-Netz, G die erreichbaren Zustände

i) (G, a_Σ) ist der Zustandsgraph von $\Sigma(sg(\Sigma))$

ii) $L(\Sigma) = L(sg(\Sigma))$ ist die Sprache von Σ

Beispiel eines Transfer-Systems (vgl. Abb. 9) :

- Zustandsgraphen von Petri-Netzen wachsen exponentiell.
- Gegeben sei $\Sigma^n = (P, T, F)$ mit n Plätzen. Dieses Netz sei so, dass $sg(\Sigma^n) \quad 2^n$ Knoten besitzt.
  Es gilt: $|P| = n \Rightarrow |\mathcal{P}(P)| = 2^n \quad (\mathcal{P}(P)$ ist die Potenzmenge von P).
- Für jedes Paar $A, B \subseteq P$ sei t eine Transition in Σ' mit $^\bullet t = A, t^\bullet = B$.
  Von einem beliebigen Anfangszustand ist jedes $0 \subseteq P$ erreichbar.
- Zustandsgraphen haben genügend Eigenschaften, um ein es-Netz bis auf Aequivalenz zu rekonstruieren.

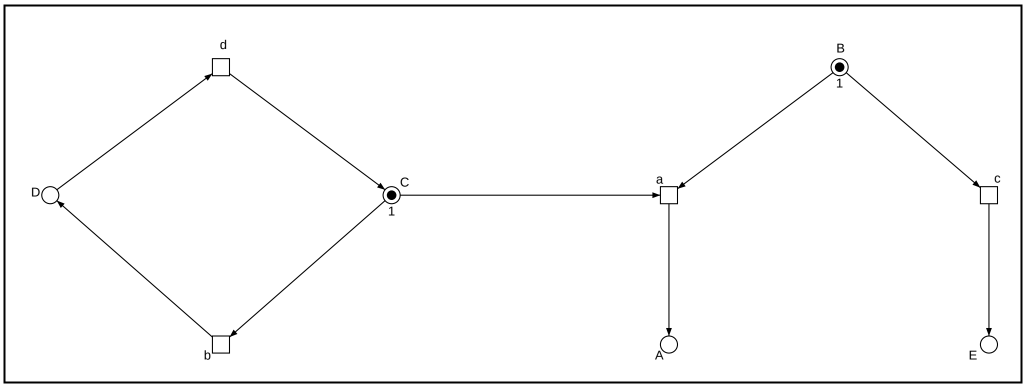


Abbildung 9: Ein Transfer-System

- Zustandsgraphen sind deterministische Transitions-Systeme.

Definition 3.16 *Ein Transitions-System ist deterministisch, falls für alle Kanten* $h \xrightarrow{\iota} k$ *und* $h' \xrightarrow{\iota} k'$ *gilt:* $h = h' \Rightarrow k = k'$.

Korollar 3.17 *Sei* Σ *ein es-Netz*

i) $sg(\Sigma)$ *ist deterministisch*

ii) jedes $\omega \in L$ *hat einen eindeutigen akzeptierenden Endzustand (dies folgt aus den Definitionen von Schritt und Zustandsgraphen).*

Anmerkungen (vgl. Abb. 10

- Es gilt: $L(\Sigma_1) = L(\Sigma_2)$.
- Zu Σ_3 : Mit gegenseitigem Ausschluss (Mutual Exclusion).

Definition 3.18 *Ein Graph* $\mathcal{G}$ *ist ein abstrakter Zustandsgraph falls ein es-Netz existiert mit* $sg(\Sigma)$.

Synthese-Problem

- Ist ein gegebender Graph $\mathcal{G}$ ein abstrakter Zustandsgraph ?
- Falls Ja: Konstruiere ein 'kleines' es-Netz mit $\mathcal{G} = sg(\Sigma)$.

3.7 Causal runs

Ein Beobachter kann nicht unterscheiden zwischen kausaler Ordnung des Systems und einer zusätzlich eingebrachten Ordnung. Unabhängigkeit kann nicht beobachtet werden, aber sie kann repräsentiert werden.
Es ist zu unterscheiden zwischen:

- Unabhängigen Ereignissen
- Ereignissen in beliebiger Reihenfolge
  (Beispiel in Σ_1 : a, c unabhängig)
  (Beispiel in Σ_2 : a, c Reihenfolge unabhängig)

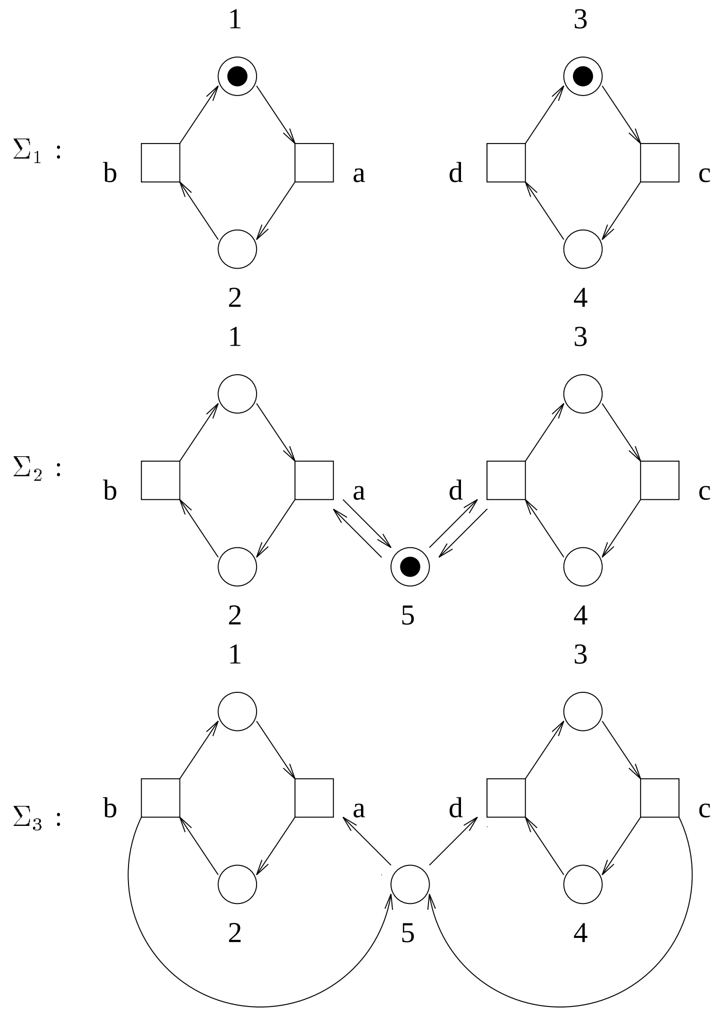


Abbildung 10: Beispiel zu Interleaved Runs

3.7.1 Konflikte bedeuten Entscheidungen

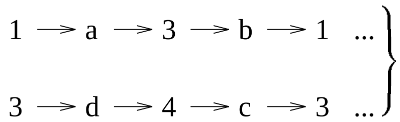
$\Rightarrow$ Verschiedene Abläufe (runs)
(Beispiel in Σ_1 : Unendlich viele verschiedene runs)
(Beispiel in Σ_1 : Ein eindeutiger unendlicher Ablauf)
Ein run kann durch das Aufschreiben von auftretenden Ereignissen beschrieben werden ($\Rightarrow$ Kausale Ordnung).
$\Rightarrow$ **Occurrence-Net**

Definition 3.19 (Occurence-Net) *Ein Netz K heisst Occurrence-Net, falls F_k^+ irreflexiv ist und $\forall p \in P_k$ gilt: $|{}^\bullet p| \leq 1$ und $|p^\bullet| \leq 1$.*
F_k^+ : $xF_k^+y \Rightarrow x \neq y$, oder $F_k^+ \cap id = \emptyset$

UNIVERSITÄT BERN	KURS	TYP	BLATT	AUSGABE
INFORMATIK	PN	V	24	1994

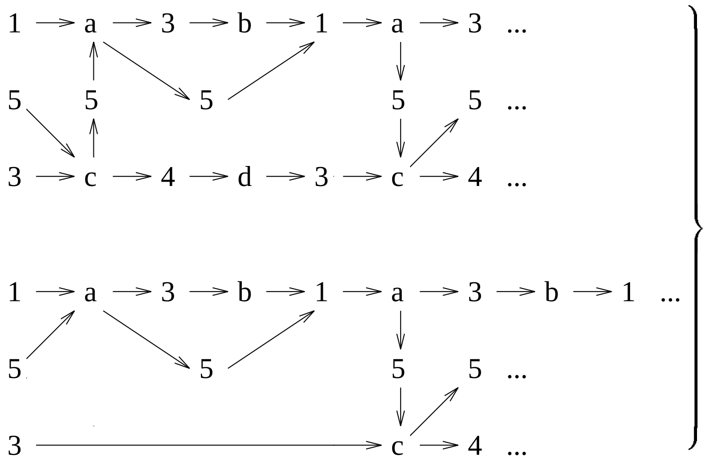
Beispiel (vgl. Abb. ??) :

Σ_1 :



unabhängig

Σ_2 :



2 verschiedene runs

Satz 3.20 *Sei K ein Occurrence-Net. Dann ist F_k^+ eine strenge partielle Ordnung (transitiv, reflexiv).*
$F_k^+ : <_k, <\quad (<^{-1} = >)$

Nicht vergleichbare Elemente, Unordnung $\Rightarrow$ Concurrency, Parallelität.

Definition 3.21 (concurrency) *Sei K Occurrence-Net. Zwei Elemente $a, b \in K$ heissen concurrent, falls weder $a < b$ noch $b < a$ gilt. Schreibweise: $a \overset{o}{\rightarrow} b$*
(reflexiv, symmetrisch, nicht transitiv)

- *$S \subseteq K$ heisst co-Menge, falls alle Elemente von S paarweise concurrent sind.*
- *Eine co-Menge S heisst Schnitt von K, falls S keine echte Teilmenge einer co-Menge S' von K ist.*

Definition 3.22 (casual run) *Sei Σ ein es-Netz und K ein Occurrence-Net. Eine Abbildung $v : K \rightarrow \Sigma$ heisst causal run von Σ, falls:*

i) $r({}^\bullet k) = a_\Sigma$

ii) $h, k \in K$ *: falls h co-Menge, dann* $r(h) \neq r(k)$.

iii) $\forall t \in T_k : r({}^\bullet t) = {}^\bullet r(t), \quad r(t^\bullet) = r^\bullet(t)$

Anmerkung:

${}^\bullet k$: Menge, die ein kleinstes Element bez. der kausalen Ordnung enthält.

$k^\bullet$: Menge, die ein grösstes Element bez. der kausalen Ordnung enthält.

UNIVERSITÄT BERN	KURS	TYP	BLATT	AUSGABE
INFORMATIK	PN	V	25	1994

Substrukturen

Definition 3.23 *Sei Σ ein es-Netz und $r : K \to \Sigma$, $r' : k' \to \Sigma$ seien causal runs von Σ.*

i) r' ist ein subrun von r, falls k' ein sub-Netz von K ist und $\forall k \in K'$ gilt: $r(k) = r'(k)$.

ii) r ist ein maximaler run von Σ , falls r ein subrun von r selbst ist.

Beispiel (vgl. Abb. 8):

$$\begin{tikzcd} rp \arrow[r] & p \arrow[r] & rd \arrow[r] & rp \arrow[r] & p \\ em \arrow[urr] & & & & \\ rg \arrow[rr] & & g \arrow[uu] & & \end{tikzcd}$$

Buffer hat genau einen maximalen run.

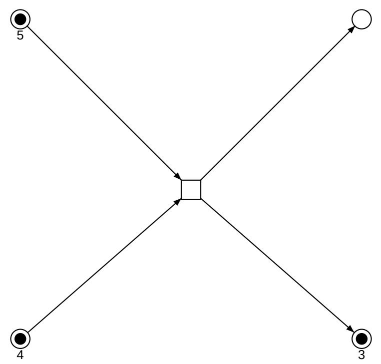
Definition 3.24 *Ein Netz Σ ist deterministisch, falls nur genau ein maximaler run von Σ existiert.*

3.8 kausale/concurrent runs

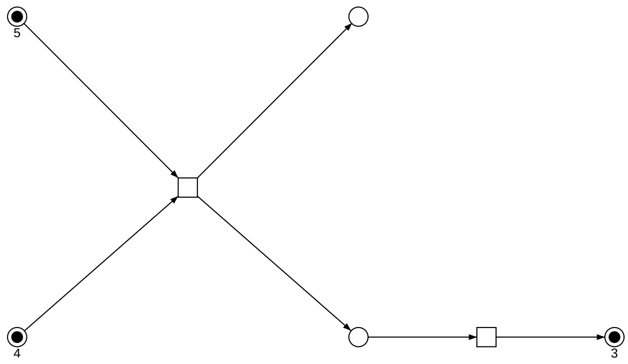
Σ kontaktfrei (nur noch $^\bullet e$ als Vorbedingung)

- genügend Marken im Vorbereich ($^\bullet e$)
- genügend Platz im Nachbereich ($e^\bullet$)

Beispiel eines nicht zugelassenen, nicht kontaktfreien Netzes:



Allerdings ist eine Transformation möglich, um daraus ein kontaktfreies Netz zu machen:



Definition 3.25 (kontaktfrei) *Σ ist kontaktfrei, falls für jeden Schnitt S jedes beliebigen Runs k eines Netzes Σ und für beliebige Transformationen $e \in T$ gilt:*
$^\bullet e \subseteq l(s) \Rightarrow e^\bullet \cap l(s) = \emptyset$
l: beliebiges labeling

Die Feuerungsregeln sind nur für Input definiert. Voraussetzung: $e^\bullet$ immer frei.
C: Menge aller Fälle (Zustandsmengen, Markierungen)
$^\bullet e \subseteq C \Rightarrow e^\bullet \cap C = \emptyset$

3.9 Fundamentale Situationen bei kontaktfreien Netzen

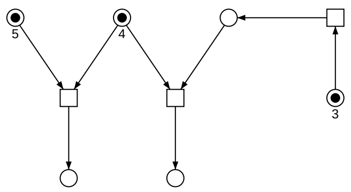
t_1, t_2 Ereignisse können in einem Zusammenhang stehen.

i) Sequenz: t_1 kann bei $c \subseteq P_\Sigma$ auftreten, nicht aber t_2

ii) Konflikt: t_1 und t_2 können bei $c \subseteq P_\Sigma$ getrennt auftreten, aber nicht beide zusammen

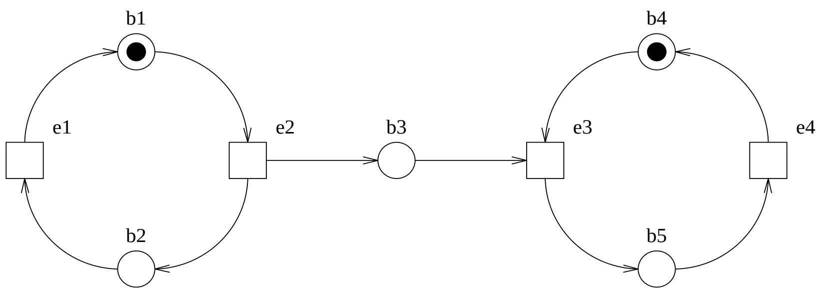
iii) concurrency: t_1 und t_2 können in beliebiger Reihenfolge auftreten (in einem Schritt)

iv) Konfusion: lässt sich auf Konflikt und concurrency zurückführen:

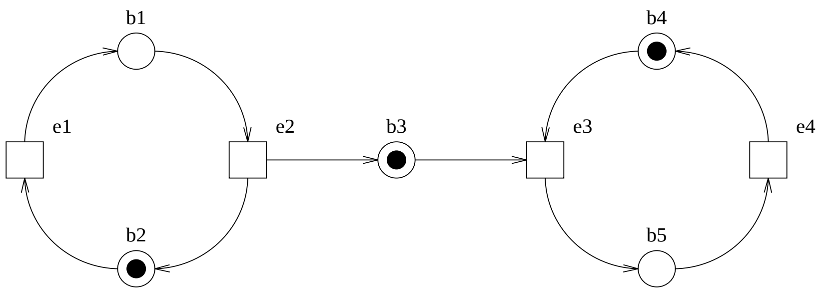


e_1, dann $e_2 \Rightarrow$ Konflikt nicht gelöst
e_2, dann $e_1 \Rightarrow$ Konfliktlösung

$M = (1, 0, 0, 1, 0)$ **Ereignis e_2 aktiviert**



Markierungen nach Eintritt von e_2



3.10 Unterklassen von es-Netzen

Definition 3.26 *Sei Σ ein kontaktfreies es-Netz (vgl. Abb. 11).*

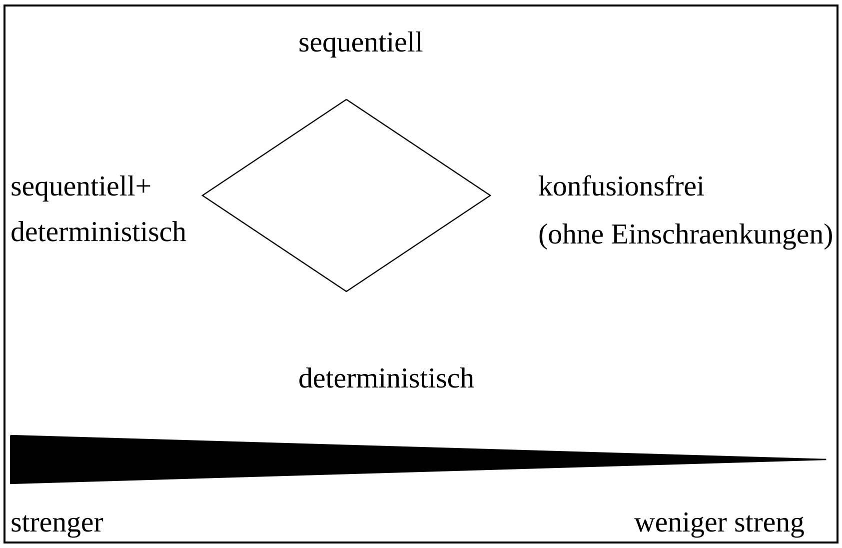


Abbildung 11: Klassifikationsschema

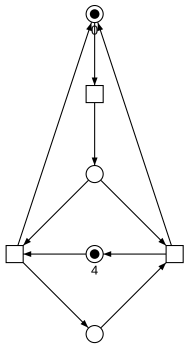
i) Σ *ist* *sequentiell, falls* $|u| = 1 \quad \forall u \subset U_\Sigma$ *mit* $U_\Sigma \subseteq 2^T$
$|u|$ = *aktivierte Transitionen* U_Σ = *Menge aller aktivierten* t *für jeden* $c \subseteq P_\Sigma$

ii) Σ *ist* *deterministisch, falls* $\forall c \subseteq P_\Sigma$ *und* $\forall t_1, t_2 \in T$ *gilt :*
$c \overset{t_1}{\to} d$ *und* $c \overset{t_2}{\to} e \Rightarrow c \overset{t_1,t_2}{\to} c'$

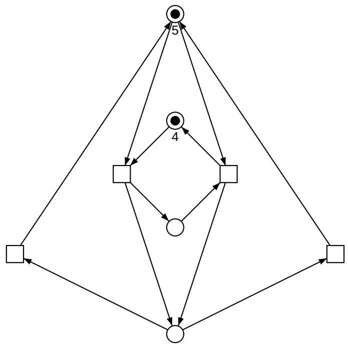
iii) Σ *ist* *konfusionsfrei, falls* Σ *ohne Konfusion ist.*

Beispiele:

a) sequentiell, deterministisch, konfusionsfrei:



b) sequentiell, nicht deterministisch, konfusionsfrei:



UNIVERSITÄT BERN	KURS	TYP	BLATT	AUSGABE
INFORMATIK	PN	V	28	1994

4 Dynamisches Verhalten

4.1 Markierung

Definition 4.1 *Markierung M von Σ ist $M : P \to \{0,1\}$*
$M = (M(P_1), M(P_2), ..., M(P_n))$
$\Rightarrow$ *n-dimensionaler Vektor $n = |P|$ (Anzahl Zustände/Plätze)*

Definition 4.2 (aktiviert) *$e \in T$ ist aktiviert, falls:*
$(b,e) \in F \Rightarrow M(b) = 1 \quad \forall b \in {}^{\bullet}e$ *und* $(e,b) \in F \Rightarrow M(b) = 0 \quad \forall b \in e^{\bullet}$
Tritt das Ereignis e ein:
$\Rightarrow$ *neue Markierung M'*
$M'(b) = 0, falls \quad (b,e) \in F$
$M'(b) = 1, falls \quad (e,b) \in F$
$M'(b) = M(b)$ *sonst*
Bsp: $M = (1,0,0,1,0) \to (0,1,1,1,0)$
aus $c \xrightarrow{t} d$ *folgt eine neue Notation:*
$M[e]M'$ *Eintritt eines Ereignises e von M nach M'*

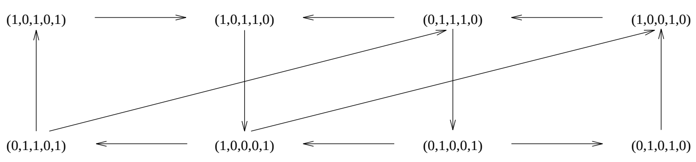
Definition 4.3 (Erreichbarkeitsraum) *Σ sei ein Netz, M_0 eine Anfangsmarkierung.*
Die minimale Menge $[M_0]$ mit
- $M_0 \in [M_0]$
- *falls $M' \in [M_0]$ und $\exists e$, M'' mit $M'[e]M''$, dann gilt $M'' \in [M_0]$*

heisst Erreichbarkeitsraum.

Erreichbarkeitsraum mit Vektoren :

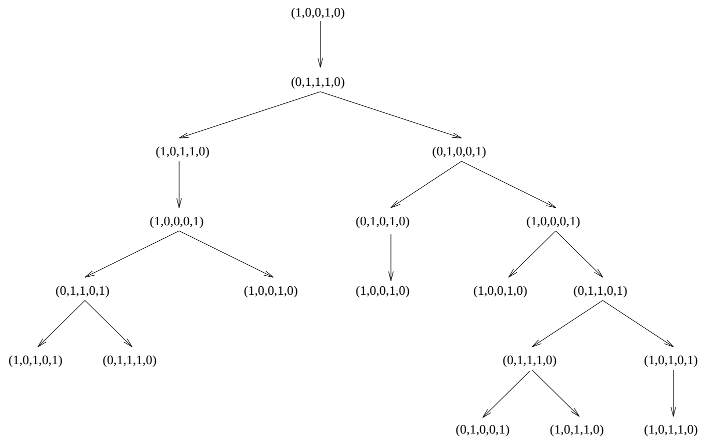
$$M_0 = \left\{ \begin{pmatrix}1\\0\\0\\1\\0\end{pmatrix}, \begin{pmatrix}0\\1\\1\\1\\0\end{pmatrix}, \begin{pmatrix}1\\0\\1\\1\\0\end{pmatrix}, \begin{pmatrix}1\\0\\0\\0\\1\end{pmatrix}, \begin{pmatrix}0\\1\\0\\0\\1\end{pmatrix}, \begin{pmatrix}0\\1\\0\\1\\0\end{pmatrix}, \begin{pmatrix}0\\1\\1\\0\\1\end{pmatrix}, \begin{pmatrix}1\\0\\1\\0\\1\end{pmatrix} \right\}$$

Erreichbarkeitsraum grafisch :



UNIVERSITÄT BERN INFORMATIK	KURS	TYP	BLATT	AUSGABE
	PN	V	29	1994

Erreichbarkeitsbaum :



5 Petri-Netze (Net-Systems), S/T-Netze

Frage: Wie beeinflusst die Struktur eines Netzes das Verhalten? Wie kann man dynamische Eigenschaften eines Netzes aus seiner Struktur ablesen?
Erweiterung eines es-Netzes: Mehrmarkennetze

Definition 5.1 (Petri-Netz) *Ein Petri-Netz ist ein Netz ohne isolierte Elemente wobei* $N = (S, T, F)$ *mit* $dom(F) \cup codom(F) = S \cup T$

Definition 5.2 (S/T-Netz (Net-System)) *Ein S/T-Netz* Σ *ist ein Netz, wobei:*

$\Sigma = (S, T, F, K, W, M_0)$
(S,T,F) = *Petri-Netz*
K = *Kapazitätsfunktion* $K : S \to \mathcal{N}^+ \cup \infty$ *Kapazität der Stellen / Zustände*
W = *Kapazität / Gewicht der Kanten* $W : F \to \mathcal{N}^+$
M_0 = *Gewicht der Stellen / Initialisierungsmarkierung* $M_0 : S \to \mathcal{N} \quad M_0(s) \leq K(s) \quad \forall s \in S$

- Elementare (es-)Netze: $K(s) = 1 \quad \forall s \in S$
  $W(x, y) = 1 \quad \forall (x, y) \in F$
  $\Rightarrow$ es-Netz ohne isolierte Elemente.
- Markierung/Schaltregeln:
  Σ sei ein S/T-Netz, $M : S \to \mathcal{N}$ mit $M(s) \leq K(s) \quad \forall s \in S$
- aktivierte Transition bei M:
  $W(s, t) \leq M(s) \quad \forall s \in {}^\bullet t$
  $M(s) \leq K(s) - W(t, s) \quad \forall s \in t^\bullet$
- Folgemarkierung M:
  $M'(s) = M(s) + W(t, s) - W(s, t)$

5.1 Dynamische Eigenschaften

Lebendigkeit $\to$ Auftreten von t-Elementen
Sicherheit $\to$ Anzahl der Token in den s-Elementen

Definition 5.3 *Sei* $N = (S,T,F)$ *ein Petri-Netz, M eine Markierung*

(i) $t \in T$ *ist lebendig bei M, falls* $\forall M' \in [M] \quad \exists M'' \in [M']$ *und t ist bei* M'' *aktiviert*
oder
t ist lebendig bei M, falls t nach endlich vielen Schritten aktiviert wird.

(ii) N ist lebendig bei M, falls jedes $t \in T$ *lebendig ist bei M.*

(iii) $s \in S$ *ist sicher bei M, falls* $M'(s) \leq 1 \quad \forall M' \in [M]$
n-Sicherheit (Mehrmarkennetze): $M'(s) \leq n$

(iv) N ist sicher bei M, falls $s \in S$ *sicher ist bei M. (analog n-Sicherheit)*

(v) N ist zyklisch bezüglich M_0, falls sich jede erreichbare Markierung reproduzieren lässt.

(vi) t ist tot bei M, falls t nicht aktiviert bei M' $\quad \forall M' \in [M]$

(vii) s ist unbegrenzt bei M, falls $\forall k \in \mathcal{N}_0 \quad \exists M' \in [M] M'(s) > k$

Eine tote Transition ist sicher nicht lebendig (bei M), aber das umgekehrte gilt nicht, d.h. nicht jede nicht tote Transition ist auch lebendig (ev. wird die Transition erst nach ∞ vielen Schritten aktiviert!!!).

5.2 Free Choice Netze

Free Choice Netze (FC-Netze) sind eine Subklasse der Petri Netze, in der keine Konfusionen vorkommen dürfen.
Konflikte zwischen Transitionen sollen nur gelöst werden können, indem eine der Transitionen schaltet. Neben einer gemeinsamen Vorstelle $s \in {}^\bullet t$ können die Transitionen keine weiteren Konflikte haben:
$\Rightarrow$ keine Vorwärts- und Rückwärtsverzweigungen zusammen!
Beispiel :

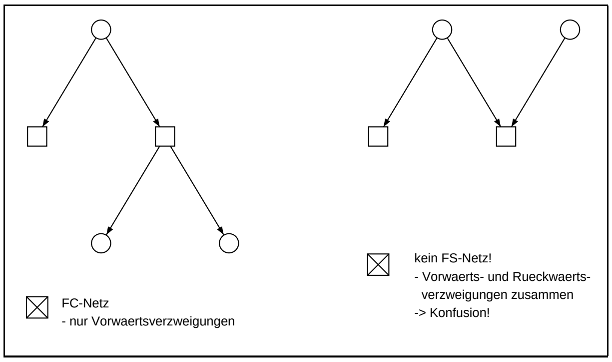


Abbildung 12: FC-Netze

Definition 5.4 (FC-Netz) *N ist ein FC-Netz falls:*
$\forall s \in S, \quad \forall t \in T, (s,t) \in F \Rightarrow s^\bullet = \{t\}$ *oder* ${}^\bullet t = \{s\}$.

5.2.1 Lebendigkeit in FC-Netzen

Es gibt zwei potentiell gefährliche Situationen:

1. Teile eines Netzes, die nie markiert werden: evtl. Deadlock.
2. Teile eines Netzes, die nie alle Marken verlieren: evtl. Trap. (entspricht einer Endlosschleife)

Deadlocks / Traps Eine Menge $D \subset S$ kann, wenn einmal unmarkiert, nie mehr Marken erhalten, falls keine Transition mehr schalten kann, die im Nachbereich eine Stelle aus D hat.
${}^\bullet D \subset D^\bullet$, dann ist D ein Deadlock.

UNIVERSITÄT BERN	KURS	TYP	BLATT	AUSGABE
INFORMATIK	PN	V	32	1994

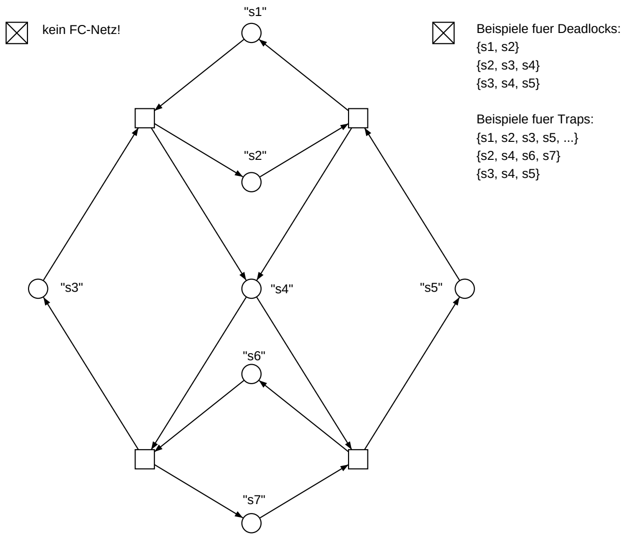
Eine Stellenmenge Q, bei welcher jede Transition, die eine Marke aus Q entnimmt, auch wieder eine zurückgibt, kann nie alle Marken verlieren.
$^\bullet Q \subset {}^\bullet Q$, dann ist Q ein Trap.

Definition 5.5 $N = (S, T, F), \quad D, Q \subset S, \quad D, Q \neq \emptyset :$

i) D ist ein Deadlock, falls $^\bullet D \subset D^\bullet$.

ii) Q ist ein Trap, falls $Q^\bullet \subset {}^\bullet Q$.

Beispiel :



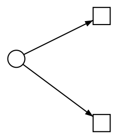
Satz 5.6 *Ein FC-Netz* $S = (S, T, F, M_0)$ *ist lebendig, falls jeder Deadlock D von S einen Trap Q enthält mit # Marken in Q bei* $M_0 \geq 0$.

1. Beispiel (vgl. Kapitel 6 und 6): System mit Trap und Deadlock.
Frage: Gibt es eine Anfangsmarkierung, so dass der obige Satz anwendbar ist?
2. Beispiel:
Deadlock: $(s1, s2, s3, s4)$,
Trap: $(s2, s4)$,
Anfangsmarkierung: {s4}.

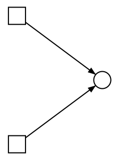
6 Strukturelle Analyse

6.1 Konflikte

Verzweigung: Markenmangel
Wettbewerb: Markenüberangebot
Fragestellung:"Welche Strukturen implizieren Konflikte ?



Preconflict

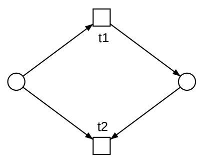


Postconflict

Definition 6.1 *Σ ein Netz, $t_1, t_2 \in T$*

1. *t_1 und t_2 sind in einem <u>strukturellen</u> Preconflikt falls:*
   ${}^\bullet t_1 \cap {}^\bullet t_2 \neq \emptyset$ *und* $t_1^\bullet \cap {}^\bullet t_2 = t_2^\bullet \cap {}^\bullet t_1 = \emptyset$
2. *t_1 und t_2 sind in einem <u>strukturellen</u> Postconflikt falls:*
   $t_1^\bullet \cap t_2^\bullet \neq \emptyset$ *und* $t_1^\bullet \cap {}^\bullet t_2 = t_2^\bullet \cap {}^\bullet t_1 = \emptyset$

Beispiel:



t_1, t_2 nicht im strukt. Konflikt

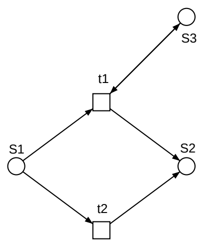
6.2 Invarianten

Darstellung von S/T Netzen als Inzidenz-Matrix

Definition 6.2 *Sei $\Sigma = (S, T, F, K, W, M_0)$ ein S/T-Netz, $|S| = n, |T| = m$*
Inzidenzmatrix von Σ: $n \times m$-Matrix, $\underline{\Sigma} = (\tau_{st})$

$$\tau_{st} = W(t, s) - W(s, t)$$

Beispiel:



$$\begin{pmatrix} -1 & -1 \\ 1 & 1 \\ 0 & 0 \end{pmatrix}$$

Definition 6.3 (S-Invariante) *Σ ist ein S/T-Netz. Eine S-Invariante ist ein Stellenvektor i, Funktion $i : S \to \mathcal{Z}$ mit*

$$\begin{pmatrix} -1 & 1 & 0 \\ -1 & 1 & 0 \end{pmatrix} \begin{pmatrix} i_1 \\ i_2 \\ i_3 \end{pmatrix} = 0 \qquad \underline{\Sigma}^{T} i = 0$$

$$\Rightarrow \qquad i = x \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} + y \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \qquad z.B.also \qquad x \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

Satz 6.4 *Σ sei ein S/T-Netz, i eine S-Invariante, M_0 eine Initialmarkierung.*
Dann gilt für jede Folgemarkierung $M \in \langle M_0 \rangle$:

$$\sum_{s \in S} i(s)M(s) = \sum_{s \in S} i(s)M_0(s)$$

$\Rightarrow$ *gewichtete Anzahl Marken bleibt konstant !*

Satz 6.5 *Falls S-Invariante keine Marken enthält, dann bleibt sie immer unmarkiert.*
$\Rightarrow$ *Deadlock*

Satz 6.6 *Falls alle Stellen einer S-Invariante markiert sind, dann bleiben sie immer markiert.*
$\Rightarrow$ *Trap*

Definition 6.7 (T-Invariante) *Σ sei ein S/T-Netz. Eine T-Invariante δ ist eine Funktion $\delta : T \to \mathcal{Z}$ mit $\underline{\Sigma} = 0$*

Satz 6.8 *$\Sigma = (S, T, F, K, W, M_0)$. M eine Markierung, δ eine T-Invariante.*
Falls bei M jede Transition aus T-Invarianten genau $\delta(t)$ mal schalten kann (Reihenfolge irrev.), dann wird die Markierung M nach dem Schalten all dieser Transitionen wieder erreicht.
$\Rightarrow$ *zyklisches Verhalten.*

Referat: Parallelität

gehalten von Dr. Rudolf Eigenmann,
CSRD (Centre for Supercomputing Research and Development),
University of Illinois

A Leistungssteigerung von parallelen Systemen

Die hauptsächliche Forschungrichtung der CSRD ist die Entwicklung eines Benchmarks für eine representative Workload.
Durch die Verwendung von Benchmarks, die keine tatsächlichen Anwendungen beinhalten, entsteht ein verzerrtes Bild der tatsächlich möglichen Leistungssteigerung. Diese Diskrepanz - die erzielten Leistungen liegen stark hinter den Erwartungen zurück - will man durch geeignete Benchmarks verkleinern.

B Kurze Geschichte der CSRD

1984: Gründung und Beginn der Entwicklung des Multiprozessor-Systems Cedar.
1986: Erstes Paper der Forschungsgruppe
1992: Abschluss des Projekts Cedar

C Beschreibung des Multiprozessor-Systems Cedar

C.1 Hardware

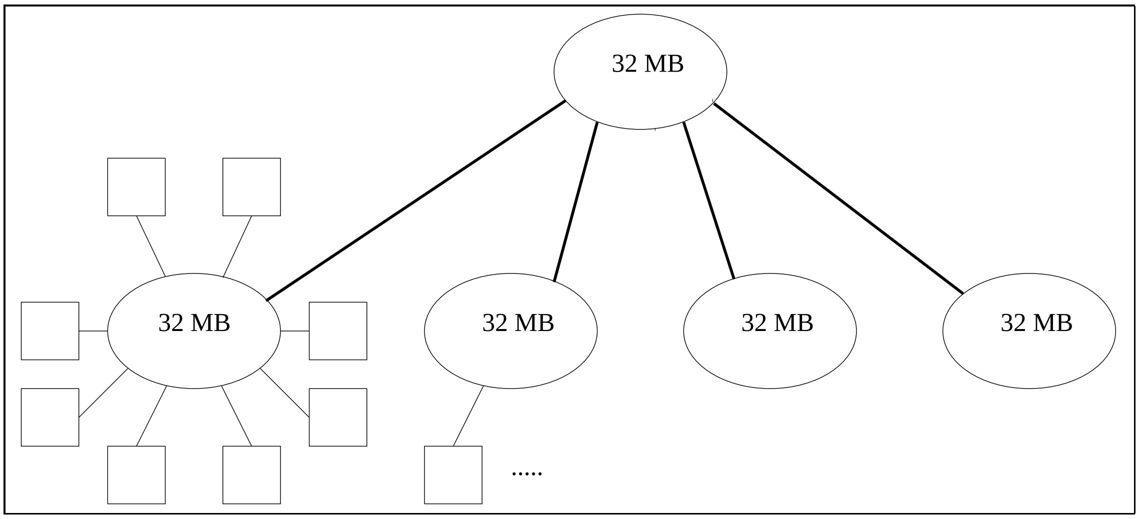


Abbildung 13: Cedar

Das System besteht aus 4 Clustern zu je 8 Prozessoren. Jedem Cluster steht ein lokaler Speicher von 32 MB zur Verfügung. Ausserdem besitzt das System einen globalen Speicher von 32 MB.
Es handelt sich somit bei diesem System um ein Shared-Memory-System (vs. Distributed-Memory-System)

C.2 System Software

Parallele Unix-Erweiterung (Xylem)
FORTRAN-Erweiterung (Cedar Fortran) Dabei wurde die Standard-Programmiersprache Fortran um Elemente zur Parallelisierung von Iterationen und einer Verwaltung von lokalem Speicher erweitert.

C.3 Applikationen

Es werden fast ausschliesslich technisch-wissenschaftliche Applikationen, davon mehrheitlich numerische Berechnungen auf parallelen Systemen betrieben.

C.4 Cedar: Zielsetzungen und Resultate

C.4.1 Können skalierbare Computersysteme gebaut werden?

Im Prinzip ja, aber die weitere Skalierbarkeit bleibt unklar. Die Komplexität eines Systems nimmt mit der Anzahl der Prozessoren sehr stark zu, so dass Voraussagen über das Verhalten bei einer bestimmten Anzahl Prozessoren sehr schwierig werden.
Ein Indiz für die Komplexität eines solchen Systems war die zeitliche Verzögerung des Projekts Cedar.

C.4.2 Können Applikationen die parallele Architektur ausnutzen?

Sequentielle Programme müssen entwender von Hand oder automatisch parallelisiert werden und können dann von der Architektur in unterschiedlichem Masse Gebrauch machen. Im Allgemeinen kann eine Leistungssteigerung (Speedup) von $n/log(n)$ erzielt werden (bei n Prozessoren).

C.4.3 Kann ein System wie Cedar leicht programmiert werden?

Verglichen mit der Programmierung von Distributed-Memory-Systemen ist diese Programmierung leichter. Wenn man die Programmierung jedoch mit Standard-Programmiersprachen vergleicht, so ist sie einiges komplexer.
Deshalb beschäftigt sich diese Forschungsgruppe heute stark mit Mechanismen zur automatischen Parallelisierung von Standard-Programmiersprachen. Hier vermutet man ein sehr grosses Forschungs-Potential, da die Industrie grosses Interesse daran hat, ihre Bibliotheken in Standard-Programmiersprachen beibehalten zu können.

D Interessen- und Applikationsgebiete des CSRD

Die betrachteten Applikationen sind meistens technisch-wissenschaftlicher Natur.

D.1 Interessen

D.1.1 Perfect Benchmarks

Dies ist eine Sammlung von ca. 70'000 Zeilen Fortran-Code aus verschiedenen Applikationen und Themenkreisen. Da diese Sammlung auch auf Workstations lauffähig sein musste, beschränkte man sich auf relativ kleine Datensätze. Daher ist dieser Test heute nicht mehr genügend aussagekräftig.

D.1.2 Perfect Benchmarks 2

Im Moment sammelt das CSRD Programme für einen zweiten, flexibleren Test, der nun mit einer Menge von verschiedenen Datensätzen arbeitet und somit auch auf grossen Maschinen vernünftige Tests ermöglicht.

D.1.3 NCSA Codes

Die NCSA ist eine weitere Forschungsgruppe für Supercomputing, von der ebenfalls Programme für Testzwecke zusammengestellt werden. Diese Bestrebungen sollen koordiniert werden.

D.1.4 Verschiedene Applikationen

Es sind verschiedene grössere Applikationen in Bearbeitung, mit denen wiederum neue Erkenntnisse gewonnen werden sollen. Beispiel sind ein Ocean-Modelling-System und ein Projekt der Electricité de France

D.1.5 Sinn des Benchmarking

Das Benchmarking soll zeigen, dass

- Supercomputer einen Speedup bringen
- die dazu entwickelten Compiler funktionieren
- die Architektur tauglich ist unter realistischen Workloads

D.1.6 Alternative Benchmark-Bestrebungen

- Euroben: operiert auf Applikationsstufe (z.B. von Perfect Benchmarks 2)
- Parkbench: entwickelt für verteilte Systeme
- Par 93: arbeitet compiler-orientiert

E Methodik und Werkzeuge für Parallel Computing

- Bei der Parallelisierung sollten nur Informationen des Programms verwendet werden. Somit muss nicht das gesamte zu realisierende Problem dazu betrachtet werden.

  Zeitlicher Gewinn: 50% - 100% gegenüber manueller Parallelisierung
- Probleme, die viel Zeit in Anspruch nahmen
  - File-Management, d.h. die Verwaltung der Experimente mit verschiedenen Optimierungsansätzen
  - Leistungsmessung

– Array (Section) Access Analysis:

a(...) = schreiben

a(...) = lesen

werden Daten ausserhalb einer Schleife generiert oder bleiben sie lokal ?

- Einfache Werkzeuge können sehr nützlich sein

Design-Prinzipien:

- auf Unix-Level bleiben
- mittels emacs-Erweiterungen arbeiten
- gängige SQL-Datenbank für die Leistungsdaten verwenden

F Automatische Parallelisierung

F.1 Motivation

Standard-Programmiersprachen sind wichtig:

- Trainingskosten können tief gehalten werden
- portable Programme können leichter generiert werden

F.2 Aktueller Stand

Die zur Zeit erhältlichen Compiler sind wenig effizient. Hier liegt ein grosses Forschungspotential.
Stossrichtungen der zukünftigen Forschung zur Realisierung eines grossen Forschungsschritts:

- verbesserte Möglichkeiten, Parallelität erkennen zu können
- die Erkenntnisse werden in einer neuen Compiler-Generation (POLARIS) implementiert. Wichtig sind dabei gute Transformationstechniken.
- ein Labor-Prototyp von POLARIS existiert bereits. Dabei steht vor allem die Beweisbarkeit der entwickelten Parallelisierungsprinzipien im Vordergrund.

F.3 Wichtige Compilations-Techniken

1. Schleifen-Arrays privatisieren

```
Real work(m)
Do i=1,n          ──>
...
```

```
CDoAll i=1,n
  Real work(m)
    ...
```

2. Parallel Reductions (Akkumulationen)

```
Do i=1,n                                 CDoAll i=1,n
  a(expr1)= a(expr1) + ...   ------->      a(expr1)= a(expr1) + ...
  ...                                     ________________________
                                            atomare Einheit
```

oder mit Hilfe von partiellen Summen (für jeden Prozessor):

```
Do i=1,n
  s = s + a(i)   ------->     s = s + SUM (a(1:n))
EndDo
```

Abbildungsverzeichnis

Tabellenverzeichnis

UNIVERSITÄT BERN	KURS	TYP	BLATT	AUSGABE
INFORMATIK	PN	ÜA	II	1994

Übungsserie 1

1. Konstruieren Sie ein Netz N, so dass N verbunden (connected), aber nicht streng verbunden ist, und dass gilt: ${}^{\bullet}N = N = N^{\bullet}$.

2. Konstruieren Sie ein kleinstes Netz N, das streng verbunden ist und mindestens ein Paar von detached Transitionen enthält.

3. N sei ein Netz; $x, y \in N$. Zeigen Sie: $x \in {}^{\bullet}y$, falls $y \in x^{\bullet}$.

4. N sei ein Netz, für jedes $x \in N$ sei ein $\bar{x}$ definiert durch ${}^{\bullet}\bar{x} = x^{\bullet}$ und $\bar{x}^{\bullet} = {}^{\bullet}x$. Zeigen Sie $\bar{\bar{x}} = x$.

5. P enthalte zwei Elemente. Konstruieren Sie das grösste T_N-einfache Netz N mit $P_N = P$.

6. Sei N ein P_N-einfaches Netz und $|T_N| = n$. Es ist zu zeigen: $2^{2n} \geq |P_N|$.

7. Sei P eine Menge mit n Elementen. Wieviele verschiedene T_N-einfache Netze mit $P_N = P$ existieren ?

8. Entwerfen Sie ein PASCAL-ähnliches Programm, welches das Produzenten-Puffer-Kosumenten Problem beschreibt. Nehmen Sie an, es existiere ein Parallel-Operator $\|$, i.e $P = (P_1 \| P_2)$.

9. Entwerfen Sie mit SystemSpecs eine Realisierung des Produzenten - Puffer - Kosumenten Problems mit mehreren Produzenten und Kosumenten, sowie einem Puffer mit verschiedener Kapazität (i.e 1 oder mehr).

UNIVERSITÄT BERN	KURS	TYP	BLATT	AUSGABE
INFORMATIK	PN	ÜA	III	1994

Übungsserie 2

1. Sei $\bar{\Sigma} = (P_\Sigma, T_\Sigma, F_\Sigma^{-1}, c_\Sigma)$.
   Zeigen Sie: $c \to^t d$ ist ein Schritt in Σ, falls $d \to^t c$ ein Schritt in $\bar{\Sigma}$ ist.

2. Zeigen Sie: Falls $c \to^t d$ ein Schritt in Σ und $e \subseteq P_\Sigma$, so dass $e \cap (c \cup d) = \emptyset$, dann ist $c \cup e \to^t d \cup e$ auch ein Schritt in Σ.

3. Beweisen Sie Lemma 3.11.

4. Falls N keine Schlaufe hat, und $c \cup e \to^t d \cup e$ ein Schritt in Σ ist, dann ist $c \to^t d$ auch ein Schritt in Σ.

5. Ein es-Netz habe n Plätze und m Transitionen. Wieviele erreichbare Zustände und wieviele Schritte hat das Netz höchstens?

6a. Die Menge der auftretenden Sequenzen ($k_0 \to$ ...) eines es-Netzes ist eine reguläre Sprache. Beweis.

6b. Existiert eine reguläre Sprache, für die kein es-Netz die Menge der auftretenden Sequenzen von Σ ist ?

7. Konstruieren Sie die Zustandsgraphen von Σ_1 und Σ_2.

UNIVERSITÄT BERN	KURS	TYP	BLATT	AUSGABE
INFORMATIK	PN	ÜL	I	1994

Musterlösungen Serie 1

Aufgabe 1

Ein Beispiel für ein Netz mit den geforderten Eigenschaften ist in Abbildung 14 dargestellt. Das Netz ist *zusammenhängend*, denn

$$x(F \cup F^{-1})y \qquad\qquad (\forall x, y \in N)$$

Das Netz ist aber *nicht streng zusammenhängend*, weil

$$(T_2, T_1) \notin F^*$$

Sicher gilt ${}^\bullet N \subseteq N$. Weil für alle $x \in N$ ein $y \in N$ existiert mit $(x, y) \in F$, gilt sicher auch $N \subseteq {}^\bullet N$. Analog dazu gilt $N \subseteq N^\bullet$.

Aufgabe 2

Das Netz mit den geforderten Eigenschaften ist in Abbildung 15 dargestellt.
T_2 und T_4 sind getrennt (detached), denn

$$({}^\bullet T_2 \cup T_2^\bullet) \cap ({}^\bullet T_4 \cup T_4^\bullet) = \emptyset$$

Aufgabe 3

Zu zeigen
$y \in x^\bullet \Rightarrow x \in {}^\bullet y$
Beweis
$y \in x^\bullet \Rightarrow (x, y) \in F \Rightarrow (y, x) \in F^{-1} \Rightarrow x \in {}^\bullet y$

Aufgabe 4

Sei $N = (T, P, F)$ ein N-einfaches Netz. Definiere $\bar{N} := (T, P, F^{-1})$. Sei $^- : N \mapsto \bar{N}$, so dass ${}^\bullet x = \bar{x}^\bullet$ und $x^\bullet = {}^\bullet \bar{x}$.
Zu zeigen

$\bar{\bar{x}} = x$

Beweis

$$\begin{aligned} {}^\bullet \bar{\bar{x}} &= \bar{x}^\bullet = {}^\bullet x \\ \bar{\bar{x}}^\bullet &= {}^\bullet \bar{x} = x^\bullet \end{aligned}$$

Aus ${}^\bullet \bar{\bar{x}} = {}^\bullet x$ und $\bar{\bar{x}}^\bullet = x^\bullet$ folgt mit der N-Einfachheit $x = \bar{\bar{x}}$.

UNIVERSITÄT BERN	KURS	TYP	BLATT	AUSGABE
INFORMATIK	PN	ÜL	II	1994

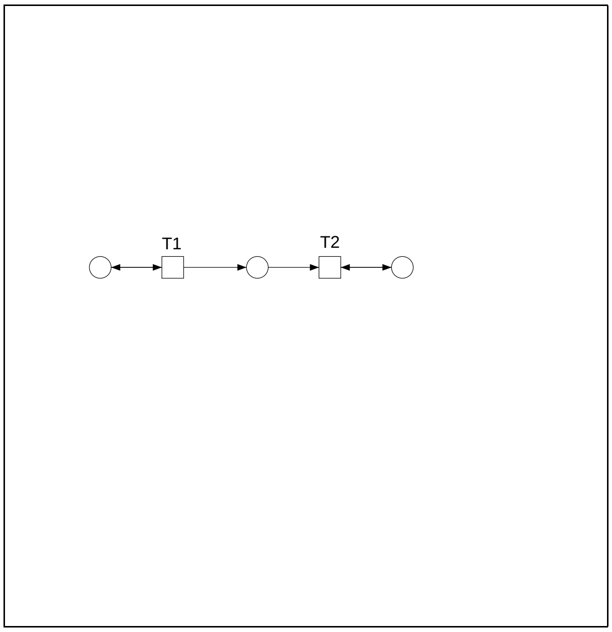


Abbildung 14: Netz zu Aufgabe 1

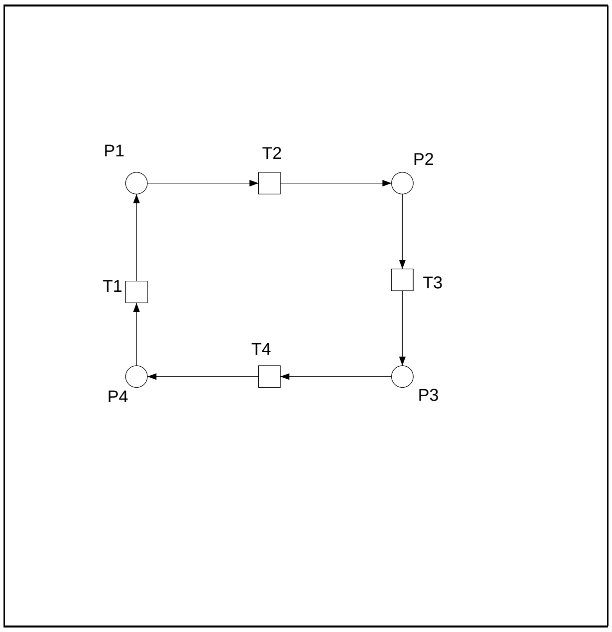


Abbildung 15: Netz zu Aufgabe 2

Aufgabe 5

Das gefragte Netz hat $2^{2n} = 16$ T-Elemente (vrgl. Aufgabe 6). Das Netz ist bis auf Isomorphie eindeutig definiert durch eine bijektive Abbildung zwischen T_N und $\mathcal{P}(P_N) \times \mathcal{P}(P_N)$.
In Abbildung 16 ist das Netz dargestellt.

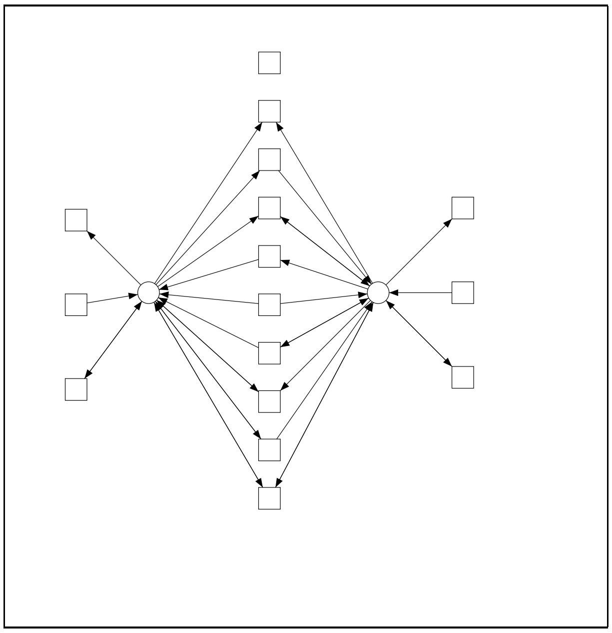


Abbildung 16: Netz zu Aufgabe 5

Aufgabe 6

In der Aufgabenstellung hat sich ein Tippfehler eingeschlichen. Die richtige Aufgabenstellung sollte heissen:
Sei N ein P_N-einfaches Netz und $T_N = n$. Zeige $|P_N| \leq 2^{2n}$.
Ein Netz ist P_N einfach, wenn für alle $x, y \in P_N$ gilt:

$$x \neq z \Rightarrow {}^\bullet x \neq {}^\bullet y \quad \text{oder} \quad x^\bullet \neq y^\bullet$$

Jedes $p \in P_N$ muss eine eindeutige Kombination von Vorgänger- und Nachfolgermenge haben. Weil ${}^\bullet x \subseteq T_N$ und $x^\bullet \subseteq T_N$, gilt:

$$m := |P_N^{\max}| = |\mathcal{P}(T_N)^2| = (2^n)^2 = 2^{2n}$$

Ein P_N-einfaches Netz mit $|T_N| = n$ und $|P_N| = m$ ist insofern maximal, als keine Kante und kein Platz hinzugefügt werden kann, ohne das die P_N-Einfachheit verlorengeht.

Aufgabe 7

Die Lösung der Aufgabe hängt davon ab, welchen Äquivalenzbegriff für T-einfache Netze man zugrunde legt. Interessant sind vor allem die *echt strukturell verschiedenen* Netze,

das heisst jene, die man durch Drehen, Strecken, Spiegeln etc. nicht zur Deckung bringen kann.
Im folgenden sind drei Äquivalenzbegriffe mit aufsteigender Strenge angegeben.
Sei $P_n := \{1, \ldots, n\}$ und $T_m := \{1, \ldots, m\}$. Sie $E(n)$ die Menge der T-einfachen Netze mit $|P| = P_n$. Sei weiter $E(n,m)$ die Menge der T-einfachen Netze mit $|P| = P_n$ und $|T| = T_m$. Nach Aufgabe 6 gilt für ein gegebenes $n : m \leq 2^{2n}$. Deshalb gilt

$$E(n) = \cup_{m=0}^{2^{2n}} E(n,m)$$

Def 1 (Äquivalenz 1) $N_1, N_2 \in E(n,m)$ *sind äquivalent, falls* $F_{N_1} = F_{N_2}$.

Def 2 (Äquivalenz 2) $N_1, N_2 \in E(n,m)$ *sind äquivalent, falls eine bijektive Abbildung* $f : T_1 \mapsto T_2$ *existiert, so dass*

- $^\bullet t = {}^\bullet f(t) \qquad (\forall t \in T_1)$
- $t^\bullet = f(t)^\bullet \qquad (\forall t \in T_1)$

Def 3 (Äquivalenz 3) $N_1, N_2 \in E(n,m)$ *sind äquivalent, falls eine bijektive Abbildung* $f : N_1 \mapsto N_2$ *existiert, so dass*

- $f(T_1) = T_2$
- $f(P_1) = P_2$
- $f({}^\bullet t) = {}^\bullet f(t) \qquad (\forall t \in T_1)$
- $f(t^\bullet) = f(t)^\bullet \qquad (\forall t \in T_1)$

$|E(n)|$ unter Äquivalenz 1

Ein T-einfaches Netz N wird charakterisiert durch eine injektive Abbildung f_N

$$f_N : T \to \mathcal{P}(P) \times \mathcal{P}(P)$$

Wie viele dieser injektiven Abbildungen gibt es bei festem n und festem m ? Wähle m Argumente aus und permutiere diese über alle Urbilder. Damit erhält man:

$$m!\binom{|\mathcal{P}(P)^2|}{m} = m!\binom{2^{2n}}{m}$$

Daraus folgt

$$|E(n)| = \sum_{m=0}^{2^{2n}} |E(n,m)| = \sum_{m=0}^{2^{2n}} m!\binom{2^{2n}}{m}$$

$|E(n,m)|$ unter Äquivalenz 2

Es gelten dieselben Überlegungen wie für Äquivalenz 1. Alle injektiven Abbildunge, die sich durch eine Permutation über T ineinander überführen lassen, werden zu einer Äquivalenzklasse zusammengefasst. Daraus folgt:
Daraus folgt

$$|E(n)| = \sum_{m=0}^{2^{2n}} \binom{2^{2n}}{m}$$

$|E(n,m)|$ unter Äquivalenz 3

Die genaue Anzahl der verschiedenen Netze unter dieser Äquivalenz ist uns nicht bekannt. Sicher wird sie jedoch durch die Anzahl unter Äquivalenz 1 nach oben abgeschätzt. Lösungen oder Hinweise dazu, nehmen wir gerne entgegen.

Musterlösungen Serie 2

Aufgabe 1

Mit $\bar{t}$ sei im folgenden eine Transition in $\bar{\Sigma}$ bezeichnet. Es gilt dann ${}^\bullet\bar{t} = t^\bullet$ und $\bar{t}^\bullet = {}^\bullet t$. Es gelte $d \xrightarrow{\bar{t}} c$ in $\bar{\Sigma}$. Das heisst

$$\begin{aligned} d &\subseteq {}^\bullet\bar{t} && (1)\\ (\bar{t}^\bullet \setminus {}^\bullet\bar{t}) \cap d &= \emptyset && (2)\\ c &= (d \setminus {}^\bullet\bar{t}) \cup \bar{t}^\bullet && (3) \end{aligned}$$

Das heisst in Σ

$$\begin{aligned} d &\subseteq t^\bullet && (4)\\ ({}^\bullet t \setminus t^\bullet) \cap d &= \emptyset && (5)\\ c &= (d \setminus t^\bullet) \cup {}^\bullet t && (6) \end{aligned}$$

Zu zeigen sind die Bedingungen für $c \xrightarrow{t} d$, das heisst $c \subseteq t^\bullet$, $(t^\bullet \setminus {}^\bullet t) \cap c = \emptyset$ und $d = (c \setminus {}^\bullet t) \cup t^\bullet$.

1. Aus Gleichung 4 und 6 folgt $c = {}^\bullet t$, also $c \subseteq {}^\bullet t$.

2. Aus Gleichung 6 folgt

$$\begin{aligned} c \cap (t^\bullet \setminus {}^\bullet t) &= ((d \setminus t^\bullet) \cup {}^\bullet t) \cap (t^\bullet \setminus {}^\bullet t)\\ &= ((d \setminus t^\bullet) \cap (t^\bullet \setminus {}^\bullet t)) \cup ({}^\bullet t \cap (t^\bullet \setminus {}^\bullet t))\\ &= (d \setminus t^\bullet) \cap (t^\bullet \setminus {}^\bullet t)\\ &= \emptyset \end{aligned}$$

3. Aus Gleichung 6 folgt

$$\begin{aligned} c &= (d \setminus t^\bullet) \cup {}^\bullet t\\ (c \setminus {}^\bullet t) &= (d \setminus t^\bullet) \setminus {}^\bullet t\\ (c \setminus {}^\bullet t) \cup t^\bullet &= (d \setminus t^\bullet) \setminus {}^\bullet t \cup t^\bullet\\ &= t^\bullet \end{aligned}$$

   Wenn man noch zeigen könnte, dass $t^\bullet \subseteq d$, wäre alles bewiesen.

Aufgabe 2

Aus $e \cap (c \cup d) = \emptyset$ folgt $e \cap c = \emptyset$ und $e \cap {}^\bullet t = \emptyset$.
Zu zeigen sind : ${}^\bullet t \subseteq (c \cup e)$, $(t^\bullet \setminus {}^\bullet t) \cap (c \cup e) = \emptyset$ und $(d \cup e) = (c \cup e) \setminus {}^\bullet t \cup t^\bullet$.

1.

$$ {}^\bullet t \subseteq c \Rightarrow {}^\bullet t \subseteq c \cup e$$

2.

$$\begin{aligned}(t^\bullet \setminus {}^\bullet t) \cap (c \cup e) &= (t^\bullet \setminus {}^\bullet t) \cap c \cup (t^\bullet \setminus {}^\bullet t) \cap e \\ &= \emptyset \cup \emptyset = \emptyset\end{aligned}$$

3.

$$\begin{aligned}(c \cup e) \setminus {}^\bullet t \cup t^\bullet &= (c \setminus {}^\bullet t) \cup (e \setminus {}^\bullet t) \cup t^\bullet \\ &= d \cup (e \setminus {}^\bullet t) \\ &= d \cup e\end{aligned}$$

Aufgabe 3

Zu beweisen sind die drei Aussagen von Lemma 3 (besprochen in der Vorlesung).

1.

$$\begin{aligned}d &= (c \setminus {}^\bullet t) \cup t^\bullet \\ d \setminus t^\bullet &= (c \setminus {}^\bullet t) \setminus t^\bullet \\ &= c \setminus {}^\bullet t \qquad\qquad \text{weil } ({}^\bullet t \setminus t^\bullet) \cap c = \emptyset\end{aligned}$$

2.

$$\begin{aligned}c \setminus {}^\bullet t &= d \setminus t^\bullet \\ c \setminus {}^\bullet t \cup {}^\bullet t &= (d \setminus t^\bullet) \cup {}^\bullet t \\ c \cup {}^\bullet t \setminus c &= (d \setminus t^\bullet) \cup {}^\bullet t \\ c \cup \emptyset &= (d \setminus t^\bullet) \cup {}^\bullet t \\ c &= (d \setminus t^\bullet) \cup {}^\bullet t\end{aligned}$$

3. Die dritte Aussage von Lemma 1 ist unter den gegebenen Voraussetzungen falsch. Betrachte das Beispiel in Abbildung 17 und setze $c := \{p_1, p_2\}$, $d := \{p_4, p_5\}$. Dann gilt :

$$c \setminus d = \{p_1, p_2\} = {}^\bullet t \setminus t^\bullet \tag{7}$$

$$d \setminus c = \{p_4, p_5\} = t^\bullet \setminus {}^\bullet t \tag{8}$$

Aber es gilt $c \overset{t}{\not\to} d$, denn ${}^\bullet t \not\subseteq c$!

Deshalb muss die Behauptung wie folgt lauten: Seien $c, d \subseteq P_\Sigma$, $t \in T_\Sigma$, ${}^\bullet t \subseteq c$, $c \setminus d = {}^\bullet t \setminus t^\bullet$, $d \setminus c = t^\bullet \setminus {}^\bullet t$.

Dann gilt ${}^\bullet t \subseteq c$ nach Voraussetzung. Zu zeigen sind dann nur noch $(t^\bullet \setminus {}^\bullet t) \cap c = \emptyset$ und $d = (c \setminus {}^\bullet t) \cup t^\bullet$.

(a)

$$(t^\bullet \setminus {}^\bullet t) \cap c = (d \setminus c) \cap c = \emptyset$$

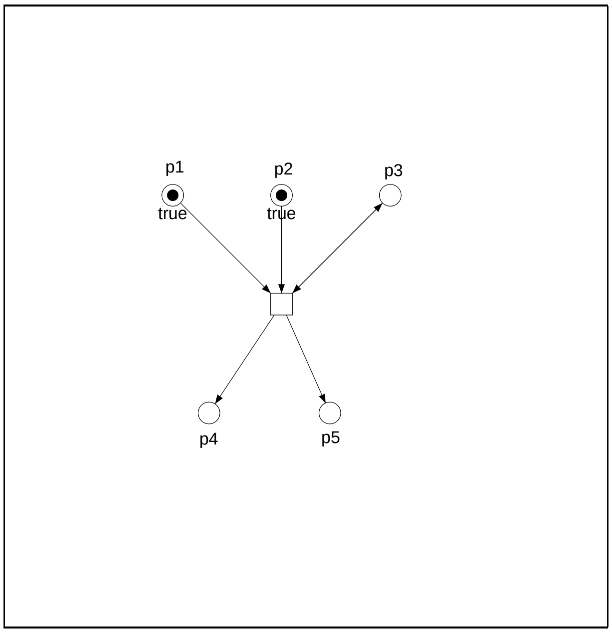


Abbildung 17: Gegenbeispiel zu Lemma 3(3)

(b)

$$
\begin{aligned}
d &= (d \setminus c) \cup (d \cap c) \\
&= (t^\bullet \setminus {}^\bullet t) \cup (d \cap c) \\
&= (t^\bullet \setminus {}^\bullet t) \cup c \setminus (c \setminus d) \\
&= (t^\bullet \setminus {}^\bullet t) \cup c \setminus ({}^\bullet t \setminus t^\bullet) \\
&= (t^\bullet \setminus {}^\bullet t) \cup (c \setminus {}^\bullet t) \cup (c \cap t^\bullet) \\
&= (c \setminus {}^\bullet t) \cup (t^\bullet \setminus {}^\bullet t) \cup (c \cap t^\bullet) \\
&= (c \setminus {}^\bullet t) \cup t^\bullet
\end{aligned}
$$

Alles in allem also $c \xrightarrow{t} d$.

Aufgabe 4

Sei Σ ein es-Netz ohne Schleifen. Das heisst

$$ {}^\bullet t \cap t^\bullet = \emptyset (\forall t \in T_\Sigma) \tag{9} $$

Es gelte nun $(c \cup e) \xrightarrow{t} (d \cup e)$. Man nehme an, dass $c \cap e = \emptyset$ und dass $d \cap e = \emptyset$. Es gilt also

$$ {}^\bullet t \subseteq (c \cup e) \tag{10} $$
$$ (t^\bullet \setminus {}^\bullet t) \cap (c \cup e) = \emptyset \tag{11} $$
$$ (d \cup e) = (c \cup e) \setminus {}^\bullet t \cup t^\bullet \tag{12} $$

Aus Gleichung 9 und 10 folgt

$$ t^\bullet \cap c = \emptyset \tag{13} $$
$$ t^\bullet \cap e = \emptyset \tag{14} $$

Es gilt weiter

$$^{\bullet}t \cap e = \emptyset \tag{15}$$

denn Gleichung 12 lässt sich umschreiben zu

$$(d \cup e) = (c \setminus {}^{\bullet}t) \cup (e \setminus {}^{\bullet}t) \cup t^{\bullet}$$

Daraus folgt sofort $e \setminus {}^{\bullet}t = e$, das heisst ${}^{\bullet}t \cap e = \emptyset$. Zu zeigen sind nun die Bedinungen für $c \xrightarrow{t} d$.

1. ${}^{\bullet}t \subseteq c$ folt aus Gleichung 10 und 15
2. $(t^{\bullet} \setminus {}^{\bullet}t) \cap c = \emptyset$ folgt direkt aus Gleichung 11.
3. $$\begin{aligned} d &= (c \setminus {}^{\bullet}t) \setminus e \cup (e \setminus {}^{\bullet}t) \setminus e \cup (t^{\bullet} \setminus e) \\ &= (c \setminus {}^{\bullet}t) \setminus e\ cup (t^{\bullet} \setminus e) \\ &= (c \setminus {}^{\bullet}t) \cup t^{\bullet} \end{aligned}$$

Alles in allem also $c \xrightarrow{t} d$.

Aufgabe 5

Sei $ESN(n, m)$ die Menge aller es-Netze mit n Plätzen und m Transitionen. Sei $C(\Sigma)$ die Menge aller Zustände von Σ. Sei $c_{\max}(n, m) := \max\{|C(\Sigma)| : \Sigma \in ESN(n, m)\}$. Weil $C(\Sigma) \subseteq \mathcal{P}(P_\Sigma)$ gilt als obere Schranke

$$c_{\max}(n, m) \leq |\mathcal{P}(P_\Sigma)| = 2^n$$

Gibt es tatsächlich ein es-Netz $\Sigma \in ESN(n, m)$, so dass $|C(\Sigma)| = 2^n$?

1. Sei $n \leq m$
   Das Netz in Abbildung 18 stellt so ein Netz dar. t_i ist aktiviert für alle i, denn $t_i \subseteq c$ und $(t_i^{\bullet} \setminus {}^{\bullet}t_i \cap c = (\emptyset \setminus \{p_i\}) \cap c = \emptyset$. Weil alle t_i unabhängig feuern können, sind alle Teilmengen von P_Σ als Zustände möglich. In diesem Fall ist also $c_{\max}(n, m) = 2^n$.
2. Sei n > m
   Mit den gleichen Überlegungen wie in 1. gibt es sicher ein Netz mit 2^m Zuständen. Als Abschätzung gilt deshalb
   $$2^m \leq c_{\max}(n, m) \leq 2^n$$
   Hinweise zu genaueren Abschätzungen werden gerne entgegengenommen.

Aufgabe 6

Ein *reguläre Sprache* ist eine Sprache, die von einem endlichen Automaten akzeptiert wird. Weil der Zustandsgraph eines Petrinetzes als endlicher Automat aufgefasst werden kann, bildet die Menge der Feuerungssequenzen eine reguläre Sprache.

Aufgabe 7

Der Zustandsgraph für Σ_1 ist in Abbildung 19, jener für Σ_2 in Abbildung 20 dargestellt.

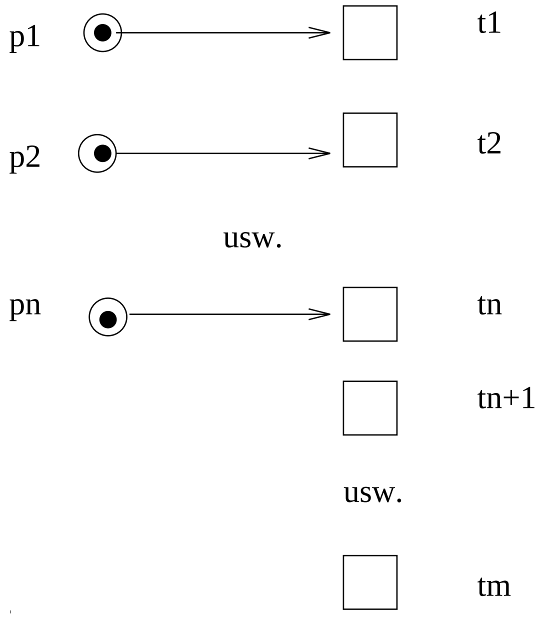


Abbildung 18: Beispielnetz für Aufgabe 5(1)

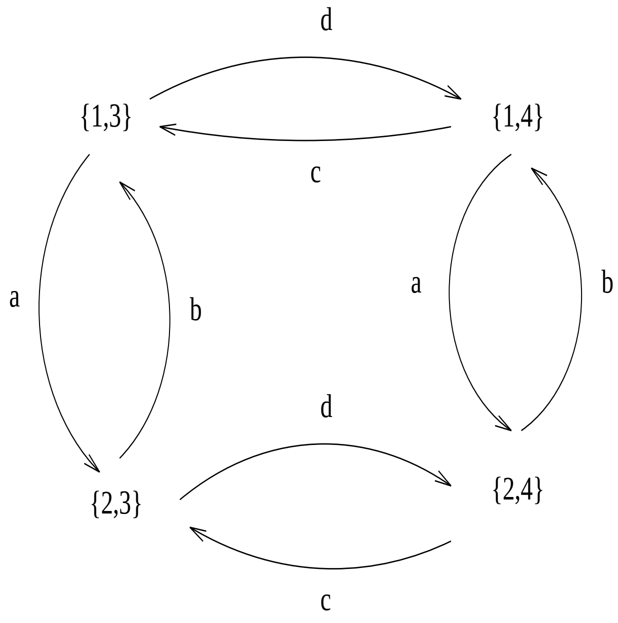


Abbildung 19: Zustandsgraph für Σ_1

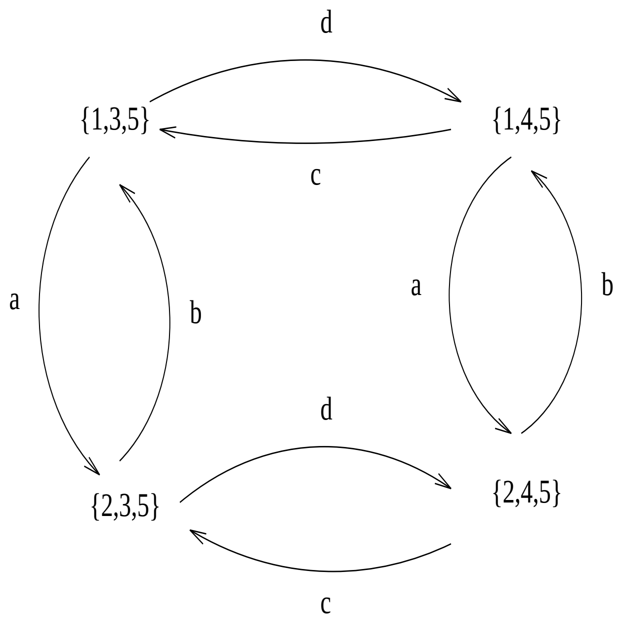


Abbildung 20: Zustandsgraph für Σ_2

Musterlösungen Serie 3

Aufgabe 1

In Abbildung 21 und in Abbildung 22 sind zwei Beispiel-Runs für Σ_0 resp. Σ_2 angegeben. Formal ist ein kausaler Run eine Abbildung von einem occurence-Netz in ein es-Netz. Diese Abbildung ist durch die Beschriftung der Plätze und Transitionen in den dargestellen Runs gegeben.

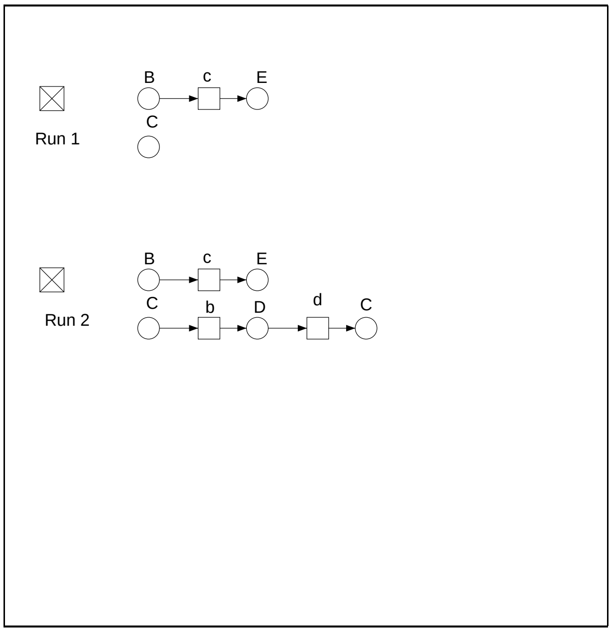


Abbildung 21: Zwei Beispiel-Runs für Σ_0

Aufgabe 2

In Abbildung 23 ist ein Buffer mit Kapazität 2 und ein möglicher kausaler Run dargestellt.

Aufgabe 3

Mögliche zulässige und unzulässige Subruns sind in Abbildung 24 dargestellt.

Aufgabe 4

Geben ein es-Netz Σ und ein zugehöriges Occurence-Netz K. Betrachte die Menge

$$^{\bullet}K := \{p \in P_K : {}^{\bullet}p = \emptyset\}$$

Dann ist $^{\bullet}K = a_{\Sigma}$.

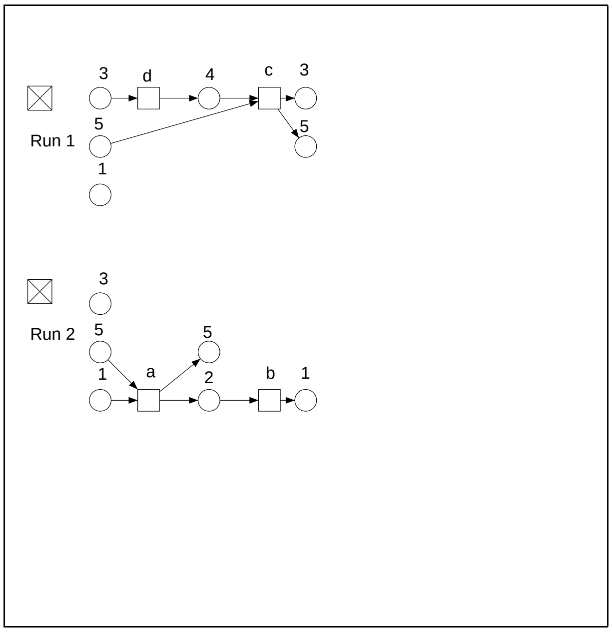


Abbildung 22: Zwei Beispiel-Runs für Σ_2

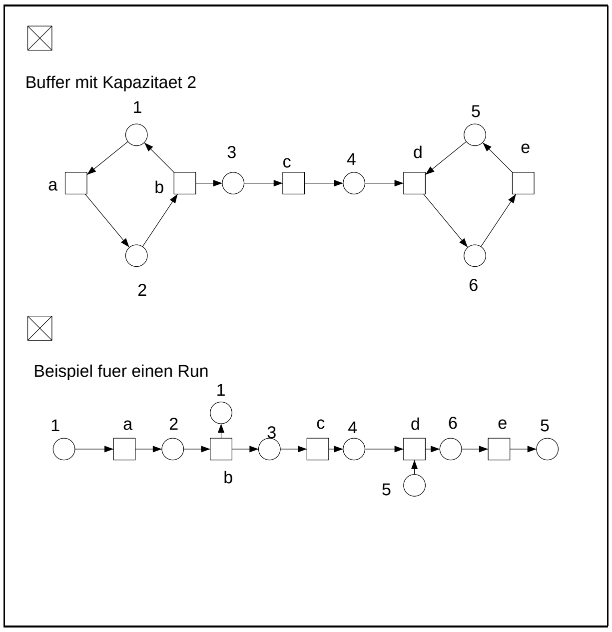


Abbildung 23: Mögliche Lösung zu Aufgabe 2

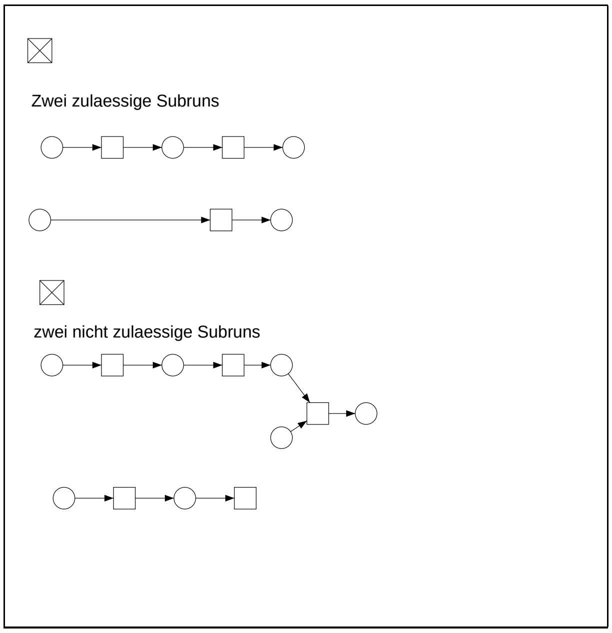


Abbildung 24: Zulässige und unzulässige Subruns