

Script zur Vorlesung
**Modelle nicht sequentieller
Rechnungen**

Lutz Priese

Dieses Script ist schon recht weit fortgeschritten aber noch nicht fertig.
Es wird im Laufe der Vorlesung korrigiert und fertig gestellt werden.
Ein Ausdruck empfiehlt sich daher erst am Ende der Vorlesung.

Inhaltsverzeichnis

1	Allgemeine Begriffe	7
1.1	Parallel-Nebenläufig	7
1.2	Abstraktes Rechenmodell	8
I	Modelle paralleler Rechnungen	11
2	Zellulare Automaten	13
2.1	Begriffe	13
2.2	Das Firing Squad Problem	19
2.3	Das French Flag Problem	21
2.4	Spracherkennung	22
2.5	Komplexität	23
2.6	Das Early Bird Problem	23
2.7	Ein Beweis des Satzes von Fredkin und Winograd	25
2.8	Artificial Life	30
2.9	Garten Eden Konfigurationen	32
2.10	Zellulare Automaten, Physik und Fiktion	37
3	Lindenmayer Systeme	41
3.1	Begriffe	42
3.2	Wachstum biologischer Organismen	43
3.3	D0L	44
3.4	Theoretische Informatik	45

4	Quantenrechner	51
4.1	Begriffe	51
4.2	C^n	54
4.3	Qubit	55
4.4	n -Qubit-System	56
4.5	Verschränkte Zustände	57
4.6	Evolution in n -Qubit-Systemen	58
4.7	Das Cloning Theorem	59
4.8	Systeme mit unbeschränkt vielen Qubits	60
4.9	Quantenschaltnetze	60
4.10	Reversible klassische Schaltnetze	65
4.11	Die Quanten-Turing-Maschine	71
	 II Modelle nebenläufiger Rechnungen	 75
5	Genetische Algorithmen	77
5.1	Motivation	77
5.2	Begriffe	79
5.3	Aufbau von genetische Algorithmen	82
5.4	Bedeutung der genetischen Operationen	85
5.5	Migration	86
5.6	Genetische Algorithmen und Evolution	87
6	Petri Netze	89
6.1	Begriffe	89
6.2	Überdeckungsgraphen und Invarianten	96
6.3	Lebendigkeit	101
6.4	Notwendige Kriterien zur Erreichbarkeit	106
6.5	Petri Netz Klassen	107
6.6	Schwache Petri Netz Berechenbarkeit	117

III Anhang: Mathematische Grundlagen 125

6.7	Komplexe Zahlen	127
6.8	Hilbert Räume	129
6.9	Lineare Abbildungen	135
6.9.1	$L(\mathcal{V}; K)$	135
6.9.2	$L(\mathcal{H}; \mathcal{H})$	138
6.10	Tensorprodukt	140

Kapitel 1

Allgemeine Begriffe

1.1 Parallel-Nebenläufig

In endlichen Automaten (e.a.) $A = (S, \Sigma, \delta, s)$ wird die Dynamik des Verhaltens üblicherweise durch eine Funktion $\delta^* : S \times \Sigma^* \rightarrow S$ definiert. δ^* sagt, wie sich ein e.a., im Zustand s gestartet, nach Verarbeitung eines Inputwortes w verhält. D.h. hier, in welchem Zustand $\delta^*(s, w)$ er sich anschließend befindet. Für Turing Maschinen hingegen wird die Dynamik durch eine direkte Nachfolgerrelation \vdash auf Konfigurationen beschrieben. Eine Konfiguration ist eine komplette Beschreibung eines Systemzustands. Man kann es sich so vorstellen, dass man bei Kenntnis der aktuellen Konfiguration ein dynamisches System unterbrechen kann. Bei einem Neustart mit dieser Konfiguration würde das System wie bei keiner Unterbrechung weiterarbeiten.

Endliche Automaten und Turing Maschinen sind Modelle von sequentiellen Rechnungen. Das bekannteste Modell eines sequentiellen Rechners ist die von Neumann Rechnerarchitektur mit einem Hauptprozessor. In dieser Vorlesung werden Modelle nicht sequentieller Rechnungen vorgestellt und untersucht. In diesen Modellen finden viele Aktivitäten “gleichzeitig” statt. “Gleichzeitig” kann dabei bedeuten, dass eine globale Uhr alle Prozesse einheitlich taktet. “Gleichzeitig” kann aber auch nur ein ideeller Begriff sein. Es existieren interessante Modelle, in denen alle Prozesse unabhängig voneinander arbeiten, ohne globale Uhr, ja völlig ohne Takt, jeweils mit ihren eignen lokalen Zeiten. Dabei ist es in diesen Modellen unerheblich, ob man sich vorstellt, dass in einer “Zeiteinheit” mehrere Prozesse gleichzeitig stattfinden, oder dass die Zeiteinheiten so kurz bemessen sind, dass stets nur ein einzelner Prozess in “Nullzeit” aktiv ist. In diesem Fall ist dann die Reihenfolge der “Nullzeitprozesse”

einem Betrachter verborgen. Unter parallelen Prozessen verstehen wir hier stets eine Vielzahl von zusammen arbeitenden Prozessen. Diese sind zumeist, aber nicht immer, über eine globale Uhr getaktet. Unter nebenläufigen Prozessen verstehen wir hingegen stets viele kooperierende Prozesse ohne einen globalen Takt. Jeder Prozess arbeitet mit unabhängigen lokalen Zeitschritten, und die Kommunikation dieser lokalen Agenten steht im Zentrum der Untersuchungen.

Besteht ein System aus mehreren Agenten, Prozessen o.ä., so verwendet man den Begriff “Zustand” meist lokal. D.h., ein einzelner Prozess ist in einem gewissen Zustand. Für den Gesamtsystemzustand, also die aktuelle Verteilung der Zustände aller Prozesse, verwenden wir einheitlich in den verschiedenen Kapiteln den Begriff der Konfiguration. In der Originalliteratur werden hierzu in den einzelnen Modellen verschiedene Begriffe gebräuchlich sind. So etwa der Begriff “Markierung” in Petri Netzen, dem wichtigsten Modell nebenläufiger Prozesse, oder der Begriff “Muster” in zellularen Automaten, ein Vertreter von hochgradig parallelen und getakteten Modellen.

1.2 Abstraktes Rechenmodell

Die Dynamik dieser nicht sequentiellen Modelle wird über eine Änderung der Konfigurationen definiert. Um modellunabhängig argumentieren zu können, führen wir den Begriff eines abstrakten Rechenmodells ein.

Definition 1.2.1 *Ein abstraktes Rechenmodell (ARM) A ist eine Tupel $A = (\mathcal{C}, \vdash)$ von*

- *einer Menge \mathcal{C} von Konfigurationen,*
- *der direkten Nachfolgerrelation $\vdash \subseteq \mathcal{C} \times \mathcal{C}$.*

\vdash^* *ist der reflexive und transitive Abschluss von \vdash .*

C' *heißt*

direkter Nachfolger von C , falls $C \vdash C'$ gilt,

Nachfolger von C , falls $C \vdash^ C'$ gilt,*

direkter Vorgänger von C , falls $C' \vdash C$ gilt.

$\mathcal{E}(C) := \{C' \mid C \vdash^* C'\}$ *ist die Erreichbarkeitsmenge von C .*

A *heißt determiniert, falls aus $C \vdash C_1$ und $C \vdash C_2$ stets $C_1 = C_2$ folgt, und indeterminiert sonst.*

A *heißt rückwärts determiniert, falls aus $C_1 \vdash C$ und $C_2 \vdash C$ stets $C_1 = C_2$ folgt, und reversibel, falls A determiniert und rückwärts determiniert ist.*

Ein Rechenbaum von A von C aus ist ein Baum (V, E) mit $V = \mathcal{E}(C)$ und $(v, v') \in E : \iff v \vdash v'$.

In einem indeterminierten ARM versteht man unter einer Rechnung entweder einen Rechenbaum selbst oder einen Ast darin. In einem determinierten ARM degeneriert ein Rechenbaum zu einem endlichen oder unendlich langen Ast.

In einem reversiblen ARM besitzt jede Konfiguration maximal eine Nachfolger- und eine Vorgänger-Konfiguration. In der Physik fordert man meist, dass genau eine Vorgänger- und Nachfolger-Konfiguration existiert. Damit wird \vdash eine bijektive Funktion auf der Menge \mathcal{C} der Konfigurationen. Rein formal sieht die Dynamik eines determinierten ARM also auch sequentiell aus, da eine Rechnung stets eine sequentielle Folge von \vdash -Übergängen ist. Zu beachten ist allerdings, dass hier Konfigurationen C, C' ein komplexes Zusammenspiel vieler paralleler Prozesse wiedergeben. Diese "sequentielle Präsentation" ist nur für uns Menschen notwendig, da wir im Bewusstsein sequentiell denken (oder zu denken glauben, obwohl im Unterbewusstsein Millionen von Prozessen parallel stattfinden).

Teil I

Modelle paralleler Rechnungen

Kapitel 2

Zellulare Automaten

2.1 Begriffe

Zellulare Automaten sind ein frühes Modell von Parallelrechnung. Ein zellulärer Automat \mathcal{Z} ist meistens ein iteratives Array von Kopien eines endlichen Automaten, A , organisiert wie ein riesiges Schachbrett. Jede Kopie von A kommuniziert mit anderen Kopien gemäß einer festgelegten Nachbarschaft. Alle Kopien arbeiten in einer globalen Taktgebung. Zum Zeitpunkt t liest jede Kopie für sich die Zustände aller Kopien in seiner Nachbarschaft und reagiert darauf mit einer Zustandsänderung. Diese Zustandsänderung führen alle Kopien gleichzeitig zum nächsten Taktschritt $t + 1$ aus. Im Zentrum des Interesses steht die globale Verteilung der Zustände aller Kopien von A . Diese Zustandsverteilung wird auch *Konfiguration*, *Muster* oder *Pattern* genannt.

Die Art, wie A auf die Zustände seiner Nachbarn reagiert, hängt ausschließlich von der lokalen Übergangsfunktion von A ab. Interessant ist aber das globale Verhalten, die Dynamik der Konfigurationen. Damit dienen zellulare Automaten als ein Modell zur Untersuchung von Fragestellungen aus dem Bereich der **Steuerung globalen Verhaltens durch einfachste lokale Regeln**. Anwendungsbereich sind daher abstrakte Fragestellungen wie

Synchronisation großer Gruppen durch lokales Verhalten der einzelnen Individuen

Selbstorganisation von Gruppen

Selbstreproduktion

Schwarmintelligenz

artificial life,

aber auch konkretere wie

Entwicklung paralleler Algorithmen

Komplexität paralleler Algorithmen

Programmierung von GPGPUs.

Bevor wir mit allgemeinen Definitionen beginnen, wollen wir uns einige Beispiele von zellularen Automaten anschauen.

Beispiel 2.1.1 Wir stellen hier ein einfaches Beispiel \mathcal{Z}_3 aus einer ganzen Schar von zellularen Automaten vor, die alle jede gegebene Konfiguration vervielfältigen. Entwickelt wurden diese Kopiermaschinen von Fredkin und verfeinert von Winograd [37].

Der endliche Automat A , der \mathcal{Z}_3 zugrunde liegt, besitzt drei Zustände $0, 1, 2$. Die Nachbarschaft jeder Kopie K von A besteht aus K selbst und den unmittelbaren vier Nachbarn von K rechts, links, oben und unten. Damit reagiert A auf fünf Zustände. Die Reaktion besteht darin, dass A die fünf Zustände zum Zeitpunkt t modulo 3 aufaddiert und das Ergebnis als seinen Zustand für den Zeitpunkt $t + 1$ setzt. Abbildung 2.1 zeigt 3 Schritte einer Rechnung von \mathcal{Z}_3 , beginnend mit der linken Konfiguration und endend mit der rechten. Dargestellt sind jeweils die Zustände der Kopien von A in den Zellen des großen Schachbretts, in dem die Kopien liegen. Alle Zellen ohne Zustandsangabe sind im Zustand 0. Das

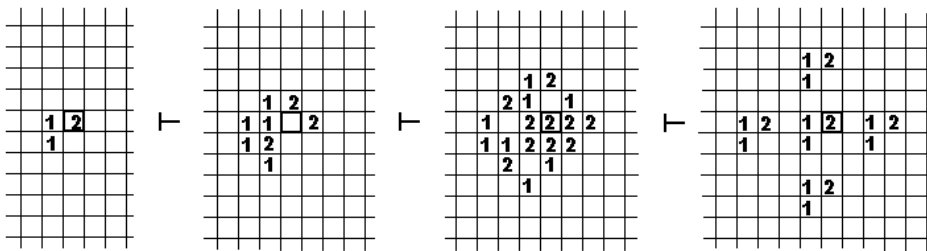


Abbildung 2.1: Eine Rechnung in \mathcal{Z}_3 .

Pattern ganz links ist damit nach drei Schritten verfünffacht wurden. \mathcal{Z}_3

vervielfacht jedes Ausgangsmuster. Natürlich dauert das für größere Muster auch länger als nur drei Schritte, stets aber ein Vielfaches von drei Schritten.

Man kann Z_3 als ein “Universumänsehen, in dem Selbstreproduktion das allgemeine physikalische Gesetz ist. Natürlich ähnelt diese Art von Selbstreproduktion eher dem starren Wachstum von Kristallen als interessanten biologischen Vorgängen. Es waren aber genau solche Fragen nach logischen Gesetzen und Möglichkeiten für Selbstreproduktion, die von Neumann und Ulam zu Beginn der Sechziger Jahre des letzten Jahrhunderts, noch vor der Entdeckung der DNA-Doppelhelix durch Crick und Watson, bewogen, das Modell der zellularen Automaten einzuführen.

Beispiel 2.1.2 (Beispiel: Game of Life) Conway [6] stellte im *Scientific America* 1970 einen zellularen Automaten namens ‘Game of Life’ vor. Zugrunde liegt ein endlicher Automat mit nur zwei Zuständen, genannt ‘lebendig’ und ‘tot’, der pro Taktschritt auf die Zustände der Nachbarkopien rechts, links, oben unten und in den vier Diagonalen reagiert. Die Überfunktionsfunktion ist denkbar einfach: A geht genau dann in den Zustand ‘lebendig’ über, falls A bereits im Zustand ‘lebendig’ ist und zwei oder drei seiner Nachbarn ebenfalls, oder falls A im Zustand ‘tot’ ist und genau drei seiner Nachbarn sind ‘lebendig’. Conway interessierte sich für ein Schwarmverhalten, in dem ein Individuum stirbt aus Überbevölkerung (mehr als drei Nachbarn) oder Unterbevölkerung (weniger als zwei Nachbarn). Genau drei Nachbarn führen zu einer Geburt. Conway vermutete, dass unter diesen Regeln Bevölkerungen (das sind nichts anderes als die Verteilungen aller lebendigen Zellen) wachsen und schrumpfen können, gegebenenfalls auch aussterben, aber nicht zu unendlicher Größe wachsen können. Diese Vermutung wurde sehr schnell widerlegt, dennoch faszinierte das Game of Life Tausende von Wissenschaftlern. W. Gosper vom MIT konnte Jahre später sogar zeigen, dass das Game of Life berechenbarkeitsuniversell ist und Konfigurationen besitzt, die sowohl jede Turing Maschine simulieren als auch sich selbst reproduzieren können. Diese Eigenschaft hatte von Neumann ein Jahrzehnt zuvor als wichtigste Eigenschaft von ‘artificial life’ gefordert. Faszinierende Konfigurationen finden sich unter <http://home.fonline.de/fo0126/spiele/denk31.htm>, wenn man den “Enjoy Life” Button anklickt.

Wir geben nun eine formale Definition eines zellularen Automaten. Die Grundstruktur, das Schachbrett in beiden Beispielen, kann auch eine andere Dimension besitzen und wird der Einfachheit halber als unendlich groß betrachtet, d.h. als der Raum \mathbb{Z}^d für ein d . In jeder Zelle des \mathbb{Z}^d liegt eine Kopie eines endlichen Automaten A . Wir interessieren uns hier nur für endlich große Konfigurationen. Dazu wird ein Sonderzustand 0,

der so genannte Ruhezustand, eingeführt. Fast alle Kopien von A sollen sich in diesem Ruhezustand befinden. 'Fast alle' heißt alle, bis auf endlich viele Ausnahmen. Um die lange Umschreibungen 'Kopie von A ' zu vermeiden, meinen wir mit dem Begriff *Zelle* sowohl ein Element des Grundraums \mathbb{Z}^d als auch die Kopie von A , die sich in dieser Zelle befindet. Keine Zelle im Ruhezustand, deren Nachbarzellen sich alle ebenfalls im Ruhezustand befinden, darf den Ruhezustand verlassen. Damit kann eine endliche Konfiguration, in der sich nur endlich viele Zellen in einem Zustand ungleich dem Ruhezustand befinden, nur in eine endliche Konfiguration übergehen. Die Nachbarschaft in einem zellularen Automaten wird nun einfach eine Liste $N[0], \dots, N[k]$ von Koordinaten des \mathbb{Z}^d . Eine Zelle $z \in \mathbb{Z}^d$ ändert damit ihren Zustand gemäß den Zuständen aller Zellen mit den Koordinaten $z, z + N[0], \dots, z + N[k]$.

Definition 2.1.1 (Definition zellularer Automat) *Ein zellularer Automat (z.a.) \mathcal{Z} ist ein Tupel $\mathcal{Z} = (d, N, 0, A)$, bestehend aus*

- einer natürlichen Zahl $d \geq 0$, der Dimension,
- einer endlichen Liste N von Elemente aus \mathbb{Z}^d , der Nachbarschaft,
- einem Element 0 , dem Ruhezustand,
- einem endlichen Automaten $A = (S, \delta)$, bestehend aus
 - einer endlichen Menge S von Zuständen mit $0 \in S$,
 - einer lokalen Überföhrungsfunktion $\delta : S^{|N|+1} \rightarrow S$, mit der Eigenschaft

$$\delta(0, \dots, 0) = 0.$$

Die Liste $N(z) := [z + N[0], \dots, z + N[k]]$ ist die Nachbarschaft von $z \in \mathbb{Z}^d$.

Definition 2.1.2 *Eine Konfiguration C von \mathcal{Z} ist eine Abbildung $C : \mathbb{Z}^d \rightarrow S$ mit endlichem Support, d.h. mit der Eigenschaft*

$$\{z \in \mathbb{Z}^d \mid C(z) \neq 0\} \text{ ist endlich.}$$

Ein endlicher Ausschnitt einer Konfiguration wird auch Muster (Pattern) genannt. Ein Zelle im Ruhezustand 0 wird auch leer genannt. Eine Konfiguration eines 1-dimensionalen z.a. wird als endliches Wort über Zuständen geschrieben, wobei alle nicht angegebenen Zellen sich im Ruhezustand befinden..

Eine Konfiguration C' heißt direkter Nachfolger einer Konfiguration C , $C \vdash C'$, falls für alle $z \in \mathbb{Z}^d$ mit $k := |N| - 1$ gilt

$$C'(z) = \delta(C(z), C(z + N[0]), \dots, C(z + N[k])).$$

Damit definiert die lokale Übergangsfunktion δ von A das globale getaktete Verhalten \vdash von \mathcal{Z} . Der Begriff des Taktes oder eines Zeitpunktes t musste dazu gar nicht eingeführt werden. Diese Begriffe gehören natürlich zu unserer Anschauung eines zellularen Automaten und sind indirekt in dieser formalen Definition versteckt. Üblicherweise nimmt man bei Verwendung des Begriffs Muster an, dass es von einer größeren Anzahl leerer Zellen umgeben ist, ohne dass alle Zellen außerhalb des Musters bereits leer sein müssen. Die Nachbarschaft sei im Folgenden stets als $N = [N[0], \dots, N[k]]$ mit $k = |N| - 1$ notiert.

Wir haben einen wichtigen Punkt noch gar nicht erwähnt, der intuitiv klar ist, aber in der formalen Definition noch nicht berücksichtigt ist. Natürlich interessieren in einer Konfiguration nicht die absoluten Koordinaten. Diese brauchen wir nur, um einfach über Konfiguration sprechen und welche angeben zu können. Verschieben wir etwa in einem 2-dimensionalen z.a. eine Konfiguration überall um 2 Zellen nach rechts und 5 Zellen nach unten, so ändert sich gar nichts am Verhalten. In Wirklichkeit sind Konfigurationen also Äquivalenzklassen unter Verschiebungen. Um es nicht zu verkomplizieren, merken wir uns einfach: **Zwei d -dimensionale Konfigurationen, die durch ein Verschieben des Nullpunktes im \mathbb{Z}^d ineinander übergehen können, werden identifiziert.**

Beispiel 2.1.3 (Signalreflektion) *Wir betrachten einen 1-dimensionalen z.a. $\mathcal{Z}_{\text{Reflektion}}$ mit*

- $N = [-1, 1]$,
- $S = \{0, >, <, |\}$,
- für die lokale Übergangsfunktion gilt
 - $\delta(0, >, 0) = >$
 - $\delta(0, >, |) = <$
 - $\delta(0, 0, <) = <$
 - $\delta(|, \cdot, \cdot) = |$
 - $\delta(\cdot, \cdot, \cdot) = 0$ in allen anderen Fällen.

Ein 'Signal' $>$ läuft von links nach rechts. Trifft es auf eine 'Barriere' $|$, so wird es als nach links laufendes Signal $<$ reflektiert. Trifft das Signal $<$ auf eine Barriere, so wird es zerstört. Schreiben wir in einem 1-dimensionalen z.a. ein Muster kanonisch als ein Wort über Zuständen, so gilt hier:

$$\begin{aligned} 0|0>00000|0 \vdash 0|00>0000|0 \vdash^* 0|00000>0|0 \vdash 0|000000<|0 \\ \vdash^* 0|<000000|0 \vdash 0|0000000|0 \end{aligned}$$

Ein Signal kann auch ein Muster aus mehr als einer Zelle sein. Findet man ein Muster nach t Takten an einer anderen Stelle in einer Entfernung d und Richtung r wieder, so sagt man auch, dass sich das Muster mit einer *Geschwindigkeit* d/t in Richtung r bewegt. Unter der *absoluten Lichtgeschwindigkeit* in einer Richtung versteht man die in Abhängigkeit der Nachbarschaft höchst mögliche Geschwindigkeit, mit der sich ein irgendein Muster in irgendeinem z.a. mit dieser Nachbarschaft über leere Zellen in leeren Umgebungen bewegen kann. Für die Moore Nachbarschaft ist die Lichtgeschwindigkeit in Richtung rechts, links, oben, unten oder in irgendeine 45° Diagonale gerade 1. In der von Neumann Nachbarschaft ist sie 1 in die Richtungen rechts, links, oben und unten, aber 2 in die diagonalen Richtungen. Häufig findet man auch die Lichtgeschwindigkeit eines konkreten z.a. in eine konkrete Richtung als die höchst mögliche Geschwindigkeit, in der sich in diesem konkreten z.a. ein Muster in diese Richtung bewegen kann. Im Game of Life ist sie in den geraden und diagonalen Richtungen jeweils $1/2$. Überraschenderweise hat man in Game of Life einen Tunneleffekt entdeckt, dass sich manche lichtschnelle Glider (spezielle kleine Muster) in nicht-leeren Gebieten überlichtschnell bewegen können.

Man kann z.B. $\mathcal{Z}_{\text{Reflektion}}$ leicht so abändern, dass sich das Signal $>$ mit einer Geschwindigkeit $1/2$ nach rechts bewegt. Allerdings benötigt man dazu einen weiteren Zustand, w , und ändert δ etwa wie folgt ab:
 $\delta(0, >, 0) = \delta(0, >, |) = w$, $\delta(0, w, 0) = >$, $\delta(0, w, |) = <$.

Definition 2.1.3 *Ein Drehung $\vartheta : \mathbb{Z}^d \rightarrow \mathbb{Z}^d$ im \mathbb{Z}^d heißt zulässig, falls sie auf die Nachbarschaft N nicht verlässt, d.h., wenn eine Permutation $\pi : \{0, \dots, k\} \rightarrow \{0, \dots, k\}$ existiert mit*

$$[\vartheta(N[0]), \dots, \vartheta(N[k])] = [N[\pi(0)], \dots, N[\pi(k)]].$$

In diesem Fall heißt π die zur Drehung ϑ gehörende Permutation.

Ein z.a. \mathcal{Z} heißt isotrop, falls für jede zu einer zulässigen Drehung gehörenden Permutation π und alle $s \in S, s^\rightarrow \in S^{|N|}$ gilt

$$\delta(s, s^\rightarrow) = \delta(s, \pi(s^\rightarrow)).$$

Von besonderem Interesse sind die beiden 'direkten' Nachbarschaften.

Definition 2.1.4 *Im \mathbb{Z}^d ist*

$$N_{vN} := \{z \in \{-1, 0, 1\}^d \mid 0 < \sum_{1 \leq i \leq d} |z_i| \leq 1\},$$

beliebig angeordnet, die von Neumann Nachbarschaft, und

$$N_M := \{z \in \{-1, 0, 1\}^d \mid z \neq (0, \dots, 0)\},$$

beliebig angeordnet, die Moore Nachbarschaft.

Das Game of Life verwendet die 2-dimensionale Moore Nachbarschaft, \mathcal{Z}_3 die 2-dimensionale von Neumann Nachbarschaft. Beide sind isotrop, im Gegensatz zu $\mathcal{Z}_{\text{Reflektion}}$. Man kann die Arbeitsweise eines jeden (d-dimensionalen) z.a. durch einen z.a. mit von Neumann Nachbarschaft simulieren. Allerdings erhöht sich dabei die Zustandszahl. Beim Game of Life ginge die Schönheit und Einfachheit der lokalen Überföhrungsfunktion verloren.

Übung 2.1.1 *Geben Sie einen 2-dimensionalen z.A. mit von Neumann Nachbarschaft an, der das Game of Life simuliert. Ein Schritt im Game of Life wird dabei zu mehreren Schritten im simulierenden z.a.*

Übung 2.1.2 *Ändern Sie $\mathcal{Z}_{\text{Reflektion}}$ so ab, dass sich $>$ mit einer Geschwindigkeit $2/3$ nach rechts bewegt.*

Übung 2.1.3 *Ändern Sie $\mathcal{Z}_{\text{Reflektion}}$ so ab, dass sich vom Ausgangswort $0^n \mid$ von \mid aus zwei Signale $<, <'$ nach links lösen, wobei sich $<$ mit der Geschwindigkeit 1 und $<'$ mit der Geschwindigkeit $1/2$ nach links bewegt.*

2.2 Das Firing Squad Problem

Das Firing Squad Problem (FSP) ist eine typische Selbstorganisationsaufgabe in einer politisch unkorrekten Fragestellung: Ein General soll sicherstellen, dass ein Schwadron von Soldaten gleichzeitig feuert, ohne dass er global alle Soldaten erreichen kann. Die Kommunikation ist lokal, immer nur zwischen zwei Soldaten. Die Größe des Schwadrons ist nicht festgelegt. Es handelt sich also um ein Synchronisationsproblem ohne globalen Zugriff auf die Teilnehmer. Im Modell der z.a. sieht das FSP damit wie folgt aus.

Definition 2.2.1 *Ein Firing Squad z.a. \mathcal{Z}_{FS} ist ein 1-dimensionaler z.a. $(1, N_{\mathcal{N}}, 0, (S, \delta))$ mit der von Neumann Nachbarschaft, für den gilt*

- $g, s, f \in S,$
- $\delta(s, x, y) = s$ für $x, y \in \{s, 0\},$

- $\forall n \in \mathbb{N} : gs^{n-1} \vdash^* f^n$,
- $\forall n \in \mathbb{N} : \forall c : (gs^{n-1} \vdash^* c \implies (c = f^n \text{ oder } f \text{ kommt in } c \text{ nicht vor}))$.

g ist ein General, der einem Schwadron der Größe n vorsteht und einen Feuerbefehl gibt. Kein Soldat, der von Zellen im Zustand Soldat oder leer umgeben ist, führt irgendeine Aktion aus (es fehlt ein Befehl). Zu irgendeinem Zeitpunkt sollen alle Soldaten, einschließlich des Generals, gleichzeitig feuern (Konfiguration f^n) und niemand darf vorher feuern. Das Firing Squad Problem (FSP) ist die Frage, ob solch ein Firing Squad z.a. existiert. Dies klärt der folgende Satz.

Satz 2.2.1 *Es gilt:*

1. *Das FSP ist lösbar.*
2. *Jeder z.a., der ein Schwadron von n Soldaten, einschließlich des Generals, synchronisiert, braucht dafür mindestens $2n - 2$ Schritte.*
3. *Es existiert ein z.a., der das FSP in optimaler Zeit löst.*

Beweis.

Zur ersten Aussage. Der General sendet zwei Signale nach rechts, $>$ mit der Geschwindigkeit 1 und $>'$ mit $1/3$. $>$ wird vom letzten Soldaten mit der Geschwindigkeit 1 als $<$ reflektiert. Damit treffen sich $>'$ und $<$ in der Mitte, egal ob bei gerader oder ungerade Zahl n , allerdings in zwei etwas unterschiedlichen Situationen. Wenn die Mitte identifiziert ist, wird der mittlere Soldat - bzw die beiden mittleren Soldaten - zum General. Der oder die beiden neuen Generäle senden nach rechts und links Befehle, um die beiden Mitten der beiden Hälften zu finden. Dies wird rekursiv fortgesetzt, bis simultan alle Zellen als neue Mitten gekennzeichnet werden. In diesem Moment findet die Synchronisation statt. Dieser Algorithmus benötigt etwa $3n$ Schritte.

Eine Visualisierung der Lösung ist zu finden unter <http://motls.blogspot.com/2009/03/firing-squad-synchronization-problem.html>

Zur zweiten Aussage. Die Ausgangssituation sei ein General plus $n - 1$ Soldaten, gs^{n-1} . Ein Signal vom ganz rechten Soldaten braucht mindestens $n-1$ Schritte, um den General zu erreichen. Da kein Soldat ohne einen Befehl reagiert, bleibt der rechte Soldat im Zustand s während der ersten i Schritte für $i < n - 1$, schickt in dieser Zeit also kein Signal an den General. Wäre die Synchronisation des gesamten Schwadrons schon nach $2n - 3$ Schritten - oder früher - erreichbar, würde der General feuern,

ohne ein Antwortsignal des ganz rechten Soldaten s_r erhalten zu haben. Damit wäre es unbemerkbar, wenn rechts von s_r sich weitere Soldaten befinden, sagen wir insgesamt $2n$ viele. Da nach $2n - 3$ Schritten davon der ganz rechte noch kein Signal vom General erhalten hat, geht er nicht in den Zustand f und das Schwadron feuert nicht synchron.

Ein Beweis zu dritten Aussage 3 findet sich etwa in Vollmar [26], eine zeitoptimale Konstruktion mit nur 8 Zuständen ist von Balzer [1]. ■

Eine Visualisierung der zeitoptimalen Lösung ist unter <http://mathworld.wolfram.com/FiringSquadProblem.html>

Das FSP besitzt auch eine Lösung, falls sich der General nicht ganz links sondern irgendwo im Schwadron befindet. Es funktioniert auch in d -dimensionalen z.a., wobei das Schwadron als d -dimensionales Quader von Soldaten organisiert ist, in deren Mitte sich irgendwo der General befindet. Zusätzlich darf das Quader auch Löcher besitzen. Man kann das Problem auch die Synchronisation in beliebigen zusammenhängenden Graphstrukturen verallgemeinern.

2.3 Das French Flag Problem

Das French Flag Problem (FFP) ist die Frage, wie sich ein beliebig langes Wort von neutralen Buchstaben, a^n , selbst so organisieren kann, dass es die Farben (rot weiß, blau) der Französischen Flagge annimmt. Dazu sei $n = 3m$ und es ist ein 1-dimensionaler z.a. mit von Neumann Nachbarschaft gesucht, der $a^{3m} \vdash^* r^m w^m b^m$ leistet. In einer anderen Interpretation stellt man sich einen Regenwurm vor, der durch einen Spaten geteilt wird. In der Natur können sich beide Hälften danach zu einem vollständigen Regenwurm reorganisieren. Wie schaffen es die Zellen nur mittels lokaler Kommunikation, ohne Wissen der Länge des ursprünglichen Regenwurms, sich wieder in Kopf-, Mitte-, und Schwanzzellen umzuformen?

Zur Lösung genügt es, die beiden Drittelstellen im Wort a^n zu finden. Dazu muss nur die ganz linke Zelle a zwei Signale $>, >'$ mit den Geschwindigkeiten 1 und $1/2$ nach rechts und die ganz rechte Zelle a zwei Signale $<, <'$ mit den Geschwindigkeiten 1 und $1/2$ nach links senden. Der Treffpunkt von $>'$ mit $<$ markiert das erste Drittel, der von $>$ mit $<'$ das zweite.

Übung 2.3.1 *Geben Sie S und δ konkret an, so dass der z.a. $\mathcal{Z}_{FF} = (1, N_{vN}, 0, (S, \delta))$ das FFP löst. Versuchen Sie danach, die Laufzeit*

und/oder Zustandszahl zu optimieren.

2.4 Spracherkennung

Formale Sprachen

Hier geht es um die Fragestellung, welche Formalen Sprachen von z.a. erkannt werden können. Das Problem wird wieder mit 1-dimensionalen z.a. mit von Neumann Nachbarschaft behandelt.

Definition 2.4.1 Ein z.a. $\mathcal{Z} = (\infty, \mathcal{N}_{vN}, \iota, (\mathcal{S}, \delta))$ erkennt eine Sprache $L \subseteq \Sigma^*$, falls Zustände $Y, N \notin \Sigma$ existieren, so dass für alle Wörter $w \in \Sigma^*$ gilt

- $w \vdash^* Y$, falls $w \in L$ gilt, und
- $w \vdash^* N$, sonst.

Da Formale Sprache in der Theoretischen Informatik eine herausragende Rolle spielen, ist die Frage nach der Erkennbarkeit von Formalen Sprachen durch z.a. verständlich. Sie wird aber der eigentlichen Intention von z.a. nicht gerecht und soll hier nicht untersucht werden. Es ist auf Grund der Lösbarkeit des FSP und FFP auch nicht verwunderlich, dass auch nicht kontextfreie Sprache von z.a. erkannt werden können.

Übung 2.4.1 Zeigen Sie, dass die Sprachen

- $L_1 = \{a^n b^n c^n \mid n \in \mathcal{N}\}$, und
- $L_2 = \{ww^R \mid w \in \Sigma^*\}$

von z.a. erkannt werden. Dazu soll δ nicht explizit angegeben werden, sondern nur das Signalverhalten des z.a. beschrieben werden. Obwohl L_2 kontextfrei, L_1 aber nicht, ist, ist die Erkennung von L_1 hier leichter.

Bildsprachen

Ein Bildsprache ist einfach eine Menge von 2-dimensionalen Konfigurationen über einem Alphabet (Zustandsraum) S . So kann man ein Binärbild als Konfiguration über 0 und 1 auffassen. Ein Sprache könnte etwa die Menge aller Binärbilder sein, in denen die Zustände 1 ein Rechteck oder andere interessante geometrische Objekte bilden. Damit stellt sich die Frage, welche Muster von z.a. erkannt werden, bzw. welche Transformationen auf Muster von z.a. geleistet werden können.

2.5 Komplexität

Generell ist die Spracherkennung von z.a. vielleicht vom Standpunkt der Theorie Formaler Sprachen eine kanonische Fragestellung, genau genommen ist sie aber recht uninteressant. Der Gag in z.a. ist ja die hochgradige Parallelarbeit. Betrachten wir nur endliche Konfigurationen, wie es in der Praxis sein muss, und sind außerhalb eines Rechtecks R einer Größe $n \times n$ alle Zellen inaktiv im Ruhezustand, so können in einem Schritt nur Randzellen einer gewissen Breite, die von der Nachbarschaft N abhängt, aktiv werden. Dies sind $O(n)$ viele. Nach t Schritten sind also gerade $O(t^2)$ viele weitere Zellen aktiviert wurden. Haben wir einen sequentiellen Algorithmus, der ein Problem einer Größe n in $f(n)$ Schritten löst, so kann man dies durch Einsatz auch eines unbeschränkt großen z.a. nur um einen Faktor $O(n^2)$ beschleunigen. Damit bleiben Probleme in den Komplexitätsklassen P und NP auch bei einem Einsatz von z.a. in den gleichen Klassen. Wir werden daher nicht weiter in eine Theorie von Spracherkennung mittels z.a. eindringen.

Interessant für praktische Anwendungen in der Bildverarbeitung können z.a. werden, die in jeder Zelle einen Lichtsensor und ein endliches Schaltwerk besitzen, das mit anderen Zellen einer interessanten Nachbarschaft - etwa alle Nachbarn in einem 11×11 -Fenster - direkt kommuniziert, und programmierbar ist. Damit könnten diverse Bildvorverarbeitungsschritte direkt auf den Bildsensor ausgelagert werden. Erst die vorverarbeiteten Daten würden dann auf einer Pipeline ausgelesen. Auch hier ist der Gewinn höchstens quadratisch in der Zeit, was theoretisch nicht aber in der Praxis überzeugend sein kann.

2.6 Das Early Bird Problem

Das Early Bird Problem (EBP) ist wie das FSP und FFP eine Synchronisationsaufgabe. Ein d -dimensionaler Raum sei zu Beginn leer. Zu jedem Zeitpunkt können spontan Zellen in den Zustand v (für Vogel) übergehen. Die Vögel der ersten Generation sollen von später entstandenen Vögeln unterschieden werden. Das Problem dabei ist, dass die Vögel zu unterschiedlichen Zeitpunkten in unterschiedlichem, beliebig großem Abstand im \mathbb{Z}^d entstehen dürfen. Es wird eine Lösung mit der von Neumann Nachbarschaft für $d = 2$ skizziert.

Induktionsbeginn: Ein eben geborener Vogel, v , setzt in jede seiner Nachbarzellen simultan einen Stein, s , und geht in Phase "aktiv".

Induktionsschritt: v sei in der aktiven Phase und jeder seiner vier

Steine s , sei genau n Zellen von v entfernt. v sendet simultan je ein Signal zu jedem Stein. erreicht das Signal einen Stein, wird der Stein um eine Zelle weiter von v entfernt und zu v reflektiert. Erreichen alle vier reflektierten Signale simultan v , dann geht v wieder in Phase 0 über. Die Strecke von v zu einem s ist dabei gesondert gekennzeichnet und wird als 'Spur' bezeichnet.

Zwei Vögel heißen *projektiv*, falls ihre x- oder y-Koordinaten im \mathbb{Z}^2 übereinstimmen. Steine projektiver Vögel treffen daher in einem 180° Winkel aufeinander, sagen wir an einer Stelle S . Treffen sich im nächsten Zyklus die Signale beider Vögel wieder in S , so sind beide gleich alt. In diesem Fall heißen sie 'äquivalent'. Einer von beiden wird deaktiviert, seine Spur und Steine bleiben erhalten, werden aber für weitere Aktivitäten anderer Vögel 'durchlässig'. Die Spuren beider gleichaltriger Vögel werden als äquivalent gekennzeichnet. Alles dies ist durch Versenden geeigneter Signale auf den betreffenden Spuren möglich. Welcher deaktiviert wird ist entweder zufällig (hier wird eine Variante von indeterminierten z.a. benötigt) oder hängt von dessen Lage (rechts, über den anderen Vogel) ab. In diesem Fall ist der z.a. nicht isotrop. Eine Lösung i einem determinierten und isotropen z.a. ist mir nicht bekannt.

Treffen sich die Signale des nächsten Zyklus nicht in S , ist einer älter. Das lässt sich leicht feststellen und der jüngere und alle dazu äquivalente Vögel werden einschließlich aller ihrer Spuren und Steine deaktiviert, durchlässig und als 'junge Vögel' gekennzeichnet. Dies ist möglich, da äquivalente Vögel über geeignete Spuren verbunden sind. Innerhalb einer Äquivalenzklasse sind alle bis auf einen Vogel bereits inaktiv.

Es seien nun v_1 und v_2 zwei nicht projektive Vögel. Damit treffen sich deren Steine in einem 90° Winkel oder ein Stein des einen trifft auf eine Spur des anderen. Es ist nun ein recht simples Spiel von Laufzeitunterschieden weiterer Signale auf den Spuren beider, um feststellen zu können, welcher Stein des einen weiter vom Vogel entfernt ist als der des anderen. Der Jüngere und alle dazu Äquivalenten werden wieder deaktiviert und als jünger gekennzeichnet.

Das EBP wurde ursprünglich von Rosenstiel et. al [21] für Graphen formuliert. Für 1-dimensionale z.a. existiert eine schöne Lösung von Legendi und Katona [34], die mit fünf Zuständen auskommt.

Übung 2.6.1 *Lässt sich diese Lösungsidee auf den \mathbb{Z}^3 übertragen*

2.7 Ein Beweis des Satzes von Fredkin und Winograd

Im Kapitel 2.1 haben wir den z.a. \mathcal{Z}_3 kennen gelernt, der jedes Muster über $0,1,2$ vervielfacht. Dieser Typ von z.a. wurde von Fredkin eingeführt und von Winograd verallgemeinert zu selbstreproduzierenden Algorithmen auf Richtungsgraphen. D.h. sogar die starre Struktur des \mathbb{Z}^d wird zugunsten allgemeiner Graphstrukturen aufgegeben. Der Satz und der Beweis sind wunderschön und werden hier im Detail vorgestellt.

Definition 2.7.1 *Ein gerichteter Graph $G = (V, E, \Sigma)$ mit Knotengewichten besteht aus*

- V , eine eventuell unendliche Menge von Knoten,
- $E \subseteq V \times V$, eine Menge von gerichteten Kanten,
- Σ , einer endlichen Menge von Knotengewichten.

Eine Abbildung $C : V \rightarrow \Sigma$ heißt eine Konfiguration von G .

Für einen Knoten $v \in V$ sei $N(v) := \{v' \in V \mid (v', v) \in E\}$ die Nachbarschaft von v . Jedes $v' \in N(v)$ heißt Nachbar von v .

Ein Richtungsgraph $G = (V, E, \Sigma, c_1, \dots, c_m)$ ist ein gerichteter Graph mit Knotengewichten (V, E, Σ) , dessen Kantenmenge in eine endliche Zahl von angeordneten Klassen (auch Richtungen genannt) c_1, \dots, c_m mit $c_i < c_{i+1}$ zerlegt werden kann, so dass gilt

- *in jedem Knoten gibt es genau eine ankommende und eine abgehende Kante in der Klasse c_i für $1 \leq i \leq m$, also einer jeden Richtung,*
- *je zwei Pfade, die im selben Knoten beginnen und die gleiche Anzahl von Kanten in jeder Richtung besitzen, enden im gleichen Feld,*
- *zwei Pfade, die nur Kanten aus jeweils einer anderen Richtung $c_i \neq c_j$ enthalten, können sich in höchstens einem Knoten berühren.*

Führt eine Kante der Richtung c_i vom Knoten y zum Knoten x , so heißt y auch i -ter Nachbar von x und wird mit $N_i(x)$ bezeichnet.

Auf Richtungsgraphen können wir leicht verallgemeinerte z.a. einführen. Dazu brauchen wir eine lokale Übergangsfunktion $\delta : \Sigma^m \rightarrow \Sigma$ und erhalten kanonisch einen globalen Übergang $C \vdash C'$ auf Konfigurationen

$C, C' : V \rightarrow \Sigma$ durch

$$C'(x) = \delta(C(N_1(x)), \dots, C(N_m(x))).$$

So wie ein eine Zelle eines z.a. auf die Zustände aller seiner Nachbarzellen reagiert, soll sich die Belegung eines Knoten v anhand der Belegung aller seiner Nachbarknoten ändern. Dies erklärt den Nachbarschaftsbegriff in Richtungsgraphen. Offenbar definiert jeder d -dimensionale z.a. mit der von Neumann Nachbarschaft einen Richtungsgraphen mit den Richtungen “von rechts, links, oben, unten” und “von mir selbst”. D.h. alle parallelen Kanten liegen in der gleichen Klasse. Aber Richtungsgraphen erlauben vielfältige Strukturen. So gilt etwa

- Jede Nachbarschaft im \mathbb{Z}^d definiert einen Richtungsgraphen.
- Die Richtungen sind nicht notwendig unabhängig. Im 2-dimensionalen Schachbrett könnte man auch die Diagonalen als Richtung einführen. Ein horizontaler Schritt kann dann durch einen diagonalen und vertikalen Schritt ersetzt werden.
- Die Nachbarschaft ist nicht notwendig symmetrisch.
- Richtungsgraphen mit der gleichen Anzahl von Richtungsklassen sind nicht notwendig isomorph.
- Richtungsgraphen müssen sich nicht in den \mathbb{Z}^d einbetten lassen.
- In maximal einer Richtung darf ein Knoten mit sich selbst benachbart sein (eine Kantenschleife).

Übung 2.7.1 *Geben Sie zwei Richtungsgraphen mit der gleichen Anzahl von Richtungsklassen an, die nicht isomorph sind.*

Geben Sie einen Richtungsgraphen an, der sich nicht in den \mathbb{Z}^d einbetten lässt.

Gibt es einen Richtungsgraphen, der nicht aus einer Nachbarschaft im \mathbb{Z}^d für ein geeignetes d entsteht?

Es ist egal, ob wir einen Pfad von v aus als eine Folge von n verbundenen gerichteten Kanten in E von v aus auffassen oder als v plus einen Vektor von n Richtungn, da aus jedem Knoten genau eine Kante einer Richtung herausführt. Ein Vektor von Richtungn ohne angegebenen Startknoten v bezeichnen wir daher auch als *abstrakten Pfad*. $r(v)$ sei der Knoten, den der abstrakte Pfad r mit gewähltem Startknoten v erreicht.

Definition 2.7.2 *Es sei r ein abstrakter Pfad. Eine Konfiguration $C' : V \rightarrow \Sigma$ heißt Replik einer Konfiguration $C : V \rightarrow \Sigma$ in “Richtung” r , falls für alle Knoten $y \in V$ $C'(r(y)) = C(y)$ gilt.*

2.7. EIN BEWEIS DES SATZES VON FREDKIN UND WINOGRAD 27

Falls ein abstrakter Pfad p v mit y verbindet, so verbindet p auch $r(v)$ mit $r(y)$. Winograd führt nun folgende Verallgemeinerung des vervielfachenden z.a. \mathcal{Z}_3 ein.

Definition 2.7.3 (Der Algorithmus *Reproduziere*) *Es seien G ein Richtungsgraph mit $\Sigma = \{0, 1, \dots, p-1\}$ mit den Richtungen c_1, \dots, c_m und mit einer Primzahl p . Dann ist Repro_G der zu G gehörende verallgemeinerte z.a. mit der lokalen Übergangsfunktion*

$$\delta(n_1, \dots, n_m) = \sum_{1 \leq i \leq m} n_i \bmod p.$$

Satz 2.7.1 (Winograd) *Es seien G ein Richtungsgraph wie in Definition 2.7.3 und C eine endliche Konfiguration (d.h. C hat den Wert 0 fast überall) von G . Dann existiert eine Zahl k_p , so dass für jedes $k > k_p$ und jede Richtung c_i eine Replik von C in Entfernung p^k in der Richtung c_i in der Konfiguration C' vorkommt, die nach p^k Schritten aus C im z.a. Repro_G entsteht.*

Beweis. Es empfiehlt sich ein Funktional Ψ zu betrachten, das zu jedem t und jeder Konfiguration C sagt, wie sich C nach t Schritten entwickelt. $\Psi : \mathbb{N} \times \Sigma^V \rightarrow \Sigma^V$ definieren wir induktiv

- $\Psi(0, C) := C$,
- $\Psi(t+1, C)(x) := C'$ mit $\Psi(t, C) \vdash C'$.

Für diesem konkreten z.a. Repro_G gilt also

$$\Psi(t+1, C)(x) := \left(\sum_{y \in N(x)} \Psi(t, C)(y) \right) \bmod p.$$

Schritt 1 *Linearkombination von Elementarkonfigurationen*

Es sei δ_y die Kronecker-Funktion von irgendeiner aus dem Kontext bekannten Menge M in $\{0, 1\}$ mit $\delta_y(z)$ gleich 1, falls $y = z$, und gleich 0 sonst, für $y, z \in M$. Wir können δ_y selbst als die Konfiguration auffassen, die genau dem Knoten $y \in V$ den Wert 1 zuordnet und allen anderen den Wert 0. Analog ist $n \cdot \delta_y$, die Konfiguration, die genau im Knoten y den Wert n und 0 sonst hat. Solche eine Konfiguration, mit dem Wert 0 überall bis auf maximal einen Knoten, nennen wir eine *Elementarkonfiguration*. Damit gilt stets

$$C(x) = \sum_{y \in V} C(y) \cdot \delta_y(x).$$

Jede Konfiguration ist Summe ihrer Elementarkonfigurationen. Über Induktion zeigen wir

$$\Psi(t, C)(x) = \left(\sum_{y \in V} C(y) \cdot \Psi(t, \delta_y)(x) \right) \bmod p.$$

Induktionsbeginn.

$$\Psi(0, C)(x) := C(x) = \sum_{y \in V} C(y) \cdot \delta_y(x) = \sum_{y \in V} C(y) \cdot \Psi(0, \delta_y)(x),$$

egal ob mod p oder nicht, da $C(x)$ stets $\leq p$ ist.

Induktionsschritt.

$$\begin{aligned} \Psi(t+1, C)(x) &= \left(\sum_{z \in N(x)} \Psi(t, C)(z) \right) \bmod p, \text{ laut Def. von } \Psi \\ &= \left(\sum_{z \in N(x)} \left(\sum_{y \in V} C(y) \cdot \Psi(t, \delta_y)(z) \right) \bmod p \right) \bmod p, \text{ laut Indvor.} \\ &= \left(\sum_{y \in V} C(y) \left(\sum_{z \in N(x)} \Psi(t, \delta_y)(z) \right) \bmod p \right) \bmod p \\ &= \left(\sum_{y \in V} C(y) \cdot \Psi(t+1, \delta_y)(x) \right) \bmod p, \text{ laut Def. von } \Psi. \end{aligned}$$

Diese Gleichung interpretieren wir wie folgt:

- Betrachten wir den Spezialfall

$$\Psi(t, n \cdot \delta_y)(x) = \left(n \cdot \Psi(t, \delta_y)(x) \right) \bmod p,$$

so bedeutet das: Eine Elementarkonfiguration $n \cdot \delta_x$ entwickelt sich unter Ψ nach t Schritten im Knoten y genauso, wie die Elementarkonfiguration δ_x sich nach t Schritten in y entwickelt, und anschließend multipliziert mit $n \pmod p$ wird.

- Eine endliche Konfiguration C entwickelt sich unter Ψ nach t Schritten genauso wie die Summe (mod p) aller getrennten Entwicklungen nach t Schritten aller Elementarkonfigurationen aus denen C besteht.

Schritt 2 Pfadanzahl

Es sei $p_t(y, x)$ die Zahl der Pfade der Länge t in G vom Knoten y zum Knoten x . Dann gilt

$$\Psi(t, \delta_y)(x) = p_t(y, x) \bmod p.$$

Für $t = 0$ ist das klar, denn es gibt nur einen Pfad der Länge 0 von y nach x , nämlich wenn $y = x$ gilt. Damit schließen wir induktiv

$$\begin{aligned} p_{t+1}(y, x) &= \sum_{z \in N(x)} p_t(y, z) \\ &= \left(\sum_{z \in N(x)} \Psi(t, \delta_y)(z) \right) \bmod p, \text{ laut Indvor.} \\ &= \Psi(t+1, \delta_y)(x), \text{ laut Def. von } \Psi. \end{aligned}$$

Schritt 3 Berechnung der Pfadanzahl

Bislang wurde noch gar nicht ausgenutzt, dass G ein Richtungsgraph ist, und die bisherigen Überlegungen gelten ganz allgemein. Jetzt nutzen wir die Eigenschaften von Richtungsgraphen zur konkreten Berechnung von $p_t(y, x) \bmod p$ aus. Offensichtlich gibt es nur endlich viele Pfade vom Knoten y zum Knoten x einer vorgegebenen Länge n . Dies sei die Menge $P_t(y, x) = \{p_1, \dots, p_u\}$. p_l und p_j heißen äquivalent, wenn beide Pfade jede Richtung c_i genauso oft benutzen. $[p_l]$ sei die Menge aller zu p_l äquivalenter Pfade. Nach Definition des Richtungsgraphen enden äquivalente Pfade im gleichen Knoten. Liegt also p_l in $P_t(y, x)$, dann schon ganz $[p_l]$. Also

$$P_t(y, x) = [p_{i_1}] \cup \dots \cup [p_{i_h}].$$

Wieviele Pfade enthält nun eine Klasse $[r]$? r ist ein Pfad der Länge n von y aus nach x und r benutze für $1 \leq i \leq m$ jede Elementarrichtung c_i genau n_i mal. Dabei gilt natürlich $n_i \geq 0$ und $t = n_1 + \dots + n_m$. Damit existieren genau so viele Pfade, die jede Richtung c_i genau n_i mal benutzen, wie es Permutationen von n Objekten gibt, von denen jeweils n_i gleich sind für $1 \leq i \leq m$. Oder anders gefragt, wenn ich t Stück Münzen habe von insgesamt m verschiedenen Sorten habe, und jede Münze der Sorte i komme genau n_i mal vor, in wieviele Anordnungen kann ich diese hinlegen? Die Anzahl dieser Möglichkeiten ist

$$s = \frac{t!}{n_1! \cdot \dots \cdot n_m!}.$$

Schritt 4 Berechnung der Pfadanzahl mod p für $n = p^k$

Nun gibt es einen Sonderfall, $s \bmod p$ leicht zu berechnen, nämlich für $n = p^k$. Dann lässt sich s immer durch p teilen, es sei denn es wird nur eine Sorte von Münzen verwendet, d.h. $n_i = p^k$ gilt schon für ein i . In diesem Sonderfall ist $s = 1$. Damit ist s modulo p gerechnet genau dann 1, falls bereits ein n_i gleich p^k ist, und 0 sonst. D.h., genau dann wenn r nur in eine Richtung c_i läuft, ist $s \bmod p$ gleich 1, und 0 sonst. Mit

Schritt 2 gilt daher

$$\begin{aligned}
 \Psi(p^k, \delta_y)(x) &= p_t(y, x) \bmod p \\
 &= |P_t(y, x)| \bmod p \\
 &= |[p_{i_1}]| + \dots + |[p_{i_h}]| \bmod p \\
 &= |[p_{i_1}]| \bmod p + \dots + |[p_{i_h}]| \bmod p \\
 &= 1
 \end{aligned}$$

genau dann wenn x von y p^k Schritte in einer Richtung c_i entfernt ist. Nur Pfade in einer Richtung c_i liefern in dieser Formel den Wert 1 mod p . Zwei Pfade unterschiedlicher Richtungen c_i und c_j , die in y beginnen, können sich nicht in x treffen, wegen der dritten Bedingung von Richtungsgraphen. Dies heißt aber nicht anders als:

Es seien G ein Richtungsgraph mit m Richtungen und $k > 0$. Dann hat $\Psi(p^k, \delta_x)$ überall den Wert 0 bis auf genau m Knoten, die sich alle in Entfernung p^k in einer der m Richtungen von x entfernt befinden, wo der Wert 1 angenommen wird.

Damit haben wir alles zusammen, um den Satz von Winograd abschließend zu beweisen:

Mit Schritt 1 ist das Ergebnis von Ψ auf eine Konfiguration C nach n Schritten an jeder Stelle die Summe modulo p der einzelnen Elementarkonfigurationen von C . Für $t = p^k$ führt jede Elementarkonfiguration zu einer Replik in jeder der m Richtungen in einer Entfernung p^k . Zwei Pfade zweier Elementarkonfiguration in Richtung i und in Richtung j können sich für $i \neq j$ in höchstens einem Punkt schneiden. Ist nun k groß genug, so liegen die m Ergebnisse einer endlichen Ausgangskonfiguration pro Richtung nach p^k Schritten in disjunkten Bereichen und bilden damit m Repliken. ■

2.8 Artificial Life

Von Neumann interessierte sich Ende der Fünfziger, Anfang der Sechziger Jahre des 20. Jahrhunderts für die Frage nach Selbstreproduktion. Ursprünglich schwebte ihm ein Modell vor von Objekten, die in einer Flüssigkeit schwimmend sich organisieren und, unter Verwenden von in der Flüssigkeit vorhandenen Grundstoffen, diese zur Reproduktion der Ausgangsobjekte veranlassen. Der ungarische Mathematiker Ulam schlug von Neumann das Modell des zellularen als Alternative vor, das von Neumann dann auch übernahm. Ein Pattern R_U in einem zellularen Automaten ist berechnungsuniversell, falls R_U die Rechnung einer beliebigen

Turing-Maschine M mit beliebigen Inputwort w simulieren kann. Dazu verbindet man R_U mit einfachen Pattern c_M und s_w , die Verschlüsselungen von M und w darstellen. Diese Codierungen müssen einfach sein, damit das Ergebnis der Rechnung von M mit Startwort w nicht in c_M oder c_w bereits verborgen ist, sondern von R_U berechnet werden muss. Unter Artificial Life verstand von Neumann nun ein Pattern A , das sowohl berechnungsuniversell ist als auch eine Kopie von sich selbst in einen zuvor leeren Teil des zellularen Raumes erzeugen kann.

Von Neumann konnte solch einen artificial life z.a. konstruieren, der mit der von Neumann Nachbarschaft und neunundzwanzig Zuständen arbeitet. Codd [3] konnte die benötigte Anzahl von Zuständen auf acht reduzieren. In seiner Doktorarbeit stellte Banks [2] verschiedene kleine z.a. vor, darunter einen artificial life z.a. mit der von Neumann Nachbarschaft und nur vier Zuständen. Will man nur Berechenbarkeitsuniversalität, ohne Selbstreproduktion, kommt Banks mit 2 Zuständen und der von Neumann Nachbarschaft aus, siehe auch

http://www.bottomlayer.com/bottom/banks/banks_commentary.htm

Im \mathbb{Z}^3 existiert ein berechenbarkeitsuniverseller reversibler z.a. mit 2 Zuständen von Miller und Fredkin [7], der allerdings unterschiedliche lokale Regeln in unterschiedlichen Phasen anwendet, siehe

<http://arxiv.org/ftp/nlin/papers/0501/0501022.pdf>

Wir wollen die verwendeten Techniken an einer abstrakteren Technik skizzieren. Ein artificial life z.a. kann z.B. neben dem Ruhezustand 0 einen weiteren Zustand X benutzen, so dass jedes Muster über nur 0 und X (im folgenden ein X -Muster genannt) tot ist, d.h. keine Nachfolgekonfiguration besitzt. Die X -Muster dienen als eine Art Skelett für ein Schaltwerk, also für Leitungen, Dioden, Gates, Transistoren, etc. Eine Aktivität findet erst statt, wenn Signale entlang der X -Muster laufen. Manche X -Muster können Signale verdoppeln, eine Sorte Signale in eine andere verwandeln, sich treffende Signale auslöschen oder passieren lassen. Eine Uhr wird z.B. durch eine Kreis von X -Leitungen realisiert, auf dem ein Signal kreist. Ein Taktgeber ist eine Uhr, die an einer Stelle das kreisende Signal verdoppelt und eines der Signale als Zeitsignal an verschiedene X -Muster schickt. Die X -Muster bilden ein totes Schaltwerk, das durch ein einzelnes Signal, das im Schaltwerk geeignet vervielfacht wird, zum Leben erweckt wird. So kann man endliche Logik realisieren. Um eine Turing-Maschine zu realisieren könnte man über weiter X -Muster spezielle Signale auf 1-dimensionale X -Folgen senden, die diese X -Folgen verlängern oder verkürzen können. Diese X -Folgen spielen dann ein Band nach. Man kann zeigen, dass man jede Turing-Maschine durch eine Mehrband-Turing-Maschine simulieren kann, in denen nur die Länge der benutzten Bänder und nicht die auf ihnen stehende Wörter

von Bedeutung sind (sogenannte Registermaschinen). Diese Registermaschinen sind also berechnungsuniversell und durch X -Muster mit einem Startsignal simulierbar.

Ferner existiert ein spezielles X -Muster C_U , universeller Konstruktor genannt, mit folgender Eigenschaft: zu jedem X -Muster M existiert eine Blaupause c_M derart, dass, wenn man die Blaupause c_M an den universellen Konstruktor C_U geeignet anhängt und C_U ein Startsignal gibt, C_U in einiger Entfernung in ein zuvor leeres Teil des \mathbb{Z}^2 eine Kopie von M erzeugt. Dazu lässt C_U , gesteuert von c_M diverse X -Arme an verschiedene Stellen im \mathbb{Z}^2 wachsen und wieder in sich zurückziehen bis auf das letzte X . So kann, wie in einem 3d Drucker, von hinten nach vorn (in Richtung der Lage von C_U) Zelle für Zelle ein totes X -Muster M erzeugt werden. Zum Schluss legt C_U ein Startsignal auf M und initiiert so M .

Im Prinzip arbeitet Banks kleiner artificial life z.a. so. Da hier aber nur vier Zustände, also nur zwei weitere neben 0 und X , zur Verfügung stehen, wird es komplizierter. Statt Registermaschinen kann Banks nur endliche Schaltwerke und einen universellen Konstruktor realisieren. Dieser kann jedoch neue Schaltelemente dem Schaltwerk hinzufügen. Wachsende Schaltwerke sind aber ebenfalls berechnungsuniversell.

Damit erhält man sofort einen berechnungsuniversellen Konstruktor U , der bei Vorlage einer Blaupause c_U eine Kopie von U erzeugen kann. Echte Selbstreproduktion muss natürlich auch die Blaupause kopieren. Das kann Banks' z.a. nicht. Hier wird ein tiefer Satz (Kleenes Rekursionstheorem) aus der Rekursionstheorie benutzt, der in einer Interpretation sagt, dass man auch ohne Blaupause auskommt. Z.B. kann man aus dem Rekursionstheorem folgern, dass eine Turing-Maschine existiert, die ihre eigene Gödelnummer ausdrückt. Ebenso existiert eine universelle Turing-Maschine, die bei geeignetem einfachen Input die Arbeitsweise jeder anderen Turing-Maschine simulieren kann - vergleiche die Vorlesung "Einführung in die Theoretische Informatik" - , und die ebenfalls ihre eigene Blaupause (hier: Gödelnummer) ausdrucken kann.

2.9 Garten Eden Konfigurationen

Im letzten Abschnitt haben wir universelle Konstrukteure kennen gelernt. Der Universelle Konstruktor C_U von Banks benutzt vier Zustände, darunter 0 und X , und kann beliebige X -Muster erzeugen (und anschließend mit einem Startsignal füttern). Für berechnungsuniverselle Selbstreproduktion ist das ausreichend. C_U kann aber nicht jedes beliebige Muster über allen vier Zuständen erzeugen. Könnte das vielleicht ein anderer,

komplizierterer universeller Konstruktor. Oder gibt es vielleicht Konfigurationen, die gar nicht generierbar sind? Eine Konfiguration C eines z.a. \mathcal{Z} wird Garten Eden Konfiguration genannt, falls C keine Vorgängerkonfiguration besitzt.

Es existiert eine reiche Literatur über Garten Eden Konfigurationen, in der eine gewisse Konfusion zu verzeichnen ist, weil unterschiedliche Arten von Konfigurationen verwendet wurden. Nach unserer Definition ist eine Konfiguration eine Abbildung $C : \mathbb{Z}^d \rightarrow S$ mit endlichem Support. Dies wird auch häufig als endliche Konfiguration bezeichnet und bei dem Begriff ‘‘Konfiguration’’ wird oft die Forderung nach endlichem Support fallen gelassen. Man erhält noch weitere Konfigurationskonzepte. So lassen sich periodische, schwach periodische oder rationale Konfigurationen definieren. Wir stellen nur zwei weitere Konzepte vor.

Definition 2.9.1 *Eine Abbildung $C : \mathbb{Z}^d \rightarrow S$ heißt (d -dimensionale) unendliche Konfiguration, oder ∞ -Konfiguration, über S . Ist C darüber hinaus Turing-berechenbar, so heißt sie rekursive Konfiguration.*

Unter Konfiguration ohne Zusatz verstehen wir also immer unsere endlichen Konfigurationen, d.h. mit endlichem Träger.

Aber auch mit dieser Unterscheidung von endlichen und ∞ -Konfigurationen verbleiben noch weitere Möglichkeiten, Garten Eden Konfigurationen zu definieren. Golze stelle in [9] über Einhundert verschiedene Garten Eden Konzepte vor. Es wird in der Originalliteratur meist nicht klar, welches Konzept von Konfiguration und Garten Eden gerade verwendet wird, mit einem großen Verwirrungspotential. Um hier nicht völlig durcheinander zu kommen, führen wir drei Begriffe ein, die so allerdings in der Literatur nicht heißen: die Moore-, echte und abstrakte Garten Eden Konfigurationen.

Definition 2.9.2 *Es seien C eine endliche oder ∞ -Konfiguration und M eine endliche Teilmenge des \mathbb{Z}^2 . $C_M : M \rightarrow S$ ist die Restriktion von C auf M als Definitionsbereich. $N(M) := \{z \in \mathbb{Z}^2 \mid \exists x \in M : y \in N(x)\}$ ist die Nachbarschaft um M .*

Eine Restriktion C_M einer Konfiguration C heißt

- Moore GE (für Moore-Garten-Eden-Konfiguration), falls keine endliche Konfiguration C' existiert mit $C' \vdash C''$ und $(C)_M = (C'')_M$
- abstrakte Moore GE, falls keine ∞ -Konfiguration C' existiert mit $C' \vdash C''$ und $(C)_M = (C'')_M$

Wir wollen den Unterschied in einem Beispiel verdeutlichen. Jede abstrakte Moore GE ist auch eine Moore GE, aber nicht notwendig umgekehrt. Wir wählen als ein Beispiel den 1-dimensionalen z.a. A mit von Neumann Nachbarschaft und zwei Zuständen 0,1 und den folgenden lokalen Übergängen, die wir der Einfachheit halber als

$$xyz \rightarrow t$$

schreiben, falls die Zelle im Zustand y mit der linken Nachbarzelle im Zustand x und der rechten im Zustand z in einem Schritt in den Zustand t überführt wird:

$$\begin{array}{cccc} 000 \rightarrow 0 & 001 \rightarrow 1 & 100 \rightarrow 1 & 101 \rightarrow 0 \\ 010 \rightarrow 1 & 011 \rightarrow 0 & 110 \rightarrow 1 & 111 \rightarrow 0 \end{array}$$

Übung 2.9.1 *Es sei δ_1 die 1-dimensionale Konfiguration mit dem Wert 1 in der Zelle 1 und den Wert 0 sonst überall. Zeigen Sie*

- Für die ∞ -Konfiguration C_δ mit $C_\delta(i) = 1$ für $i > 1$ und $C_\delta(i) = 0$ für $i \leq 1$ gilt

$$C_\delta \vdash \delta_1, \text{ (das ist leicht)}$$

- C_δ ist die einzige Vorgängerkonfiguration von δ_1 , (das ist schwieriger)
- \vdash ist injektiv auf der Menge der endlichen Konfigurationen. (Schwierig. Geben Sie einen Algorithmus an, der zu jeder endlichen Konfiguration versucht, eine Vorgängerkonfiguration zu konstruieren. Zeigen Sie dabei, dass es nur eine endliche Vorgängerkonfiguration geben kann.)

Das obige Beispiel zeigt also einen z.a., in dem eine Moore GE nicht eine abstrakte Moore GE ist. \vdash in diesem Beispiel ist injektiv aber nicht surjektiv auf der Menge aller endlichen Konfigurationen. Umgekehrt kann man aber überraschenderweise zeigen, dass jedes \vdash , das auf der Menge der endlichen Konfigurationen surjektiv ist, auf dieser Menge auch injektiv und damit bijektiv sein muss. Natürlich bietet sich eine alternative Definition von Garten Eden Konfigurationen nur über Surjektivität an.

Definition 2.9.3 *Variante von GE sind:*

- Eine abstrakte GE ist eine ∞ -Konfiguration C , für die keine ∞ -Konfiguration C' existiert mit $C' \vdash C$.

- Eine echte GE ist eine endliche Konfiguration C , für die keine endliche Konfiguration C' existiert mit $C' \vdash C$.

Damit besitzt ein z.a. eine abstrakte Garten Eden Konfiguration genau dann, falls \vdash auf der Menge der ∞ -Konfigurationen nicht surjektiv ist, und eine echte GE, falls \vdash auf der Menge der endlichen Konfigurationen nicht surjektiv ist.

Ohne Beweis wollen wir folgenden durchaus verwirrenden Zusammenhang angeben. Details und weiterführende Literatur finden sich in Amoroso, Cooper, Patt [30]. Weitere Zusammenhänge mit “mutually erasing” Konfigurationen lassen wir lieber ganz weg.

Satz 2.9.1 (Zusammenhang der GE Konzepte) *Es gilt für jeden z.a. A mit der globalen direkten Nachfolgerrelation \vdash_A :*

- \vdash_A ist nicht surjektiv auf der Menge aller ∞ -Konfigurationen
genau dann wenn
- A besitzt eine abstrakte GE
genau dann wenn
- A besitzt eine abstrakte Moore GE
genau dann wenn
- A besitzt eine Moore GE
genau dann wenn
- \vdash_A ist nicht injektiv auf der Menge der endlichen Konfigurationen,
dies impliziert aber nur
- \vdash_A ist nicht surjektiv auf der Menge der endlichen Konfigurationen
genau dann wenn
- \vdash_A ist nicht bijektiv auf der Menge der endlichen Konfigurationen
genau dann wenn
- A besitzt eine echte GE.

Man findet für das Game of Life leicht zwei simple Konfigurationen, die in einem Schritt in die leere Konfiguration übergehen. Damit ist \vdash im Game of Life nicht injektiv und es existieren echte Garten Eden Konfigurationen. Es soll hier noch ein schönes Abzählungsargument von Banks und Ward vorgestellt werden, das zeigt, dass im Game of Life bereits in einem Quadrat der Seitenlänge 40 Garten Eden Konfigurationen existieren müssen.

Zur Erinnerung, eine Zelle schaltet im nächsten Schritt in den Zustand 1 (lebendig), falls sie bereits im Zustand 1 ist und genau 2 oder 3 ihrer acht Nachbarn ebenfalls, oder falls sie im Zustand 0 (tot) ist und genau 3 ihrer Nachbarn lebendig sind. D.h., es werden 3×3 -Fenster betrachtet. In einem Fenster können sich $2^9 = 512$ verschieden Muster über 0,1 befinden. Bei

$$\binom{8}{3} + \binom{8}{3} + \binom{8}{2} = 140$$

Mustern davon schaltet das Zentrum auf 1, in den restlichen 372 Fällen auf 0. Ein *1-Elementarfenster* sei ein 3×3 -Fenster mit der Zentrumszelle im Zustand 1. Wir betrachten nun ein Makroquadrat, M_l dass aus l^2 vielen aneinander liegenden 1-Elementarfenstern besteht. M_l besteht damit aus $9 \cdot l^2$ vielen Zellen, von denen l^2 viele im festen Zustand 1 sein müssen. Der Rand von M_l besteht aus $4 \cdot 3 \cdot l - 4$ vielen Zellen, also $12 \cdot l - 4$. Es sei R_l gerade M_l ohne diese Randzellen. R_l besteht also aus $9 \cdot l^2 - (12 \cdot l - 4)$ vielen Zellen. Damit liegt in R_l in l^2 vielen Zellen der Zustand 1 fest (die Zentren der 1-Elementarfenster) und in den $f = 8 \cdot l^2 - 12 \cdot l + 4$ vielen Restzellen in R_l kann der Zustand 0 oder 1 sein. Das ergibt 2^f viele erlaubte Konfigurationen innerhalb von R_l und 140^{l^2} viele Konfigurationen in M_l können die gewünschte 1-Verteilung in den Zentren der 1-Elementarzellen bewirken. D.h., nur 140^{l^2} viele dieser erlaubten Konfigurationen besitzen eine Vorgängerkonfiguration. Bereits für $l > 13.44$ gilt $140^{l^2} < 2^f$. Da mit $l = 13.44$ ein Quadrat R_l mit $3 \cdot l - 2 > 40$ vielen Randzellen ausgelegt wird, muss in einem Quadrat von 40×40 Zellen bereits eine echte Garten Eden Konfiguration existieren.

Füllt man ein 40×40 Quadrat zufällig mit Zuständen 0,1, so beträgt die Chance **keine** Garten Eden Konfiguration zu treffen gerade 1 zu 128. Nur jede 128. Konfiguration in diesem Quadrat besitzt eine Vorgängerkonfiguration.

Es sei noch bemerkt, dass es für 1-dimensionale z.a. die Existenz von Garten Eden Konfigurationen entscheidbar ist, da zu jeder endlichen Konfiguration eine endliche Vorgängerkonfiguration algorithmisch berechnet werden kann. Für höher dimensionale z.a. ist dies Problem aber unentscheidbar (Golze [9]).

2.10 Zellulare Automaten, Physik und Fiktion

Eine physikalische Theorie besagt, dass auch der Raum und die Zeit selbst quantifizierbar sind und kleinste Raum- und Zeiteinheiten existieren. Die kleinste Länge, Elementarlänge oder Plancklänge genannt, wird auf $10^{-35}m$ vermutet. Um von dieser Winzigkeit eine Vorstellung zu erhalten: in einen Atomkern können mehr dieser Raumatome Platz finden als Staubteilchen im Sonnensystem. Ein Konsequenz solcher kleinster Raumeinheiten ist es, dass der Urknall nicht eine mathematische Singularität gewesen sein kann. Ein vor dem Urknall implodierendes Universum hätte sich nicht auf einen Nullraum zusammenziehen können. Alle Raumatome eines kleinen Gebiets wären irgendwann vom implodierenden Universum gefüllt gewesen und die Implosion des Raumes wäre davon abgeprallt.

Schon sehr früh kam der Gedanke auf, dass das Raum-Zeit-Gefüge, wenn es schon kleinste Raum- und Zeiteinheiten besitzt, wie ein zellulare Automat arbeiten könne. Jedes Raumzeitatom kann in einem von endlich vielen möglichen Zuständen sein. Jedes Raumzeitatom kommuniziert mit endlich vielen anderen mittels Austausch der Information über die jeweiligen Zustände. Diese Kommunikation bewirkt neue Informationen und damit einen neuen Zustand in jeder Raumeinheit im nächsten Zeitschritt. Grundobjekte sind also nicht mehr Elementarteilchen, die sich in einem leeren kontinuierlichem Raum bewegen. Solche Elementarteilchen sind vielmehr stabile Muster in einem quantisierten rechnenden Raum. Da sich nur Zustände ändern kann sich ein Muster bewegen, ohne dass sich die Raumzeitatome selbst bewegen. Eine neue Interpretation des Stadion-Paradoxon von Zeno sagt, dass Bewegung im Raum gar nicht möglich sei, wenn sich der Raum nicht beliebig weit unterteilen ließe. Die Argumentation ist wie folgt: Es sei $abcd$ ein Block von vier Raumatomen linear angeordnet, der sich pro Zeitatom um ein Raumatom nach rechts bewegt. Ein weiterer Block $efgh$ bewege sich genauso nach links, beide begegnen sich im Abstand von einem Raumatom. Wir haben also die Situation

Zeit t	0	a	b	c	d	0	0	0
		0	0	0	e	f	g	h
Zeit $t + 1$	0	0	a	b	c	d	0	0
		0	0	e	f	g	h	0

Damit hat sich e nach links gehend an b vorbei bewegt, ohne b zu passieren (denn es gibt keinen Zeitschritt zwischen t und $t + 1$ und keinen weiteren Raum zwischen den bezeichneten Raumeinheiten. Damit führen sich bewegende kleinste Raumzeitatome zu einem Paradoxon. Ein z.a., in

dem sich die Raumzeitatome gar nicht bewegen, wo nur neue Zustände berechnet werden, löst dieses Problem sofort.

Bekannte Sympathisanten dieses Denkansatzes sind Konrad Zuse (siehe sein Büchlein "Der rechnende Raum" [13]), Fredkin (ehemaliger Direktor des Project MAC am MIT) und Steve Wolfram (Wolfram Industries mit dem Produkt Mathematica). Das über 1000 Seiten mächtige Buch "A new kind of science" von Wolfram [31] behandelt diese Thematik. Bereits Zuse stellte die Frage, wie die nachgewiesenen Isotropie der Lichtgeschwindigkeit in einem zellularen Raum erklärbar sei, da in diesem stets Vorzugsrichtungen vorhanden sein sollten. Vielleicht sollte man sich den zugrunde liegenden Raum nicht als den \mathbb{Z}^3 mit extrem kleiner Kantenlänge der 3d-Quader vorstellen, sondern eher als ein Gas von elementaren Raumteilchen, als eine Art Raumschaum, ähnlich wie der Quantenschaum, dessen Nachbarschaftsstruktur flexibel ist. Dieser Gedanke mag merkwürdig sein, dennoch sind viele zuerst merkwürdige Ideen Bestandteil der Physik geworden (Unschärfe, Quantenschaum, virtuelle Teilchen). Sicherlich ist solch eine Vorstellung nicht phantastischer als die Stringtheorie. Wir haben schon im Kapitel über Fredkins und Winograds Beweis eine Verallgemeinerung von z.a. auf Richtungsgraphen kennengelernt. Zuse ging 1975 in [14] weiter und schreibt darüber in einem kleinen 1977 veröffentlichten Bericht [38]:

"Demgegenüber entwickelte ich in einer weiteren Arbeit 'Ansätze einer Theorie der Netzautomaten' die Idee wachsender und veränderlicher Netze mit unregelmäßiger Struktur. Der Aufbau des Netzes selbst unterliegt dabei laufenden Änderungen, die wiederum mit den im Netz ablaufenden Vorgängen in Beziehung stehen. Anstelle des gekrümmten Raumes tritt dabei der geknickte und zerknitterte Raum. Der Beziehung zwischen Materie und Raumkrümmung der allgemeinen Relativitätstheorie entspricht dabei die Beziehung zwischen Schaltungsmustern und Netzstruktur.

Signalfortpflanzung in derartigen unregelmäßigen Netzen erfolgen mit im statistischen Mittel gleicher Geschwindigkeit in verschiedenen Richtungen."

Trotz Heisenbergs Unschärferelation, trotz der akzeptierten Idee des Quantenschaums vertreten viele Physiker noch die Idee, dass die Entwicklung des Universums determiniert sei. Nicht nur vorwärts in der Zeit sondern auch rückwärts. D.h. man glaubt, dass zu jeder Konfiguration genau eine Nachfolge- und genau eine Vorgängerkonfiguration existieren muss. Derartige Systeme werden reversibel genannt. Eine exakte Definition mit Beispielen von reversibel abstrakten Rechenmodellen findet sich im Theoriebuch von Erk, Priese [12]. Somit stellt sich auch die Frage nach reversiblen z.a. In diesen physikalisch motivierten Fragestellungen wer-

den meist auch ∞ -Konfigurationen zugelassen, in denen unendlich viele Zellen in einem Zustand ungleich 0 sein dürfen. Ein z.a. heißt nun reversibel, falls nicht nur jede ∞ -Konfiguration maximal eine ∞ -Vorgängerkonfiguration besitzt, sondern falls dieser Vorgänger auch lokal durch einen anderen z.a. berechnet werden kann.

Toffoli [35] konnte zeigen, dass reversible z.a. berechnungsuniversell sind. Die Frage, ob z.a. reversibel sind, ist für 1-dimensionale z.a. entscheidbar (Golze [9]), für 2-dimensionale hingegen unentscheidbar (Kari [15]), ähnlich wie die Entscheidbarkeit der Frage nach Garten Eden Konfigurationen.

In einer Vorstellung des physikalischen Universums als z.a. stellt sich die Frage, wie ein globaler Takt möglich sein soll. Petri hat in seiner berühmten Dissertation [24] bereits den Gedanken eines universellen Taktes für beliebig groß wachsende Schaltwerke ad absurdum geführt. Würde man etwa eine Turing-Maschine mit unbeschränkt wachsenden Band in einem Schaltwerk realisieren, an das man bei Bedarf (Wachsen des Bandes) beliebig viele weitere Schaltwerke andocken kann, so wird entweder die globale Taktzeit immer langsamer oder man verzichtet auf einen globalen Takt generell. Damit führte Petri Untersuchungen zur nicht getakteten, 'nebenläufigen' Kommunikation beliebig vieler Agenten in die Informatik ein. Petri Netze sind das bekannteste und weitgehendst untersuchte Modell solcher nebenläufiger Parallelarbeit.

Man kann versuchen, in z.a. auf einen globalen Takt zu verzichten und diesen durch 'lokal synchrone' Arbeitsweise zu ersetzen. Wachsmuth konnte allerdings in seiner Dissertation [36] zeigen, dass sich die Möglichkeit des Überholens von zeitlich unterschiedlich ausgesendeten Signalen in nicht global getakteten aber lokal synchronen z.a. nicht verhindern lässt.

Zuse hat in seinem Büchlein "Der rechnende Raum" bereits die Möglichkeit von selbstreproduzierenden Robotern vorgestellt, deren Nachkommen kleiner als sie selbst sind. Dieser Gedanke ist in der Science Fiction Welt unterdessen verbreitet, siehe die Replikanten in der Fernsehserie "Stargate" oder der selbstreproduzierende Staub, aus dem der Wächterroboter im Remake "Der Tag, an dem die Erde still stand" besteht.

Einem der Pioniere der Theorie z.a., nämlich John Holland, gelang es dann, die starre Struktur in z.a. vollkommen zu überwinden, in dem er als Erster eine Theorie eines auf dem ersten Blick völlig verschiedenen Modells schuf: eine Theorie genetischer Algorithmen.

Kapitel 3

Lindenmayer Systeme

Zellulare Automaten beschreiben hochparallele Anordnungen von endlichen Automaten. In der Theorie formaler Sprachen sind Grammatiken das Gegenstück zu Maschinenmodellen. So wie normalerweise Maschinen sequentiell arbeiten, finden Ersetzungen in Grammatiken sequentiell, allerdings indeterminiert, statt. Lindenmayer Systeme sind nun hochparallel arbeitende Grammatiken. Sie wurden Ende der Sechziger Jahre des Zwanzigsten Jahrhunderts von ungarischen Biologen Aristid Lindenmayer [17] eingeführt und führten rasch zu einem großen Interesse in Biologie und, viel mehr noch, in der Theorie formaler Sprachen. Lindenmayer Systeme eignen sich sehr gut zur Beschreibung und Generierung filigraner Organismen, wie etwa Farne oder Baumstrukturen. In der Arbeitsgruppe zur medizinischen Bildverarbeitung am Labor Bilderkennen wurden Lindenmayer Systeme noch vor kurzer Zeit erfolgreich von Sturm [25] eingesetzt zur Generierung 3-dimensionaler “naturnaher” Nervengeflechte, die als Groundtruth zur Verifikation von 3D Analyse Methoden dienten. Siehe

http://www.uni-koblenz.de/lb/lb_research/db/index.html

Ohne Lindenmayer Systeme wäre eine Erzeugung solcher künstlicher Nervengeflechte deutlich aufwändiger geworden. In der Theoretischen Informatik führten Lindenmayer Systeme zu einer großen Begeisterung. Der Grund ist aber wohl mehr trivialer Art: alle Forscher, die damals in formalen Sprachen bereits ihre ersten Sporen sich verdient hatten, konnte ihre Kenntnisse und Erfahrung unmittelbar auf Lindenmayer Systeme übertragen und leicht zu weiteren Publikationen gelangen. Es wurde eine recht umfangreiche Hierarchie von verschiedensten Modifikationen von Lindenmayer Systemen aufgestellt.

3.1 Begriffe

Eine formale Grammatik G ist ein Tupel $G = (V, T, R, S)$ von *Variablen*, *Terminalen*, *Regeln* und einem *Startsymbol*. Eine Regel $P \rightarrow Q$ aus R ist auf ein Wort w über V - und T -Buchstaben anwendbar, falls w P als ein Teilwort enthält. In einem Anwendungsschritt $w \vdash w'$, wird P in w durch Q ersetzt. Es muss also $w = uPv$, $w' = uQv$ für Wörter u, v über $V \cup T$ gelten. Die von G generierte Sprache $L(G)$ besteht nun aus allen Wörtern nur über T , die aus S abgeleitet werden können, also $L(G) = \{w \in T^* \mid S \vdash^* w\}$. Der Indeterminismus ergibt sich, da in w mehrere Prämissen P verschiedener Regeln an verschiedenen Stellen vorkommen können. Eine mögliche Ersetzung wird indeterminiert in einem Schritt gewählt. In einem Lindenmayer System sollen nun im Prinzip alle möglichen Ersetzungen simultan ausgeführt werden. Dies führt offensichtlich zu einigen Problemen. Das erste ist das von Überlappungen. Betrachten wir die beiden Regeln $ab \rightarrow ba$, $bc \rightarrow cb$ und das Wort abc . Wie soll eine Anwendung dieser beiden Regeln auf w aussehen? Soll das Ergebnis $bacb$ werden? Dieses Überlappungsproblem umgeht man in sogenannten *Typ 0* Lindenmayer Systemen (0L-Systeme), indem man nur Regeln der Form $P \rightarrow Q$ mit $|P| = 1$ zulässt, also stets nur das Ersetzen eines Buchstaben erlaubt, dies allerdings parallel für alle Buchstaben im Wort. Das zweite Problem ist Indeterminismus, wenn für einen Buchstaben a zwei (oder mehr Regeln) Regeln $a \rightarrow P_1$, $a \rightarrow P_2$ existieren. Eine Lösung dieses Problems ist trivial, indem man einfach eine indeterminierte Nachfolgerrelation \vdash zulässt, wie es in der Theorie formaler Sprachen völlig normal ist. Weiter sind noch zwei recht harmlose Veränderungen zu normalen Grammatiken zu nennen: es wird nicht mehr zwischen Variablen und Terminalen unterschieden und statt eines Startsymbols sind ganze Startwörter zulässig. Dies führt zu folgender Definition.

Definition 3.1.1 *Ein Lindenmayer System vom Typ 0 (0L-System) G ist ein Tupel $G = (V, R, u)$ von*

- *einem endlichen Alphabet V ,*
- *einer endlichen Menge R von Regeln der Form $a \rightarrow P$ mit $a \in V$, $P \in V^*$,*
- *einem Startwort $u \in V^+$.*

$v \in V^*$ heißt direkter Nachfolger von $w \in V^*$, $w \vdash v$, falls Buchstaben $a_1, \dots, a_n \in V$ mit $n > 0$ und Regeln $a_i \rightarrow Q_i \in R$ existieren mit $w = a_1 \dots a_n$ und $v = Q_1 \dots Q_n$.

$L(G) := \{v \in V^* \mid u \vdash^* v\}$ ist die von G generierte Sprache.

G heißt deterministisch (*D0L-System*), falls zu jedem $a \in V$ genau eine Regel $a \rightarrow Q$ in R existiert.

G heißt propagierend (*P0L-System*) oder ε -frei, falls für jede Regel $a \rightarrow P \mid P \mid > 0$ gilt.

Ein DP0L-System ist dementsprechend ein deterministisches und propagierendes 0L-System und eine von einem DP0L-System erzeugte Sprache heißt natürlich DP0L-Sprache, etc.

Es existieren noch etliche weitere Klassen von Lindenmayer Systemen, die hier nicht betrachtet werden. Diese Klassen befinden sich bereits in dem schönen Lehrbuch [32] von Salomaa aus dem Jahr 1973. Das Startwort soll nicht das leere Wort sein.

3.2 Wachstum biologischer Organismen

Hierzu betrachten wir folgendes Beispiel eines deterministischen propagierenden 0L-Systems aus [32]. Es seien $V := \{1, 2, 3, 4, 5, 6, 7, 8, (,)\}$, $u = 1$ und folgenden Regeln, die wir senkrecht statt waagerecht schreiben

1	2	3	4	5	6	7	8	()
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
2223	2	43	5	6	7	8(1)	8	()

Damit ergibt sich folgendes determiniertes Verhalten
 $1 \vdash 2223 \vdash 22243 \vdash 222543 \vdash 2226543 \vdash 22276543 \vdash 2228(1)76543$
 $\vdash 2228(2223)8(1)76543$

und nach jeweils $n + 6$ Schritten erhält man das Wort

$$v_{n+6} = 2228(v_n)8(v_{n-1})8\dots 8(v_0)76543,$$

dabei ist v_i die Herleitung nach i Schritten, $v_0 = u (= 1)$.

Hier interpretiert man jede Zahl als eine Zelle in diesem Zustand, (und) als Klammern, deren eingeschlossener Teil aus der die Klammern beinhaltenden Hauptstruktur abwechselnd rechts oder links herauswächst. Damit beschreibt dieses DP0L-System einen Organismus mit einem Stammansatz (Zustand 2) der Höhe 3 (der Anfang 222), anschließend Verknospungszellen (Zustand 8), an denen abwechselnd rechts und links ein weitere Organismus der gleichen Art herauswächst, und 76543 ist das wachsende Ende (die Spitze des Organismus, allerdings an allen Ende aller Ableger, wobei die Spitze nicht bei allen Ablegern bereits kom-

plett ausgebildet sein muss). Die Spitze muss erst in fünf Schritten von 3 auf 76543 weit wachsen, um einen weiteren Ableger erzeugen zu können.

Zeichnet man neu entstehende Strukturen im abnehmenden Maßstab, kann man algorithmisch hübsche Fraktale und biologische Formen erzeugen. Ein schönes Spielzeug zu solch einer “algorithmischen Biologie” findet man unter

<http://de.wikipedia.org/wiki/Fraktal>

http://www.lab4web.com/chelmiger/lyndyhop/lh_start_de.html

und <http://lindenmayer.berlios.de/>.

Übung 3.2.1 *Finden Sie im www ein Java-Applet zum Generieren von biologischen Organismen mittels Lindenmayer Systemen. Spielen Sie damit und generieren Sie einige Muster. Gewinnen Sie dabei ein Gefühl, welche Effekte welche Regeln haben. Führen Sie in der nächsten Übungsstunde vor, was Sie herausgefunden haben.*

3.3 D0L

Interessant ist das Wachstumsverhalten von DP0L-Systemen. Man kann leicht exponentielles, quadratisches, kubisches oder Fibonacci Wachstum erreichen. Betrachten wir als Alphabet hier 1, 2, 3, 4, als Startwort nur 1, dann erhalten wir mit folgenden Regelmengen folgendes Wachstum

- exponentiell: $1 \rightarrow 11$:

$$L = \{1, 11, 1111, 11111111, \dots, 1^{2^n}, \dots\},$$

- quadratisch: $1 \rightarrow 1233, 2 \rightarrow 233, 3 \rightarrow 3$:

$$L = \{1, 1233, 123323333, 123323333233333, \dots\},$$

- kubisch: $1 \rightarrow 12344444, 2 \rightarrow 234444, 3 \rightarrow 3444444, 4 \rightarrow 4$:

$$L = \{1, 1234^5, 1234^5234^534^64^5, \dots\},$$

- Fibonacci: $1 \rightarrow 2, 2 \rightarrow 12$:

$$L = \{1, 2, 12, 212, 12212, 21212212, \dots\},$$

Definition 3.3.1 *Für ein D0L G sei v_i das im i -ten Ableitungsschritt aus dem Startwort erzeugte Wort. $f_G(i) := |v_i|$ ist dann die Wachstumsfunktion von G .*

Die Beispiele zeigen, dass natürliche Wachstumsfunktionen von PD0L-Systemen angenommen werden. Besonders einfach sind die Regeln für Fibonacci-Wachstum, das man auch an vielen Stellen in der Natur wiederfindet, wie in Spiralen in Blütenkelchen oder auf Schneckenhäusern. Paz und Salomaa [23] zeigten 1975, dass die Wachstumsfunktionen von D0L-Systemen stets Summationen von Polynom- und Exponentialfunktionen sind. Da in der Natur auch andere Wachstumsprozesse zu beobachten sind, wie etwa logarithmische, muss zu deren Modellierung das Konzept der Typ 0 Lindenmayer Systeme zu *kontext-abhängigen* Lindenmayer Systemen erweitert werden.

Letztlich sind D0L-Systeme nichts anderes als iterierte Anwendungen eines Homomorphismen. Zu einem D0L-system $G = (V, R, u)$ definieren wir den Homomorphismus $h_G : V^* \rightarrow V^*$ durch $h(a) := Q$, falls $a \rightarrow Q$ eine Regel in R ist. Eine simultane Ersetzung aller Buchstaben durch ein neues Wort gemäß den Regeln ist nichts anderes als eine einmalige Anwendung von h_G . Insbesondere ist $v_i = h^i(u)$. Man darf sich also nicht wundern, wenn man in der Literatur unter dem Stichwort "D0L" Aussagen über Homomorphismen findet.

3.4 Theoretische Informatik

Neben Anwendungen in einer formalen Biologie wurden Lindenmayer Systeme sofort von der Theoretischen Informatik beschlagnahmt und die Standardfragen der Theorie wurde an Lindenmayer Systeme gestellt. Beginnen wir mit einigen Beispielen von 0L und nicht 0L Sprachen.

0L	nicht 0L
$L_1 = \{a\}$	$L'_1 = \{\varepsilon\}$
$L_2 = \{aa\}$	$L'_2 = \{a, aa\}$
$L_3 = \{\varepsilon, a, aa\}$	$L'_3 = \{\varepsilon, aa, aaaa\}$
$L_4 = \{aa\} \cup \{b^{2^n} n > 1\}$	L_4^+

$(\{a\}, \{a \rightarrow \varepsilon\}, aa)$ erzeugt gerade L_3 und $(\{a, b\}, \{a \rightarrow bb, b \rightarrow bb\}, aa)$ gerade L_4 . L'_1 ist nicht 0L, da das Startwort nicht leer sein darf. Dass L_4^+ nicht 0L ist, sieht man wie folgt: Angenommen ein 0L-System $G = (\{a, b\}, R, u)$ erzeuge L_4^+ . Die Regeln $a \rightarrow \varepsilon$ und $b \rightarrow \varepsilon$ sind nicht erlaubt, da sonst mit $aa, bbbb \in L_4^*$ auch ε in L_4^+ sein müsste. Also ist G propagierend und aa muss das Startwort sein. Damit muss $aa \vdash^* aaaa$ und $aa \vdash^* bbbb$ gelten. Also existiert eine Regel $a \rightarrow b^i$ und eine Regel $a \rightarrow a^j$ mit $1 \leq i, j \leq 3$. Damit gilt auch $aa \vdash a^j b^i \notin L_4^+$ für $1 \leq i, j \leq 3$.

0L-Systeme besitzen also keine gescheiterten Abschlusseigenschaften, können aber sogar nicht-kontextfreie Sprachen enthalten, siehe L_4 .

Lemma 3.4.1 *Jede 0L-Sprache ist kontext-sensitiv.*

Beweis. L werde von dem 0L-System (V, R, u) erzeugt, dann auch von der kontext-sensitiven Grammatik

$$(\{S, E, X, Y, Z\}, V, R', S)$$

mit $S, E, X, Y, Z \notin V$ und den Regeln

$$\begin{aligned} S &\rightarrow EuE \\ Ea &\rightarrow XYa \text{ für alle } a \in V \\ Ya &\rightarrow QY \text{ für alle } a \rightarrow Q \in R \\ YE &\rightarrow ZE \\ aZ &\rightarrow Za \text{ für alle } a \in V \\ XZ &\rightarrow E \\ E &\rightarrow \varepsilon \end{aligned}$$

■

Folgende Erweiterungen von L-Systemen sind üblich:

Definition 3.4.1 *Ein extended Lindenmayer System $G = (V, R, u, T)$ ist ein Lindenmayer System (V, R, u) , in dem eine Teilmenge $T \subseteq V$ als terminales Alphabet ausgezeichnet wird. Die von einem extended Lindenmayer System (G, T) generierte Sprache wird jetzt auf $L(G) \cap T^*$ gesetzt. In den Abkürzungen wird ein E hinzugefügt.*

Ein Table Lindenmayer System ist ein Tupel $G = (V, T, u)$, wobei T eine endliche Menge $T = \{R_1, \dots, R_n\}$ von Regelmengen ist, so dass jedes (V, R_i, u) ein Lindenmayer System ist. In einem Schritt $w \vdash w'$ werden indeterminiert Regeln aus nur einer dieser Regelmengen angewendet. In den Abkürzungen wird ein T hinzugefügt.

Ein interaktives Lindenmayer System besteht aus Regeln der Form

$$a_1, a, a_2 \rightarrow Q \text{ mit } a, a_1, a_2 \in V, Q \in V^*.$$

In einer Regelanwendung $w \vdash w'$ wird in w jeder Buchstabe wieder ersetzt, und zwar ein Buchstabe a durch Q , falls a in w von a_1 links und von a_2 rechts umgeben ist und $a_1, a, a_2 \rightarrow Q$ eine Regel ist. Interaktive Lindenmayer Systeme reagieren also kontext-sensitiv. Damit man auch den ersten und letzten Buchstaben eines Wortes ersetzen kann, ist eine Regel $\varepsilon, a, a_2 \rightarrow Q$ erlaubt und genau dann auf a anwendbar, falls links von a keine weiterer Buchstabe steht. Analog für rechts. Manchmal wird

in den Abkürzungen 0 durch 1 ersetzt, manchmal 0 durch 2 und 0 durch 1 , falls alle Regeln ausschließlich auf den linken oder ausschließlich auf den rechten Kontext reagieren.

Eine Greisen-Sprache eines Lindenmayer System G besteht aus allen Wörtern $w \in L(G)$, die sich nicht mehr fortpflanzen können, also mit

$$w \vdash^* v \iff w = v.$$

Ein TDP1L-system wäre also ein deterministisches, propagierendes Table Lindenmayer System, in dem die Regeln kontext-sensitiv sind, aber ausschließlich auf rechte oder ausschließlich auf linke Nachbarn bei der Wahl der Ersetzungsregel reagieren dürfen. Da in einem EL-System $T \subseteq V$ gilt, können die terminalen Buchstaben selbst verändert werden. Manche Definitionen schließen das aber aus, d.h. für die Buchstaben $t \in T$ gibt es nur die Regel $t \rightarrow t$. Beide Ansätze sind äquivalent. Man braucht nur für jedes $t \in T$ eine neue Variable V_t hinzuzunehmen und in jeder Regelkonklusion Q die terminalen Buchstaben t durch V_t zu ersetzen. Hinzu kommen dann noch die Regeln $V_t \rightarrow t$ und $t \rightarrow t$.

Beispiel 3.4.1 Die nicht 0L-Sprache $L'_2 = \{a, a^2\}$ ist eine 1L-Sprache und wird erzeugt von dem Startwort aa aus mit den beiden Regeln $\varepsilon, a, a \rightarrow a$ und $a, a, \varepsilon \rightarrow \varepsilon$.

Einen 1-dimensionalen z.a. mit von Neumann Nachbarschaft kann man als ein spezielles 2L-System auffassen, in dem der Ruhezustand 0 außerhalb eines Wortes zu dem leeren Wort ε und innerhalb eines Wortes zu einem speziellen Zustand 0 übersetzt wird. Jeder Buchstabe wird durch genau einen Buchstaben ersetzt (auch 0 ist dabei erlaubt), wenn er einen linken und rechten Nachbarbuchstaben besitzt, und durch maximal zwei Buchstaben, wenn er keinen linken oder keinen rechten Nachbarbuchstaben besitzt.

Folgende Resultate finden man von unterschiedlichen Forschern in dem Lehrbuch [32] und dem Überblicksartikel [29] von Rozenberg:

Satz 3.4.1 Es gilt

- Die Klasse der 0L-sprachen ist nicht abgeschlossen gegen
 - Vereinigung,
 - Durchschnitt,
 - Durchschnitt mit regulären Sprachen,
 - Komplement,

- *Konkatenation,*
- *Kleene-Stern $*$,*
- *ε -freie Homomorphismen.*
- *Die Klasse der E0L-Sprachen ist abgeschlossen gegen*
 - *Vereinigung,*
 - *Durchschnitt mit regulären Sprachen,*
 - *Konkatenation,*
 - *Kleene-Stern $*$,*
 - *Homomorphismen.*
- *Jede 0L-Sprache ist kontext-sensitiv.*
- *Es existieren 2L-Sprachen, die nicht 1L, und 1L-Sprachen, die nicht 0L sind.*
- *Die kontext-freien Sprachen sind genau die von E0L-Systemen (V, R, u, T) erzeugten Sprachen, in denen zu jedem Buchstaben $a \in V$ eine Regel $a \rightarrow a$ in R vorkommt.*
- *Die kontext-freien Sprachen sind genau die Greisen-Sprachen von 0L-Systemen.*
- *EIL-Sprachen stimmen mit den rekursiv aufzählbaren (= Turing Maschinen akzeptierbaren) Sprachen überein.*
- *Für ET0L-Systeme sind folgende Fragen entscheidbar*
 - *Leerheitsproblem (ist $L(G) = \emptyset$?)*,
 - *Endlichkeitsproblem (ist $L(G)$ endlich?)*,
 - *Wortproblem (liegt w in $L(G)$?)*.
- *Für 0L-Systeme ist das Äquivalenzproblem (gilt $L(G_1) = L(G_2)$?) unentscheidbar.*
- *Für D0L ist das Äquivalenzproblem entscheidbar.*

Das letztgenannte Resultat war lange offen. Culik und Fris [11] konnten in einem sehr aufwändig Beweis zeigen, dass das Homomorphismen-äquivalenzproblem (ist $h_{G_1}^n(u_1) = h_{G_2}^n(u_2)$ für alle n ?) für D0L entscheidbar ist. Schon vorher war bekannt, dass das Äquivalenz und Homomorphismen-äquivalenzproblem für D0L äquivalent sind, siehe Nielsen [20].

Letztlich muss man aber sagen, dass diese sprachtheoretischen Untersuchungen nicht der inhärenten Parallelität in Lindenmayer Systemen gerecht werden. Die biologisch motivierten Fragestellungen passen schon besser.

Übung 3.4.1 • *Geben Sie mehr oder weniger formalisierte Argumente, wie man die Arbeit von Turing Maschinen in Lindenmayer Systemen simulieren kann.*

- *Sind alle von Turing Maschinen akzeptierbare Sprachen bereits ETOL? Oder sind ETOL auch stets kontextsensitiv?*
- *Finden Sie ein (besonders einfaches) Lindenmayer System, das die Sprache $\{a^n b^n c^n \mid n \in \mathbb{N}\}$ erzeugt.*

Kapitel 4

Quantenrechner

4.1 Begriffe

Quantenrechner sind ein aktuelles Forschungsgebiet. Der Quantenrechner wurde als Gedankenexperiment zuerst 1982 vom Physiknobelpreisträger R. Feynman [8] auf der MIT-Tagung “Physics of Computation” vorgestellt. Der Quantenrechner folgt den Gesetzen der Quantenmechanik. Außer absolut trivialen Quantenrechnern, die eigentlich nichts können, ist noch kein Quantenrechner gebaut worden. Es ist auch umstritten, ob ein Quantenrechner überhaupt jemals gebaut werden kann. Um dies wirklich verstehen zu können, ist ein Studium der Physik erforderlich. Selbst ein Mathematikstudium reicht hier nicht, und ein Informatikstudium schon gar nicht. Allerdings kann man sich auf die algorithmischen Aspekte eines Quantenrechners einschränken und sich damit auch als Informatiker beschäftigen. Und dabei ergeben sich überraschende und interessante Resultate. In diese Richtung werden wir uns hier dabei etwas bewegen, aber nur an der Oberfläche bleiben. Dennoch kommen wir nicht umhin, einige Begriffe aus der Physik und Mathematik zu benutzen.

In der klassischen Mechanik ist der Zustand eines Teilchen durch seinen Ort und seinen Geschwindigkeitsvektor eindeutig beschrieben, die beide gemessen werden können. Durch eine Messung kann der Zustand festgestellt werden. Die Genauigkeit der Messung hängt vom Können des Experimentators ab. Mit Heisenbergs Unschärferelation ist das im atomaren Bereich aber prinzipiell anders. Mit den Begriff *Observable* werden die messbaren Eigenschaften von Teilchen, wie etwa deren Ort, Impuls, Drehimpuls, bezeichnet. Das Unschärfepinzips besagt, dass nicht alle Observablen gleichzeitig genau gemessen werden können. Misst man etwa den Ort eines Teilchen exakt, so muss dessen Impuls unbekannt bleiben.

Es sei o ein Quantenobjekt, das in einer Observablen zwei Werte q_0 und q_1 annehmen kann. Ein Beispiel ein solchen *2-Zustand-Systems* ist ein Elektron, dessen Spin den Wert “up” oder “down” annehmen kann, oder die Polarisation eines Photons in senkrechte oder waagerechte Schwingungen. Die beiden messbaren Werte eines 2-Zustand-Systems heißen auch die *Basiszustände* oder *Eigenzustände* und werden üblich mit $|0\rangle$ und $|1\rangle$ bezeichnet. Im Gegensatz zur Makrowelt, wo die Zustände $|0\rangle$ und $|1\rangle$ sich ausschließen, kann in der Quantenwelt o alle Zustände $z_1|0\rangle + z_2|1\rangle$ für beliebige komplexe Zahlen z_1, z_2 mit $|z_1|^2 + |z_2|^2 = 1$ annehmen. Bei einer Messung kann aber im Zustand $z_1|0\rangle + z_2|1\rangle$ nur der Wert $|1\rangle$ oder $|0\rangle$ gemessen werden, und zwar mit der Wahrscheinlichkeit $|z_1|^2$ der Wert $|0\rangle$ und mit der Wahrscheinlichkeit $|z_2|^2$ der Wert $|1\rangle$. Betrachten wir $|0\rangle, |1\rangle$ als Basis, so bewegen wir uns im \mathbb{C}^2 . Und für Zustände muss $|z_1|^2 + |z_2|^2 = 1$ gelten. Ist also in der klassischen Informatik ein Bit ein Wert aus $\{0, 1\}$, so wird in der Quanteninformatik ein *Qubit* (ein Quanten-Bit) also ein zwei-dimensionaler komplexer Vektor der Norm 1.

In einem n -Qubit-System spielen n Qubits zusammen, die 2^n klassische Zustände repräsentieren können. So wie ein Qubit als Einheitsvektor im \mathbb{C}^2 aufgefasst werden kann, werden die Zustände als Einheitsvektoren in einen 2^n -dimensionalen Vektorraum über den komplexen Zahlen aufgefasst. Wir brauchen also einen 2^n -dimensionalen Vektorraum. Dessen Basis sei $\{|i\rangle \mid 1 \leq i \leq 2^n\}$. Ein Zustand im n -Quanten-System hat dann die Form $\sum_{1 \leq i < 2^n} z_i \cdot |i\rangle$ mit $\sum_{0 \leq i < 2^n} |z_i|^2 = 1$. Der Unterschied zur Makrowelt ist nun, dass im n -Qubit-System mehrere Zustände gleichzeitig angenommen und manipuliert werden können. Es existieren sogenannte verschränkte Zustände, die globale Zustände des n -Qubit-System sind, die nicht in einzelne Zustände der beteiligten Qubits zerlegt werden können. Die Manipulation eines einzelnen Qubits in einem verschränkten Zustand ändert nun die Zustände aller Qubits. Damit kann ein Quantenrechner einen Zeitgewinn gegen klassische Rechner aufweisen.

Shor [33] konnte 1994 zeigen, dass auf einem Quantenrechner ein Algorithmus zur Zerlegung von natürlichen Zahlen in deren Primfaktoren mit solch einem Zeitgewinn implementierbar ist, dass die Berechnung der Primfaktoren einer großen, mehrere Hundert Stellen langen natürlichen Zahl in kurzer Zeit möglich ist. Dies gab dem Gebiet einen riesigen Auftrieb aus einem ganz simplen Grund: Mit einem funktionierenden Quantenrechner lassen sich kodierte Daten nach der RSA-Verschlüsselung dekodieren. Diese Verschlüsselung ist nach ihren Entdeckern Rivest, Shamir, Adleman benannt und arbeitet mit einem öffentlichen Schlüssel. Ein öffentlicher Schlüssel ist dabei ein öffentlich bekanntes Produkt p zweier großer Primzahlen p_1, p_2 , wobei die beiden Primzahlen nicht be-

kannt sind. Man kann nun mittels der bekannten Zahl p eine Meldung so verschlüsseln, dass man zur Entschlüsselung die beiden Primzahlen p_1, p_2 benötigt. In der Praxis besitzen p, p_1, p_2 einige Hundert Dezimalstellen und es ist keine Methode bekannt, aus p die Zahlen p_1 und p_2 in sinnvoller Zeit zurück zu berechnen. Ein Quantenrechner aber könnte das. Es sei k eine natürliche Zahl mit n Stellen als Binärzahl. Ein n -Qubit-System könnte dann die Primfaktorzerlegung von k in $p(n)$ vielen Schritten für ein Polynom p berechnen. Physikalisch realisiert werden konnte ein n -Quanten-System bisher nur für $n = 8$ und die Primfaktor von 15 in 3 und 5 gelang damit.

Um etwas mehr zu sagen, als auf dem Niveau dieser kleinen Einleitung, kommen wir aber um etwas Wissen aus der Quantenmechanik nicht herum. Das Verständnis in der Physik ist nicht, wie wir es gerade vereinfacht dargestellt haben, dass man vom Zustand $z_1|0\rangle + z_2|1\rangle$ nur den Wert $|1\rangle$ mit der Wahrscheinlichkeit $|z_1|^2$ oder $|0\rangle$ mit der Wahrscheinlichkeit $|z_2|^2$ messen kann, und zwar den Wert $|0\rangle$ und mit der Wahrscheinlichkeit $|z_2|^2$ den Wert $|1\rangle$, sondern dass die Messung das Quantenobjekt abändert, dass es durch die Messung einen der beiden Eigenzustände $|0\rangle$ oder $|1\rangle$ mit den genannten Wahrscheinlichkeiten annimmt. Es empfiehlt sich, das kleine Büchlein “QED” [28] von Feynman zu lesen. Es sind hier Vorträge über Quantenelektrodynamik nieder geschrieben, die Feynman für Laien hielt. Diese Vorträge sind verständlich und trotzdem auf hohem Niveau.

Die Physiker haben eine sehr gut verstandene und allgemein akzeptierte mathematische Theorie der Quantenelektrodynamik entwickelt. Nach den grundlegenden Arbeiten von John von Neumann ist dies die Theorie der linearen Operatoren auf Hilbert Räumen. Das kann man sogar als Laie leicht nachvollziehen. Das mathematische Objekt, in dem die Physik ihre Theorie entwickelt, ist der Vektorraum. In der Physik sind stetige und differenzierbare Prozesse entscheidend. Also braucht der Vektorraum eine Metrik, um Begriffe der Konvergenz einführen zu können. Quantenzustände besitzen ferner wegen $\sum_{0 \leq i < 2^n} |z_i|^2 = 1$ die Norm 1, also brauchen wir Räume, deren Metrik durch eine Norm definiert wird. Dann wollen wir die normalen geometrischen Eigenschaft der Welt auch im Modell haben. Insbesondere soll das Parallelogrammgesetz der Vektorrechnung gelten. Das bedeutet, dass die Norm durch ein Skalarprodukt definiert sein muss. Ferner soll der Grenzwert einer konvergierenden Folge auch selbst im Raum liegen. Dies bedeutet, dass der Raum topologisch vollständig sein muss. Als mathematisches Modell kommen also nur vollständige Vektorräume über \mathbb{C} mit einem Skalarprodukt in Frage, und das sind genau die Hilbert Räume. Ein Zustand in der Quantenmechanik ist dann ein Einheitsvektor in einem Hilbert Raum. Manipulationen von

Zuständen sollen durch lineare Operationen beschrieben werden. Die linearen Operatoren in Hilbert Räumen, die die Norm 1 nicht verändern, sind aber gut studiert. Es sind genau die sogenannten unitären Operatoren, die im endlich-dimensionalen Fall genau die Multiplikationen mit unitären Matrizen sind. Um verschränkte Zustände modellieren zu können, braucht man ferner das Tensorprodukt von Hilbert Räumen.

Wir werden uns also etwas mit komplexen Zahlen, Hilbert Räumen, linearen Operatoren und dem Tensorprodukt beschäftigen müssen. Ab jetzt wird es unvermeidlicherweise schwer. Hilbert Räume sind mir den mathematischen Kenntnissen des Bachelor-Studiums Informatik oder CV durchaus zu verstehen. Nur, wer sie noch nicht kennt, hat es nicht leicht, sich darüber schlau zu machen. Der Grund ist, dass Hilbert Räume spezielle Banach Räume, diese wieder spezielle Normierte Räume und diese spezielle Vektorräume sind. In Mathematikbüchern stehen im Kapitel Hilbert Räume üblicherweise diese Eigenschaften von Hilbert Räumen, die in Banach Räumen nicht gelten. Im Kapitel über Banach Räume findet man die spezifischen Eigenschaften, die in Normierten Räumen nicht allgemein gelten, und so weiter. Damit muss man, um etwas über Hilbert Räume zu verstehen, auch die Kapitel über Vektorräume, Normierte Räume und Banach Räume lesen und kommt so leicht auf deutlich über 100 Seiten. Daher ist im Anhang, Kapitel III, ein Crash-Kurs über komplexe Zahlen, über die für Quantenrechner wichtigsten Eigenschaften von Hilbert Räumen, ohne zu unterscheiden, ob diese schon in Normierten Räumen gelten oder erst in Hilbert Räumen, und über Tensorprodukte. Der besseren Lesbarkeit halber versuchen wir hier direkt weiter zu machen, und geben nur kurze Erklärungen. Für mathematisch ausgebildete Studierende sollten die Erklärungen ausreichend sein. Ansonsten schlägt man im III nach oder liest es direkt vorher.

4.2 \mathbb{C}^n

Der $\mathbb{C}^n = \{(z_1, \dots, z_n) \mid z_i \in \mathbb{C} \text{ für } 1 \leq i \leq n\}$ ist der Raum aller n -dimensionaler Vektoren über \mathbb{C} . Die Addition zweier Vektoren ist komponentenweise erklärt, die Multiplikation mit einem Skalar auch, also

$$\begin{aligned} z \cdot (z_1, \dots, z_n) &= (z \cdot z_1, \dots, z \cdot z_n), \\ (z_1, \dots, z_n) + (z'_1, \dots, z'_n) &= (z_1 + z'_1, \dots, z_n + z'_n), \text{ und} \\ \langle (z_1, \dots, z_n) \mid (z'_1, \dots, z'_n) \rangle &:= (\bar{z}_1 \cdot z'_1, \dots, \bar{z}_n \cdot z'_n) \\ &= (\bar{z}_1, \dots, \bar{z}_n) \cdot (z'_1, \dots, z'_n)^T \text{ in Matrizenschreibweise,} \end{aligned}$$

ist das Skalarprodukt auf \mathbb{C}^n . \bar{z} ist das Konjugierte von z , also $\overline{a + bi} = a - bi$. In der Physik wird ein Vektor $u \in \mathbb{C}^n$ als ket-Vektor bezeichnet

und als $|u\rangle$ notiert. Ein bra-Vektor $\langle u|$ hingegen ist die lineare Abbildung

$$\langle u| : \mathbb{C}^n \rightarrow \mathbb{C} \text{ mit } \langle u|(|v\rangle) := \langle u|v\rangle.$$

$|u| := \sqrt{\langle u|u\rangle}$ definiert eine Norm auf \mathbb{C}^n und $E_1^n := \{u \in \mathbb{C}^n \mid |u| = 1\}$ ist der n -dimensionale komplexe Einheitskreis im \mathbb{C}^n .

4.3 Qubit

Ein Mathematiker würde einen Qubit vielleicht einfach als den zweidimensionalen komplexen Einheitskreis E_1^2 definieren, und einen Zustand eines Qubits als ein Element von E_1^2 . Die Physik geht anders vor. Qubits kann man in der Physik durch quantenmechanische Zweizustandssysteme realisieren, etwa senkrechte und waagerechte Polarisation eines Photons oder Spin “up” und “down” eines Elektron, etc. $|0\rangle$ und $|1\rangle$ seien die sogenannten *Eigenzustände* des Systems, also Polarisation nur in der Senkrechten bzw. nur in der Waagerechten. Nach der *Superposition* in der Quantenmechanik sind dann aber auch alle überlagerten Zustände

$$|u\rangle = z_1|0\rangle + z_2|1\rangle$$

für komplexe Zahlen z_1, z_2 mit $|z_1|^2 + |z_2|^2 = 1$ möglich. Diese Eigenzustände schließen sich aber bei Messung aus, da sie “senkrecht” aufeinander stehen. Mathematisch gesehen ist die Menge $\{|0\rangle, |1\rangle\}$ einfach eine Orthonormalbasis des \mathbb{C}^2 , d.h. es gilt $\langle u|v\rangle = 1$ für $u = v$ und $\langle u|v\rangle = 0$ für $u \neq v$, jeweils mit $u, v \in \{|0\rangle, |1\rangle\}$. Damit wird folgende formale Definition möglich.

Definition 4.3.1 *Ein 1-Qubit-System \mathcal{H}_2 ist der \mathbb{C}^2 mit einer Orthonormalbasis $\{|0\rangle, |1\rangle\}$.*

$|0\rangle, |1\rangle$ nennt man die Eigenzustände von \mathcal{H}_2 .

Ein Zustand eines 1-Qubit-System ist ein Einheitsvektor im \mathcal{H}_2 .

Ein Qubit ist dann formal einfach ein 1-Qubit-System. Wegen

$$|u| = \sqrt{\langle u|u\rangle} = \sqrt{|z_1|^2 \cdot \langle 0|0\rangle + |z_2|^2 \cdot \langle 1|1\rangle} = \sqrt{|z_1|^2 + |z_2|^2} = |(z_1, z_2)|$$

können wir die Zustände eines Qubits mit den Einheitsvektoren im E_1^2 identifizieren.

Bei einer Messung eines Qubits kann aber nur eine der beiden “Polarisationen” $|0\rangle$ oder $|1\rangle$ als Messwert erhalten. Im Zustand $|u\rangle = z_1|0\rangle + z_2|1\rangle$ liefert eine Messung mit der Wahrscheinlichkeit $|z_1|^2$ den Wert $|0\rangle$

und mit der Wahrscheinlichkeit $|z_2|^2$ den Wert $|1\rangle$. z_i werden die *Amplituden* im Zustand $|u\rangle$ genannt. Wichtig für das Verständnis von Qubits ist, dass eine Messung den Zustand den Qubits verändert. Man kann prinzipiell nicht mehrfach den gleichen unbekanntem Zustand messen, um über die Verteilung der Messergebnisse auf die Amplituden zu schließen. Eine Messung verwandelt den Zustand in einen Basiszustand, der dann allerdings bekannt ist und auch mehrfach gemessen werden kann.

4.4 n -Qubit-System

Für ein physikalisches n -Qubit-System nimmt man als mathematisches Modell den 2^n -dimensionalen Hilbert Raum

$$\mathcal{H}_{2^n} := \mathcal{H}_2 \otimes \dots \otimes \mathcal{H}_2,$$

das n -fache Tensorprodukt des \mathcal{H}_2 , siehe Kapitel 6.10 für eine ausführliche Einführung in das Tensorprodukt. ${}_i\mathcal{H}_2$ bezeichne das i -te Vorkommen von \mathcal{H}_2 in diesem n -fachen Tensorprodukt mit der Orthonormalbasis $|0\rangle_i, |1\rangle_i$. Die 2^n Elemente einer Basis von \mathcal{H}_{2^n} werden mit

$$|e_1 \dots e_n\rangle, |e_1\rangle \otimes \dots \otimes |e_n\rangle, \text{ oder } |e_1\rangle_1 \otimes \dots \otimes |e_n\rangle_n, \text{ für } e_i = 0 \text{ oder } 1$$

bezeichnet oder einfach als

$$|0\rangle, \dots, |2^n - 1\rangle,$$

wobei $|i\rangle$ den Basisvektor $|e_{i_1} \dots e_{i_{2^n}}\rangle$ mit der Binärzahldarstellung $i_1 \dots i_{2^n}$ von i bezeichnet. Das Skalarprodukt zweier Basisvektoren u_i, u_j wird auf $\langle u_i | u_j \rangle = \delta_i(j)$ gesetzt. Für zwei Vektoren $u = \sum_{1 \leq i \leq 2^n} x_i \cdot |e_i\rangle, v = \sum_{1 \leq j \leq 2^n} y_j \cdot |e_j\rangle$ mit $x_i, y_j \in \mathbb{C}$ ist das Tensorprodukt definiert als

$$u \otimes v := \sum_{1 \leq i \leq 2^n} \sum_{1 \leq j \leq 2^n} x_i y_j \cdot |e_i\rangle \otimes |e_j\rangle.$$

Analog wird $V \otimes W$ erklärt, wenn V, W zwei Hilbert Räume sind. Damit gelten folgende Regeln für $u, u_1, u_2 \in V, v, v_1, v_2 \in W, z \in \mathbb{C}$:

$$\begin{aligned} (u_1 + u_2) \otimes v &= u_1 \otimes v + u_2 \otimes v, \\ u \otimes (v_1 + v_2) &= u \otimes v_1 + u \otimes v_2, \\ (zu) \otimes v &= z \cdot (u \otimes v) \\ &= u \otimes (zv), \end{aligned}$$

und \otimes verhält sich wie ein Produkt. Allerdings gilt kein Kommutativgesetz.

Ein Zustand im \mathcal{H}_{2^n} ist nun jeder Vektor der Form

$$u = \sum_{1 \leq i \leq 2^n} z_i \cdot |e_i\rangle, \text{ mit } \sum_{1 \leq i \leq 2^n} |z_i|^2 = 1,$$

also mit $|u| = 1$. Die Zustände sind also gerade die Einheitsvektoren auf dem Einheitskreis $\{u \in \mathcal{H}_{2^n} \mid |u| = 1\}$ des \mathcal{H}_{2^n} , den wir mit $E_1^{2^n}$ identifizieren können. Zur Verwirrung trägt bei, dass auch das die sogenannte *Dichtematrix*, das ist das Funktional $P[u] = |u\rangle\langle u|$ mit $P[u](v) = |u\rangle\langle u|(|v\rangle) = |u\rangle\langle u|v\rangle = \langle u|v\rangle \cdot |u\rangle$ anstelle von u als Zustand bezeichnet wird (für $|u| = 1$). Zustände sind dann keine Einheitsvektoren in einem Hilbert Raum sondern Funktionale mit speziellen Eigenschaften. Wir bleiben hier aber bei der Auffassung von Zuständen als Einheitsvektoren. Damit haben wir folgende formale Definition

Definition 4.4.1 *Ein n -Qubit-System ist der \mathcal{H}_{2^n} .*

Ein Zustand eines n -Qubit-System ist ein Einheitsvektor im \mathcal{H}_{2^n} .

4.5 Verschränkte Zustände

Entscheidend für einen Quantenrechner sind *verschränkte* oder im Englischen *entangled* Zustände. Dies sind Zustände eines n -Qubit-Systems, die sich nicht als n -faches Tensorprodukt aus Vektoren des \mathcal{H}_2 schreiben lassen. D.h., in denen einem einzelnen Qubit kein Zustand zugewiesen werden kann, sondern nur dem Gesamtsystem.

Definition 4.5.1 *Ein verschränkter Zustand eines n -Qubit-Systems ist ein Einheitsvektor im \mathcal{H}_{2^n} , so dass keine n Einheitsvektoren u_i im \mathcal{H}_2 existieren mit $u = u_1 \otimes \dots \otimes u_n$.*

Beispiel 4.5.1 *Verschränkte und nicht-verschränkte Zustände.*

$|u\rangle = \sqrt{\frac{1}{3}} |0\rangle - \sqrt{\frac{2}{3}} i |1\rangle$ und $|v\rangle = \sqrt{\frac{1}{4}} |0\rangle + \sqrt{\frac{3}{4}} |1\rangle$ sind Zustände im \mathcal{H}_2 . Damit gilt

$$\begin{aligned} |u\rangle \otimes |v\rangle &= \left(\sqrt{\frac{1}{3}} |0\rangle_1 - \sqrt{\frac{2}{3}} i |1\rangle_1 \right) \otimes \left(\sqrt{\frac{1}{4}} |0\rangle_2 + \sqrt{\frac{3}{4}} |1\rangle_2 \right) \\ &= \sqrt{\frac{1}{12}} |00\rangle + \sqrt{\frac{3}{12}} |01\rangle - \sqrt{\frac{2}{12}} i |10\rangle - \sqrt{\frac{6}{12}} i |11\rangle \end{aligned}$$

und auch $|u\rangle \otimes |v\rangle$ ist ein (nicht-verschränkter) Zustand. Da das Tensorprodukt zweier Einheitsvektoren ein Einheitsvektor ist, bildet es Zustände wieder auf Zustände ab.

$|w\rangle = \sqrt{\frac{1}{2}}|00\rangle + \sqrt{\frac{1}{2}}|11\rangle$ besitzt die Länge 1 und ist damit ein Zustand im $\mathcal{H}_4 = \mathcal{H}_2 \otimes \mathcal{H}_2$. $|w\rangle$ ist aber verschränkt, da sich $|w\rangle$ nicht als Tensorprodukt zweier Zustände des \mathcal{H}_2 ergibt, wie man leicht nachrechnet: Annahme es gilt $|w\rangle = |u_1\rangle \otimes |u_2\rangle$ mit $|u_i\rangle = z_{i,0}|0\rangle + z_{i,1}|1\rangle$. Dann folgt

$$\begin{aligned} |w\rangle = |u_1\rangle \otimes |u_2\rangle &= z_{1,0}z_{2,0}|00\rangle + z_{1,0}z_{2,1}|01\rangle + z_{1,1}z_{2,0}|10\rangle + z_{1,1}z_{2,1}|11\rangle \\ &= \sqrt{\frac{1}{2}}|00\rangle + \sqrt{\frac{1}{2}}|11\rangle. \end{aligned}$$

Das führt zu folgenden offensichtlichen Widerspruch

$$z_{1,0}z_{2,1} = 0 \text{ und } z_{1,0}z_{2,0} = \sqrt{\frac{1}{2}} = z_{1,1}z_{2,1}.$$

Ebenfalls verschränkt ist $\sqrt{\frac{1}{2}}|01\rangle + \sqrt{\frac{1}{2}}|10\rangle$.

Verschränkte Zustände besitzen interessante physikalische Eigenschaften. Messen wir z.B. im obigen Zustand w das erste Qubit und erhalten dabei mit Wahrscheinlichkeit 0.5 den Basiszustand $|0\rangle$, so muss das nicht gemessene zweite Qubit auch in den Zustand $|0\rangle$ gehen, da ein gemischter Zustandsanteil $z|01\rangle$ nicht vorkommt. So kann man zwei in der Polarisation verschränkte Photonen erzeugen und in unterschiedliche Richtung fliegen lassen. Misst man nun von einem der beiden die Polarisation, muss das andere auch die gemessene Polarisation annehmen, egal wie weit es sich entfernt hat. Das ist das Prinzip der *Nichtlokalität* verschränkter Zustände. Manipuliert man ein Qubit eines n -Qubit-Systems in einem verschränkten Zustand, werden die anderen $n-1$ Qubits mit manipuliert. Wenn man eine Quantenrechner bauen könnte, der mit verschränkten Zuständen rechnet, kann man durch Manipulation eines Qubits andere Qubits mit manipulieren, eine offensichtlich hochinteressante Fähigkeit, die Nicht-QuantenRechner nicht besitzen.

Beispiel 4.5.2 Was ist mit den Zustand $\sqrt{\frac{1}{2}}|00\rangle + \sqrt{\frac{1}{2}}|01\rangle$? Er ist gleich $|0\rangle_1 \otimes \sqrt{\frac{1}{2}}|0\rangle_2 + \sqrt{\frac{1}{2}}|1\rangle_2$ und somit nicht verschränkt. Eine Messung des ersten Qubits liefert immer den Wert 0 ohne einen Wert für das zweite festzulegen.

4.6 Evolution in n -Qubit-Systemen

Unter einer Evolution in einem n -Qubit-Systemen versteht man eine lineare Transformation eines Zustandes in einen anderen. Eine lineare

Abbildung eines Vektorraumes in sich wird als *Operator* bezeichnet. Da Zustände Einheitsvektoren im \mathcal{H}_{2^n} sind, muss dieser Operator Einheitsvektoren in Einheitsvektoren abbilden. Für Operatoren T , die Einheitsvektoren auf Einheitsvektoren abbilden, gilt $|\frac{1}{|u|}| \cdot |Tu| = |T(\frac{1}{|u|}u)| = 1$, also $|Tu| = |u|$, und der Operator ist Norm erhaltend und muss damit unitär sein.

Wenn wir die Evolution als ein abstraktes Rechenmodell beschreiben wollen, wird ein n -Qubit-System S einfach eine m^2 -dimensionale unitäre Matrix A mit $m = 2^n$. Genauer: Zu S existiert eine unitäre Matrix A , so dass wir S als das ARM $S = (\mathcal{C}, \vdash)$ auffassen mit

- $\mathcal{C} = E_1^{2^n}$, der Raum aller Vektoren der Norm 1 im \mathcal{H}_{2^n} ,
- $u \vdash v : \iff v = Au$.

Da A unitär ist, ist A auch invertierbar und S ist reversibel. Die Erreichbarkeitsmenge von einem Zustand u ist einfach $\mathcal{E}(u) = \{A^t u \mid t \in \mathbb{N}\}$.

4.7 Das Cloning Theorem

Das Cloning Theorem, oder besser gesagt Nicht-Cloning Theorem, besagt, dass ein unbekannter Zustand nicht verdoppelt werden kann. Es existiert also keine unitäre Transformation, die zu jedem Zustand $|u\rangle$ den Zustand $|u0\rangle$ in $|uu\rangle$ überführt. Das beweist man wie folgt. Annahme es existiere eine unitäre Transformation U mit $U|u0\rangle = |uu\rangle$. Es seien $|v\rangle, |v'\rangle$ zwei linear unabhängige, orthogonale Zustände in \mathcal{H}_2 und $|u\rangle := \sqrt{\frac{1}{2}}(|v\rangle + |v'\rangle)$. Damit gilt

$U|v0\rangle = |vv\rangle$, $U|v'0\rangle = |v'v'\rangle$, $U|u0\rangle = |uu\rangle$ und damit auch

$$\begin{aligned} \frac{1}{4}(|vv\rangle + |vv'\rangle + |v'v\rangle + |v'v'\rangle) &= |u\rangle \otimes |u\rangle = |uu\rangle = U|u0\rangle \\ &= U \sqrt{\frac{1}{2}}(|v0\rangle + |v'0\rangle) \\ &= \sqrt{\frac{1}{2}}(U|v0\rangle + U|v'0\rangle) \\ &= \sqrt{\frac{1}{2}}(|vv\rangle + |v'v'\rangle), \end{aligned}$$

und wir haben $|uu\rangle$ auf zwei verschiedene Weisen als Linearkombination linear unabhängiger Zustände dargestellt, ein Widerspruch.

Andererseits existiert eine unitäre Transformation U mit $U|u0\dots0\rangle = z_1|00\dots0\rangle + z_2|11\dots1\rangle$ für $u = z_1|0\rangle + z_2|1\rangle$. Das Nicht-Cloning Theorem verbietet nur eine unitäre Transformation, die $|u0\dots0\rangle$ in $|u\rangle \otimes \dots \otimes |u\rangle$ überführt.

4.8 Systeme mit unbeschränkt vielen Qubits

Es ist umstritten, ob Systeme mit unbeschränkt vielen Qubits überhaupt physikalisch sinnvoll sein können, da fraglich ist, ob Superposition mehr als ein lokales Phänomen sein kann. Als Modell eines Systems mit unbeschränkt vielen Qubits wird der ℓ^2 gewählt. Dabei ist

$$\ell^2 = \{(u_n)_{n \in \mathbb{N}} \mid u_n \in \mathbb{C} \wedge \sum_{n \in \mathbb{N}} |u_n|^2 \text{ konvergiert} \} \text{ mit}$$

$$\langle u|v \rangle = \sum_{n \in \mathbb{N}} \overline{u_n} v_n \text{ für } u, v \in \ell^2$$

ein unendlich dimensionaler Hilbertraum mit einer Basis $\{\delta_i \mid i \in \mathbb{N}\}$, wobei die Folge (oder Funktion) δ_i an i -ter Stelle eine 1 und sonst überall eine 0 enthält. Wegen $|\overline{u_n} v_n| \leq 0,5 \cdot (|u_n|^2 + |v_n|^2)$ konvergiert das innere Produkt.

Zustände sind nun die Einheitsvektoren im unendlich-dimensionalen Hilbert Raum ℓ^2 . Eine Zeitevolution $u \mapsto v$ ist dann gegeben durch Anwendung eines *unitären Operators* U , d.h. $v = Uu$. Ein Operator heißt unitär, falls er invertierbar ist und $U^{-1} = U^*$ gilt mit dem zu U adjungierten Operator U^* . Das bedeutet, dass für alle $u, v \in \ell^2$ gelten muss

$$\langle Uu|v \rangle = \langle u|U^{-1}v \rangle.$$

4.9 Quantenschaltnetze

Eine Manipulation eines n -Qubit-System ist eine Multiplikation im \mathcal{H}_{2^n} mit einer 2^{2^n} -dimensionalen unitären $2^n \times 2^n$ -Matrix. Da dies ein recht hochdimensionaler Raum ist, empfiehlt es sich, diese Matrizen in kleinere zu zerlegen. das führt zum Modell des *Quantenschaltnetzes*. Jedes der n Qubits erhält eine Leitung. Manche der Leitungen führen in eine Black Box, aus der die gleiche Anzahl von Leitungen wieder herausführt. Die restlichen Leitungen passieren neben dieser Box. Es können auch mehrere Boxen übereinander in einem Abschnitt liegen. Solche Abschnitte von jeweils n Leitungen mit eventuellen Boxen werden ohne Rückkopplung

hintereinander geschaltet. Für ein Beispielbild eines Quantenschaltnetzes, siehe

http://en.wikipedia.org/wiki/Quantum_circuit

Besitzt eine Black Box $k < n$ Leitungen als Ein- und Ausgänge, so repräsentiert sie eine Operation auf nur k Qubits, also eine 2^{2k} -dimensionale unitäre Matrix. Nehmen wir einen Abschnitt, in dem eine Box mit 2 Ein- und Ausgängen über 4 Leitungen liegt. In diesem 6-Qubit-System operiert damit eine unitäre $2^2 \times 2^2$ -Matrix A_2 auf den oberen beiden Qubits, die vier anderen bleiben unverändert, d.h. auf ihnen operiert die $2^4 \times 2^4$ -Einheitsmatrix $\mathbf{1}_4$, die in der Diagonalen 1 enthält und 0 sonst. Die unitäre $2^6 \times 2^6$ -Matrix, die auf diesem Abschnitt operiert ist dann $A_2 \otimes \mathbf{1}_4$. Würde auf den unteren vier Qubits eine unitäre $2^4 \times 2^4$ -Matrix B_4 , so operiert auf diesem Abschnitt die unitäre Matrix $A_2 \otimes B_4$. Zu bedenken ist, dass das Tensorprodukt zweier Matrizen die Dimension des Produktes beider Dimensionen hat. Zur Definition des Tensorproduktes oder Kroneckerprodukt zweier Matrizen siehe Abschnitt 6.10 oder

<http://de.wikipedia.org/wiki/Kronecker-Produkt>

Ist hinter dem Abschnitt, auf dem $X_1 = A_2 \otimes B_4$ operiert, ein weiterer Abschnitt, auf dem etwa $X_2 = C_3 \otimes D_3$ operiert, so operiert also zuerst X_1 und dann X_2 , d.h. insgesamt operiert die unitäre $2^6 \times 2^6$ -Matrix $X_2 \cdot X_1 = (C_3 \otimes D_3) \cdot (A_2 \otimes B_4)$. Sie liefert im Quantenschaltnetz zwar auch nur eine große unitäre Matrix, aber zerlegt in Produkte von Tensorprodukten von einfacheren unitären Matrizen. Die kleineren unitären Matrizen, die Black Boxes, sind dann die *Quantengates* des Quantenschaltnetzes. Es gilt nun, dass jeder unitäre Operator im \mathcal{H}_{2^n} als Tensorprodukt von 2×2 -Matrizen geschrieben werden kann.

Betrachten wir Beispiele von Quantengates, die auf einem Qubit operieren.

Das *Nicht-Gate* N ist die unitäre Matrix $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ und operiert auf einem Zustand $u = \alpha|0\rangle + \beta|1\rangle$ als

$$Nu = N(\alpha|0\rangle + \beta|1\rangle) = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \beta \\ \alpha \end{pmatrix} = \beta|0\rangle + \alpha|1\rangle.$$

Das Gate \sqrt{N} ist $1/2 \begin{pmatrix} 1-i & 1+i \\ 1+i & 1-i \end{pmatrix}$,

und das *Walsh-Gate* W_2 ist $1/\sqrt{2} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$.

Es gilt

$$\begin{aligned}\sqrt{N} \sqrt{N} u &= \frac{1}{2} \begin{pmatrix} 1-i & 1+i \\ 1+i & 1-i \end{pmatrix} \cdot \frac{1}{2} \begin{pmatrix} 1-i & 1+i \\ 1+i & 1-i \end{pmatrix} u \\ &= \frac{1}{4} \begin{pmatrix} 0 & 4 \\ 4 & 0 \end{pmatrix} u \\ &= N u\end{aligned}$$

Für den Zustand $u_0 = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ gilt:

$$\begin{aligned}W_2 u_0 &= W_2 \left(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \right) = \frac{1}{\sqrt{2}} W_2 |0\rangle + \frac{1}{\sqrt{2}} W_2 |1\rangle \\ &= \frac{1}{\sqrt{2}} \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle) + \frac{1}{\sqrt{2}} \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle) = |0\rangle.\end{aligned}$$

Oder in Matrixschreibweise

$$W_2 u_0 = \frac{1}{\sqrt{2}} \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = |0\rangle.$$

Ein Quantenschaltnetz im \mathcal{H}_4 operiert auf einem 2-Qubit-System und besitzt entsprechend 2 Leitungen. Es sei G ein Gate $\begin{pmatrix} c_{00} & c_{01} \\ c_{10} & c_{11} \end{pmatrix}$ in einem 1-Qubit-System.

$$G^c = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & c_{00} & c_{01} \\ 0 & 0 & c_{10} & c_{11} \end{pmatrix} \text{ ist dann } G \text{ kontrolliert von einem anderen}$$

Qubit. Die Wirkung von G^c ist

$$\begin{aligned}|00\rangle &\rightarrow |00\rangle, \\ |01\rangle &\rightarrow |01\rangle, \\ |10\rangle &\rightarrow |1\rangle U|0\rangle = |1\rangle \otimes (c_{00}|0\rangle + c_{10}|1\rangle) = c_{00}|10\rangle + c_{10}|11\rangle, \\ |11\rangle &\rightarrow |1\rangle U|1\rangle = |1\rangle \otimes (c_{01}|0\rangle + c_{11}|1\rangle) = c_{01}|10\rangle + c_{11}|11\rangle.\end{aligned}$$

Schauen wir uns das kontrollierte Nicht-Gate $CN = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$ ge-

nauer an. Es operiert auf dem \mathcal{H}_4 mit den Basisvektoren

$$\begin{aligned}|0\rangle &= |00\rangle = (|0\rangle)_1 \otimes (|0\rangle)_2, \\ |1\rangle &= |01\rangle = (|0\rangle)_1 \otimes (|1\rangle)_2, \\ |2\rangle &= |10\rangle = (|1\rangle)_1 \otimes (|0\rangle)_2, \\ |3\rangle &= |11\rangle = (|1\rangle)_1 \otimes (|1\rangle)_2.\end{aligned}$$

Das erste Qubit formt die erste, das zweite die zweite Leitung. Im Basiszustand $|0\rangle$ der ersten Leitung (Signal 0 liegt an), soll die zweite Leitung durchschalten, bei Signal 1 auf Leitung 1, also im Basiszustand $|1\rangle$ des ersten Qubits, soll das Nicht-Gate auf der zweiten Leitung aktiv sein. Wir wünschen also folgendes Verhalten

$$\begin{aligned} |00\rangle &\rightarrow |00\rangle \\ |01\rangle &\rightarrow |01\rangle \\ |10\rangle &\rightarrow |11\rangle \\ |11\rangle &\rightarrow |10\rangle, \end{aligned}$$

also gerade $|0\rangle \rightarrow |0\rangle$, $|1\rangle \rightarrow |1\rangle$, $|2\rangle \rightarrow |3\rangle$, $|3\rangle \rightarrow |2\rangle$. Oder in Vektorschreibweise,

$$\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \rightarrow \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \rightarrow \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \quad \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \rightarrow \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}, \quad \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}. \quad \text{Genau}$$

dies leistet die Multiplikation mit der Matrix CN .

Betrachten wir weitere Quantennetzwerke mit 2 Leitungen und dem Nicht-Gate N . Q_1 soll auf Leitung 1 Signale weiterleiten und auf Leitung 2 als Nicht-Gate operieren, Q_2 soll auf beiden Leitungen als Nicht-Gate operieren. Damit erhalten Q_1 und Q_2 folgende Matrizen:

$$\begin{aligned} Q_1 &= \mathbf{1}_1 \otimes N = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \\ Q_2 &= N \otimes N = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \end{aligned}$$

Die Wirkungen beider Netzwerke sind

$$\begin{aligned} Q_1 = \mathbf{1}_1 \otimes N &: |0\rangle \rightarrow |1\rangle, |1\rangle \rightarrow |0\rangle, |2\rangle \rightarrow |3\rangle, |3\rangle \rightarrow |2\rangle, \\ Q_2 = N \otimes N &: |0\rangle \rightarrow |3\rangle, |1\rangle \rightarrow |2\rangle, |2\rangle \rightarrow |1\rangle, |3\rangle \rightarrow |0\rangle, \end{aligned}$$

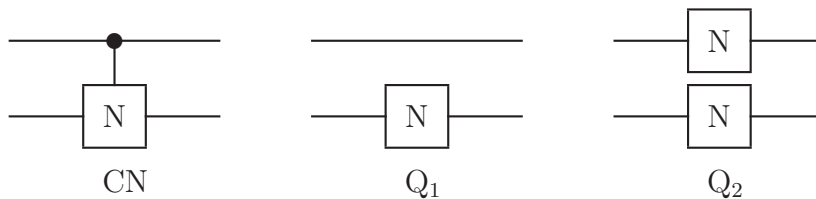
und man sieht erst dann etwas, wenn man nicht diese Basisdarstellung wählt, sondern

$$Q_1 = \mathbf{1}_1 \otimes N : |00\rangle \rightarrow |01\rangle, |01\rangle \rightarrow |00\rangle, |10\rangle \rightarrow |11\rangle, |11\rangle \rightarrow |10\rangle,$$

(4.2)

$$Q_2 = N \otimes N : |00\rangle \rightarrow |11\rangle, |01\rangle \rightarrow |10\rangle, |10\rangle \rightarrow |01\rangle, |11\rangle \rightarrow |00\rangle.$$

In Q_1 beeinflusst das Signal auf Leitung 1 nicht die Arbeit von N auf Leitung 2; in Q_2 operiert je ein N auf beiden Leitungen. In einer graphischen Darstellung sehen die drei Quantenschaltnetze CN , Q_1 und Q_2 wie folgt aus:



Interessant ist die Frage, ob es *universelle* Quantengates gibt, aus denen sich jeder Quantenschaltnetz zusammensetzen lässt (besser gesagt: beliebig genau approximieren lässt, da es überabzählbare viele unitäre Matrizen über \mathbb{C} gibt; \mathbb{C} selbst ist überabzählbar und jede Drehung auf dem Einheitskreis ist eine unitäre Abbildung.) Die Antwort ist “ja”, es gibt mehrere Quantengates, die jedes allein schon universell sind. Ein solches universelles Gate ist das *Deutsch-Gate*, das von folgender Matrix beschrieben wird, wobei θ/π irrational sein muss. Es beschreibt ein 3-Qubit-System und besitzt damit drei Leitungen.

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & i \cos \theta & \sin \theta \\ 0 & 0 & 0 & 0 & 0 & 0 & \sin \theta & i \cos \theta \end{pmatrix}$$

Faszinierend ist natürlich, dass ein Eingriff auf nur ein Qubit, d.h. eine Manipulation an nur einer Leitung, alle anderen Qubits mit ändern kann, wenn das System in einen verschränkten Zustand ist. Ein Quantenschaltnetz wird damit zu einem Algorithmus. Selbst wenn es noch keine physikalische Realisierung für $n > 8$ gibt, kann man dennoch schon Quantenalgorithmen entwickeln.

4.10 Reversible klassische Schaltnetze

In diesem Abschnitt stellen wir das Fredkin-Gate vor und zeigen wir mit den Mitteln der Informatik, dass es universell ist im Sinn der Informatik: jedes Schaltelement läßt sich damit simulieren. Da alle Rechnungen in Quantenschaltnetzen reversibel sind, führen wir zuerst den Begriff des reversiblen Schatwerks ein. Hierbei bewegen wir uns nicht in der Quantenwelt, sondern in der ganz normalen Informatik. Schaltwerke und Gats sind hier klassisch zu verstehen: sie manipulieren Bits 0 und 1 auf den Leitungen und nicht Quantenzustände.

Zur Erinnerung, ein ARM (\mathcal{C}, \vdash) heißt reversibel, wenn $\vdash \subseteq \mathcal{C} \times \mathcal{C}$ determiniert und rückwärts determiniert ist, d.h. wenn jede Konfiguration maximal eine Nachfolger- und Vorgänger-Konfiguration besitzt. In der Physik würde man statt “maximal eine” genau eine Nachfolger- und Vorgänger-Konfiguration fordern und \vdash wird dann zu einer bijektiven Funktion auf \mathcal{C} . Jetzt geht es um *Schaltelemente* wie NICHT oder UND, die zu *Schaltwerken* verbunden sind. Allerdings wollen wir reversible Schaltelemente untersuchen, bei denen man aus der Ausgabe immer eindeutig auf die Eingabe zurückschließen kann. Diese verbinden wir zu reversiblen Schaltwerken, die endlich berechnungsuniversell sind – das heißt, dass man mit ihnen jedes Schaltelement simulieren kann.

Ein *Schaltelement* mit m Eingängen und n Ausgängen ist eine Funktion $f : \{0, 1\}^m \rightarrow \{0, 1\}^n$ für $m, n \in \mathbb{N}$. Gilt $n = 1$, dann sprechen wir auch von einer *Boole’schen Funktion*. Beispiele von Boole’schen Funktionen sind etwa

$$\begin{aligned} \text{nicht} : \{0, 1\} &\rightarrow \{0, 1\} && \text{mit } \text{nicht}(x) = \bar{x} = 1 - x \\ \text{und} : \{0, 1\}^2 &\rightarrow \{0, 1\} && \text{mit } \text{und}(x, y) = x \cdot y \\ \text{oder} : \{0, 1\}^2 &\rightarrow \{0, 1\} && \text{mit } \text{oder}(x, y) = \overline{\bar{x} \cdot \bar{y}} = 1 - (1 - x)(1 - y) \\ \text{nand} : \{0, 1\}^2 &\rightarrow \{0, 1\} && \text{mit } \text{nand}(x, y) = 1 - x \cdot y \\ &&& \text{mit} \end{aligned}$$

jeweils für $x \in \{0, 1\}$. \bar{x} steht hier für $1 - x$ und darf nicht mit dem konjugiert Komplexen verwechselt werden. Ein *kombinatorisches Schaltwerk* ist ein rückkopplungsfreies Netzwerk mit Boole’schen Funktion oder „Lötstellen“ als Knoten. Auf jeder Leitung liegt ein Signal 0 oder 1 an. Die Signale auf der Eingabeseite der Boole’schen Funktionen werden entweder verzögerungsfrei weitergeleitet – d.h. in dem Moment, wo die Eingabe am Schaltwerk anliegt, liegt auch schon das Ausgabesignal auf dem einzigen Ausgang – oder synchron getaktet mit einer Einheitsverzögerung – d.h. alle Schaltwerke des Netzes schalten jeweils gleichzeitig, für die Eingabe im Takt t liegt im Takt $t + 1$ die Ausgabe vor.

Es ist nicht schwer zu zeigen, daß man ein beliebiges Schaltelement durch ein kombinatorisches Schaltnetz über NICHT und UND (bzw. zusätzlich DELAY und FAN-OUT, je nach Verzögerungsmodell) realisieren kann: Das Ausgabe-Verhalten jedes einzelnen Ausgangs läßt sich durch eine aussagenlogischen Formel über alle Eingangsleitungen beschreiben. Und jede AL-Formel besitzt eine äquivalente disjunktive Normalform (DNF), in der man dann $x \vee y$ durch $\neg(\neg x \wedge \neg y)$ weiter ersetzen kann. Das kombinatorische Schaltnetz in Abb. 4.2 folgt genau diesem Konstruktionsprinzip. Abb. 4.1 zeigt das Schaltelement $cn : \{0, 1\}^2 \rightarrow \{0, 1\}^2$ mit $cn(c, x) = (c, \bar{c}x + c\bar{x})$ und ein kombinatorisches Schaltnetz über NICHT, UND und ODER, das CN realisiert. In diesem Schaltnetz sollen die Boole'schen Funktionen verzögerungsfrei arbeiten. cn ist hier die klassische Variante des kontrollierten Nicht-Gates CN aus dem letzten Kapitel.

CN:

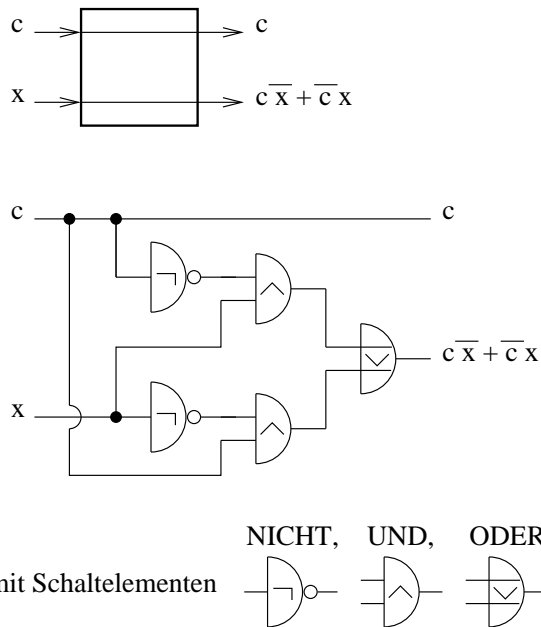


Abbildung 4.1: Beispiel eines verzögerungsfreien kombinatorischen Schaltnetzes

Wenn wir stattdessen in einem Modell arbeiten, in dem alle Schaltelemente eine Einheitsverzögerung haben, kann das Netz aus Abb. 4.1 nicht mehr CN realisieren, da die Ausgabe auf c sofort, die bei $\bar{c}x + c\bar{x}$ erst nach drei Verzögerungsschritten anliegt. In diesem Fall braucht man Verzögerungselemente, zum Beispiel $delay : \{0, 1\} \rightarrow \{0, 1\}$ mit $delay(x) = x$. Allerdings gibt es von dem Modell mit Einheitsverzögerung zwei Varianten: Entweder die „Lötstellen“ von Leitungen sind Schaltelemente und unterliegen der Einheitsverzögerung, oder sie arbeiten verzögerungs-

frei. In der ersten Variante wird eine Lötstelle als ein Schaltelement $fan-out : \{0, 1\} \rightarrow \{0, 1\}^2$ mit $fan-out(x) = (x, x)$ modelliert. In diesem Modell kombinatorischer Schaltnetze lassen sich dann alle Schaltelemente durch Schaltnetze aus den atomaren Schaltelementen NICHT, UND, DELAY und FAN-OUT realisieren. Abb. 4.2 zeigt ein Schaltnetz mit Einheitsverzögerung, die das Schaltelement CN realisiert, unter Verwendung von FAN-OUT und DELAY. Dieses Schaltnetz hat eine Gesamtverzögerung von 5 Takten von der Eingabe in CN bis zur Reaktion auf der Ausgabe-seite.

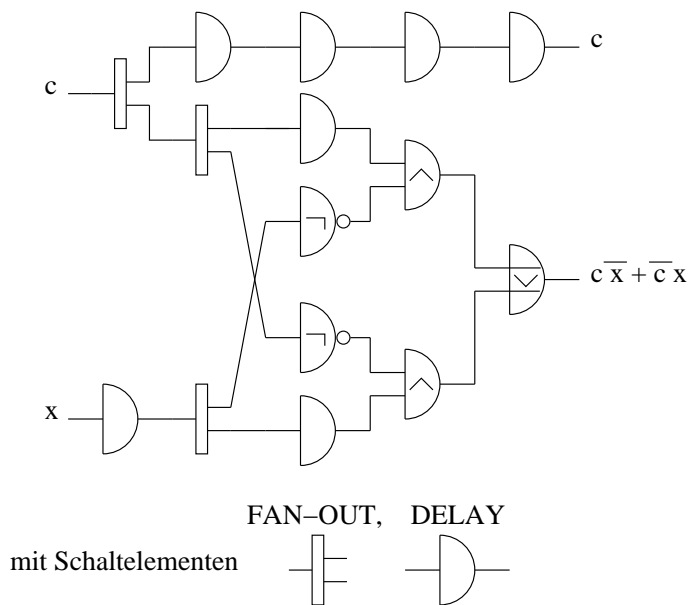


Abbildung 4.2: Das Schaltelement CN aus Abb. 4.1, realisiert durch ein getaktetes kombinatorisches Schaltnetz mit Einheitsverzögerung auf allen Schaltelementen

Der Begriff der Reversibilität überträgt sich kanonisch auf Schaltelemente. Sie sind als Funktionen grundsätzlich vorwärts determiniert. Rückwärts determiniert sind sie genau dann, wenn sie injektiv sind. Von den bereits vorgestellten Schaltelementen sind NICHT, DELAY und FAN-OUT reversibel, aber UND ist es nicht.

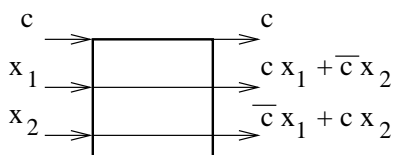


Abbildung 4.3: Ein Fredkin-Gate

Das reversible Schaltelement *Fredkin-Gate* FG in Abb. 4.3 ist definiert als $fg : \{0, 1\}^3 \rightarrow \{0, 1\}^3$ mit

$$fg(c, x_1, x_2) = (c, cx_1 + \bar{c}x_2, \bar{c}x_1 + cx_2) \text{ f\u00fcr } c, x_1, x_2 \in \{0, 1\}.$$

Wie alle Schaltnetze ist das Fredkin-Gate per Definition vorw\u00e4rts determiniert. Und es ist auch r\u00fcckw\u00e4rts determiniert: Wenn man die Ausgabesignale eines Fredkin-Gates noch einmal in ein Fredkin-Gate eingibt, erh\u00e4lt man

$$\begin{aligned} fg(c, cx_1 + \bar{c}x_2, \bar{c}x_1 + cx_2) &= \\ (c, c(cx_1 + \bar{c}x_2) + \bar{c}(\bar{c}x_1 + cx_2), \bar{c}(cx_1 + \bar{c}x_2) + c(\bar{c}x_1 + cx_2)) &= \\ (c, cx_1 + \bar{c}x_1, cx_2 + \bar{c}x_2) &= (c, x_1, x_2) \end{aligned}$$

Also ist $fg \circ fg = id$. Das hei\u00dft, ein Fredkin-Gate FG ist gleich seinem Inversen. Damit ist es r\u00fcckw\u00e4rts determiniert und somit auch reversibel.

Mit nur einem Fredkin-Gate kann man ein UND-, NICHT-, DELAY- oder FAN-OUT-Element realisieren, wie Abb. 4.4 zeigt. Damit haben wir insgesamt bewiesen:

Satz 4.10.1 *Reversible kombinatorische Schaltnetze aus Fredkin-Gates sind endlich berechnungsuniversell.*

Satz 4.10.1 gilt f\u00fcr beide Varianten von Schaltnetzen, sowohl f\u00fcr das Modell, in dem alle Schaltelemente bis auf DELAY verz\u00f6gerungsfrei schalten, als auch f\u00fcr das Modell mit Einheitsverz\u00f6gerung in allen Schaltelementen. Um UND-, NICHT-, DELAY- und FAN-OUT-Elemente zu simulieren, braucht das Fredkin-Gate zus\u00e4tzlich 0- und 1-Signale als Eingabe. Es liefert zus\u00e4tzliche Ausgaben, die f\u00fcr die physikalische Reversibilit\u00e4t sorgen: Sie spiegeln wider, welche Eingangssignalverteilung die Ausgangssignalverteilung bewirkt hat. Man k\u00f6nnte auch sagen, die zus\u00e4tzlichen Ausgaben liefern ein PROTOKOLL der Schaltung. Die zus\u00e4tzlichen Eingaben k\u00f6nnen wir als ENERGIE auffassen, die f\u00fcr die Rechnung gebraucht wird. Es ergibt sich damit die Situation aus Abb. 4.5.

Betrachten wir das Schaltnetz $\hat{E} : \{0, 1\}^2 \rightarrow \{0, 1\}^3$ mit $\hat{E}(s, t) = (s, st, \bar{s}t)$ f\u00fcr $s, t \in \{0, 1\}$. Damit haben wir

$$\begin{aligned} \hat{E}(0, 0) &= (0, 0, 0) & \hat{E}(0, 1) &= (0, 0, 1) \\ \hat{E}(1, 0) &= (1, 0, 0) & \hat{E}(1, 1) &= (1, 1, 0) \end{aligned}$$

Also ist \hat{E} injektiv mit Wertebereich $\{(0, 0, 0), (0, 0, 1), (1, 0, 0), (1, 1, 0)\}$, also vorw\u00e4rts und r\u00fcckw\u00e4rts determiniert und somit auch reversibel. Das inverse Schaltelement zu \hat{E} ist \hat{E}^{-1} , ein Element mit partieller \u00dcbergangsfunktion. Es ist die Umkehrfunktion $\hat{E}^{-1} : W(\hat{E}) \rightarrow \{0, 1\}^2$ von

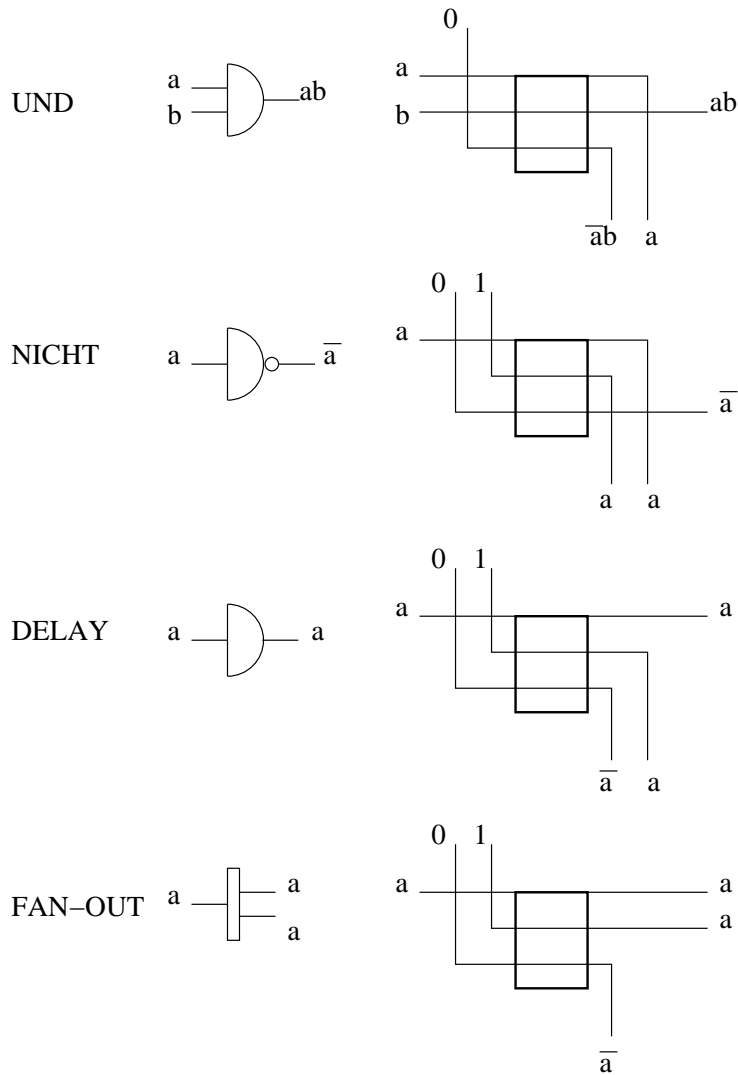


Abbildung 4.4: Schaltelemente und ihre Simulation durch ein Fredkin-Gate

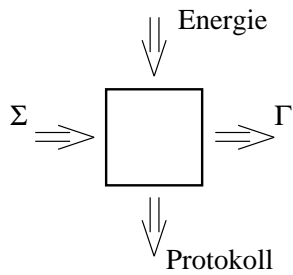
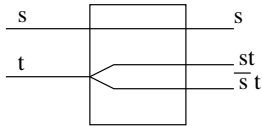
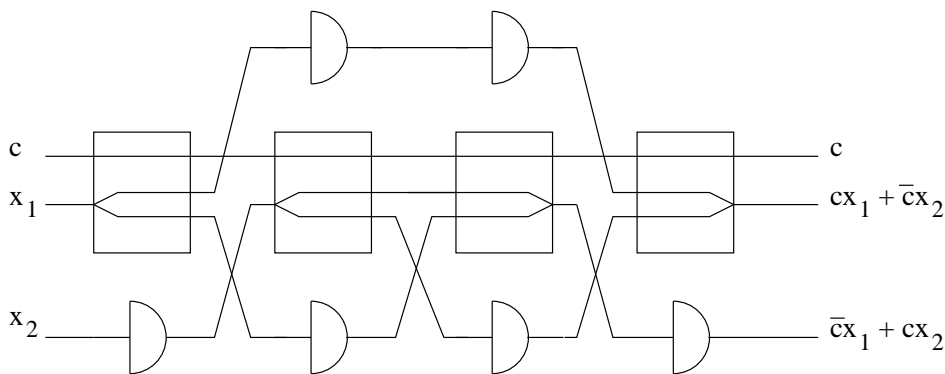


Abbildung 4.5: Realisierung eines Schaltnetzes durch ein reversibles Schaltnetz

Abbildung 4.6: Das Schaltelement \hat{E} .

\hat{E} , die nur auf dem Wertebereich $W(\hat{E})$ von \hat{E} definiert ist. Mit \hat{E} ist natürlich auch \hat{E}^{-1} ein reversibles Schaltelement, wenn es nur mit Eingangssignalverteilungen aus $W(\hat{E})$ benutzt wird.

Abbildung 4.7: Realisierung eines Fredkin-Gates mit \hat{E} , \hat{E}^{-1} , und DELAY-Elementen.

Interessanterweise kann man mit nur zwei Vorkommen von \hat{E} , zwei Vorkommen von \hat{E}^{-1} und sechs Vorkommen von DELAY-Elementen ein Fredkin-Gate realisieren: Abb. 4.7 zeigt das Schaltnetz, in der Variante mit Einheitsverzögerung in allen Schaltelementen. (In dem anderen Schaltnetz-Modell, wo alle Elemente verzögerungsfrei arbeiten, werden die DELAY-Elemente natürlich nicht gebraucht.) Da sich das DELAY-Element aus \hat{E} - und \hat{E}^{-1} -Elementen realisieren läßt, gilt:

Satz 4.10.2 *Reversible kombinatorische Schaltnetze aus \hat{E} - und \hat{E}^{-1} -Elementen sind endlich berechnungsuniversell.*

Es könnte interessant werden, mittels \hat{E} optische Computer zu realisieren. In dem DIA-Vortrag “Optical Conservative Reversible and Nearly Reversible Gates” der Portland Quantum Logic Group finden man in www.ee.pdx.edu/~mperkows/2005-quantum-class/2005-0099-optical.PPT einen reversiblen Volladdierer aufgebaut aus sieben \hat{E} Bausteinen, die auch *Priese-Gate* genannt werden.

4.11 Die Quanten-Turing-Maschine

Eine klassische Turing Maschine M ist ein Tupel $M = (K, \Sigma, \delta, s_0)$ von einer endlichen Menge K von Zuständen, einem *Bandalphabet* Σ mit einem ausgezeichneten Symbol $\#$, einem Startzustand s_0 und einer Übergangsfunktion $\delta : K \times \Sigma \rightarrow K \times \Sigma \times \{L, R\}$. Dabei bedeutet $\delta(q, a) = (q', b, X)$, dass M im Zustand q wenn es den Buchstaben a auf dem Arbeitsfeld liest dort b schreiben, ein Feld in Richtung rechts (für $X = R$) oder links (für $X = L$) gehen und in den Zustand q' wechseln soll. Eine Konfiguration ist dann eine Beschreibung des aktuellen Zustandes, Bandinhaltes und der Position des Arbeitsfeldes. Es werden nur endliche Konfigurationen zugelassen, in den auf dem Band nur endlich viele Symbole ungleich $\#$ stehen dürfen. Für zwei Konfigurationen C, C' gilt dann $C \vdash C'$, wenn sich die Konfiguration C bei Ausführung der in δ angegebenen Vorschrift in einem Schritt in die Konfiguration C' ändert. Es ist bekannt, dass man zur Berechnung beliebiger berechenbarer Funktionen immer mit dem Bandalphabet $\{|\, \#\}$ bei Codierung von Zahlen als Unärzahl, bzw. mit $\{0, 1, \#\}$ bei Codierung als Binärzahl auskommt. Wir lassen ein beidseitig unendliches Band zu. Damit wird die Bandinschrift eine Funktion $B : \mathbb{Z} \rightarrow \Sigma$ mit endlichem Support, d.h. mit $B(i) = \#$ nur für endlich viele $i \in \mathbb{Z}$, die Position P eine Zahl in \mathbb{Z} , und eine Konfiguration ist ein Tupel $C = (q, i, B)$ aus dem aktuellen Zustand, der aktuellen Position des Arbeitsfeldes und der aktuellen Bandinschrift. Die Startkonfiguration C_w mit Input $w = a_1 \dots a_n$ mit $a_i \in \Sigma - \{\#\}$ ist $(s_0, 1, B_w)$ mit $B_w(i) = a_i$ für $1 \leq i \leq n$ und $B_w(x) = \#$ sonst. D.h., das Inputwort wird auf das Band beginnend mit dem Feld auf Position 1 geschrieben, alle anderen Felder sind im Zustand $\#$.

Von einer quantenmechanisch arbeitenden Turing Maschine erwarten wir, dass sie in der Startkonfiguration beginnend nach t Schritten in einer Überlagerung von Konfigurationen ist. Erst beim Auslesen, einer Messung, wird eine der überlagerten Konfigurationen mit der Wahrscheinlichkeit ihres Amplitudenquadrats angenommen. Anstelle einer Endkonfiguration endet eine Rechnung mit dem Messen der aktuellen Konfigurationsüberlagerung, das zu einer konkreten Konfiguration führt. Dies führt zu folgender Definition.

Definition 4.11.1 *Eine Quanten-Turing-Maschine (QTM) M ist ein Tupel $M = (K, \Sigma, \delta, s_0)$ mit K, Σ, s_0 wie zuvor und einer lokalen Amplitudenfunktion*

$$\delta : K \times \Sigma \times K \times \Sigma \times \{L, R\} \rightarrow \mathbb{C}.$$

Eine Konfiguration C von M ist ein Tupel $C = (q, i, B)$ von einem Zu-

stand q , einer Position $i \in \mathbb{Z}$ und einer Funktion $B : \mathbb{Z} \rightarrow K$ mit endlichem Träger.

Die Dynamik \vdash soll jetzt den Gesetzen der Quantenmechanik folgen. Dazu geht man wie folgt vor. Wegen des endlichen Supports von B bleibt die Menge aller Konfigurationen abzählbar und wir wählen eine feste Abzählung mit C_i als i -ter Konfiguration, $i \in \mathbb{N}$.

Definition 4.11.2 Der Zeitevolutionsoperator \mathbb{M}^δ von M ist eine Matrix mit unendlich vielen Zeilen und Spalten mit

$$\mathbb{M}_{i,j}^\delta = \delta(q, a, q', b, X) \text{ für}$$

- $i, j \in \mathbb{N}$, $C_i = (q, k, B)$, $a = B(k)$, und
- $C_j = (q', l, B')$, $b = B'(k)$, $B'(x) = B(x)$ für $x \neq k$, und
- $l = k - 1$ für $X = L$ und $l = k + 1$ für $X = R$.

Deren Einträge $\mathbb{M}_{i,j}^\delta$ heißen Überföhrungsamplituten. \mathbb{M}^δ fassen wir als Operator auf ℓ^2 auf. Eine QTM mit einer lokalen Amplitudenfunktion δ heißt zulässig, falls \mathbb{M}^δ unitär ist.

Wir betrachten nur noch zulässige QTMen. Damit stellt $|M_{i,j}^\delta|^2$ die Wahrscheinlichkeit dar, in einem Schritt von Konfiguration C_i nach C_j zu gelangen. In einem sauberen Modell sind Zustände Überlagerungen von Konfigurationen, deren Summe der Quadrate der Normen der Amplituten 1 sein muss. Man kann dabei endliche oder unendliche Überlagerungen betrachten. Damit hat ein Zustand die Form

$$u = \sum_{i \in \mathbb{N}} z_i \cdot C_i \text{ mit } \sum_{i \in \mathbb{N}} |z_i|^2 = 1,$$

und wir identifizieren einen Zustand einer QTM mit einer Folge $(z_i)_{i \in \mathbb{N}}$ der Norm 1, also wieder mit einem Einheitsvektor im ℓ^2 aus dem Einheitskreis $E_1^{\ell^2} = \{u \in \ell^2 \mid |u| = 1\}$ im ℓ^2 . Als ARM wird eine QTM M damit zu

$$M = (E_1^{\ell^2}, \vdash) \text{ mit } u \vdash v \iff v = \mathbb{M}_\delta u.$$

Hier ist allerdings die Startkonfiguration C_w noch nicht berücksichtigt, was man auf verschiedene Weise machen kann.

Nehmen wir an, wir wollen die QTM M zum Zeitpunkt $t = 0$ mit einer Startkonfiguration $C_w = (s_0, 1, B_w)$ starten. Also im Zustand s_0 , mit dem Arbeitsfeld an Position 1, und mit der Bandbeschriftung B_w . C_w sei die i_0 -te Konfiguration C_{i_0} in unserer Aufzählung. Damit soll

zum Zeitpunkt 0 noch keine Überlagerung von Konfigurationen vorliegen sondern die Konfiguration $C_w = C_{i_0}$ mit der Wahrscheinlichkeit 1. Daher liegt der Zustand $e_{i_0} \in \ell^2$ vor, wobei e_{i_0} die Folge im ℓ^2 ist, die genau an Stelle i_0 den Wert 1 und sonst überall den Wert 0 besitzt. Da sich M in jedem Arbeitsschritt um ein Feld nach links oder rechts bewegen muss, ist die Überførungsamplitude $M_{i_0,j}^\delta$ nur für solche Konfigurationen $C_j = (q, i, B)$ ungleich 0, in denen die Position i die Werte 2 oder 0 besitzt und B sich nur in Position 1 von B_w unterscheiden kann. Das sind nur endlich viele Möglichkeiten. Ebenso erreichen wir nach t -Schritten den Zustand $v^t = (M^\delta)^t e_{i_0}$, in dem sich wiederum nur endlich viele Konfigurationen überlagern. D.h. v^t ist ein Einheitsvektor in ℓ^2 , in dem nur endlich viele Folgenglieder ungleich Null sind. Als ARM wählen wir für M nun den Konfigurationsraum $\mathcal{C} = \{(M^\delta)^t e_{i_0} \mid t \in \mathbb{N}\}$, die Erreichbarkeitsmenge von C_w .

Will man eine QTM realisieren, so muss man eine Technik finden, die Buchstaben auf den Bandfelder sich überlagern zu lassen. Dazu wählt man als Bandalphabet $\Sigma := \{0, 1, \#\}$ und lässt nur zusammenhängende Wörter über $\{0, 1\}$ umgeben von lauter $\#$ links und rechts als Bandinschriften zu. Damit schränkt man die Berechnungsmöglichkeiten nicht ein. Auf dem endlichen, zusammenhängenden Bandstück, auf dem Buchstaben 0 und 1 stehen, werden diese durch je ein Qubit realisiert. $\#$ steht nun für ein nicht vorhandenes Qubit. Damit könnten man in einem n -Quanten-System ein Band der Länge n simulieren.

Da unitäre Operatoren invertierbar sind, sind QTM stets reversibel. Damit bietet sich als Vorstellung auch eine Zeit \mathbb{Z} an. Zu einem Zeitpunkt t_0 greift man durch eine Manipulation in das ansonsten geschlossene System ein und stellt die Startkonfiguration ein. Dann arbeitet die QTM als geschlossenes System. Mit einer weiteren Messung von außen wird von dem aktuellen überlagerten Zustand aus $E_1^{\ell^2}$ eine Konfiguration mit positiver Amplitude zufällig ausgelesen, dabei natürlich die Rechnung der QTM zerstört. Man hat also nur ein zufälliges Resultat. Durch viele neue Rechnungen erhält man dann eine Wahrscheinlichkeitsverteilung. Umso weniger positive Amplituden ein Zustand beim Auslesen hat, um so sicherer wird die Messung bzw. Messreihe. Dies gilt natürlich auch für Quantenschaltnetze.

In der Informatik kann man sich unabhängig von diesen physikalischen Überlegungen mit der QTM befassen. Natürlich definiert man die von einer QTM akzeptierte Sprache und untersucht den Zusammenhang mit Sprachhierarchien. Eine normale Turing Maschine akzeptiert ein Inputwort $w \in \{0, 1\}^*$ nach t Schritten, falls sie mit C_w gestartet nach t Schritten im Haltezustand ist, und sie entscheidet eine Sprache L , falls

sie bei jeder Startkonfiguration C_w irgendwann in einen Haltezustand gelangt mit 1 auf dem ersten Bandfeld (Antwort "Ja") für $w \in L$ und 0 auf dem ersten Bandfeld (Antwort "Nein") für $w \notin L$. In der QTM haben wir keinen Haltezustand. Es sei $t : \mathbb{N} \rightarrow \mathbb{N}$ eine Schrittzahlfunktion. Wir erreichen bei der Startkonfiguration $C_w = C_{i_0}$ nach $t(|w|)$ Schritten in einer QTM M die überlagerte Konfiguration

$$C_w^t := \mathbb{M}_\delta^{t(|w|)} e_{i_0} = \sum_{i \in \mathbb{N}} z_i \cdot C_i.$$

Man betrachtet nun alle Konfigurationen $C_i = (q_i, p_i, B_i)$ mit positiven z_i , in den auf dem ersten Feld der Wert 1 (bzw. 0) steht, d.h. in denen $B_i(0) = 1$ (bzw. $=0$) gilt. Gilt

$$\begin{aligned} w \in L &\implies \frac{2}{3} \leq \sum_{B_i(0)=1} z_i, \text{ und} \\ w \notin L &\implies \frac{2}{3} \leq \sum_{B_i(0)=0} z_i, \end{aligned}$$

dann sagt man, dass die QTM M die Sprache L mit beschränkter Fehlerwahrscheinlichkeit akzeptiert. BQP ist die Klasse aller Sprachen, die von Quanten Turing Maschinen in polynomieller Zeit mit beschränkter Fehlerwahrscheinlichkeit akzeptiert werden können.

Teil II

Modelle nebenläufiger Rechnungen

Kapitel 5

Genetische Algorithmen

5.1 Motivation

Evolution, zuerst beschrieben von Darwin und Wallace, ist ein wissenschaftlicher Fakt. Daran ändern auch mehr oder weniger wütende Angriffe aus religiös motivierten Ecken nichts. Evolution ist nicht nur wissenschaftlich beobachtbar und analysierbar, sondern sogar experimentell nachvollziehbar und logisch begründbar. Und genauso, wie die astronomischen Entdeckungen von Kepler, Galileo u.v. nichts über Gott aussagen, Galileo von der katholischen Kirche wieder rehabilitiert ist, wird auch die Kirche ihren Frieden mit der Evolution machen müssen. Überraschend ist meist nur die Geschwindigkeit, mit der sich Evolution vollziehen kann. Während der Laie oft Evolution akzeptiert aber glaubt, dass dazu Jahrtausende notwendig sind, ist evolutionäre Anpassung in wenigen Generationen möglich. Ein Beispiel sei hier der Birkenspanner. Vor der einer Industrialisierung folgenden Verschmutzung besaßen Birken eine helle Grundfarbe mit dunklen Einsprengsel. Zu Beginn des 19ten Jahrhunderts wurden aber die Birken in der Gegend Manchesters von einer starken Rußschicht bedeckt und die sie besiedelten Flechten starben ab. Damit änderten die Birken ihre Farbe zu Dunkel mit hellen Einsprengsel. Entsprechend war die Färbung der Birkenspanner: eine helle Grundfarbe mit dunklen Einsprengsel bis 1850. Um 1850 beobachtete man die ersten dunklen Birkenspanner und keine 50 Jahre später waren 98% aller Birkenspanner im Raum Manchester sehr dunkel gefärbt auf Grund eines genetisch weitergegebenen höheren Melaninanteils in der Haut. Der Grund ist eigentlich völlig trivial: ein höherer Melaninanteil war immer schon genetisch möglich, die hellen Birkenspanner hatten aber eine erhöhte Überlebenswahrscheinlichkeit. Nach der Verschmutzung konnten der Grundfarbe der Birken nicht angepasste Birkenspanner leichter von Vögeln erkannt

und somit zur Beute werden. Zufällig farblich angepasste Falter wurden weniger leicht zur Beute und hatten damit eine höhere Fortpflanzungswahrscheinlichkeit. Natürlich funktioniert die Evolution nur, wenn die Vorteile eines Elternteils auch den Nachkommen weiter gegeben werden. In unserem Beispiel heißt das, dass die zufälligen vorteilhaften erhöhten Melaninanteile auch genetisch gespeichert sind und bei Fortpflanzung erhalten bleiben. Dieses Phänomen ist als Industriemelanisierung berühmt geworden (mit wütenden Angriffen von Kreationisten). Die gesamten Formen der Tarnung in Tierreich, Mimese und Mimikry, sind mittels Evolution offensichtlich leicht und logisch konsistent erklärbar.

Als einer der ersten nutzte Ingo Rechenberg eine evolutionäre Strategie zur Entwicklung technischer Systeme, siehe etwa [27]. Bekannt wurde ein Versuch von ihm, eine optimale Rohrbiegung um 90° zu erreichen. Gegeben sei zwei Rohre im 90° Winkel zueinander. Die Frage ist nach der strömungsgünstigsten Form der Verbindung beider Rohre. Offensichtlich ist das keine abrupte Übergang, indem man beide Rohre schräg ansägt und verschweißt. Aber auch eine Viertelkreisverbindung zwischen beiden ist nicht optimal. Rechenberg fügte nun zwischen beide Rohre einen flexiblen Schlauch gleichen Durchmessers, dessen Form in Abstand von einigen Zentimetern durch Schieber verändert werden konnte. Zufällig wurde nun der Parameter eines Schiebers verändert und damit auch die Schlauchkrümmung. Es lässt sich leicht messen, ob diese Änderung den Durchströmungswiderstand verbessert oder verschlechtert. Bei Verbesserung wird der neue Wert beibehalten und weitere Mutationen der Schieberparameter getestet.

Man kann dieses Experiment als einen sehr einfachen genetischen Algorithmus bei ungeschlechtlicher Vermehrung ansehen. Ungeschlechtlich, weil hier nur Mutation eine Rolle spielt, wie sie auch bei ungeschlechtlicher Vermehrung durch Teilung bei Bakterien auftritt. In einer geschlechtlichen Vermehrung kombinieren sich die Chromosome zweier (oder mehrerer) Eltern neu. Dabei ist Mutation auch erlaubt ist, aber sie ist nur eine von mehreren möglichen genetische Operation. Zum ersten Mal wurden solche allgemeineren genetischen Algorithmen theoretisch von John Holland [10] untersucht. Interessant ist hierbei, dass zu dieser Zeit Holland bereits ein bekannter Name in der Theorie zellulärer Automaten war. Obwohl dies nicht offensichtlich ist, sind genetische Algorithmen auch eine Überwindung der starren Nachbarschaftsstrukturen in zellulären Automaten unter Beibehaltung deren hoher Parallelität. Man kann sich streiten, ob man genetische Algorithmen als parallele oder als nebenläufige Systeme betrachten will.

5.2 Begriffe

Wir müssen zuerst mal die Klasse der Probleme etwas genauer verstehen, die zur Lösung mittels genetischer Algorithmen in Frage kommen. Betrachten wir eine bekannte Schulaufgabe: Man soll möglichst schnell von einem Ort A zu einem Ort B gelangen. A liegt an einer geraden Straße, die nahe an B vorbeiführt. B liegt auf einer Wiese. Die Entfernung von A zu B sei s , der orthogonale Abstand von B zur Straße sei a , die Entfernung vom Lotpunkt auf der Straße zu A sei b . Auf der Straße kann man sich mit einer Geschwindigkeit v_1 und auf der Wiese mit $v_2 < v_1$ bewegen. An welcher Stelle S verlasse ich die Straße und laufe direkt auf der Wiese zu B, um insgesamt von A nach B in kürzester Zeit zu gelangen? Dies ist eine einfache Differentialaufgabe mit einer simplen analytischen Lösung $s = f(s, a, b, v_1, v_2)$. Hier gibt es keinerlei Grund, eine Lösung mit genetischen Algorithmen oder irgendwelchen Suchalgorithmen anzugehen. Anders ist es in fast allen Optimierungsaufgaben im Bereich Operations Research. Oder auch in dem Beispiel der optimalen Krümmung. Es ist keine analytische Formel f bekannt, die bei gegebenem Durchmesser d der Rohre und des Raumes x, y, z , in dem die Krümmung stattfinden darf, als $f(d, x, y, z)$ die optimale Krümmung berechnet. Nebenbei, genau dieses Problem bewirkte die jahrelange Verzögerung in der Auslieferung der A380. Im Computermodell waren die Krümmungen der Kabelbäume falsch modelliert und die berechneten Kabelbäume waren zu kurz und liesen sich im Flugzeug nicht installieren.

Betrachten wir eines der bekanntesten NP-vollständigen Probleme, das Travelling-Salesman-Problem (TSP). Gegeben seien n Orte und eine Tabelle der Entfernungen $d(x, y)$ von je zwei Orten x und y . Gesucht ist eine minimale (bzgl der zurückgelegten Gesamtstrecke) Rundreise, die jeden Ort mindestens einmal besucht. Überraschenderweise ist das einzig bekannte Lösungsverfahren, das mit Sicherheit eine optimale Rundreise findet, das Ausprobieren aller möglichen Rundreisen. Das sind $n!$ viele. Dieser triviale Suchalgorithmus braucht also exponentiell viele Schritte. Zwar gibt es diverse Strategien, die Suche sinnvoll einzuschränken oder gezielter vorzugehen, dennoch kann man im ungünstigen Fall gezwungen sein, alle Rundreisen auszuprobieren.

Was ist nun eine "Lösung" des TSP? Man sollte meinen, das sei die Rundreise, die alle Orte besucht und von allen möglichen Rundreisen die kürzeste Gesamtentfernung zurücklegt. Oder eine solche Rundreise, wenn es mehrere mit optimaler zurückgelegter Strecke gibt. Allerdings sind die Notationen in genetischen Algorithmen etwas gegen die Intuition. In einem genetischen Algorithmus würde jede Permutation der n Städte als Lösung bezeichnet (potentielle Lösung wäre ein besserer Na-

me), Lösungen mit relativ wenig Gesamtstrecke sind **gute** Lösungen, die mit minimaler Gesamtstrecke **optimale** Lösungen.

Nehmen wir ein Beispiel mit 5 Städten A,B,C,D,E. Zu jedem Paar X, Y zweier Städte sei deren Entfernung $d(X, Y)$ bekannt. Dann sind die Permutationen (B,E,A,D,C) und (C,E,B,A,D) Lösungen, egal wie schlecht oder gut deren Gesamtstrecken sind. $r=(B,E,A,D,C)$ bedeutet in Stadt B zu starten, dann in dieser Reihenfolge zu den Städten E,A,D,C zu fahren, und schließlich zum Ausgangsort B zurückzukehren. Die *Güte* $g(r)$ der Rundreise r ist dann die Gesamtentfernung in r , also

$$g(r) = d(B, E) + d(E, A) + d(A, D) + d(D, C) + d(C, B).$$

Häufig werden Reiseverläufe wie $r'=(A,C,E,A,B,D)$ oder $r''=(A,C,B,D)$ auch als Lösungen, dann aber als *unzulässige* Lösungen bezeichnet. In r' wird die Stadt A zweimal besucht, was in einer optimalen Lösung nicht notwendig wäre. In r'' wird die Stadt E gar nicht besucht, was nicht erlaubt ist. In diesem Beispiel soll g minimal werden. Allgemein werden wir allerdings annehmen, dass die Güte maximalisiert werden soll. Lösungen sind i.A. Vektoren, wobei zwei verschiedene Lösungen durchaus als zwei Vektoren unterschiedlicher Dimension modelliert werden können. Wir werden daher statt mit Vektoren lieber mit Wörtern arbeiten.

Definition 5.2.1 *Ein genetischer Algorithmus $G = (M, d, g)$ besteht aus einer Menge M , einer Metrik d auf M und einer Gütefunktion $g : M^* \rightarrow \mathbb{R}$, die in polynomieller Zeit berechnet werden kann.*

Jedes Wort $w \in M^$ heißt eine Lösung von G . Eine Lösung w heißt besser als eine Lösung w' , falls $g(w) > g(w')$ gilt. Eine Lösung w heißt optimal, falls keine bessere Lösung als w existiert.*

Eine Population (der Größe n) ist eine Menge von n Lösungen.

Im folgenden seien $u, v, w, u', v', w', x, y, z, u_i, v_i, u'_i, v'_i \in M^$, $a, b \in M$. Unter einer genetischen Operation verstehen wir eine der folgenden ein- oder zweistelligen Funktionen von M^* oder $M^* \times M^*$ nach M^* :*

- Mutation $\mu : M^* \rightarrow M^*$ mit

$$\mu(uav) = ubv,$$

mit der Weite $d(a, b)$,

- Inversion $\iota : M^* \rightarrow M^*$ mit

$$\iota(uvw) = uv^Rw,$$

mit der Weite $|v|$,

- Verschiebung $t : M^* \rightarrow M^*$ mit

$$t(uvw) = u'vw'$$

mit $uw = u'w'$ und mit der Weite $|v|$ und Entfernung $||u| - |u'||$,

- Verkürzung $k : M^* \rightarrow M^*$

$$k(uvw) = uw,$$

mit der Weite $|v|$,

- Verlängerung $l : M^* \rightarrow M^*$

$$l(uvw) = xvyvz,$$

mit $xyz = uw$, mit der Weite $|v|$,

- Einfügung $e : M^* \times M^* \rightarrow M^*$ mit

$$e(uv, u'vw') = uvv,$$

mit der Weite $|w|$,

- einfaches Crossing-Over $ec : M^* \times M^* \rightarrow M^*$

$$ec(uvw, u'v'w') = wv'w,$$

mit $|v| = |v'|$ und der Weite $|v|$,

- mehrfaches Crossing-over $c : M^* \times M^* \rightarrow M^*$

$$c(u_1v_1u_2v_2\dots u_nv_nw, u'_1v'_1u'_2v'_2\dots u'_nv'_nw') = u_1v'_1u_2v'_2\dots u_nv'_nw,$$

mit $|v_i| = |v'_i|$ für alle i und der Weite $\max_i |v_i|$. n ist die Rate dieser Operation.

Eine mehrfach Mutation der Rate n besteht aus n Anwendungen der Operation Mutation. Die dabei verwendete maximale Weite ist die Weite der mehrfach Operation. Analog für alle anderen Operationen werden mehrfach Operationen definiert. Der Zusatz "mehrfach" wird häufig weggelassen ist nur aus dem Kontext erkennbar.

Wird in einem konkreten Problem eine mögliche Lösung als ein Wort aus M^* kodiert, so nennt man alle Wörter aus M^* , die keine mögliche Lösung des konkreten Problem codieren, auch unzulässige oder letale Lösungen. Ein Reparaturmechanismus r ist eine Funktion $r : M^* \rightarrow M^*$, die eine letale Lösung in eine zulässige Lösung umformt.

5.3 Aufbau von genetische Algorithmen

Will man eine konkrete Aufgabe mittels genetischer Algorithmen lösen, so geht man wie folgt vor. Das Problem muss erst überhaupt für genetische Algorithmen geeignet sein. D.h. einfach, dass man zulässige Lösungen als Wörter $w \in M^*$ einer geeigneten metrischen Menge M verschlüsseln kann und zu jedem Wort w die Güte $g(w)$ der in w verschlüsselten möglichen Lösung schnell berechnen können muss. Ist die Berechnung von $g(w)$ schwierig, ist eine Verwendung von genetischen Algorithmen nicht sinnvoll.

Man wählt nun eine zufällige Menge von k Wörtern aus M^* als initiale Population P_1 . Im Induktionsschritt liege bereits eine n -te Population P_n vor. Jede Lösung in P_n heißt auch *Individuum*. Ein Individuum w wird nun gemäß seiner *Fitness*, das ist $g(w)$, zu einer Fortpflanzung ausgewählt. Je höher der Wert $g(w)$ desto höher die Wahrscheinlichkeit, dass w ausgewählt wird. Der Zusammenhang von g zur Auswahlwahrscheinlichkeit heißt *Selektionsdruck*. Auf w wird nun zufällig eine oder mehrere der genetischen Operationen angewendet, bei geschlechtlichen Operationen mit einem auch nach Fitness ausgewählten Partner. Die gewählten Parameter des genetischen Algorithmus steuern die Auswahl der genetischen Operation(en). Das Ergebnis w' der Anwendung einer oder mehrerer Operationen auf w heißt *Nachkomme* von w . Entsteht $w' = uvw$ durch Crossing-Over aus uvw und $u'v'w'$, so wird häufig auch $u'vw'$ als zweiter Nachkomme zugelassen. Diese Methode Nachkommen zu generieren wird nun mehrfach, sagen wir m mal angewendet, resultierend in einer Menge P'_n von m Individuen. Es gibt nun unterschiedliche Ansätze, die nächste Generation P_{n+1} zu definieren. Man kann einfach $P_{n+1} := P'_n$ setzen, oder einige der besten Individuen aus P_n auf jeden Fall in P_{n+1} übernehmen. Oder man setzt $P_{n+1} := P_n \cup P'_n$. Da hierbei die Generationen stets in der Größe wachsen, wird in jedem Schritt oder alle paar Schritte eine Sterberate eingeführt, in der mit einer Wahrscheinlichkeit reziprok zur Fitness Individuen entfernt werden. Hierbei sind zahlreiche Variationen denkbar und auch verwendet wurden.

Zahlreiche Parameter können einen genetischen Algorithmus steuern. Die wichtigsten sind die durchschnittliche oder feste Größe einer Population und die Durchschnitte und Varianzen der Raten, Weiten, Entfernung und Anwendungshäufigkeiten der einzelnen genetischen Operationen.

Beispiel 5.3.1 *In diesem Sinn ist das Experiment von Rechenberg zur Bestimmung der optimalen Krümmung ein genetischer Algorithmus, allerdings ein extrem entarteter. M ist hierbei \mathbb{R} mit dem Euklid'schen Abstand. Ein Wort $w = x_1 \dots x_n$ mit $x_i \in \mathbb{R}$ sagt, in welche Richtung (+*

oder -) und mit welcher Kraft x_i der i -te Schieber das flexible Rohr verbiegt. Alle genetischen Operationen sind hier sinnvoll. Eine Verlängerung $l(u_1u_2vw) = u_1vu_2vw$ z.B. würde $|v|$ viele neue Schieber einfügen. Ein Reparaturmechanismus würde extreme Schieberstellungen, deren Realisierung das flexible Rohr nicht mehr zulässt, etwa auf deren maximal möglichen Ausschlag begrenzen. Rechenberg verwendet letztlich aber das Modell der genetischen Algorithmen gar nicht, oder nur in einer extrem entarteten Form. Seine Generationen bestehen stets nur aus einem Individuum, die einzige verwendete Operation ist die einfache Mutation, und genau der bessere der beiden, altes Individuum oder Nachkomme, bestimmt die neue Generation. Die Güte eines Individuum bestimmt Rechenberg trivial durch eine Messung der Strömung.

Natürlich bestimmt die Größe einer Generation die Parallelität in diesem Schritt. Jedes Individuum kann mit jedem anderen zur geschlechtlichen Fortpflanzung mittels Verlängerung oder Crossing-Over ausgewählt werden, unabhängig von dessen "Lage" in der Population. In diesem Sinn heben genetische Algorithmen die starre Nachbarschaft in zellularen Automaten auf, ein "Lagebegriff" spielt keinerlei Rolle.

Beispiel 5.3.2 *Ein überzeugendes Beispiel für einen Einsatz genetischer Algorithmen ist die Verteilung von Standorten. Nehmen wir als Beispiel Deutschland. Die Verteilung der Bevölkerung nach Wohn- oder Arbeitsorten ist weitgehendst bekannt, die Verkehrsverbindungen sind in jedem Navisystem aktualisierbar. Sagen wir, ein neues Kommunikationssystem soll eingerichtet werden, wie damals bei UMTS. Dazu sind Standorte für Sende- und Empfangseinrichtungen erforderlich, die nur in einer begrenzten Entfernung funktionieren. Ferner hat jede dieser Einrichtungen eine begrenzte Kapazität. Die Frage ist nun, wieviele dieser Einrichtungen sollen wo aufgestellt werden. Ziel ist es, einem möglichst hohen Prozentsatz der Bevölkerung (deutlich über 85% bzgl der Wohnverteilung, deutlich über 95% bzgl der Arbeitsplatzverteilung) zu über 95% der Zeit (kapazitative Einschränkung pro Kommunikationseinrichtung) eine Kommunikation zu ermöglichen, unter Minimierung der Einrichtungs- und Unterhaltskosten. Oder ein Unternehmen mit einem festen Produktionsort sucht optimale Standorte für Auslieferungslager. Die Anzahl der Auslieferungslager liege nicht a priori fest. Die Wahrscheinlichkeiten, in einen bestimmten Wohn- oder Arbeitsort auszuliefern seien bzgl der Bevölkerungsverteilung und des Benutzungsverhaltens bekannt. Es soll eine komplexe Zielfunktion optimiert werden: die Einrichtungskosten (einmalig) und Unterhaltskosten pro Auslieferungslager, die Transportkosten vom Produktionsort zu den Lagern und die Transportkosten von den Lagern zu den Verbrauchern gemäß der Benutzungswahrscheinlichkeit ist zu mi-*

nimieren, die Zufriedenheit der Kunden zu maximieren.

Natürlich existiert für solche eine komplexe Zielfunktionen keine geschlossene analytische Formel. Wählt man aber als "Lösung" eine mögliche Verteilung der Standorte, so lässt sich die Kostenfunktion für diese Lösung relativ einfach berechnen. Damit bietet sich ein genetischer Algorithmus an. M ist damit die Menge der möglichen Standorte, eine Lösung ein Vektor unbekannter Dimension, bzw. ein Wort unbekannter Länge, über M . Mutation verändert die Lagen der Standorte in einer Lösung leicht. Verlängerung und Verkürzung ändert die Anzahl der Standorte. Einfügung und Crossing-Over überträgt Teillösungen eines Elternteils auf ein anderes (bzw. Teillösungen von zwei Elternteilen auf den Nachkommen, was logisch aber ähnlich ist). Da Eltern gemäß ihrer Fitness zur Fortpflanzung ausgewählt werden, ist zu erwarten, dass gute Teillösungen in der geschlechtlichen Fortpflanzung eher als schlechte übertragen werden. Inversion und Verschiebung scheinen in dieser Anwendung nicht so wichtig zu sein. Deren Bedeutung wird noch diskutiert. Eine Reparaturfunktion wird kaum benötigt, spielt aber bei der genetischen Operation Einfügung eine gewisse Rolle zur Vermeidung von Redundanzen.

Beispiel 5.3.3 Im ersten Moment ist es nicht offensichtlich, wie man das Travelling Salesman Problem als genetischen Algorithmus modellieren kann. Als M kann man die Menge der Städte selbst nehmen mit der diskreten Metrik, d.h. der Abstand einer Stadt mit sich selbst ist 0, zu jeder anderen 1. Jedes Wort über M^* ist damit formal eine Lösung. Kommt in einem Wort $w \in M^*$ aber eine Stadt gar nicht oder mehrfach vor, betrachten wir diese Lösung als unzulässig. Ein höchst simpler Reparaturmechanismus ist es, in einem Wort eine mehrfach vorkommende Stadt bis auf ein Vorkommen zufällig oder systematisch zu entfernen, sowie eine gar nicht vorkommende Stadt an einer Stelle des Wortes zufällig oder systematisch einzufügen. Damit könnten alle Operationen mit solch einem Reparaturmechanismus prinzipiell verwendet werden. Dies scheint aber nicht sehr sinnvoll zu sein. Die beiden Operationen, die ein zulässiges Wort in ein zulässiges Wort ohne Reparatur überführen, sind Inversion und Verschiebung. Damit lässt sich sofort ein genetischer Algorithmus aufbauen. Sicherlich sollte man im TSP auf eine Mutation verzichten, aber ein Verzicht auf Crossing-Over ist schon schmerzhaft. Anstelle eines Crossing-Over mit der aufgezeigten trivialen Reparatur bietet sich hier ein intelligenteres Vorgehen an. Als Ersatz für die geschlechtliche Fortpflanzung mittels Crossing-Over wählen wir die Verlängerung. Es seien $w_1 = uw$ und $w_2 = u'v'w'$ gemäß ihrer Fitness ausgewählt und $w' := uv'w$ sei der aus w_1 und w_2 durch Verlängerung gewonnene Nachkomme, in dem der Abschnitt v' aus w_2 zusätzlich eingefügt wurde. v'

wird nun so in w' belassen und alle anderen Vorkommen von Städten aus v' in w' werden einfach entfernt. Dies scheint ein sinnvolle geschlechtliche Fortpflanzung zu sein, in der Erbinformation von w_2 konsistent nach w_1 übernommen wird.

Am Beispiel des TSP ist es ersichtlich, dass die Anordnung als Wort (bzw als Liste) in einer Codierung der Aufgabe nicht zwingend ist. In einer Rundreise kommt jede Stadt genau einmal vor. Für die Güte ist daher der Startort der Rundreise irrelevant und zur Codierung kommt auch die Datenstruktur Schlange in Betracht. Da wir uns aber als Wörter zur Codierung festgelegt haben, könnten wir einfach Wörter u_1u_2 und u_2u_1 identifizieren und in den genetischen Operationen als gleichberechtigt betrachten.

Genetische Algorithmen zeigen typische Verhaltensmuster. In den ersten Generationen steigt die durchschnittliche und maximale Güte rasch. Dieser Anstieg lässt aber schnell nach und nach einigen Hundert Generationen ist eine Verbesserung in der durchschnittlichen oder maximalen Güte nicht mehr oder nur noch sehr selten zu sehen. Dennoch kann man nicht wissen, ob in einer späteren Generation nicht noch eine, vielleicht sogar substantielle, Verbesserung erreicht wird. Hier zeigt sich die größte Schwäche von genetischen Algorithmen: es existiert kein überzeugendes **Abbruchkriterium**.

5.4 Bedeutung der genetischen Operationen

Unter einem *greedy* Algorithmus versteht man ein Suchverfahren, in dem man in jedem Schritt alle Nachbarn (bzgl der Metrik d auf M) einer Lösung $w \in M^*$ betrachtet, und mit dem Nachbarn mit der besten Güte g weiterarbeitet. Greedy Verfahren sind sehr schnell, können aber leicht in einem lokalen Nebenmaxima hängen bleiben. Eine Methode, lokale Nebenmaxima möglichst auszuschließen, sind Methoden des *simulated annealing*. Hier wird die Temperatursteuerung zur Auskristallisierung in einer Schmelze simuliert. Höhere Temperatur bedeutet Suche in weiter entfernten Nachbarn. Auch genetische Algorithmen versuchen das Hängenbleiben in lokalen Nebenmaxima zu vermeiden, im Wesentlichen durch ihre hohe Parallelität, d.h. die Größe einer Generation. Gerät ein genetischer Algorithmus allerdings in eine Generation, deren Individuen alle Variationen eines einzelnen Individuum sind, so hängt auch der genetische Algorithmus in der Nähe eines Maximas, das durchaus ein

Nebenmaxima sein kann.

Die Bedeutung der einzelnen genetischen Operationen zur Überwindung der Schwächen von greedy Algorithmen dürfte offensichtlich sein.

Die **Größe** einer Population bestimmt die Parallelität des Algorithmus.

Mutation erlaubt eine Suche in der Nachbarschaft. Allein mittels Mutation und einer zeitlichen Steuerung der Mutationsraten lässt sich simulated annealing erreichen.

Einfügung und **Crossing-Over** sind geschlechtliche Operation, die einen Nachkommen aus zwei Individuen entstehen lassen. Dabei werden größere zusammenhängende Erbinformationen von einem Partner auf den anderen übertragen.

Verlängerung, **Verkürzung** und **Einfügung** ändern die Zahl der Gene. Man sagt häufig, das dadurch eine neue Art entsteht. Häufig ist ein Crossing-Over zwischen zwei verschiedenen Arten nicht mehr möglich, bzw. führt zu letalen Nachkommen.

Die Bedeutung von **Verschiebung** und **Inversion** sind auf den ersten Blick nicht so klar. Häufig wird eine einzelne Eigenschaften eines Individuum nicht von einem sondern von mehreren Genen gesteuert. Dann kann die Lage der Gene auf den Chromosomen eine wichtige Bedeutung erlangen. Es kann vorteilhaft sein, wenn verschiedene Gene für eine Eigenschaft nahe benachbart angeordnet sind, um eine vorteilhafte Kombination davon mit höherer Wahrscheinlichkeit gemeinsam vererben zu können. Genau so gut kann aber eine möglichst weit entfernte Anordnung Vorteile in der Sicherheit bringen, wichtige Eigenschaften durch mehrere Gene redundant zu codieren. In diesem Fall ist dann die Kombination der Ausprägungen dieser Gene nicht so entscheidend.

5.5 Migration

Das Problem der *Inzucht*, dass eine Generation fast nur aus Variationen eines Individuums besteht, muss vermieden werden. Das ist eine Kunst. Dazu gehört eine je nach Aufgabe unterschiedliche Wahl der vielfachen Parameter.

Ein weiteres Hilfsmittel zur Vermeidung von Inzucht ist die *Migration*. Hier werden mehrere Populationen gebildet, die sich unabhängig voneinander entwickeln. Nur recht selten gelangen Individuen einer Population "zur Blutauffrischung" in eine andere.

Übung 5.5.1 *Dies ist eine Gruppenaufgabe, die von jedem Mitglied einer Übungsgruppe unabhängig zu lösen ist. Von der Gruppe sind dann die Lösungen zusammen zu bewerten und integrieren.*

- *Implementieren Sie einen genetischen Algorithmus für das TSP. Ihr Algorithmus soll als Input eine Textdatei benutzen, in der die Anzahl n der Städte in der ersten Zeile steht und in Zeile i die Entfernungen von Stadt i zu Stadt j für $1 \leq j < i$, jeweils durch ein Semikolon getrennt. Die Städte sollen 1 bis n heißen. Der Output Ihres Algorithmus soll die beste gefundene Rundreise, dargestellt als Permutation p von n , und die Güte $g(p)$ sein. p sei dabei eine Liste von n Integerwerten, in der jeder Wert von 1 bis n genau einmal vorkommt.*
- *Stellen Sie sicher, dass Ihr Algorithmus in einer Übungsstunde von Ihnen auf ein in der Übungsgruppe angegebenes TSP, dargestellt als die genannte Liste L , angewendet werden kann. Gehen Sie davon aus, dass diese Liste etwa 20 bis 30 Städte mit Entfernungswerten von 0 bis $2^{16} - 1$ enthalten wird. Ihr Algorithmus darf höchstens 10 Minuten lang laufen. Ziel ist es, die Parameter der genetischen Algorithmen, die die besten Ergebnisse geliefert haben, zu vergleichen und zu diskutieren. Entscheidend ist daher, dass die einzelnen Lösungen **unabhängig** entwickelt werden.*
- *Stellen Sie sicher, dass wir Ihren Algorithmus mit der von Ihnen gefundenen letzten Generation neu starten können, wenn wir k (wird als Variable in der Übung vorgegeben) gemäß der Fitness schlechte Individuen entfernen und k andere, die von anderen Übungsteilnehmern als beste gefunden wurden neu aufnehmen. Die k neuen Individuen werden ebenfalls als Textdatei von k Zeilen übergeben, wobei auf Zeile i die i -te Rundreise in der Form $x_1; \dots; x_n$ stehen soll für eine Permutation x_1, \dots, x_n von $\{1, \dots, n\}$. So wollen wir Effekte von Migration studieren. Sie brauchen also in Ihren individuellen Algorithmen keine Migration einzubauen und können mit nur einer Population arbeiten.*

5.6 Genetische Algorithmen und Evolution

In der biologischen Evolution ist eine geschlechtliche Fortpflanzung deutlich später als die ungeschlechtliche entstanden. Auch ist der Vorteil einer

geschlechtlichen Fortpflanzung nicht so viel größer, dass sie die ungeschlechtliche verdrängt hätte. Die Gesamtbio­masse aller Lebewesen, die sich ungeschlechtlich fortpflanzen, wie Bakterien, ist auf der Erde höher als die der sich geschlechtlich fortpflanzenden.

Man kann davon ausgehen, dass diverse genetische Algorithmen in der Evolution ausprobiert wurden und dass die verwendeten eine gewisse Optimalität erreicht haben, die den Lebensbedingungen auf der Erde gut angepasst sind. Auf anderen Welten mit anderen Umweltbedingungen wären möglicherweise ganz andere Mutations- und Crossing-Over-Raten besser geeignet. Daher ist es auch nicht a priori vorteilhaft, exakt die aus der menschlichen Fortpflanzung bekannten Raten der genetischen Operatoren auf genetische Algorithmen zur Lösung technischer Probleme zu übernehmen. Je nach Aufgabe scheinen manche der genannten genetischen Operatoren weniger als andere geeignet zu sein. Ein allgemeines “Kochrezept”, welche Parameterkombinationen besonders gut sind, existiert nicht.

Kapitel 6

Petri Netze

6.1 Begriffe

Petri Netze sind eines, wenn nicht *das*, Standardmodell für nebenläufige Rechnungen oder Prozesse. Nebenläufig bedeutet, dass parallel mehrere Aktionen (Prozesse, Agenten, etc.) möglich sind, diese aber nicht von einer globalen Uhr getaktet werden. Man kann es sich so vorstellen, dass jeder Agent seine eigne lokale Uhr besitzt, nach der er sich richtet. Daraus darf aber nicht gefolgert werden, dass nebenläufige Agenten unabhängig sind. Greifen z.B. mehrere Agenten auf eine gemeinsame Resource zurück, so verlieren alle die Zugriffsmöglichkeit, falls einer die Resource benutzt. Verfügen mehrere nicht-getaktete Prozessoren über einen gemeinsamen Speicher, so werden Lese- und Schreibkonflikte gern mittels Petri Netze modelliert. Eine theoretische Analyse des Verhaltens von Petri Netze hat in der Vergangenheit hier Lösungsmöglichkeiten aufgezeigt. Eingeführt wurden Petri Netze in der berühmten Dissertation “Kommunikation mit Automaten“ von Carl Adam Petri. Ausgangspunkt war die Beobachtung, dass in der Größe unbeschränkt wachsende Schaltwerke, etwa durch Anflanschen weitere Module, zur Simulation beliebiger Turingmaschinen nicht sinnvoll global getaktet werden können. Mit Wachsen des Schaltwerkes müsste sich die globale Taktung stets Verlangsamen. Petri schlug als Lösung den Verzicht auf den globalen Takt und eine Einführung lokaler Zeiten vor. Etwa gleichzeitig wurden in den USA, hauptsächlich am MIT, zu Petri Netzen äquivalente Konzepte wie Vektor-Ersetzungs-(bzw. -Additions-)Systeme entwickelt. Ein früher Austausch der Resultate der GMD um Petri und des MIT um Holt hat das Gebiet stark befruchtet.

Wir werden zuerst diverse Konzepte im Rahmen von Petri Netzen formal definieren und dabei immer wieder auf die Ausgangsmotivation

zurückkommen.

Definition 6.1.1 *Ein Petri Netz, PN, ist ein Tupel $\mathcal{N} = (P, T, F)$ von*

- *einer endlichen, angeordneten Menge $P = \{p_1, \dots, p_n\}$ von Places,*
- *einer endlichen, angeordneten Menge $T = \{t_1, \dots, t_m\}$ von Transitionen,*
- *mit $P \cap T = \emptyset$, und*
- *einer Abbildung $F : (P \times T) \cup (T \times P) \rightarrow \mathbb{N}$ von Kanten.*

$F(p, t)$ sagt, wie viele gerichtete Kanten vom Place p zur Transition t führen, und $F(t, p)$ wieviele gerichtete Kanten von der Transition t zum Place p führen. Kanten verbinden also stets Places mit Transitionen oder Transitionen mit Places, aber niemals Places mit Places oder Transitionen mit Transitionen. Damit kann man alternativ die statische Struktur eines Petri Netzes als einen bipartiten, gerichteten, kanten- und knotengewichteten, endlichen Graphen auffassen. Bipartit, weil die Knoten in zwei Klassen, Transitionen und Places, unterteilt sind. Die Knotengewichte sind gerade $\{t_1, \dots, t_m\}$ und $\{p_1, \dots, p_n\}$, und Kantengewichte sind natürliche Zahlen, gegeben durch F , die Multiplizitäten“ der Kanten.

Generell empfiehlt es sich bei der Einführung von Petri Netzen mit Multimengen zu arbeiten, da die Definitionen damit viel übersichtlich werden. Eine Menge ist nur dadurch charakterisiert, welche Elemente in ihr vorkommen. So gilt $\{1, 2, 3\} = \{2, 1, 3\} = \{1, 2, 1, 2, 3, 1\}$. Spielt die Anordnung der Elemente eine Rolle, so kann man statt Mengen besser das Konzept der Listen verwenden. Ist die Anordnung egal aber die Anzahl der Vorkommen der Elemente wichtig, so empfehlen sich Multimengen.

Definition 6.1.2 *Eine Multimenge m über einer Grundmenge M ist eine Abbildung $m : M \rightarrow \mathbb{N}$. Für $a \in M$ mit $m(a) > 0$ sagt man, dass a in m vorkommt, und zwar mit der Multiplizität $m(a)$.*

Für $x, y \in P \cup T$ sind $\bullet x$, der Vorbereich, und x^\bullet , der Nachbereich, von x zwei Multimengen definiert durch

$$\bullet x(y) = F(y, x) \text{ und } x^\bullet(y) = F(x, y).$$

Für $X \subseteq P \cup T$ ist $\bullet X := \sum_{x \in X} \bullet x$, analog für X^\bullet .

Die Vereinigung von Mengen, \cup , wird bei Multimengen kanonisch zur Summation, \sum . Ein Place p heißt etwa *Inputplace* einer Transition, falls p im Vorbereich von t vorkommt, falls also $F(p, t) > 0$ gilt. Analog für

Outputplace. Die Multiplizität von p in $\bullet t$ ist die Zahl der Kanten von p nach t und wird auch interpretiert als “wie oft“ p als Inputplace von t vorkommt.

Bislang haben wir nur die statische Struktur eines Petri Netzes kennengelernt, noch nicht aber dessen Verhalten. Dazu brauchen wir einen Begriff einer Konfiguration eines Petri Netzes und eines direkten Nachfolgers, \vdash , für die Dynamik. Diese Dynamik wird durch ein Tokenspiel gegeben. Places sind Platzhalter für eine Anzahl von *Token*, auch *Marken* genannt. Eine Transition t kann *schalten* oder *feuern*, falls auf jedem Inputplace p mindestens so viele Token liegen wie dessen Multiplizität angibt, d.h. wie viele Kanten von p nach t führen. t feuert, indem t von jedem Inputplace für jede Kante einen Token wegnimmt und auf jeden Outputplace p' so viele Token zusätzlich legt, wie Kanten von t nach p' führen. Die Verteilung der Token auf den Places nennt man *Markierung*. Diese Markierungen sind die hier interessierenden Konfigurationen. Das Feuern definiert die Dynamik der Nachfolgerrelation \vdash . Zwei Transitionen t, t' können durchaus einen gemeinsamen Inputplace p besitzen. Sind nun t und t' in einer Konfiguration feuerebar, so kann das Feuern der einen so viele Token von p entfernen, dass die andere Transition ihre Aktivierung verliert und nicht mehr feuern kann. In diesen Fall sagt man, dass t und t' *im Konflikt* stehen. Lässt das Feuern der einen genügend Token auf p , dass auch die andere Konfiguration noch feuern kann, so können beide unabhängig voneinander feuern und man sagt, dass beide *nebenläufig* sind. Liegen genügend Token auf alle Inputplaces, dass eine Transition mehrfach feuern könnte, so sagt man, dass sie zu sich selbst nebenläufig ist. Diese Begriffe beziehen sich dabei stets auf eine gegebene Konfiguration. Um eine präzise aber einfache formale Definition zu erhalten, empfiehlt sich gleich das Feuern von mehreren Transitionen zu definieren.

Definition 6.1.3 *Eine Konfiguration s eines Petri Netzes $\mathcal{N} = (P, T, F)$ ist eine Multimenge über P .*

Eine Multimenge Γ von Transition $t \in T$ ist (in s) aktiviert, in Zeichen $s \vdash_{\Gamma}$, falls $s \geq \bullet \Gamma$ gilt. $s \vdash_{\Gamma} s'$ gilt genau dann, falls

- Γ ist in s aktiviert, und
- $s' = s - \bullet \Gamma + \Gamma \bullet$.

Ist Γ (in s) aktiviert, so heißen alle Transitionen in Γ zueinander (in s) nebenläufig.

Kommt t in Γ mit einer Multiplizität > 1 vor, so heißt t auto-concurrent.

Gilt $s \vdash_t$ und $s \vdash_{t'}$ aber nicht $s \vdash_{\{t,t'\}}$, so sind t und t' (in der Konfiguration s) im Konflikt.

Besteht Γ nur aus einer einzigen Transition t der Multiplizität 1, so schreiben wir \vdash_Γ auch einfach nur als \vdash_t , etc. Ein Wort σ über T heißt Feuersequenz (in s) mit $s \vdash_\sigma s'$, falls gilt

- σ ist das leere Wort und $s' = s$, oder
- $\sigma = \sigma't$ und es existiert ein s'' mit $s \vdash_{\sigma'} s'' \vdash_t s'$.

σ heißt *feuerbar* in s , $s \vdash_\sigma$, falls σ eine Feuersequenz in s ist.

Abbildung 6.1 zeigt ein Petri Netz und Abbildungen 6.2, 6.3 zwei Schritte im Verhalten.

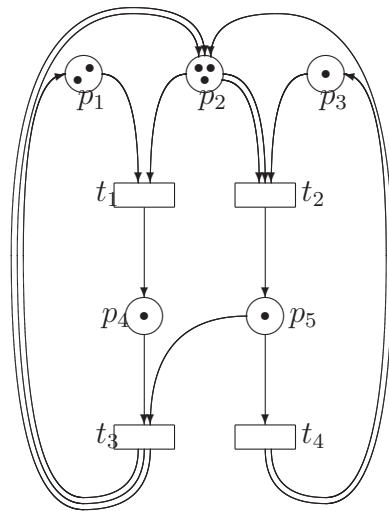
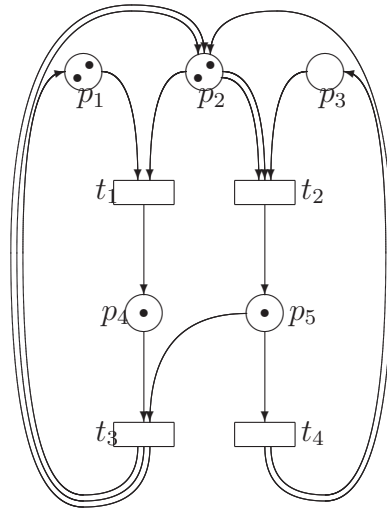


Abbildung 6.1: Ein Petri-Netz N

Es gibt sehr viele unterschiedliche Begriffe in der Literatur. Ist eine Transition t (in einer Konfiguration s) aktiviert, so sagt man auch, dass t *feuerbar* oder *enabled* sei. Für $s \vdash_t s'$ sagt man oft, dass t von s nach s' *schaltet* oder *feuert*. Nebenläufige Transitionen heißen im Englischen *concurrent*. Transitionen selbst heißen auch manchmal *Aktionen*, *Ereignisse* oder ähnlich. Konfigurationen heißen häufig auch *Markierung*, *marking* (im Englischen), *Zustand* oder ähnlich. Eine Übersetzung von auto-concurrent in etwa *selbst-nebenläufig* ist hingegen unüblich. Der Grund ist, dass sich die deutsche Petri-Netz-Schule der GMD nie für auto-concurrency interessiert hat, die internationale Schule aber schon. Statt $s \vdash_t$ und $s \vdash_t s'$ findet man meist $s [t >$ und $s [t > s'$. Eine Konfiguration ist also eine Abbildung von P auf \mathbb{N} , und wird, da P endlich

Abbildung 6.2: \mathcal{N} nach Feuern von t_1, t_2, t_3

und angeordnet ist, einfach als $|P|$ -dimensionaler Vektor $s = (x_1, \dots, x_n)$ aufgefasst mit $s(p_i) = x_i$.

Definition 6.1.4 Ein initiales Petri Netz $\mathcal{N} = (P, T, F, s_0)$ ist ein Petri Netz mit einer ausgezeichneten initialen Konfiguration s_0 . Die Erreichbarkeitsmenge $\mathcal{E}(s)$ von einer Konfiguration s aus ist

$$\mathcal{E}(s) := \{s' \mid s \vdash_{\mathcal{N}}^* s'\}.$$

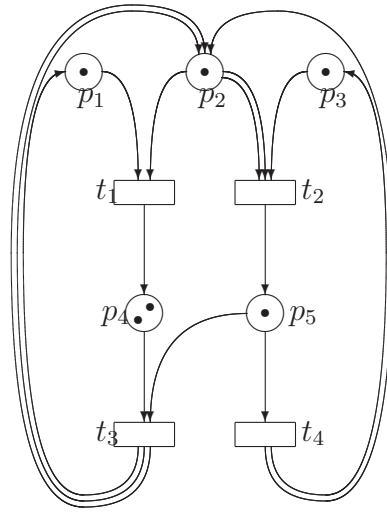
Die Erreichbarkeitsmenge $\mathcal{E}(\mathcal{N})$ von \mathcal{N} ist $\mathcal{E}(s_0)$.

Das Erreichbarkeitsproblem ist die Frage, ob für \mathcal{N}, s, s' gilt, dass in \mathcal{N} s' von s aus erreichbar ist, also ob $s' \in \mathcal{E}(s)$ gilt.

Definition 6.1.5 \mathcal{N} heißt

- k -beschränkt, falls jede Koordinate jeder erreichbaren Konfiguration in $\mathcal{E}(\mathcal{N})$ einen Wert $\leq k$ besitzt,
- beschränkt, falls \mathcal{N} k -beschränkt für irgendein k ist, und
- sicher, falls es 1-beschränkt ist.

In der deutschen Petri Netz Schule der GMD wurden fast ausschließlich sichere Petri Netze untersucht. Places heißen dann meist *Bedingungen*, Transitionen *Ereignisse* und Konfigurationen *Fälle*. Untersucht werden häufig Systemeigenschaften nebenläufiger Prozesse, wie Konflikten,

Abbildung 6.3: N nach Feuern von t_1, t_2, t_4

Konfusion, Synchronisationsabstand, etc. Im Gegensatz dazu untersuchte die amerikanische Schule hauptsächlich unbeschränkte Netze, deren Sprachen, Komplexität und Modellierfähigkeit zur Berechenbarkeit.

Im ersten Moment sieht das Erreichbarkeitsproblem recht einfach aus. Laut Definition gilt $m \vdash_{\Gamma} m'$ genau dann, wenn gilt

$$m \geq \bullet\Gamma \quad (6.1)$$

$$m' = m + \Delta_{\Gamma} \quad (6.2)$$

mit $\Delta_{\Gamma} = \Gamma\bullet - \bullet\Gamma$. Was kann man über das Tokenspiel sofort sagen? Es sei Γ eine Multimenge über $T = \{t_1, \dots, t_m\}$ und σ irgendeine Feuersequenz, in der jedes t_i genau $\Gamma(t_i)$ mal vorkommt, dann folgt sofort für beliebige Konfigurationen m, m', m'' :

- $m \vdash_{\Gamma} m' \iff m \vdash_{\sigma} m'$,
- $m \vdash_{\Gamma} m' \implies m + m'' \vdash_{\Gamma} m' + m''$.

Eine Konsequenz ist, dass man sich im wesentlichen auf das Feuern einzelner Transition konzentrieren kann. Untersuchungen zum Feuern einer ganzen Multimenge von nebenläufigen Transitionen in einem Schritt sind für fast alle elementaren Fragen in der Petri Netz Theorie unerheblich. Wir werden uns daher im folgenden auf Fragen beim Feuern einer einzelnen Transition oder einer Sequenz von einzelnen Transitionen beschränken.

Damit sieht das Tokenspiel wie ein lineares Additionssystem aus. Das ist aber falsch wegen der Bedingung (1), dass in der Konfiguration m

eine Transition t nur gefeuert werden darf, wenn $m \geq^\bullet t$ gilt. Eine Feuersequenz darf also niemals eine negative Tokenzahl “zwischen durch” auf Places bewirken. Es sei für ein Wort $\sigma = a_1 \dots a_k$ von Transitionen in T

$$\Delta_\sigma := \sum_{1 \leq i \leq k} a_i^\bullet - \bullet a_i,$$

die Summe aller Tokenänderungen in σ . Δ_σ heißt der *Wechsel* von σ . Wechsel, weil sich so die Tokenverteilung bei Feuern von σ ändert (wechselt). Die kleinste (bzgl. \leq in allen Koordinaten gemessen) Konfiguration, in der ein Wort σ von Transitionen feuertbar ist, ist eindeutig bestimmt und heißt die *Hürde* H_σ von σ . Für ein weiteres Wort σ' über T , das nur eine Permutation von σ ist, gilt dann natürlich

- $\Delta_{\sigma'} = \Delta_\sigma$,
- $H_\sigma \neq H_{\sigma'}$, im Allgemeinen, und
- $m \vdash_\sigma$ impliziert nicht $m \vdash_{\sigma'}$.

Kurz gesagt, es gilt $m \vdash_\sigma m'$ genau dann, wenn $m \geq H_\sigma$ und $m' = m + \Delta_\sigma$ ist.

Das Erreichbarkeitsproblem stellte sich als eines der schwierigsten Probleme in der Theoretischen Informatik überhaupt heraus. Es wurde erst nach ca 20 Jahren von Mayr gelöst, der einen Entscheidungsalgorithmus für diese Frage angeben konnte. Dieser Algorithmus ist allerdings äußerst schwierig und seine Komplexität kann noch nicht einmal durch eine Funktion abgeschätzt werden, die primitiv rekursiv ist, d.h., die durch ein LOOP-Programm berechnet werden kann. Es ist offen, ob es ein primitiv rekursives Entscheidungsverfahren für das Erreichbarkeitsproblem geben kann. Bekannt ist, dass jedes Entscheidungsverfahren dazu mindestens einen exponentiellen Speicherbedarf hat (Lipton [18]).

Petri Netze sind auch gern als ARM (vergleiche Definition ??) untersucht wurden. Dazu wird aber eine Verfeinerung des ARM, das sogenannte Transitionssystem, herangezogen.

Definition 6.1.6 *Ein Transitionssystem $A = (\mathcal{C}, \Sigma, \vdash)$ ist ein ARM (\mathcal{C}, \vdash) zusammen mit einer endlichen Menge Σ (von Transitionen), so dass zu jedem $t \in T$ ein ARM (\mathcal{C}, \vdash_t) existiert mit*

$$\vdash = \bigcup_{t \in \Sigma} \vdash_t.$$

$s \vdash_{tt'} s'$ ist eine Abkürzung für $\exists \hat{s} : s \vdash_t \hat{s} \vdash_{t'} s'$, und $s \vdash_{tt'}$ für $\exists \hat{s} : s \vdash_{tt'} \hat{s}$. Ein Transitionssystem $A = (\mathcal{C}, \Sigma, \vdash)$ heißt

- lokal determiniert, falls $s \vdash_t s' \wedge s \vdash_t s'' \implies s' = s''$,
- kommutativ, falls $s \vdash_{tt'} \wedge s \vdash_{t't} \implies \exists \hat{s} : s \vdash_{tt'} \hat{s} \wedge s \vdash_{t't} \hat{s}$,
- persistent, falls $t \neq t' \wedge s \vdash_t \wedge s \vdash_{t'} \implies s \vdash_{tt'}$,
- konfluent, falls $s \vdash^* s' \wedge s \vdash^* s'' \implies \exists \hat{s} : s' \vdash^* s''' \wedge s'' \vdash^* \hat{s}$,
- monoton, falls auf \mathcal{C} eine Addition $+$ erklärt ist und $s \vdash_t s' \implies (s + s'') \vdash_t (s' + s'')$,

jeweils für alle $t, t' \in \Sigma, s, s', s'' \in \mathcal{C}$.

Ein Petri Netz $\mathcal{N} = (P, T, F)$ fassen wir als das Transitionssystem $\mathcal{N} = (\mathcal{C}, T, \vdash)$ auf mit $\mathcal{C} = \mathbb{N}^P$ und \vdash_t wie gerade erklärt für $t \in T$. Für ein initiales Petri Netz kann man alternative $\mathcal{C} = \mathcal{E}(\mathcal{N})$ setzen. Damit sind Petri Netze lokal determiniert, kommutativ und monoton, im Allgemeinen aber nicht persistent oder konfluent. Konfluenz wird auch als *Church-Rosser-Eigenschaft* bezeichnet.

6.2 Überdeckungsgraphen und Invarianten

Wir stellen hier zwei wichtige Analysetechniken für Petri Netze vor, den Überdeckungsgraphen und Invarianten von Petri Netzen.

Definition 6.2.1 *Der Erreichbarkeitsgraph $ER(\mathcal{N})$ eines Petri Netzes $\mathcal{N} = (P, T, F, s_0)$ ist der kantengewichtete, gerichtete Graph $ER(\mathcal{N}) = (V, E, T)$ mit*

- $V = \mathcal{E}(\mathcal{N})$,
- eine gerichtete Kante (m, t, m') mit Gewicht t führt genau dann von dem Knoten m zum Knoten m' , wenn $m \vdash_t m'$ gilt.

Offensichtlich ist $ER(\mathcal{N})$ genau dann ein endlicher Graph, wenn \mathcal{N} beschränkt ist. Da ein endlicher Erreichbarkeitsgraph nur eine andere Darstellung für einen endlichen Automaten ist, haben beschränkte Petri Netze also keine größere Ausdrucksfähigkeit als endliche Automaten. Natürlich erlauben beschränkte Petri Netze aber eine elegantere Modellierung nebenläufiger Prozesse, wie z.B. Konflikte, etc. Unendliche Erreichbarkeitsgraphen sind natürlich kein schönes Analysetool. Ein Ersatz sind hier so genannte Überdeckungsgraphen, die stets endlich sind, und Erreichbarkeit annäherungsweise modellieren. Dabei wird für einen Place

ein ‐Zustand‐ ω zugelassen, falls auf diesen Place unbeschränkt viele Token gelegt werden können. Dem Überdeckungsgraphen kann man auf den ω -Places nicht mehr ansehen, welche Tokenanzahl dort exakt erreichbar ist, nur, dass unbeschränkt viele erreichbar sind, d.h., dass jede endliche Zahl von Token dort ‐überdeckt‐ werden kann.

Definition 6.2.2 *Es sei $\mathcal{N} = (P, T, F, s_0)$ ein initiales Petri Netz. Ein Überdeckungsgraph $UG(\mathcal{N})$ von \mathcal{N} ist ein kantengewichteter, gerichteter Graph $G = (V, E, T)$, der wie folgt konstruiert wird.*

$V := \{s_0\};$

$E := \emptyset;$

$Neu := V;$

solange $Neu \neq \emptyset$

 wähle ein $\hat{s} \in Neu;$

 für alle $t \in T$

 falls $\hat{s} \vdash_t:$

 berechne \hat{s}' mit $\hat{s} \vdash_t \hat{s}';$

 falls ein $s' \in (\mathbb{N} \cup \omega)^P$ auf Pfad von s nach \hat{s} im bereits

 erzeugten Teilgraphen (V, E, T) existiert mit $s' < \hat{s}'$

 setze $\hat{s}'(p) := \omega$ für $p \in P$ mit $s'(p) < \hat{s}'(p)$

$E := E \cup \{(\hat{s}, t, \hat{s}')\};$

 falls $\hat{s}' \notin V$ setze $V := V \cup \{\hat{s}'\}$ und $Neu := Neu \cup \{\hat{s}'\};$

$Neu := Neu \setminus \{\hat{s}\}$

Existiert in $UG(\mathcal{N})$ ein Knoten s' und ein Place p mit $s'(p) = \omega$, so heißt p auch ω -Place oder ω -Koordinate.

Das Ergebnis dieser Konstruktion hängt von der Reihenfolge ab, in der man ein \hat{s} aus Neu und die Transitionen in der ‐für alle $t \in T$ ‐ Schleife auswählt. Es ist aber für eine weitere Analyse unerheblich, welche der Varianten man wählt, und man spricht deshalb auch einfach, aber fälschlich, von ‐dem Überdeckungsgraphen‐ von \mathcal{N} . In Abbildung 6.4 ist ein Petri Netz angegeben, aus dem man, je nach Reihenfolge, in der man den Algorithmus anwendet, zwei verschiedene Überdeckungsgraphen erhalten kann, die beide in Abbildung 6.5 dargestellt sind.

Lemma 6.2.1 *Es gilt für alle Überdeckungsgraphen G eines jeden Petri Netzes \mathcal{N} :*

- G ist stets endlich,
- alle Überdeckungsgraphen besitzen die gleichen ω -Places,

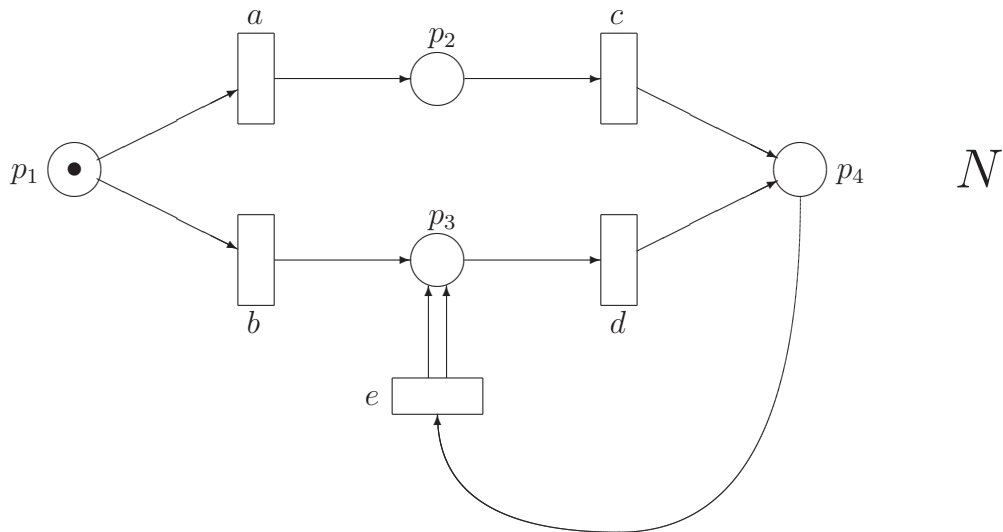


Abbildung 6.4: Ein Petri-Netz N mit verschiedenen Überdeckungsgraphen

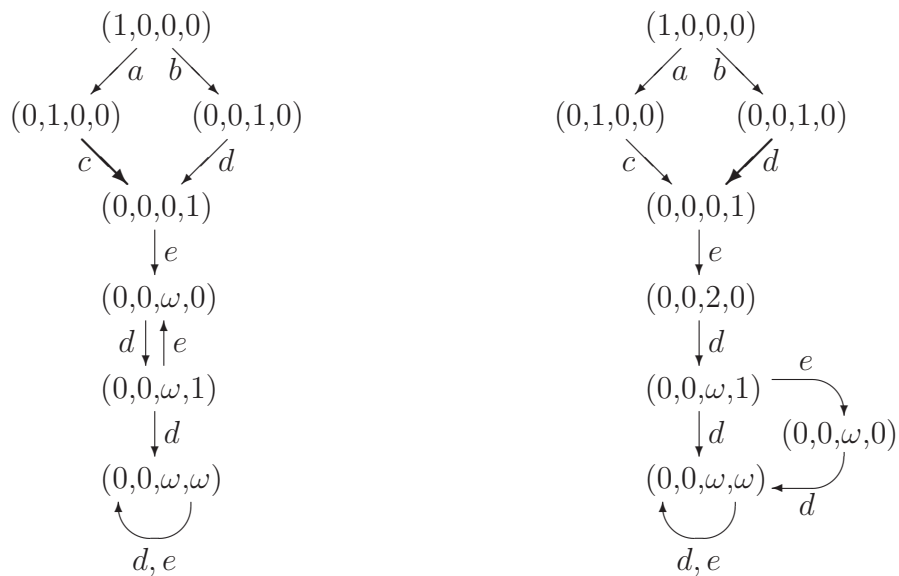


Abbildung 6.5: Überdeckungsgraphen zum Petri-Netz aus Abbildung 6.4. Links wird $(0,1,0,0)$ zuletzt betrachtet, rechts $(0,0,1,0)$. Die fett gedruckten Kanten werden daher zuletzt eingefügt

- ist p ein ω -Place, so existiert zu jedem $k \in \mathbb{N}$ eine erreichbare Konfiguration $s \in \mathcal{E}(\mathcal{N})$ mit $s(p) > k$ (d.h. der Place p ist unbeschränkt),
- ist x ein Knoten in G , in dem sowohl die Koordinaten p und p' den Wert ω besitzen, so existiert zu jedem $k \in \mathbb{N}$ eine erreichbare Konfiguration $s \in \mathcal{E}(\mathcal{N})$ mit $s(p) > k$ und $s(p') > k$ (p und p' sind simultan unbeschränkt),
- \mathcal{N} ist genau dann beschränkt, wenn G keinen ω -Place besitzt.

Ein Überdeckungsgraph ohne ω -Places ist im Wesentlichen der endliche Erreichbarkeitsgraph eines beschränkten Petri Netzes. Beide Graphen besitzen nur eine etwas unterschiedliche Darstellung, da im Überdeckungsgraphen Konfigurationen mehr als einmal auftreten dürfen. Die Beweise zu dem Lemma sind nicht schwer, aber auch nicht direkt trivial. Für die Endlichkeit braucht man z.B. eine interessante Variante von Königs Lemma. Zum ersten Mal wurden Überdeckungsgraphen von Karp und Miller im Zusammenhang mit Programm Schemata eingeführt.

Invarianten

Invarianten beschreiben unveränderliche Eigenschaften in dynamischen Systemen. Die Invariantentheorie von Petri Netzen geht im Wesentlichen auf Kurt Lautenbach [16] zurück. Sie lässt sich am besten mittels Matrizenrechnung vermitteln.

Hierzu unterteilt man F eines Petri Netzes (P, T, F) in zwei Matrizen, \mathbb{F} , die Vorwärtsmatrix, und \mathbb{B} , die Rückwärtsmatrix, die jeweils vom Standpunkt der Places aus sagen, wie viele Kanten vorwärts (in \mathbb{F}), also aus einem Place heraus, zu welchen Transitionen führen, bzw. wie viele Kanten rückwärts (in \mathbb{B}), also zu dem Place von welchen Transitionen aus, führen.

Definition 6.2.3 Die Vorwärtsmatrix \mathbb{F} und Rückwärtsmatrix \mathbb{B} zu einem Petri Netz $\mathbb{N} = (P, T, F)$ sind zwei $|P| \times |T|$ -Matrizen über \mathbb{N} definiert als

$$\mathbb{F}(p, t) := F(p, t) \quad \text{und} \quad \mathbb{B}(p, t) := F(t, p).$$

Eine T -Invariante ist ein $|T|$ -dimensionaler **Spaltenvektor** I_T über \mathbb{Z} mit

$$\mathbb{F} \cdot I_T = \mathbb{B} \cdot I_T.$$

Eine P -Invariante ist ein $|P|$ -dimensionaler **Zeilenvektor** I_P über \mathbb{Z} mit

$$I_P \cdot \mathbb{F} = I_P \cdot \mathbb{B}.$$

Insbesondere gilt $\bullet t = \mathbb{F}(t)$, $t\bullet = \mathbb{B}(t)$. Eine T -Invariante ist eine Gewichtung aller Transitionen, bei der die gewichteten Summen aller Inputplaces mit denen der Outputplaces übereinstimmen. Hat man also eine Feuersequenz, in der jede Transition genauso häufig vorkommt wie in der Gewichtung angegeben, so verändert das Feuern dieser Sequenz den Zustand des Petri Netzes nicht. Eine P -Invariante ist eine Tokengewichtung für jeden Place, so dass sich bei Feuern einer Transition die gewichtete Summe aller Token im Netz nicht ändert. Zu beachten ist, dass in beiden Invarianten negative Gewichte auftreten können. Für nicht negative T -Invarianten gilt

Lemma 6.2.2 Sei $\mathcal{N} = (P, T, \mathbb{F}, \mathbb{B})$. Ein nicht-negativer Vektor $I_T \in \mathbb{N}^T$ ist eine T -Invariante, genau dann, wenn eine Feuersequenz $\sigma \in T^*$ mit $\#_t(\sigma) = I_T(t)$ für alle $t \in T$ existiert, so dass für $s \geq H(\sigma)$ gilt $s \vdash_\sigma s$.

Beweis. Wir rechnen aus: $\Delta_\sigma = \sum_{t \in T} \#_t(\sigma) \cdot (\mathbb{F}(t) - \mathbb{B}(t)) = \sum_{t \in T} I_T(t) \cdot (\mathbb{F}(t) - \mathbb{B}(t)) = \sum_{t \in T} I_T(t) \cdot (\mathbb{F} - \mathbb{B})(t) = (\mathbb{F} - \mathbb{B}) \cdot I_T = \mathbb{F} \cdot I_T - \mathbb{B} \cdot I_T$. Damit gilt $s \vdash_\sigma s \iff \Delta_\sigma = 0 \iff I_T$ ist T -Invariante. ■

Dies hat eine hübsche Konsequenz: findet sich im Erreichbarkeitsgraph ein Kreis mit Beschriftung σ , so gilt $s \vdash_\sigma s$ für $s \geq H_\sigma$, und σ definiert damit die T -Invariante I_T mit $I_T = \#_t(\sigma)$.

P -Invarianten sind Tokengewichtungen, so dass sich beim Feuern einer Transition die gewichtete Summe der Token im Gesamtnetz nicht ändert. Für P -Invarianten gilt

Lemma 6.2.3 Seien $\mathcal{N} = (P, T, F)$ ein Petri Netz, $I_P \in \mathbb{Z}^P$ eine beliebige P -Invariante von N , $s, s' \in \mathbb{N}^P$ Konfigurationen und $\sigma \in T^*$ eine Feuersequenz mit $s \vdash_\sigma s'$, so gilt $I_P \cdot s = I_P \cdot s'$.

Beweis. Es gelte $s \vdash_t s'$. Damit gilt auch

$$I_P \cdot s' = I_P \cdot (s + \mathbb{F}(t) - \mathbb{B}(t)) = I_P \cdot s + I_P \cdot (\mathbb{F}(t) - \mathbb{B}(t)) = I_P \cdot s.$$

■

Da P - und T -Invarianten Lösungen homogener linearer Gleichungssysteme

$$(\mathbb{F} - \mathbb{B}) \cdot I_P = 0 \quad I_T \cdot (\mathbb{F} - \mathbb{B}) = 0$$

sind, gilt sofort

Lemma 6.2.4 *Sind I_1 und I_2 (P - oder T -) Invarianten und ist $n \in \mathbb{Z}$, so sind auch $I_1 + I_2$ und $n \cdot I_1$ Invarianten.*

Ein weiteres Kriterium zur Beschränktheit von Petri Netzen lässt sich aus den P -Invarianten ableiten. Dieses Kriterium ist zwar nur hinreichend, d.h. es existieren beschränkte Netze, die es nicht erfüllen, hat jedoch einen entscheidenden Vorteil gegenüber dem Ansatz über den Überdeckungsgraphen. Der Überdeckungsgraph eines k -beschränkten Netzes kann, wenn P die Menge der Places ist, bis zu $k^{|P|}$ verschiedene erreichbare Zustände besitzen. All diese Zustände müssen untersucht werden, bevor eine positive Entscheidung getroffen werden kann, bevor man also feststellen kann, dass das Petri Netz k -beschränkt ist. (Ist das Netz unbeschränkt, so lässt sich dies bei Einführen einer ω -Koordinate eventuell früher feststellen.) Für den gleich vorzustellenden Ansatz muss man lediglich eine Basis des P -Invariantenraums ermitteln, und die bekommt man mittels linearer Algebra durch Lösen des Gleichungssystems $I_P(\mathbb{F} - \mathbb{B}) = 0$ selbst mit den einfachsten Algorithmen mit quadratischem Aufwand. Sodann muss man nur noch feststellen, ob es in diesem Invariantenraum Vektoren mit ausschließlich positiven Werten gibt, dies ist ebenfalls mittels linearer Algebra schnell zu lösen. Ist dies der Fall, so zeigt uns das folgende Lemma, dass das untersuchte Petri Netz beschränkt ist.

Lemma 6.2.5 *Seien $\mathcal{N} = (P, T, \mathbb{B}, \mathbb{F}, s_0)$ ein initiales Petri Netz und I_P eine positive P -Invariante von \mathcal{N} , d.h. $I_P(p) > 0$ für alle $p \in P$. Dann ist \mathcal{N} beschränkt.*

Beweis. Wir wählen $k := I_P \cdot s_0 = \sum_{p \in P} I_P(p) \cdot s_0(p)$. Seien $s \in \mathcal{E}(\mathcal{N})$ ein erreichbarer Zustand und $\sigma \in T^*$ eine Feuersequenz mit $s_0 \vdash_\sigma s$. Dann gilt auch $I_P \cdot s = k$. Also ist $I_P \cdot s = k$ für alle $s \in \mathcal{E}(\mathcal{N})$. Wegen $I_P(p) > 0$ für alle $p \in P$ muss damit $s(p) \leq k$ für alle $p \in P$ gelten, also ist \mathcal{N} k -beschränkt. ■

6.3 Lebendigkeit

Lebendigkeit ist ein gutartiges Verhalten eines Petri Netzes, das einer Transition die Feuerbarkeit zu einem späteren Zeitpunkt zusichert. Je nach Stärke dieser Zusicherung unterscheiden wir vier verschiedene Grade an Lebendigkeit.

Definition 6.3.1 Eine Transition t heißt im Zustand s

- 1-lebendig $\iff \exists s' \in \mathcal{E}(s) : s' \vdash t$,
- 2-lebendig $\iff \forall n \in \mathbb{N} \exists \sigma_1, \dots, \sigma_n \in T^* : s \vdash_{\sigma_1 t \sigma_2 t \dots \sigma_n t}$,
- 3-lebendig $\iff \exists \sigma \in T^\omega$ (unendlich lang) mit $s \vdash_\sigma$ und
 t kommt in σ unendlich oft vor,
- 4-lebendig $\iff \forall s' \in \mathcal{E}(s) \exists \sigma \in T^* : s' \vdash_{\sigma t}$
- lebendig $\iff t$ ist 4-lebendig,
- tot $\iff t$ ist nicht 1-lebendig.

\mathcal{N} selbst heißt

- lebendig $\iff \mathcal{N}$ ist im initialen Zustand lebendig,
- wohlgeformt $\iff \mathcal{N}$ ist sicher und lebendig.

Dabei definiert man für die 3-Lebendigkeit, dass ein unendlich langes Wort $\sigma \in T^\omega$ von s aus feuerebar ist, $s \vdash_\sigma$, falls $s \vdash_{\sigma'}$ für jedes endliche Anfangsstück σ' von σ gilt.

Zu beachten ist, dass tot nicht das Gegenteil von lebendig ist. Tot bedeutet, nie mehr feuern zu können. Lebendigkeit meint hier “ewig” lebendig, egal wie das Petri Netz arbeitet, die Transition wird niemals tot.

Lemma 6.3.1 Für beliebige Petri Netze \mathcal{N} , Transitionen t und Zustände s gilt:

$$\begin{array}{ccccccc}
 t \text{ ist in } s & \implies & t \text{ ist in } s & \implies & t \text{ ist in } s & \implies & t \text{ ist in } s \\
 4\text{-lebendig} & \not\iff & 3\text{-lebendig} & \not\iff & 2\text{-lebendig} & \not\iff & 1\text{-lebendig}.
 \end{array}$$

Übung 6.3.1 Beweisen Sie das obige Lemma. Die Richtungen \implies sind leicht. Für die Rückrichtungen geben Sie bitte Gegenbeispiele an. Ein Gegenbeispiel für eine 3-lebendigen Netzes, das nicht 4-lebendig ist, ist nicht wirklich schwer, bedarf aber schon einiger Überlegung.

Lebendige Netze sind in der Praxis oft von großer Bedeutung. Beschreibt ein Petri Netz zum Beispiel den Ablauf eines sich (evtl. mit Abweichungen) wiederholenden Fertigungsprozesses, so möchte man natürlich Fehlerquellen ausschließen, die den Prozess zum Halten bringen. Sind solche Fehlerquellen im Prozess vorhanden, so ist das beschreibende Petri Netz nicht lebendig. Dies würde man gern von vornherein feststellen können.

Lebendigkeit und der Überdeckungsgraph

Mittels des Überdeckungsgraphen lassen sich einige Lebendigkeitseigenschaften feststellen, so die 1- und 2-Lebendigkeit von Transitionen.

Lemma 6.3.2 *Es ist mit dem Überdeckungsgraphen entscheidbar, ob eine Transition eines Petri Netzes in einem Zustand s 1-lebendig oder 2-lebendig ist.*

Beweis. Eine Transition t ist offensichtlich im Zustand s 1-lebendig genau dann, wenn im Überdeckungsgraphen $UG(N, s)$ von \mathcal{N} eine Kante mit Beschriftung t existiert.

Eine Transition t ist im Zustand s 2-lebendig genau dann, wenn im Überdeckungsgraphen $UG(N, s)$ ein Kreis mit einer mit t beschrifteten Kante existiert. ■

Die 3- und 4-Lebendigkeit kann man hingegen am Überdeckungsgraphen nicht ablesen, wie wir gleich sehen werden. Das heißt natürlich nicht, dass diese Probleme unentscheidbar sind.

Lemma 6.3.3 *Sei $\mathcal{N} = (P, T, \mathbb{B}, \mathbb{F}, s_0)$ ein initiales Petri Netz. Mit Hilfe des Überdeckungsgraphen $UG(N)$ lässt sich nicht entscheiden, ob*

1. eine Transition t 3-lebendig ist,
2. eine Transition t lebendig ist,
3. das Netz \mathcal{N} lebendig ist,
4. eine gegebene Konfiguration erreichbar ist.

Beweis. Wir zeigen jeden dieser Punkte durch Angabe zweier Petri Netze, von denen eins die jeweilige Bedingung erfüllt und das andere nicht, und die denselben Überdeckungsgraphen besitzen. Zu 1. und 2. sind in Abbildung 6.6 zwei Petri Netze zu sehen, in denen die Transition t_3 im ersten Netz lebendig, im zweiten jedoch nicht einmal 3-lebendig ist. Der gemeinsame Überdeckungsgraph ist ebenfalls abgebildet. Zu 3. sehen wir uns Abbildung 6.7 an. Das erste Netz ist lebendig, im zweiten führt jedoch $(1, 0, 0) \xrightarrow{t_1 t_2 t_3} (0, 0, 1)$ in einen Zustand, in dem das Netz tot ist. Für Punkt 4. schließlich zeigt Abbildung 6.8 zwei Petri Netze mit unterschiedlichen Erreichbarkeitsgraphen aber gleichem Überdeckungsgraphen. Insbesondere ist der Zustand $(1, 1)$ im ersten Netz erreichbar, im zweiten jedoch nicht.

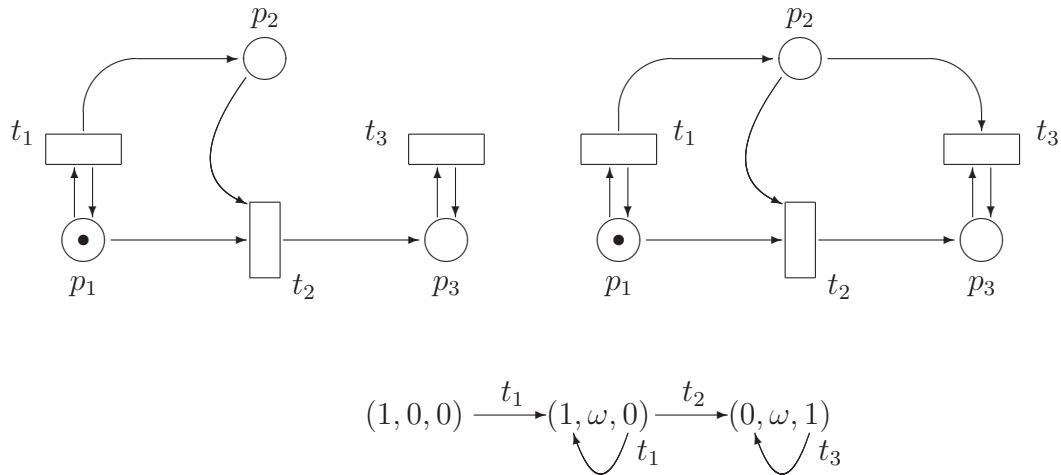


Abbildung 6.6: Zwei Petri-Netze mit demselben Überdeckungsgraphen. Im linken Netz ist t_3 lebendig, im rechten nur 2-lebendig

■
■

Betrachten wir jedoch Lebendigkeit zusammen mit der Sicherheit, also die Wohlgeformtheit von Netzen, so lässt sich die Entscheidung wiederum mit Hilfe des Überdeckungsgraphen fällen.

Lemma 6.3.4 *Es ist anhand des Überdeckungsgraphen entscheidbar, ob Petri Netze wohlgeformt sind.*

Beweis. Ein Petri Netz $\mathcal{N} = (P, T, \mathbb{B}, \mathbb{F}, s_0)$ ist genau dann beschränkt, falls $UG(N, s_0) = EG(N, s_0)$ ist. In diesem Fall ist die Lebendigkeit jetzt leicht in $EG(N, s_0)$ testbar: \mathcal{N} ist nämlich genau dann lebendig, wenn es für jede Transition von jedem Knoten im Erreichbarkeitsgraphen aus einen Weg gibt, an dem diese Transition als Kanteninschrift vorkommt.

■

Lebendigkeit und T -Invarianten

Auch T -Invarianten können zu Lebendigkeitsuntersuchungen herangezogen werden, insbesondere wenn das zu untersuchende Petri Netz beschränkt ist.

Lemma 6.3.5 *Sei $\mathcal{N} = (P, T, \mathbb{B}, \mathbb{F}, s_0)$ ein lebendiges, beschränktes Petri Netz. Dann existiert eine positive T -Invariante I_T von \mathcal{N} , d.h. $\forall t \in T$:*

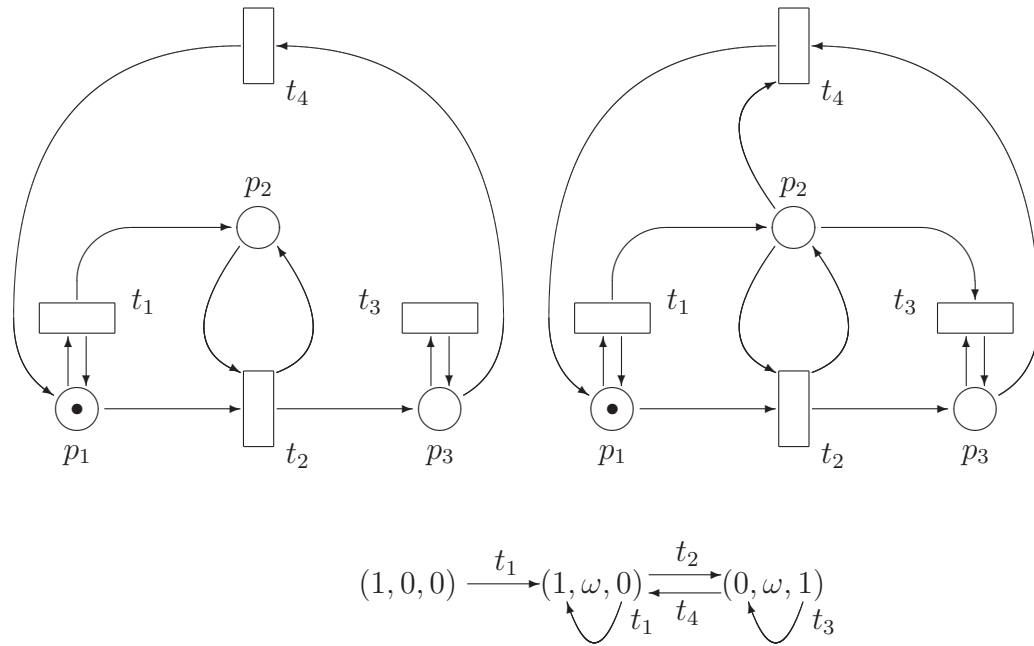


Abbildung 6.7: Zwei Petri-Netze mit demselben Überdeckungsgraphen. Nur das linke Netz ist lebendig, das rechte ist nach Feuern von $t_1 t_2 t_3$ tot

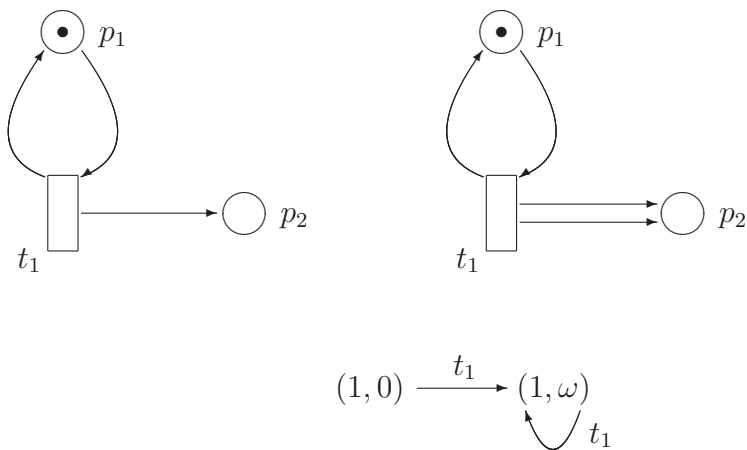


Abbildung 6.8: Zwei Petri-Netze mit verschiedenen Erreichbarkeitsmengen, aber gleichem Überdeckungsgraphen

$I_T(t) > 0$.

Beweis. Da \mathcal{N} beschränkt ist, ist der Erreichbarkeitsgraph $EG(N)$ endlich. Sei $t \in T$ eine feste Transition. Wegen der Lebendigkeit von t muss nun im Erreichbarkeitsgraphen ein Kreis existieren, der t enthält und von s_0 aus erreichbar ist. Der Kreis stellt offensichtlich eine Feuersequenz σ mit $W_\sigma = 0$ dar. Damit existiert eine T -Invariante I_t , in der jeder Transition die Anzahl ihrer Vorkommen auf dem Kreis zugeordnet wird. Diese Invariante ist nicht-negativ und es gilt $I_t(t) > 0$. Wir ermitteln für jede Transition t eine solche T -Invariante I_t und erhalten durch Summation die neue, positive T -Invariante $I_T = \sum_{t \in T} I_t$. ■

Man benutzt nun den Umkehrschluss: Existiert keine positive T -Invariante in einem beschränkten Netz, so kann das Netz nicht lebendig sein.

6.4 Notwendige Kriterien zur Erreichbarkeit

Es erweist sich, wie wir auch im kommenden Kapitel noch sehen werden, als sehr schwierig, zu entscheiden, ob eine bestimmte Konfiguration erreichbar ist. Hin und wieder bieten aber Überdeckungsgraph oder P -Invarianten eine einfache Möglichkeit, festzustellen, ob eine Konfiguration **nicht** erreichbar ist. Diese jetzt vorgestellten Methoden funktionieren aber nicht immer, da wir hier nur notwendige aber nicht notwendig hinreichende Kriterien für die Erreichbarkeit ausnutzen.

Unerreichbarkeit via Überdeckungsgraph

Lemma 6.4.1 *Seien $\mathcal{N} = (P, T, \mathbb{B}, \mathbb{F}, s_0)$ ein Petri Netz und $s \in \mathbb{N}^P$ eine Konfiguration. Gilt $s \in \mathcal{E}(\mathcal{N})$, so existiert im Überdeckungsgraph ein Knoten s' mit $s' \geq s$.*

Wir können also am Überdeckungsgraphen trivialerweise erkennen, dass eine Konfiguration unerreichbar ist, wenn sie von keinem Knoten von $UG(N)$ überdeckt wird. Sei \mathcal{N} ein Petri Netz einer Größe n , so ist die Größe $f(n)$ eines Überdeckungsgraphen $UG(\mathcal{N})$ stets endlich und von \mathcal{N} abhängig, aber es gibt Fälle in denen f noch nicht einmal primitiv rekursiv ist. In solchen Fällen ist die Konstruktion von $UG(\mathcal{N})$ nur

theoretisch möglich aber nicht praktisch für etwas größere Petri Netze \mathcal{N} .

Unerreichbarkeit via P -Invarianten

Vom Aufwand her weit günstiger ist wiederum der Ansatz über Invarianten, da man bei der Berechnung von P -Invarianten mit dem Lösen eines linearen Gleichungssystems auskommt. Ist etwa s ein Zustand, von dem wir zeigen möchten, dass er unerreichbar ist, so können für die uns bekannten P -Invarianten I_P prüfen, ob $I_P \cdot s_0 \neq I_P \cdot s$ gilt. Sollte dies der Fall sein, so kann nach Lemma 6.2.3 s nicht von s_0 aus erreichbar sein.

Keine der in diesem Kapitel vorgestellten Techniken erlaubt eine unmittelbare Lösung des Erreichbarkeitsproblems oder des Lebendigkeitsproblems (die Frage, ob ein Petri Netz lebendig ist). Allerdings sind Überdeckungsgraphen ein entscheidendes Hilfsmittel im Beweis der Entscheidbarkeit der Erreichbarkeitsproblems, den wir hier nicht vorstellen werden.

6.5 Petri Netz Klassen

Wir werden hier Petri Netze nach Unterschieden in der Struktur ihrer Graphen charakterisieren, erstmal unabhängig von den Konfigurationsmöglichkeiten.

Definition 6.5.1 *Ein Petri Netz $\mathcal{N} = (P, T, F)$ heißt*

- markierter Graph, falls $|\bullet p| = |p^\bullet| = 1$ gilt,
- Zustandsmaschine, falls $|\bullet t| = |t^\bullet| = 1$ gilt,
- free choice oder FC Netz, falls für $t' \in p^\bullet$ bereits $|\bullet t'| = 1$ oder $|p^\bullet| = 1$ folgt,
- simpel, falls für $p', p'' \in \bullet t$ mit $|p^\bullet| > 1$ und $|p''^\bullet| > 1$ bereits $p' = p''$ folgt,

jeweils für alle $p \in P, t \in T$.

In markierten Graphen kommen Token auf einem Place von genau einer Transition und können auch nur zu einer Transition führen. Damit sind Konflikte und lokaler Indeterminismus ausgeschlossen. Lokaler Indeterminismus meint hier, dass zwei Transitionen aus p^\bullet feuern können, und es indeterminiert ist, welche feuert. Ein globaler Indeterminismus

kann nur durch Wettrennen verschiedener Token an verschiedenen Orten des Netzes stattfinden. Places sind trivial, sie könnten durch gerichtete kanten im Graphen ersetzt werden, die allerdings jetzt Token (Markierungen) tragen dürfen, daher der Name. In Zustandsmaschinen sind die Transitionen komplett trivial und nichts anderes als ‐Leitungen‐, die Signale (Token) weitergeben. Lokaler Indeterminismus im obigen Sinn ist möglich, nicht aber Konflikte oder Konfusion, etc. In einem FC Netz ist jede gerichtete Kante die einzige ausgehende Kante aus einem Place oder die einzige in eine Transition führende Kante. Ist in einem FC Netz ein Place p Inputplace zweier verschiedener Transitionen t_1, t_2 , so besitzen beide ausschließlich p als Inputplace. Trägt p einen Token, sind beide Transitionen feuerbar. Beide können nun selbst entscheiden (free choice), welche feuert, da dies von keinen weiteren Bedingungen mehr abhängen kann. Dies ist in einem simplen Netz nicht mehr der Fall. Eine feuerbare Transition t mit einem Token auf ihren Inputplcae p kann ihre Feuereigenschaft verlieren, weil eine weitere Transition t' den Token von p entfernen kann. Besitzt t mehrere Inputplaces, so kann von einer fremden Transition t' ein Token nur von maximal einem Inputplace von t entfernt werden. In einem simplen Netz kann eine Transition mehrere Inputplaces besitzen, von denen aber höchstens einer noch Inputplace einer weiteren Transition sein darf. Abbildung 6.9 zeigt einige Beispiele.

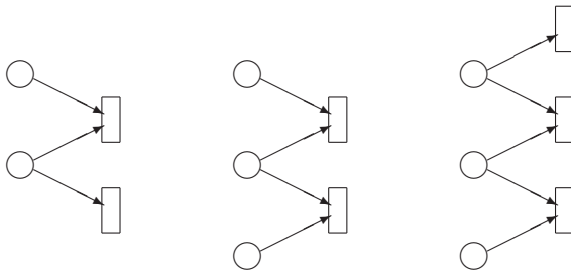


Abbildung 6.9: Links: nicht FC, mitte: simples PN, rechts: nicht simpel

Offensichtlich gilt syntaktisch, dass markierte Graphen und Zustandsmaschinen echte Teilklassen von FC Netzen, diese eine echte Teilklassse von simplen Netzen, und diese wieder eine echte Teilklassse aller Petri Netz bilden. Dies gilt nicht nur syntaktisch sondern auch semantisch: bei vielen, aber nicht allen, sinnvollen Verhaltensbegriffen erhält man ebenfalls diese Klasseneinteilung.

Ein Verhaltensbegriff, in dem man stets mit FC Netzen auskommt, ist indeterminiertes Raten. Wenn man z.B. sagt, dass ein Petri Netz \mathcal{N}' ein Petri Netz \mathcal{N} simuliert, falls die Erreichbarkeitsmenge von \mathcal{N} sich in der

von \mathcal{N}' codiert wieder findet, dann kann man jedes Petri Netz durch ein FC simulieren. Das mittlere Netz in Abbildung 6.9 wird dann durch das FC aus Abbildung 6.10 simuliert.

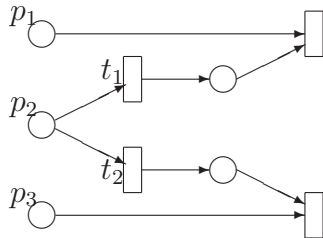


Abbildung 6.10: “Simuliert” das mittlere simple Netz aus Abbildung 6.9

Diese Simulation kann bei falschem Raten allerdings Blockierungen einführen. Liegt z.B. je ein Token auf p_1 und p_2 und t_2 rät fälschlich zu feuern, so kann dadurch das gesamte Netz blockiert werden, wohingegen bei “korrekten” Feuern von t_1 eine Fortsetzung möglich wird. In manchen Verhaltensbegriffen, wie in diversen Sprachkonzepten von Petri Netzen, können diese Blockierungen maskiert werden, da dort nur diejenigen Feuersequenzen betrachtet werden, die von gewissen Start- zu gewissen Endmarkierungen führen. Damit lassen sich dann alle Petri Netz Sprachen bereits von FC Netzen erzeugen. Fordert man allerdings sinnvolle Simulationskonzepte, die keine Blockierungen oder Endlosschleifen einführen können, so sind FC Netzwerke weniger leistungsfähig als simple, und simple weniger leistungsfähig als allgemeine Petri Netze.

Wohlgeformtheit von markierten Graphen und FCNetzen

Zwei Transitionen t_1, t_2 stehen im Konflikt, wenn beide feuierbar sind, das Schalten der einen aber das Schalten der anderen verhindert. Eine Konfiguration heißt hingegen *konfus*, wenn zwei Transitionen t_1, t_2 im Konflikt stehen, dieser Konflikt aber durch Feuern einer weiteren Transition t_3 aufgelöst wird. Konfusion ist in FC Netzen nicht möglich. Dies ist der tiefere Grund, weshalb eine Analyse von Wohlgeformtheit (das initiale Petri Netz ist sicher und lebendig) in FC Netzen recht einfach möglich ist. Ob markierte Graphen wohlgeformt sind wurde bereits in einer bekannt gewordenen Arbeit von Commoner, Holt, Even und Pnueli [5] 1971 untersucht.

Für Fragen der Lebendigkeit sind Sperren (Deadlocks) und Fallen (Traps) kritische Bestandteile eines Petri Netzes. Dabei ist eine Sper-

re eine Menge von Places, die, wenn sie einmal alle Token verloren hat, niemals mehr Token erhalten kann. Umgekehrt ist eine Falle eine Menge von Places, die, wenn sie einmal einen Token enthält, niemals alle Token verlieren kann.

Definition 6.5.2 *Es sei $\mathcal{N} = (P, T, F)$ ein Petri Netz. Eine Teilmenge $S \subseteq P$ von Places heißt*

- Sperre, falls $\bullet S \subseteq S^\bullet$ gilt, und
- Falle, falls $S^\bullet \subseteq \bullet S$ gilt,
- markiert in der Konfiguration s , falls mindestens ein $p \in S$ existiert mit $s(p) > 0$.

Der Name Falle ist klar, da “gefangene” Token eine Falle nicht komplett verlassen können. Eine markierte Falle kann nie leer werden. Der Name Sperre deutet an, dass eine leere, d.h. unmarkierte, Sperre niemals wieder einen Token enthalten kann, also eine Tokensperre darstellt.

Lemma 6.5.1 *Es seien $\mathcal{N} = (P, T, F)$ ein Petri Netz, $S \subseteq P$ eine Menge von Places, $s \in S \subseteq \mathcal{N}^P$ eine Konfiguration und $s' \in \mathcal{E}(s)$. Dann gilt*

- $s = 0 \implies s' = 0$, falls S eine Sperre ist,
- $s \neq 0 \implies s' \neq 0$, falls S eine Falle ist.

Die Vereinigung und der Durchschnitt von Sperren ist wieder eine Sperre. Die Vereinigung und der Durchschnitt von Fallen ist wieder eine Falle.

Damit sind für Fragen der Lebendigkeit Fallen wünschenswerte Teilstrukturen und Sperren gefährliche. Eine Sperre ist immer dann harmlos, wenn sie eine markierte Falle enthält. Dann kann die Sperre nicht geleert werden und operiert nicht als Sperre. Eigentlich wäre also “potentielle Sperre” ein besserer Name als Sperre.

Commoner [4] charakterisiert nun lebendige und wohlgeformte FC Netze. Die Beweise sind recht aufwendig aber klar strukturiert in Hack [19] zu finden. Wir geben nur die Ergebnisse an.

Satz 6.5.1 *Ein FC Netz ist (bzgl einer Konfiguration s) lebendig genau dann wenn jede Sperre eine markierte (in s) Falle besitzt.*

Aufwändiger ist eine Analyse der Wohlgeformtheit, also der Frage, ob ein Netz sicher (d.h. 1-beschränkt) und lebendig ist. Diese Frage kann mit dem Überdeckungsgraphen entschieden werden, allerdings mit einer unverhältnismäßigen Komplexität in Vergleich zu folgender Lösung.

Satz 6.5.2 *Ein FC Netz \mathcal{N} ist wohlgeformt genau dann, wenn es von streng zusammenhängenden, mit jeweils einem Token markierten, Zustandsmaschinen in \mathcal{N} überdeckt werden kann und jede minimale Sperre eine markierte streng zusammenhängende Zustandsmaschine in \mathcal{N} bildet.*

Eine streng zusammenhängende Zustandsmaschine in einem Petri Netz \mathcal{N} ist hierbei eine Teilmenge von Places $S \subseteq P$, die so als $S = \{p_1, \dots, p_k\}$ angeordnet werden können, dass k unterschiedliche Transitionen t_1, \dots, t_k in \mathcal{N} existieren mit $p_i \in \bullet t_i$ und $p_{i+1} \in t_i \bullet$ für $1 \leq i \leq k$ und $k+1 := 1$. Überdeckt bedeutet, dass jeder Place des FC Netzes in mindestens einer dieser Zustandsmaschinen vorkommt.

Eine derartige Charakterisierung von Lebendigkeit oder Wohlgeformtheit ist für die nächst komplexere Teilklasse der simplen Netzen schon nicht mehr bekannt.

Simple Netze und das Problem der dinierenden Philosophen

Bei dem Problem der dinierenden Philosophen handelt es sich um eine Synchronisationaufgabe, die in ein Spiel um Philosophen verkleidet ist: Drei Philosophen sitzen an einem kleinen runden Tisch. Vor jedem ein Teller Spaghetti, zwischen den drei Tellern liegt je eine Gabel, insgesamt also drei Gabeln, je eine links und rechts vor jedem der Philosophen. Jeder Philosoph benötigt zum Verspeisen seiner Spaghetti zwei Gabeln. Ein Philosoph, der einmal eine Gabel ergriffen hat, legt diese erst dann wieder weg, wenn er seine Spaghettimalzeit beendet hat.

Das sieht ganz harmlos aus. Ein Problem ergibt sich in einem nebenläufigen Modell, wenn jeder Philosoph unabhängig von einer zentralen Steuerung seine Aktion ausführen darf. Nimmt etwa ein Philosoph als einziger seine linke und rechte Gabel, isst und legt anschließend die Gabeln zurück, so könnte der nächste Philosoph speisen. Greift jeder Philosoph aber zu seiner linken Gabel und nimmt diese an sich, so fehlen jedem Philosophen die zweite Gabel zu Speisen. Da ein Philosoph nur dann eine Gabel zurücklegt, wenn er sein Mal beendet hat, keiner aber überhaupt mit der einen Gabel in der Hand sein Mal beginnen kann, liegt ein Deadlock vor. Die gesamte Situation ist blockiert.

Die gleiche Situation wird auch als 3-Raucher-Problem beschrieben.

Drei Raucher sitzen an einem Tisch, auf dem sich Zigarettenpapier, Tabak und Streichhölzer befinden. Raucher A besitzt darüber hinaus eigenes Zigarettenpapier, Raucher B eigenen Tabak, Raucher C eigene Streichhölzer. Jeder der drei Raucher braucht also noch zwei weitere andere Teile um Rauchen zu können. Und natürlich legt jeder seine Ingredienzien erst dann zurück, wenn er gemütlich mit seinem Rauchen beginnen kann. Ein Deadlock ergibt sich wieder, wenn jeder der Raucher zu nur einem fehlenden Zubehör greift und damit jedem anderen das zweite fehlende Zubehör blockiert.

Die ernsthafte Frage ist, wie man diese Synchronisationaufgabe sinnvoll modellieren und lösen kann. Überraschenderweise zeigte sich mit Hilfe einer Analyse von Petri Netzen, dass eine Synchronisation mittels Dijkstra's Semaphoren nicht möglich ist. Der Beweis wurde so geführt, dass man

- eine Lösung mit Petri Netzen angibt,
- zeigt, dass eine Lösung mit simplen Petri Netzen nicht möglich ist,
- und zeigt, dass Semaphoren stets mit simplen Petri Netzen modelliert werden können.

Eine Modellierung mit Petri Netzen ist leicht und in Abbildung 6.11 gezeigt.

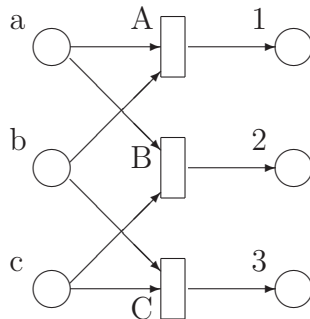


Abbildung 6.11: 3RN

Die Places a, b, c stehen für Zigarettenpapier, bzw. Tabak, bzw. Streichhölzer (oder für je eine Gabel), Transition A (B,C) für den Raucher mit Streichhölzern (mit Tabak, bzw. mit Zigarettenpapier) (bzw. für je einen Philosophen), die Places 1, 2, 3 dafür, dass Akteur A (bzw. B, bzw. C) seine Aktion beendet hat und die Ingredienzien wieder frei gibt. Dieses Netz nennen wir das 3-Raucher-Netz (3RN). Man kann 3RN leicht in ein größeres Petri Netz \mathcal{N} einbetten, so dass \mathcal{N} bei Start je einen Token

zufällig auf a und b oder auf a und c oder auf b und c verteilt und ansonsten keine Token mehr besitzt. Durch einen Token auf 1,2 oder 3 wird \mathcal{N} neu gestartet.

3RN ist kein simples Netz. Wir wollen nun zeigen, dass man die obige Situation nicht mit einem simplen Netz modellieren kann. Oder anders ausgedrückt, dass kein simples Netz das 3RN simulieren kann, oder die gleiche Semantik wie das 3RN besitzt.

Hier ergibt sich ein schwieriges prinzipielles Problem: um so einen Beweis führen zu können, muss man die Begriffe “Modellieren”, “Simulieren” oder “Semantik” präzise definieren. Schließlich geht es um eine Unmöglichkeitssatz. So konnte man in der Grundlagenforschung der Mathematik und Logik erst dann zeigen, dass es unentscheidbare Probleme gibt, nachdem man den Begriff der Entscheidbarkeit tief verstanden und sich über dessen präzise Definition geeinigt hatte. Ohne eine präzise Definition des Entscheidbarkeitsbegriffes konnte man zwar auch schon positive Ergebnisse erzielen. Etwa war stets unbestritten, dass die Frage, ob eine Zahl eine Primzahl ist, entscheidbar ist, da man immer schon Verfahren kannte, die zu jeder Zahl diese Frage nach endlich vielen Schritten beantworten. Dass diese bekannten Algorithmen eine Entscheidung in dieser Frage bewirken, war auch ohne präzise Definition der Entscheidbarkeit unbestritten. Um aber etwa zu zeigen, dass das 10. Hilbertsche Problem unentscheidbar ist, war diese Präzisierung des Entscheidungsbegriffs unumgänglich.

Also müssen wir auch einen der genannten Begriffe (Modellieren, Simulieren, Semantik) präzisieren, um das obige Unmöglichkeitsergebnis erhalten zu können. Das würde aber den Umfang dieser Vorlesung sprengen. Hinzu kommt, dass in der Gemeinde absolut keine Einigung besteht, was genau unter diesen Begriffen zu verstehen sei. Was also tun?

Es gibt einen Ausweg. Allen Informatikern sind Protokolle wohl vertraut. Wir werden Protokolle definieren, die sinnvollerweise das 3RN beachten soll, die aber von keinem simplen Netz erfüllt werden können. Leider brauchen wir dazu noch einige Begriffe, die nur in diesem Paragraphen benötigt werden.

Definition 6.5.3 *Ein Petri Netz mit Interface $\mathcal{N} = (P, T, F, I, O)$ ist ein Petri Netz (P, T, F) mit ausgezeichneten Mengen $I \subseteq P$ von Input- und $O \subseteq P$ von Output-Places von \mathcal{N} mit $\bullet I = O \bullet = \emptyset$.*

Eine Input-Output-Prozedur (I-O-Prozedur) P ist eine Relation $P \subseteq \mathbb{N}^P$ auf Konfigurationen mit $sPs' \implies$

1. $s(p) = s'(p) \forall p \in P - (I \cup O)$,

2. $s(p) \geq s'(p) \forall p \in O$,
3. $s(p) \leq s'(p) \forall p \in I$.

Ein P -Protokoll von \mathcal{N} (von s_0 aus) ist ein Wort $s_0s_1\dots s_k$ von Konfigurationen mit $s_i \vdash s_{i+1}$ oder $s_i P s_{i+1}$ für $0 \leq i < k$.

Eine I-O-Prozedur legt damit fest, wann Token von Output- und wann Token auf Input-Places von \mathcal{N} von außen entfernt bzw gelegt werden dürfen. Ein Protokoll ist eine Beschreibung des Verhaltens, bei dem in einem Schritt ein interner Übergang \vdash des Petri Netz oder eine äußere I-O-Prozedur ausgeführt werden kann.

Definition 6.5.4 Es seien $\mathbb{N} = (P, T, F, I, O)$ ein Petri Netz mit Interface, P eine I-O-Prozedur und $d = s_0\dots s_k$ ein P -Protokoll von \mathcal{N} von s_0 aus. Mit P/d bezeichnen wir alle P -Protokolle, die mit d beginnen. $B(d)$ sind alle Places p in \mathcal{N} mit $s_k(p) > 0$, die also am Ende des Protokolls d einen Token besitzen. $B(P/d) := \bigcup_{d' \in P/d} B(d')$ bezeichnet alle Places, die nach Ausführung des Protokolls d noch in einem P -Protokoll einen Token erhalten können.

Definition 6.5.5 Ein Petri Netz $\mathcal{N} = (P, T, F, I, O)$ heißt nicht monoton, falls eine Start-Konfiguration s_0 , zwei I-O-Prozeduren P_1, P für \mathcal{N} von s_0 aus und ein Output-Place $p_0 \in O$ existieren mit

- $P_1 \subseteq P$,
- für alle $d \in P_1/s_0$ gilt
 - $I \subseteq B(P_1/d)$,
 - $p_0 \in B(P/d) - B(P_1/d)$.

Anderenfalls heißt \mathcal{N} monoton.

Der Name “nicht monoton” erklärt sich damit, dass P_1 (und damit auch P) alle Input-Places mit Token belegen, P_1 aber auf weniger Output-Places als P einen Token bewirken kann. Dabei kann man jedes P_1 -Protokoll so als P -Protokoll fortsetzen, dass p_0 einen Token enthält. Dies verallgemeinert die Lebendigkeitbedingung für geschlossene Petri Netze zu diesen “offenen” Petri Netzen mit Interface und externer Kommunikation mittels Protokollen.

Wir betrachten nun irgendein Petri Netz \mathcal{N}_{3RN} , das die Rolle des $3RN$ aus Abbildung 6.11 übernehmen soll. Was fordern wir für “die Rolle übernehmen”? Wir fordern

- $\mathcal{N}_{3\text{RN}}$ besitzt als Interface genau $I = \{a, b, c\}$ und $O = \{1, 2, 3\}$ und eine Start-Konfiguration s_0 , und
- es existiert eine I-O-Prozedur P_1 für $\mathcal{N}_{3\text{RN}}$ von s_0 aus mit $sP_1s' \implies$
 - $\sum_{p \in O} s(p) = 1, \sum_{p \in O} s'(p) = 0,$
 - $\sum_{p \in I} s(p) = 0, \sum_{p \in I} s'(p) = 2,$
 - $s'(a) = s'(b) = 1$ und $s'(c) = 0,$ oder
 - $s'(a) = s'(c) = 1$ und $s'(b) = 0,$ und
 - $3 \notin B(P_1/s_0),$
- es existiert eine I-O-Prozedur P für $\mathcal{N}_{3\text{RN}}$ von s_0 aus mit $sPs' \implies$
 - $\sum_{p \in O} s(p) = 1, \sum_{p \in O} s'(p) = 0,$
 - $\sum_{p \in I} s(p) = 0, \sum_{p \in I} s'(p) = 2,$
 - $s'(a) = s'(b) = 1$ und $s'(c) = 0,$ oder
 - $s'(a) = s'(c) = 1$ und $s'(b) = 0,$ oder
 - $s'(b) = s'(c) = 1$ und $s'(a) = 0,$ und
 - $3 \in B(P, d)$ für jedes $d \in P_1/s_0.$

Sowohl P_1 als auch P arbeiten sequentiell: es muss ein Token auf einem Output-Place liegen um jeweils genau zwei Token auf die Input-Places zu legen. Dabei darf P_1 die beiden Token nur auf die Places a und b oder auf a und c , nicht aber auf b und c legen. Gefordert ist, dass der Output-Place 3 dann nicht über die I-O-Prozedur P_1 aber über P erreichbar ist. Im Unterschied zu P_1 darf P auch auf die Kombination b, c je einen Token legen.

3RN erfüllt diese Bedingungen offensichtlich, und diese Bedingungen sind gerade die Essenz, die jedes Netz, dass die Rolle von 3RN übernimmt, erfüllen muss. Diese Bedingungen zwingen allerdings $\mathcal{N}_{3\text{RN}}$ ein nicht monotones Petri Netz zu sein. Damit kann $\mathcal{N}_{3\text{RN}}$ nicht ein simples Netz sein, denn wir zeigen folgenden Satz, der ein Resultat von Patil [22] verallgemeinert.

Satz 6.5.3 *Jedes simple Netz ist monoton.*

Beweis. Wir nehmen an, dass ein nicht monotones simples Petri Netz \mathcal{N} existiert. s_0, p_0, P_1 und P seien gemäß Definition 6.5.5 gewählt. Wir konstruieren induktiv P_1 -Protokolle d_n wie folgt.

$d_0 := s_0$.

Sei d_n bereits definiert. Wir wollen d_{n+1} als Verlängerung von d_n definieren und gehen dazu wie folgt vor:

Für $p \in P - B(P_1/d_n)$ seien

$$L(p) := \begin{cases} \infty & : p \notin B(P/d_n) \\ \min\{|d| \mid p \in B(d) \wedge d \in P/d_n\} & : \text{sonst,} \end{cases}$$

$$L := \min\{L(p) \mid p \in P - B(P_1/d_n)\}.$$

Wegen $L(p_0) < \infty$ gilt $L < \infty$. Es seien $\hat{d} \in P/d_n$ und $\hat{p} \in B(\hat{d}) - B(P_1/d_n)$ mit $L(\hat{p}) = |\hat{d}| = L$, also von minimaler Länge. \hat{d} habe die Form $\hat{d} = d^+ s' s''$. Dann gilt wegen der minimalen Länge von \hat{p} : $s''(\hat{p}) > 1$ und $s'(\hat{p}) = 0$. \hat{p} kann kein Input-Place sein, da P_1 alle Input-Places belegen kann. Also muss eine Transition t existieren mit $s' \vdash_t s''$. Wir nehmen an, dass t die beiden Input-Places i_1, i_2 besitzt. Der allgemeine Fall mit n Input-Places von t geht völlig analog zur folgenden Argumentation. Falls $i_1 \notin B(P_1/d_n)$ gilt, hätten wir mit i_1 einen Place in $B(P/d_n) - B(P_1/d_n)$ einer kleineren Länge als \hat{p} gefunden, im Widerspruch zur Minimalität von \hat{p} . Also gilt $i_1 \in B(P_1/d_n)$ und analog auch $i_2 \in B(P_1/d_n)$.

Sowohl i_1 als auch i_2 können durch ein P_1 -Protokoll einen Token erhalten. Es gibt aber kein P_1 -Protokoll, das gleichzeitig auf i_1 und i_2 einen Token legen kann, da dann t feuerebar wäre und \hat{p} somit über ein P_1 -Protokoll einen Token erhalten könnte, im Widerspruch zur Konstruktion von \hat{p} .

\mathcal{N} ist ein simples Petri Netz. Damit kann nur einer der beiden Input-Places i_1, i_2 von t noch Input-Place einer weiteren Transition t' sein. Dies sei i_2 . i_1 ist also nur von t ein Input-Place. Es sei $d' \in P_1/d_n$ ein kürzestes P_1 -Protokoll, das einen Token auf i_1 legt.

Fall 1. $i_2 \in B(P_1/d')$. Dann existiert ein kürzestes Protokoll d'' in P_1/d' , das einen Token auf i_2 legt. Der Token auf i_1 ist immer noch vorhanden, da keine andere Transition als t diesen Token entfernen kann. Dann wäre t feuerebar und \hat{p} könnte wieder fälschlich in einem P_1 -Protokoll einen Token erhalten. Fall 1 führt als unmittelbar zu einem Widerspruch. Damit gilt

Fall 2: $i_2 \notin B(P_1/d')$.

Wir setzen

$$d_{n+1} := d'.$$

Damit gilt aber $i_2 \in B(P_1/d_n) - B(P_1/d_{n+1})$ und die Zahl der in P_1/d_n belegbaren Places nimmt mit wachsendem n stets ab. Also kann Fall 2

nur endlich oft eintreten und irgendwann muss Fall 1 gelten, mit einem Widerspruch. Damit ist die Annahme der Existenz eines simplen nicht monotonen Netzes widerlegt. ■

6.6 Schwache Petri Netz Berechenbarkeit

Wir werden hier einen Begriff vorstellen, wie man mit Petri Netzen “schwach” rechnen kann. “Schwach” bezieht sich darauf, dass man hier nicht alle Aspekte von “Rechnen” erfasst. Interessant ist der Begriff des schwachen Rechnens, weil man mit ihm zeigen kann, dass Überdeckungsgraphen riesig und nicht mehr handhabbar werden können.

Definition 6.6.1 *Ein Petri Netz $\mathcal{N} = (P, T, F)$ berechnet eine Funktion $f : \mathbb{N}^r \rightarrow \mathbb{N}$ schwach genau dann, wenn folgende Bedingungen erfüllt sind:*

- \mathcal{N} besitzt eine Menge $In = \{in_1, \dots, in_r\} \subseteq P$ von r ausgezeichneten sogenannten Eingabe-Places,
- einen ausgezeichneten Ausgabe-Place out,
- einen ausgezeichneten Start-Place start,
- und einen ausgezeichneten Halte-Place halt,
- für $x_1, \dots, x_r \in \mathbb{N}$ sei s_{x_1, \dots, x_r} die eindeutig bestimmte Konfiguration in \mathbb{N}^P mit $s_{x_1, \dots, x_r}(in_i) = x_i$ für $1 \leq i \leq r$, $s_{x_1, \dots, x_r}(start) = 1$ und $s_{x_1, \dots, x_r}(p) = 0$ sonst,
- für alle $x_1, \dots, x_r, m \in \mathbb{N}$ soll gelten:
 1. $m \leq f(x_1, \dots, x_r) \iff \exists s \in \mathcal{E}(s_{x_1, \dots, x_r}): (s(halt) = 1 \wedge s(out) = m)$,
 2. $\forall s \in \mathcal{E}(s_{x_1, \dots, x_r}) \exists s' \in \mathcal{E}(s): s'(halt) = 1$,
 3. $\forall s \in \mathcal{E}(s_{x_1, \dots, x_r}): (s(halt) > 0 \implies \mathcal{N} \text{ ist tot in } s)$,
 4. $\forall s \in \mathcal{E}(s_{x_1, \dots, x_r}) \forall p \in P: s(p) \leq f(x_1, \dots, x_r)$.

Existiert ein Petri Netz \mathcal{N} , das eine Funktion f schwach berechnet, so sagen wir auch, f ist schwach PN-berechenbar und \mathcal{N} ist ein schwacher PN-Computer für f .

Wir teilen einem schwacher PN -Computer die Argumente x_i der zu berechnenden Funktion f durch x_i viele Token auf dem i -ten Inputplace in_i mit und starten die Rechnung mit einem Token auf $start$. Wegen der Monotonie von Petri Netzen kann das Netz manche Input-Token einfach ignorieren. Bei einem Token auf $halt$ hält das Petri Netz mit jeglicher Arbeit an (Forderung 3) und auf der Ausgabe out liegen m Token mit $m \leq f(x_1, \dots, x_r)$ (Forderung 2). Generell sollen nirgends im Petri Netz Computer mehr als $f(x_1, \dots, x_r)$ Token vorkommen (Forderung 4). Andererseits muss nach Forderung 1 auch eine Rechnung möglich sein, die genau $f(x_1, \dots, x_r)$ viele Token auf out legt. Schwache PN -Computer besitzen mit Forderung 2 eine interessante Lebendigkeitseigenschaft: Egal, wie man in \mathcal{N} feuert ($\forall s \in \mathcal{E}(s_{x_1, \dots, x_r})$), man kann so fortsetzen ($\exists s' \in \mathcal{E}(s)$), dass ein Token $halt$ erreicht ($s'(halt) = 1$). N kann also nur sterben, wenn ein Token $halt$ erreicht.

Abbildung 6.12 zeigt ein Petri Netz, das die konstante Funktion b von $\mathbb{N}^0 \rightarrow \mathbb{N}$ erzeugt. Da per Definition auf out k Token für jedes $k \leq b$ liegen können sollen, werden t_2 und t_3 zur Entfernung beliebiger Token von out verwendet. Man könnte die konstante Funktion also auch sicher berechnen, so dass immer genau b Token auf out liegen müssen. Für weitere Funktionen wird uns das aber nicht gelingen.

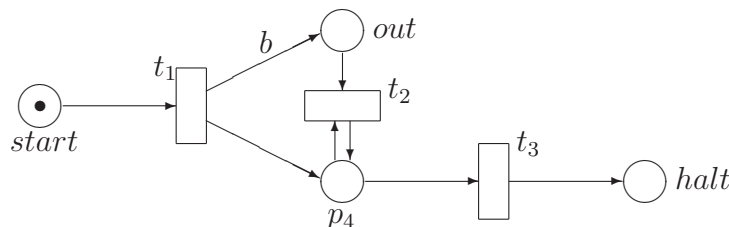


Abbildung 6.12: Ein Petri-Netz zur Erzeugung einer Konstanten b

Man überlegt sich leicht, dass eine schwach berechenbare Funktion $f : \mathbb{N}^r \rightarrow \mathbb{N}$ stets monoton ist, d.h. $\forall m, n \in \mathbb{N}^r : (m \leq n \implies f(m) \leq f(n))$.

Lemma 6.6.1 *Die Konstanten, die Addition und die Multiplikation sind schwach PN -berechenbar.*

Beweis. Das Netz in Abbildung 6.13 berechnet die Addition schwach, das in 6.14 die Multiplikation für $x > 0$. Soll die Multiplikation auch für $x=0$ funktionieren, braucht man noch eine zusätzliche Transition, die den Token von $start$ direkt auf $halt$ legen kann. ■

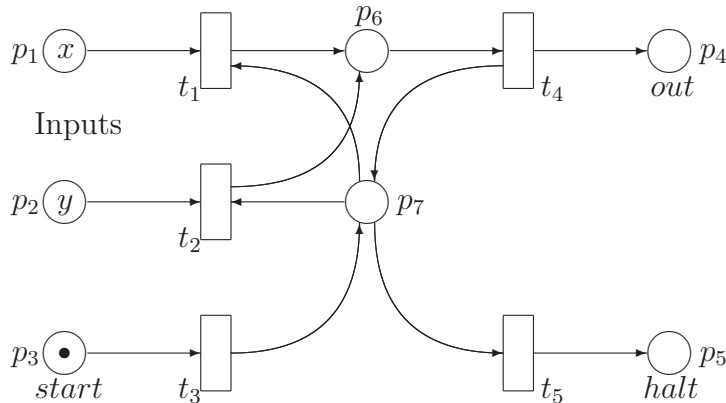


Abbildung 6.13: Ein schwacher PN -Computer für die Addition

Ein Polynom der Ordnung n über \mathbb{N} ist eine Funktion der Form

$$f(x_1, \dots, x_r) = \sum_{(i_1, \dots, i_r) \in \{0, \dots, n\}^r} a_{i_1, \dots, i_r} x_1^{i_1} \cdot \dots \cdot x_r^{i_r}$$

mit natürlichzahligen Koeffizienten a_{i_1, \dots, i_r} . Aus den Grundbausteinen Konstante, Addition und Multiplikation können wir alle Polynome aufbauen. Hierbei ist auch die Größe der Petri Netze, die ein solches Polynom f berechnen, proportional zur Größe von f . Dabei definieren wir die Größe eines Petri Netzes als die Summen aller im Netz auftretenden Places, Transitionen, Pfeile und Token, und die Größe von f als die Summe aller in f vorkommenden Koeffizienten a_{\dots} und Potenzen i_j , die mit Koeffizienten ungleich 0 auftreten.

Definition 6.6.2 Die Funktionen $A_i : \mathbb{N} \rightarrow \mathbb{N}$ sind induktiv definiert durch

$$A_0(x) := x + 1, \quad A_{n+1}(0) := A_n(1) \quad \text{und} \quad A_{n+1}(x + 1) := A_n(A_{n+1}(x)).$$

Die Ackermann-Funktion $a : \mathbb{N} \rightarrow \mathbb{N}$ ist

$$a(n) := A_n(n).$$

Hilbert vermutete, dass jede berechenbare Funktion bereits primitiv rekursiv ist. Dies konnte Ackermann durch Angabe einer berechenbaren Funktion, die nicht primitiv rekursiv sein kann, widerlegen. Diese hier Ackermann-Funktion genannten Funktion ist eine von der ungarischen Mathematikerin Péter gefundene Variante der ursprünglichen Funktion von Ackermann. Die Ackermann-Funktion ist nicht primitiv

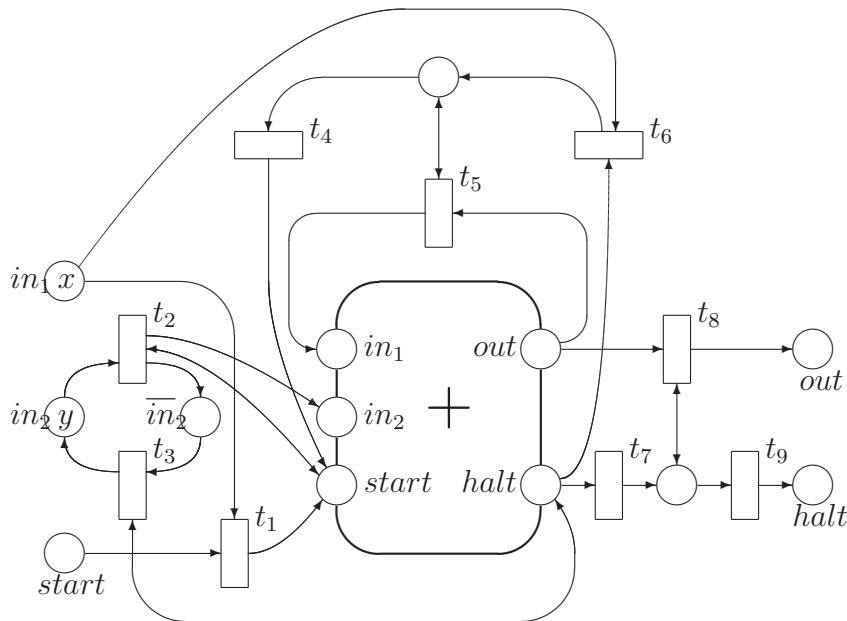


Abbildung 6.14: Ein schwacher PN -Computer für die Multiplikation

rekursiv. Insbesondere existiert kein LOOP-Programm, das sie berechnen kann, und die Ackermann-Funktion wächst ab irgendeiner Stelle schneller als eine beliebig vorgegebene LOOP-berechenbare Funktion. Für $n = 3$ wächst A_n bereits so schnell wie die Exponentialfunktion. Ackermann-Funktionen werden gern als Benchmark Tests verwendet, da sie extrem geschachtelte rekursive Aufrufe und extrem hohe natürliche Zahlen benötigen. Es ist gar nicht auf dem ersten Blick ersichtlich, dass die Funktionen A_n und a überhaupt berechenbar sind. Zu zeigen ist, dass die Rekursionsaufrufe stets abbrechen. Wir zeigen

Lemma 6.6.2 $A_n(m)$ ist für jedes $n, m \in \mathbb{N}$ wohl definiert.

Beweis. Wir führen den Beweis durch eine geschachtelte Induktion. Äußere Schleife: Induktion über n :

Induktionsbeginn $n = 0$. Offensichtlich ist $A_0(m)$ stets definiert.

Induktionsschritt $n \rightarrow n + 1$: Es sei $A_n(m)$ für jedes $m \in \mathbb{N}$ definiert.

Dann zeigen wir in einer inneren Schleife mittels Induktion über m , dass auch $A_{n+1}(m)$ definiert ist.

Induktionsbeginn $m = 0$: $A_{n+1}(0) = A_n(1)$ ist nach Induktionsvoraussetzung für n definiert.

Induktionsschritt $m \rightarrow m+1$: $A_{n+1}(m+1) = A_n(A_{n+1}(m))$ ist definiert, da nach Induktionsannahme für m bereits $A_{n+1}(m)$ definiert ist, und nach Induktionsannahme für n dann auch $A_n(x)$ für jedes x definiert ist. ■

Übung 6.6.1 Schreiben Sie ein Programm, das die Ackermann-Funktion berechnet. Bauen Sie einen Notstopp ein. Versuchen Sie, $a(n)$ für kleine Werte n zu berechnen. Was ist die größte Zahl $n \in \mathbb{N}$, für die Ihr Rechner noch $a(n)$ in "vernünftiger" Zeit berechnen kann?

Die A_n -Funktionen sind nun trotz der Tatsache, dass sie extrem schnell wachsen, immer noch schwach PN -berechenbar.

Lemma 6.6.3 Für jedes $n \in \mathbb{N}$ ist A_n schwach PN -berechenbar.

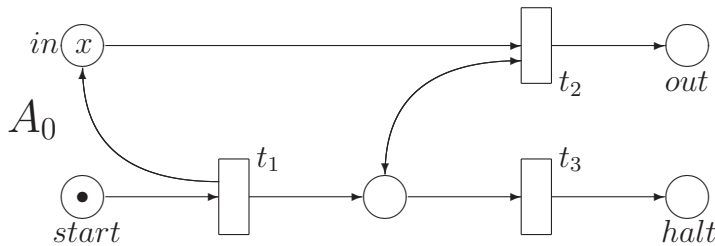


Abbildung 6.15: Ein Petri-Netz, das $A_0(x) = x + 1$ schwach berechnet

Beweis. Das Petri Netz aus Abbildung 6.15 berechnet offensichtlich A_0 . Sei A_n bereits schwach PN -berechenbar. Abbildung 6.16 zeigt, wie dann auch A_{n+1} schwach PN -berechenbar ist. Es gilt

$$A_{n+1}(m) = \underbrace{A_n(\dots A_n(1) \dots)}_{m+1\text{-mal}}.$$

Es sei $p_1 = start_{n+1}$, $p_2 = in_{n+1}$, $p_3 = out_{n+1}$, $p_4 = halt_{n+1}$. Wir zeigen, dass zu jedem i mit $0 \leq i \leq m$ eine Feuersequenz σ_i und eine Konfiguration s_i existieren mit $(1, m, \vec{0}) \vdash_{\sigma_i} s_i$ und

- $s_i(p) = 0$ für $p \notin \{in_{n+1}, out_n, halt_n\}$,
- $s_i(out_n) = A_{n+1}(i)$,
- $s_i(in_{n+1}) = m - i$, und
- $s_i(halt_n) = 1$.

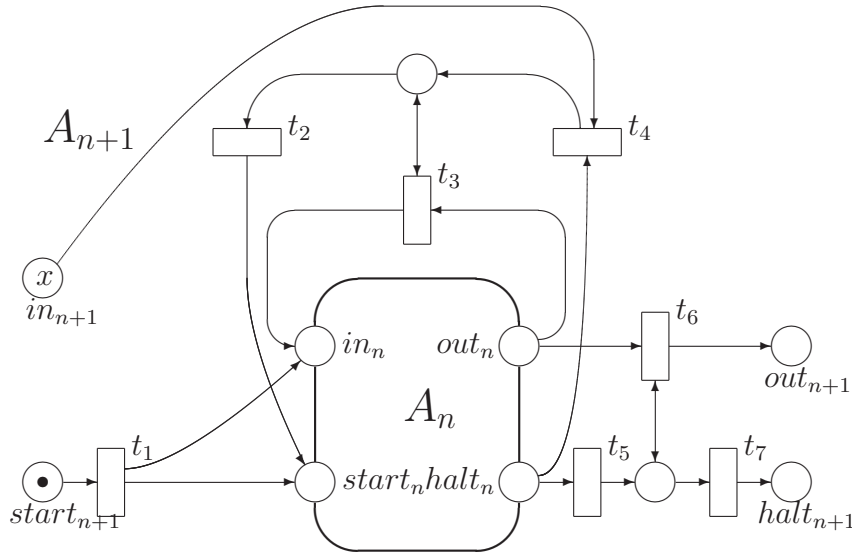


Abbildung 6.16: Ein Petri-Netz, das $A_{n+1}(x)$ schwach berechnet

Für $i = 0$ setzen wir $\sigma_0 = t_1\sigma$, wobei σ die Rechnung innerhalb A_n ist, die $A_n(1)$ viele Token auf out_n legt und hält. Wegen $A_{n+1}(0) = A_n(1)$ gilt damit $(1, m, \vec{0}) \vdash_{\sigma_0} s_0$.

Es sei σ_i für $0 \leq i < m$ bereits definiert, dann erhalten wir σ_{i+1} als $\sigma_i t_4 t_3^{s_i(out_n)} t_2 \sigma$, wobei σ die Rechnung innerhalb von A_n ist, die $A_n(s_i(out_n))$ viele Token auf out_n legt und hält. Zu beachten ist nun, dass $A_n(s_i(out_n)) = A_n(A_{n+1}(i)) = A_{n+1}(i+1)$ gilt. Also gilt

$$(1, m, \vec{0}) \vdash_{\sigma_i} s_i \vdash_{t_4 t_3^{s_i(out_n)} t_2 \sigma} s_{i+1}.$$

Damit erreicht man durch Feuern von σ_m gerade s_m mit $s_m(out_n) = A_{n+1}(m)$, $s_m(halt_n) = 1$ und alle anderen Places sind leer. Ein Feuern von $s_m \vdash_{t_5 t_6^{A_{n+1}(x)} t_7} s$ liefert eine Konfiguration s mit $s(out_{n+1}) = A_{n+1}(m)$, $s(halt_{n+1}) = 1$ und keine Token sonst im Netz, die gerade die Forderung 1 der schwachen PN -berechenbarkeit erfüllt. Weniger als $A_{n+1}(m)$ Token bekommen wir, indem wir in A_0 (dem innersten Petri Netz) beim letzten Durchlauf einige Token “verschlucken”. Mehr als $A_{n+1}(m)$ Token sind aber nicht möglich. Token können nur auf dem in_i - oder out_i -Place irgendeines A_i liegen bleiben. Wegen $A_i(m) + 1 \leq A_i(m+1)$ trägt ein liegen gebliebener Token (der nicht durch t_3 zurückgelegt oder durch t_6 weitergereicht wird) aber weniger zur Gesamtzahl der Token bei als einer der in der Berechnung benutzt wird. Dieser Effekt verstärkt sich bei jedem Start eines A_i -Netzes, da alle A_i streng monoton wachsend sind. ■

Satz 6.6.1 *Für alle $n \in \mathbb{N}$ existiert ein Petri Netz \mathcal{N}_n der Größe $O(n)$, das die konstante Funktion $a(n) = A_n(n)$ schwach berechnet.*

Achtung, hier ist n kein Argument sondern ein Parameter. Für festes n wird die Existenz eines Petri Netzes ausgesagt, dass die Zahl b schwach berechnet, mit $b = a(n)$. Es wird nicht gesagt, dass ein PN -Computer für die Funktion a existiert.

Beweis. Es sei n fest. Wir stellen durch direktes Nachzählen in Abbildung 6.15 und 6.16 fest, dass die Größe von A_n gerade $34n + 18$ ist. Zur Erzeugung des Argumentes n benutzen wir den schwachen Computer für die Konstante n der Größe $n + 15$, vergleiche Abbildung 6.12. Insgesamt kommen wir zu schwachen Berechnung der Konstanten $a(n)$ mit einem Netz einer Größe von $35n + 31$ aus. ■

Insbesondere ist der Erreichbarkeitsgraph von \mathcal{N}_n gleich dem Überdeckungsgraphen von \mathcal{N}_n und besitzt mindestens $a(n)$ -viele Knoten. Dann kann keine primitiv rekursive Funktion f existieren, so dass alle beschränkten Petri Netze einer Größe n auch einen Erreichbarkeitsgraphen einer Größe (= Anzahl der Kanten plus Knoten) $\leq f(n)$ besitzen. Damit haben wir folgenden überraschenden Satz bewiesen:

Satz 6.6.2 *Die Größe eines Überdeckungsgraphen $UG(\mathcal{N})$ oder Erreichbarkeitsgraphen $ER(\mathcal{N})$ in Abhängigkeit der Größe von \mathcal{N} ist nicht durch eine primitiv rekursive Funktion beschränkbar.*

Insbesondere sind alle Algorithmen zu Petri Netzen, die den Erreichbarkeits- oder Überdeckungsgraphen explizit nutzen, nicht primitiv rekursiv. In der Praxis bedeutet das aber nicht unbedingt eine Katastrophe, da diese Graphen bei Petri Netzen, die praktische Anwendungen modellieren, nicht notwendig explodieren müssen. Man muss nur diese Gefahr kennen.

Teil III

Anhang: Mathematische Grundlagen

6.7 Komplexe Zahlen

Bekannte mathematische Körper sind \mathbb{R} und \mathbb{C} , die Körper der reellen bzw. komplexen Zahlen. Das besondere an Körpern ist die Möglichkeiten, in Ihnen die Grundrechenarten Addition und Multiplikation (bzw. andere, der Addition und Multiplikation artverwandte Operationen) und deren Inverses, also Subtraktion und Division (mit Ausnahme der Null) durchführen zu können. Zusätzlich besitzen \mathbb{R} und \mathbb{C} eine Metrik, sogar eine Norm, und beide sind vollständig in dem Sinn, dass jede Cauchy-Folge einen Grenzwert besitzt. Die wichtigsten Unterschiede von \mathbb{R} und \mathbb{C} sind, dass \mathbb{R} angeordnet ist, \mathbb{C} aber nicht angeordnet werden kann, und dass in \mathbb{C} jede Zahl eine Quadratwurzel besitzt, in \mathbb{R} aber nicht, siehe die Zahl -1 . Wir wiederholen kurz die wichtigsten Eigenschaften von komplexen Zahlen.

\mathbb{R}^n ist der Vektorraum der n -dimensionalen Vektoren von reellen Zahlen. Im zwei-dimensionalen können wir uns das sehr einfach geometrisch vorstellen. Zwei Punkte $P_1 = (1, 2)$, $P_2 = (-2, 2)$ im \mathbb{R}^2 kann man als Vektoren vom Nullpunkt $(0, 0)$ nach P_1 und nach P_2 auffassen. Damit besteht ein Winkel α zwischen P_1 und P_2 , den man mit der bekannten Formel

$$\cos \alpha = \frac{\langle P_1 | P_2 \rangle}{|P_1| \cdot |P_2|}$$

ausrechnet. Dabei ist $|P|$ die Länge des Vektors P , und $\langle P_1 | P_2 \rangle$ das Skalarprodukt beider, im Beispiel also

$$\begin{aligned} |P_1| &= \sqrt{1^2 + 2^2} \approx 2,23, \\ |P_2| &= \sqrt{2^2 + 2^2} \approx 2,89, \\ \langle P_1 | P_2 \rangle &= 1 \cdot -2 + 2 \cdot 2 = 2, \\ \cos \alpha &\approx 2 / (2,23 \cdot 2,89) \approx 0,310, \text{ d.h.} \\ \alpha &\approx 72^\circ \end{aligned}$$

Komplexe Zahlen werden oft als $z = a + bi$ geschrieben, mit Realteil $Re(z) = a$ und Imaginärteil $Im(z) = b$. i ist die imaginäre Einheit $\sqrt{-1}$, die zusätzliche Wurzel von -1 , mit der Multiplikationsregel $i \cdot i = -1$. Dieser kartesischen Darstellung entspricht die Vorstellung von \mathbb{C} als Zahlenebene, isomorph zum $\mathbb{R} \times \mathbb{R}$. Damit kann man eine komplexe Zahl als 2-dimensionalen Vektor über \mathbb{R} auffassen. In der Elektrotechnik wird i als j geschrieben, da das Symbol i bereits für Wechselstrom benutzt wird. In der Physik wird i für Wechselstrom geschrieben und j für die imaginäre Einheit. $\bar{z} = a - bi$ ist die konjugierte Zahl von z , d.h. gerade die Spiegelung im \mathbb{R}^2 an der x-Achse. Es gelten folgende Rechenregeln

- $(a + bi) + (c + di) = (a + c) + (b + d) i,$
- $(a + bi) - (c + di) = (a - c) + (b - d) i,$
- $(a + bi) \cdot (c + di) = (ac - bd) + (ad + bc) \cdot i,$
- $\frac{a+bi}{c+di} = \frac{(a+bi)(c-di)}{(c+di)(c-di)} = \frac{ac+bd}{c^2+d^2} + \frac{bc-ad}{c^2+d^2} \cdot i,$
- $z \cdot \bar{z} = (a + bi)(a - bi) = a^2 + b^2 = |z|^2,$
- $|z| = \sqrt{a^2 + b^2}$ ist die Norm von $z,$
- $z + \bar{z} = (a + bi) + (a - bi) = 2a = 2 \operatorname{Re}(z),$
- $z - \bar{z} = (a + bi) - (a - bi) = 2bi = 2i \operatorname{Im}(z).$

In Polarkoordinaten schreibt man komplexe Zahlen als

$$z = r \cdot e^{i\varphi} = r \cdot (\cos \varphi + i \cdot \sin \varphi),$$

mit der Norm $r = |z|$ als Entfernung vom Nullpunkt. $\{r \cdot e^{i\varphi} | \varphi \in \mathbb{R}\} = \{r \cdot (\cos \varphi + i \cdot \sin \varphi) | \varphi \in \mathbb{R}\}$ ist der Kreis um 0 mit Radius r . Damit gelten folgende Umrechnungen von Polar $z = r \cdot e^{i\varphi}$ nach kartesisch $z = a + bi$ und die Rechenregeln

- $a = r \cdot \cos \varphi, \quad b = r \cdot \sin \varphi,$
- $r = |z| = \sqrt{a^2 + b^2},$
- $\varphi = \arg(z) = \begin{cases} \arccos \frac{a}{r} & \text{für } b \geq 0 \\ -\arccos \frac{a}{r} & \text{für } b < 0, \\ \text{unbestimmt} & \text{für } r = 0 \end{cases}$
- $(r \cdot e^{i\varphi}) = r \cdot e^{i(\varphi+2\pi)},$
- $(r \cdot e^{i\varphi}) \cdot (s \cdot e^{i\psi}) = (r \cdot s) \cdot e^{i(\varphi+\psi)},$
- $\frac{(r \cdot e^{i\varphi})}{(s \cdot e^{i\psi})} = \frac{r}{s} \cdot e^{i(\varphi-\psi)},$
- $z^n = r^n \cdot e^{in\varphi} = r^n \cdot (\cos n\varphi + i \cdot \sin n\varphi),$
- $\sqrt[n]{z} = \sqrt[n]{r} \cdot e^{i\frac{\varphi+2k\pi}{n}}.$

Siehe auch folgendes Java-Applet <http://www.walter-fendt.de/m14d/komplz.htm>

Damit ist die Addition und Subtraktion in kartesischen Koordinaten, die Multiplikation und Division aber in Polarkoordinaten leichter durchführbar. Die Addition entspricht genau der Vektoraddition im \mathbb{R}^2 . Bei der Multiplikation werden die Normen beider komplexen Zahlen multipliziert und deren Winkel addiert. Ist $z = r \cdot e^{i\varphi}$, dann ist $\bar{z} = r \cdot e^{-i\varphi}$ und damit $z\bar{z} = r^2 \cdot e^0 = r^2$ eine reelle Zahl, das Quadrat der Norm von z . Da sich \mathbb{C} nicht anordnen läßt, spielt die Norm einen Ersatz dafür. Das Skalarprodukt $\langle z_1 | z_2 \rangle$ zweier komplexer Zahlen ist üblich definiert als

$$\langle z_1 | z_2 \rangle := z_1 \bar{z}_2.$$

Es ist in der ersten Komponente linear wegen

$$\langle z_1 + z_2 | z \rangle = (z_1 + z_2)\bar{z} = z_1\bar{z} + z_2\bar{z} = \langle z_1 | z \rangle + \langle z_2 | z \rangle.$$

Und genauso mit $z_1 \cdot z_2$ statt $z_1 + z_2$. In der Physik wünscht man jedoch ein in der zweiten Komponente lineares Skalarprodukt. Dazu muss man nur setzen

$$\langle z_1 | z_2 \rangle := \bar{z}_1 z_2.$$

Vorsicht, für 2-dimensionale Vektoren über \mathbb{R} berechnet $\langle x | y \rangle / (|x| \cdot |y|)$ den Cosinus des Winkels zwischen beiden Vektoren x und y , in \mathbb{C} wäre dann $\langle z_1 | z_2 \rangle / (|z_1| \cdot |z_2|)$ wegen $\langle z_1 | z_2 \rangle := \bar{z}_1 z_2$ der Winkel zwischen dem an der x -Achse gespiegelten Vektor z_1 und z_2 . Wir werden im folgenden diese Variante des Skalarproduktes benutzen, dessen Ergebnis der konjugierte Wert des alten, "mathematischen" Skalarproduktes ist. In der Theorie der Hilbert Räume sind als Skalarprodukte aber auch andere sogenannte hermitesche Formen zulässig.

6.8 Hilbert Räume

Hilbert Räume sind Vektorräume mit besonders schönen Eigenschaften: sie besitzen ein inneres Produkt, und damit eine Norm, also auch eine Metrik und damit eine Topologie, und sie sind vollständig in dieser Topologie. Hilbert Räume sind mir den mathematischen Kenntnissen des Bachelor-Studiums Informatik oder CV durchaus zu verstehen. Nur, wer sie noch nicht kennt, hat es nicht leicht, sich darüber schlau zu machen. Der Grund ist, dass Hilbert Räume spezielle Banach Räume, diese wieder spezielle Normierte Räume und diese spezielle Vektorräume sind. In Mathematikbüchern stehen im Kapitel Hilbert Räume üblicherweise diese Eigenschaften von Hilbert Räumen, die in Banach Räumen nicht gelten.

Im Kapitel über Banach Räume findet man die spezifischen Eigenschaften, die in Normierten Räumen nicht allgemein gelten, und so weiter. Damit muss man, um etwas über Hilbert Räume zu verstehen, auch die Kapitel über Vektorräume, Normierte Räume und Banach Räume lesen und kommt so leicht auf deutlich über 100 Seiten. Daher machen wir hier einen Crash-Kurs über die für Quantenrechner wichtigsten Eigenschaften, ohne zu unterscheiden, ob diese schon in Normierten Räumen gelten oder erst in Hilbert Räumen.

Definition 6.8.1 Ein Vektorraum (oder Linearer Raum) $\mathcal{V} = (V, 0, \oplus, \odot)$ über einem Körper K besteht aus einer Menge V mit $0 \in V$ und $\oplus : V \times V \rightarrow V$ und $\odot : K \times V \rightarrow V$ sind Abbildungen mit

- (V, \oplus) ist eine kommutative Gruppe mit Nullelement 0 und inverser Operation \ominus von \oplus ,
- $z \odot (z' \odot u) = (z \cdot z') \odot u$,
- $z \odot (u \oplus v) = z \odot u \oplus z \odot v$,
- $(z + z') \odot u = z \odot u \oplus z' \odot u$,
- $1 \odot u = u$,

für alle $z, z' \in K, u, v \in V$, wobei \odot stärker bindet als \oplus und 1 das Einselement der Multiplikation in K ist.

In einem Vektorraum \mathcal{V} über einem Körper K heißen die Elemente in V *Vektoren* und die Elemente in K *Skalare*. Der wesentliche Unterschied zu Körpern ist, dass in einem Vektorraum keine Multiplikation von Vektoren erklärt sein muss, bzw., falls eine Multiplikation vorhanden ist, diese nicht umkehrbar zu einer Division zu sein braucht. Es seien $\mathcal{V}_i = (V_i, 0_i, \oplus_i, \odot_i)$ für $1 \leq i \leq 2$ zwei Vektorräume über dem gleichen Grundkörper K . Ein Vektorraumhomomorphismus $h : \mathcal{V}_1 \rightarrow \mathcal{V}_2$ ist dann eine Abbildung von V_1 in V_2 , die mit den Vektorraumoperatoren kommutiert, für die also für $c \in K, v, v_1, v_2 \in V_1$ gilt

$$\begin{aligned} h(0_1) &= 0_2, \\ h(v_1 \oplus_1 v_2) &= h(v_1) \oplus_2 h(v_2), \\ h(c \odot_1 v) &= c \odot_2 h(v). \end{aligned}$$

Wegen ihrer Wichtigkeit gibt man Vektorraumhomomorphismen einen kürzeren Namen und nennt sie *lineare Abbildungen*.

Eine Menge B von Vektoren aus V heißt *Basis*, falls sich jeder Vektor $v \in V$ eindeutig als Linearkombination der Basiselemente darstellen lässt, d.h. wenn es eindeutige Skalare x_e für $e \in B$ gibt, so dass gilt

$$v = \sum_{e \in B} x_e \odot e.$$

Diese Darstellung eines Vektors als Linearkombination von Basiselementen ist eindeutig. Ist die Basis B bekannt und endlich mit $n = |B|$, so wird v üblich als Spaltenvektor

$$v = \begin{pmatrix} x_1 \\ \cdot \\ \cdot \\ \cdot \\ x_n \end{pmatrix}$$

angegeben.

$|B|$ ist für jede Basis gleich und heißt die *Dimension* von \mathcal{V} . Es sei $L : V \rightarrow V$ eine lineare Abbildung. Ein Skalar a heißt *Eigenwert* und ein Vektor u *Eigenvektor* von L , falls gilt

$$L(u) = a \cdot u.$$

Definition 6.8.2 Ein Normierter Raum ist ein Vektorraum über dem Körper \mathbb{R} oder \mathbb{C} für den eine Norm existiert. Dabei ist eine Norm $\|\cdot\|$ eine Abbildung $\|\cdot\| : \mathcal{V} \rightarrow \mathbb{R}_{\geq 0}$ mit

- $\|u\| = 0 \iff u = 0$,
- $\|zu\| = |z|\|u\|$,
- $\|u + v\| \leq \|u\| + \|v\|$,

für z in \mathbb{R} oder \mathbb{C} , u, v in V .

Statt $\|u\|$ wird in der Mathematik meist $\|u\|$ bei Vektoren u geschrieben und $|c|$ wird für Körperelemente c benutzt. Jede Norm definiert mittels $d(u, v) = \|u \ominus v\|$ eine Metrik und damit eine Topologie. Die Bedeutung von Normierten Räumen ist, dass man in ihnen über die Metrik die Begriffe der Stetigkeit und Konvergenz einführen und somit eine Theorie analytischer Funktionen entwickeln kann.

Definition 6.8.3 Ein Banach Raum ist ein Normierter Raum, der bzgl. seiner Metrik vollständig ist. D.h., ist $(u_n)_{n \in \mathbb{N}}$ eine Cauchy-Folge, so dass zu jedem $\epsilon > 0$ ein $n_0 \in \mathbb{N}$ existiert mit $\|u_n \ominus u_m\| < \epsilon$ für $n, m > n_0$, dann konvergiert sie gegen ein Element $u \in V$, d.h. zu jedem $\epsilon > 0$ existiert ein $n_0 \in \mathbb{N}$ mit $\|u_n \ominus u\| < \epsilon$ für $n > n_0$.

Definition 6.8.4 (Hilbert Raum) Ein (komplexer) Hilbert Raum $\mathcal{H} = (H, 0, \oplus, \odot, \langle | \rangle)$ ist

- ein Vektorraum $(H, 0, \oplus, \odot)$ über dem Körper \mathbb{C} , mit
- einer hermiteschen Form (auch inneres Produkt oder Skalarprodukt genannt) $\langle | \rangle : H \times H \rightarrow \mathbb{C}$ mit den Eigenschaften
 - $\langle u|u \rangle$ liegt in \mathbb{R} und ist ≥ 0 ,
 - $\langle u|u \rangle = 0 \iff u = 0$,
 - $\langle u|v \rangle = \overline{\langle v|u \rangle}$,
 - $\langle u|z \odot v \rangle = z \cdot \langle u|v \rangle$,
 - $\langle u|v_1 \oplus v_2 \rangle = \langle u|v_1 \rangle + \langle u|v_2 \rangle$,
 für alle $z \in \mathbb{C}, u, v, u_1, u_2 \in H$,
- der bzgl der Norm $|u| = \sqrt{\langle u|u \rangle}$ vollständig ist.

Ein Hilbert Raum ist also einfach ein vollständiger normierter Vektorraum, dessen Norm über ein Skalarprodukt definiert wird. Statt \oplus, \ominus und \odot schreiben wir auch für Vektoren einfacher $+, -$ und \cdot und lassen \cdot auch manchmal weg. \cdot ist die Multiplikation eines Vektors mit einem Skalar (eine komplexe Zahl), $\langle | \rangle$ dagegen beschreibt das Produkt zweier Vektoren, deren Ergebnis aber in \mathbb{C} liegt. Die Norm $|u|$ von u wird oft auch die *Länge* von u genannt. Das Skalarprodukt multipliziert zwei Vektoren und hat als Ergebnis ein Skalar. Für das Skalarprodukt existieren unterschiedliche Bezeichnungen. Unsere Bezeichnung ist in der Quantenmechanik üblich. In der Mathematik findet man statt $\langle u|v \rangle$ auch $\langle u, v \rangle, (u, v), u \cdot v, u.v$ oder nur uv , wobei Vektoren aber typographisch von Skalaren unterschieden werden, um “zu” nicht mit “uv” zu verwechseln. Ein Skalarprodukt ist in dieser Definition also in der zweiten Komponente linear und damit automatisch in der ersten antilinear, d.h. es gilt

$$\begin{aligned} \langle z_1 u_1 + z_2 u_2 | v \rangle &= \bar{z}_1 \langle u_1 | v \rangle + \bar{z}_2 \langle u_2 | v \rangle, \\ \langle u | z_1 v_1 + z_2 v_2 \rangle &= z_1 \langle u | v_1 \rangle + z_2 \langle u | v_2 \rangle. \end{aligned}$$

Die Sache verkompliziert sich leider noch dadurch, dass in der Mathematik fast immer Linearität in der ersten und Antilinearität in der zweiten Komponente gefordert wird. Wir folgen aber der Notation in der Quantenmechanik und hoffen, nicht durcheinander zu kommen.

Beispiel 6.8.1 Folgende Räume sind Hilbert Räume:

- \mathbb{C}^n der n -dimensionale Vektorraum über \mathbb{C} mit den üblichen Operationen und dem Skalarprodukt

$$\langle u|v \rangle := \sum_{j=1}^n \overline{u_j} v_j,$$

- $\ell^2 = \{u = (u_n)_{n \in \mathbb{N}} | u_n \in \mathbb{C} \wedge \sum_{n \in \mathbb{N}} |u_n|^2 \text{ konvergiert} \}$ mit

$$\langle u|v \rangle = \sum_{n \in \mathbb{N}} \overline{u_n} v_n$$

h ist unendlich dimensional mit einer Basis $\{e_i | i \in \mathbb{N}\}$, wobei die Folge e_i an i -ter Stelle eine 1 und sonst überall eine 0 enthält. Wegen $|\overline{u_n} v_n| \leq 0,5 \cdot (|u_n|^2 + |v_n|^2)$ konvergiert das innere Produkt.

- L^2 , der Raum der 2-fach integrierbaren Funktionen mit dem inneren Produkt $\langle f|g \rangle = \int f(x)g(x)dx$.

In den Beispielen in Mathematikbüchern sind die Definitionen leicht anders, da dort Linearität in der ersten Komponente gefordert wird.

Übung 6.8.1 Rechnen Sie nach, dass in jedem Hilbert Raum die Parallelogrammgleichung

$$2(|u|^2 + |v|^2) = |u + v|^2 + |u - v|^2$$

gilt. Geben Sie eine geometrische Anschauung, was dies mit einem Parallelogramm zu tun hat.

In Banach Räumen muss die Parallelogrammgleichung nicht unbedingt gelten. Gilt sie aber, dann ist ein komplexer Banach Raum bereits ein Hilbert Raum, da dann durch

$$\langle x, y \rangle := \frac{1}{4} (\|x + y\|^2 - \|x - y\|^2) + \frac{i}{4} (\|x + iy\|^2 - \|x - iy\|^2)$$

ein Skalarprodukt definiert wird, das gerade $\|\cdot\|$ als Norm besitzt, d.h., für das $|u| = \sqrt{\langle u|u \rangle}$ gilt.

In jedem Hilbert Raum gilt die Cauchy-Schwarz'sche Ungleichung

$$|\langle u|v \rangle| \leq |u| \cdot |v| \leq \frac{1}{2}(|u|^2 + |v|^2).$$

Zwei Vektoren u, v heißen *orthogonal*, $u \perp v$, falls $\langle u|v \rangle = 0$ gilt. Sind u und v orthogonal, so ist $|u + v|^2 = |u - v|^2$. Als ein Spezialfall der Parallelogrammgleichung erhält man damit den Satz des Pythagoras:

Sind u, v orthogonal zu einander, so gilt $|u + v|^2 = |u|^2 + |v|^2$

Zwei Teilmengen $U, V \subseteq H$ heißen orthogonal, falls jeder Vektor aus U orthogonal zu jedem Vektor aus V ist. Das orthogonale Komplement U^\perp von $U \subseteq H$ ist die Menge aller zu U orthogonaler Vektoren in H . Ein *Orthogonalsystem* ist eine Menge O von Vektoren von denen je zwei verschiedene orthogonal zueinander sind. Ein *Orthonormalsystem* ist normiertes Orthogonalsystem, in dem also $|u| = 1$ für $u \in O$ gilt. Man kann aus einem Orthogonalsystem immer ein Orthonormalsystem machen, in dem man jeden Vektor u durch $\frac{1}{|u|} \cdot u$ ersetzt. Eine *Orthonormalbasis* ist ein maximales Orthonormalsystem O , das also durch Hinzufügen eines weiteren Vektors kein Orthonormalsystem mehr bleibt. Achtung: in Hilbert Räumen meint man mit *Basis* immer eine Orthonormal- oder Orthogonalbasis. In Falle endlich-dimensionaler Hilbert Räume ist eine Orthonormalbasis auch Basis im Sinn von Vektorraumbasen, im Fall von unendlich-dimensionalen Hilbert Räumen aber nicht. Im unendlich-dimensionalen Fall lässt sich eine Vektorraumbasis häufig nicht einmal orthonormieren. Für eine Orthonormalbasis B in einem Hilbert Raum \mathcal{H} gilt also $\langle e, f \rangle = 0$ für $e \neq f$ und $\langle e, f \rangle = 1$ für zwei Basisvektoren e, f , und kein weiterer Vektor kann hinzugenommen werden, ohne die Orthogonalität der Basis zu verlieren. Mit Pythagoras gilt dann für eine Orthonormalbasis B und $u \in H$ mit $u = \sum_{e \in B} z_e \cdot e$:

$$|u|^2 = \left| \sum_{e \in B} z_e \cdot e \right|^2 = \sum_{e \in B} |z_e|^2 \cdot |e|^2 = \sum_{e \in B} |z_e|^2,$$

also $|u| = \sqrt{\sum_{e \in B} |z_e|^2}$. Wir gehen im Folgenden davon aus, dass alle vorkommenden Basen von Hilbert Räumen orthonormal sind.

Es seien \mathcal{H}_n , $1 \leq n \leq n_0$, endlich viele Hilbert Räume mit den jeweiligen inneren Produkten $\langle \cdot | \cdot \rangle_n$. Die *direkte Summe* $\mathcal{H} := \mathcal{H}_1 \oplus \dots \oplus \mathcal{H}_{n_0}$ besteht aus allen Vektoren aus $H_1 \times \dots \times H_{n_0}$ mit der üblichen Addition von Vektoren und Multiplikation mit Skalaren und versehen mit $\langle u | v \rangle := \langle u_1 | v_1 \rangle_1 + \dots + \langle u_{n_0} | v_{n_0} \rangle_{n_0}$ als Skalarprodukt. Damit wird auch \mathcal{H} zu einem Hilbert Raum. Es sei nun $(\mathcal{H}_n)_{n \in \mathbb{N}}$ eine unendliche Folge von Hilbert Räumen. H sei die Menge derjenigen Folgen $(u_n)_{n \in \mathbb{N}}$ mit $u_n \in H_n$, deren Reihen $\sum_{n \in \mathbb{N}} |u_n|^2$ konvergieren. H selbst ist ein Vektorraum mit der kanonischen gliedweisen Summation zweier Folgen und der ebenfalls gliedweisen Multiplikation mit einem Skalar, also $z \cdot (u_n)_{n \in \mathbb{N}} := (z \cdot u_n)_{n \in \mathbb{N}}$. Für $u = (u_n)_{n \in \mathbb{N}}$, $v = (v_n)_{n \in \mathbb{N}}$ ist dann $\langle u | v \rangle := \sum_{n \in \mathbb{N}} \langle u_n | v_n \rangle$ ein wohldefiniertes Skalarprodukt, das H zu einem Hilbert Raum macht, der sogenannten *Hilbert Summe* oder *direkten Summe* der Räume \mathcal{H}_n .

6.9 Lineare Abbildungen

Es seien $\mathcal{V}_1, \mathcal{V}_2$ zwei endlich-dimensionale Vektorräume über dem gleichen Grundkörper K . $B_1 = \{e_1, \dots, e_n\}$ sei eine Basis von \mathcal{V}_1 und $B_2 = \{f_1, \dots, f_m\}$ eine von \mathcal{V}_2 . Jedes $u \in \mathcal{V}_1$ lässt sich eindeutig darstellen als

$$u = \sum_{1 \leq i \leq n} x_i \cdot e_i \text{ mit } x_i \in K.$$

Der Spaltenvektor $(x_1 \dots x_n)^T$ ist dann die Darstellung von u in der Basis B_1 . Für eine lineare Abbildung $L : \mathcal{V}_1 \rightarrow \mathcal{V}_2$ gilt daher

$$\begin{aligned} L(u) &= L\left(\sum_{1 \leq i \leq n} x_i \cdot e_i\right) = \sum_{1 \leq i \leq n} x_i \cdot L(e_i) \\ &= \begin{pmatrix} L(e_1) & \dots & L(e_n) \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ \cdot \\ \cdot \\ \cdot \\ x_n \end{pmatrix} \\ &= \begin{pmatrix} a_{1,1} & \dots & a_{1,n} \\ \cdot \\ \cdot \\ \cdot \\ a_{m,1} & \dots & a_{m,n} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ \cdot \\ \cdot \\ \cdot \\ x_n \end{pmatrix}, \end{aligned}$$

wobei $(a_{1,i} \dots a_{m,i})^T$ die Darstellung von $L(e_i)$ in der Basis B_2 ist. Lineare Abbildungen sind also gerade Multiplikationen mit Matrizen.

Die Menge aller linearen Abbildungen von \mathcal{V}_1 nach \mathcal{V}_2 ist selbst ein Vektorraum mit der üblichen Addition von Funktionen $(L_1 + L_2)(u) = L_1(u) + L_2(u)$ und Multiplikation mit Skalaren $(c \cdot L)(u) = c \cdot L(u)$. Dieser Vektorraum wird mit

$$L(\mathcal{V}_1; \mathcal{V}_2)$$

bezeichnet. Seine Dimension ist das Produkt der Dimensionen von \mathcal{V}_1 und \mathcal{V}_2 . Besonders interessant sind die beiden Spezialfälle $\mathcal{V}_2 = K$ und $\mathcal{V}_2 = \mathcal{V}_1$.

6.9.1 $L(\mathcal{V}; K)$

Es sei nun \mathcal{V} ein Vektorraum über dem Körper \mathbb{R} oder \mathbb{C} mit einer Norm $\|\cdot\|$. Eine lineare Abbildung $f : V \rightarrow \mathbb{C}$ heißt *beschränkt*, falls eine reelle positive Zahl a existiert mit $|f(u)| \leq a|u|$ für alle $u \in H$. Für lineare Abbildungen in Normierten Räumen stimmen die Begriffe beschränkt und stetig überein, wie wir zur Übung zeigen wollen.

Übung 6.9.1 Es seien \mathcal{V} ein Linearer Raum über $K = \mathbb{R}$ oder $K = \mathbb{C}$ und $f : V \rightarrow K$ eine lineare Abbildung. Dann ist f genau dann beschränkt, wenn f stetig ist.

Beweis. Sei f beschränkt mit $|f(u)| \leq a|u|$ für alle $u \in V$ mit einem $a \in \mathbb{R}$, $a > 0$. Dann gilt

$$|f(u) - f(u_0)| = |f(u - u_0)| \leq a|u - u_0| < \epsilon \text{ für } |u - u_0| < \delta := \epsilon/a.$$

Also ist f in u_0 stetig und u_0 war beliebig.

Umgekehrt, sei f stetig. Zu zeigen ist die Existenz eines $a \in \mathbb{R}$, $a > 0$, mit $|f(u)| \leq a|u|$ für alle $u \in V$. Es sei $|f(u)| > 0$. Insbesondere ist f stetig in 0 , und es existiert zu $\epsilon = 1$ ein $\delta > 0$ mit $|f(v) - f(0)| \leq 1$ für $|v - 0| \leq \delta$. D.h., es ist $|f(v)| \leq 1$ für $|v| \leq \delta$. Man setzt $a := 1/\delta$. Für $v := (\delta/|u|) \cdot u$ gilt $|v| = \delta$ und damit $|f(v)| \leq 1$. Also

$$1 \geq |f(v)| = |f((\delta/|u|) \cdot u)| = (\delta/|u|) \cdot |f(u)|, \text{ d.h.}$$

$$|f(u)| \leq \frac{1}{\delta}|u|.$$

Ist \mathcal{V} ein Vektorraum über einen Körper K , so heißt $L(\mathcal{V}; K)$ der (algebraische) Dualraum von \mathcal{V} und wird mit \mathcal{V}^* bezeichnet. Also gilt einfach

$$\mathcal{V}^* = L(\mathcal{V}; K) = \{F : V \rightarrow K \mid f \text{ ist linear}\}.$$

Der topologische Dualraum \mathcal{V}' eines Normierten Raums \mathcal{V} ist

$$\mathcal{V}' = \{F : V \rightarrow K \mid f \text{ ist linear und stetig}\}.$$

Beschränken wir uns wieder auf Hilbert Räume, obwohl manches des Folgenden auch für einfachere Räume gilt. Lineare Abbildungen eines Hilbert Raumes in \mathbb{C} werden auch *lineare Funktionale* oder nur *Funktionalen* genannt. In vielen Büchern wird statt mit stetigen Funktionalen nur mit beschränkten Funktionalen gearbeitet, was wir ja als äquivalent gesehen haben. Im Fall eines endlich dimensionalen Hilbert Raumes stimmen algebraischer und topologischer Dualraum überein, da dann jede lineare Abbildung von H nach \mathbb{C} bereits stetig ist. \mathcal{V}^* ist selbst ein Vektorraum und es gilt $(\mathcal{V}^*)^* = \mathcal{V}$. In der Physik heißen die Elemente aus \mathcal{V}^* auch *kovariante Vektoren* oder *Kovektoren*. Ist f ein stetiges Funktional, so existiert ein eindeutiger Vektor u_f in H mit

$$f(v) = \langle v | u_f \rangle \text{ für alle } v \in H.$$

Dabei gilt $|f| = |u_f|$. Die Elemente eines Hilbert Raumes \mathcal{H} und seines topologischen Dualraumes \mathcal{H}' werden üblicherweise identifiziert mittels der isometrischen, antilinearen Abbildung

$$\omega : H^* \rightarrow H \text{ mit } \omega(f) := u_f.$$

Antilinearität bedeutet wieder

$$\omega(z_1 f + z_2 g) = \bar{z}_1 \omega(f) + \bar{z}_2 \omega(g).$$

Damit ist $\langle f|u \rangle$ definiert als

$$\langle f|u \rangle := \langle \omega(f)|u \rangle = \langle u_f|u \rangle.$$

Man findet auch häufige folgende einleuchtende Redewendungen: “Der Dualraum von \mathbb{R}^n ist \mathbb{R}^n selbst, der Dualraum von \mathbb{C}^n ist der konjugierte Raum \mathbb{C}^n und diese Identität wird durch das Skalarprodukt $\langle x|y \rangle$ gegeben.”

In der Quantenmechanik wird üblicherweise folgende von Dirac entwickelte Notation verwendet: Ein Vektor u des Hilbertraums \mathcal{H} wird in Anlehnung an die bracket-Notation $\langle u|v \rangle$ auch als $|u \rangle$ notiert und als *ket*-Vektor bezeichnet, ein Funktional f aus \mathcal{H}' wird als $\langle f|$ notiert und als *bra*-Vektor bezeichnet. Ein bra $\langle u|$ ist damit das lineare Funktional mit

$$\langle u|(v) = \langle u|(|v \rangle) = \langle u||v \rangle := \langle u|v \rangle.$$

Mit dieser Dirac’schen Schreibweise wird klar, warum Linearität in der zweiten Komponente gefordert wird.

Schauen wir uns das im Hilbertraum \mathbb{C}^n an. Als Matrix aufgefasst ist $|u \rangle \in \mathbb{C}^n$ nichts anderes als ein n -dimensionaler Spaltenvektor von komplexen Zahlen, seine Entsprechung $\langle u|$ im Dualraum ist dann einfach \bar{u}^T , das konjugierte von u als Zeilenvektor. Dann gilt genau

$$\langle u|(|v \rangle) = \langle u| \cdot |v \rangle = \langle u|v \rangle.$$

In dieser Notation ist dann $|u \rangle \langle v|$ das Funktional mit

$$|u \rangle \langle v|(|w \rangle) = |u \rangle \langle v||w \rangle = |u \rangle \langle v|w \rangle = \langle v|w \rangle \cdot |u \rangle.$$

Dabei haben wir $u \cdot z := z \cdot u$ für eine Skalar z und einen Vektor u gesetzt. $|u \rangle \langle v|$ ist das *Tensorprodukt* von u und v und wird auch als $u \otimes v$ bezeichnet. Das Funktional $|u \rangle \langle u|$ wird auch als $P[u]$ bezeichnet. Es hat die Wirkung

$$P[u](v) = |u \rangle \langle u|v \rangle = \langle u|v \rangle \cdot |u \rangle.$$

P ist also eine Projektion auf den von $|u \rangle$ aufgespannten Unterraum von \mathcal{H} . Ein Unterraum M von \mathcal{H} ist eine Teilmenge von H , die selbst ein Vektorraum ist. D.h., mit $u, v \in M$, $z_1, z_2 \in \mathbb{C}$ muss auch $z_1 u + z_2 v$ in M liegen.

6.9.2 $L(\mathcal{H}; \mathcal{H})$

Eine lineare Abbildung in $L(\mathcal{H}; \mathbb{C})$ von einem Hilbert Raum in die komplexen Zahlen haben wir ein Funktional genannt. Eine lineare Abbildung $T : M \rightarrow H$ von einem Teilraum M von \mathcal{H} in H selbst heißt ein *Operator* in \mathcal{H} . M wird dabei als $\text{Dom}(T)$, der *domain* von T bezeichnet. In der Physik werden üblicherweise Anwendungen eines Operators auf einen Vektor als Tu statt $T(u)$ wie in der Mathematik geschrieben. Wir nutzen beide Schreibweisen.

$L(\mathcal{H}; \mathcal{H})$ ist der Raum aller Operatoren in \mathcal{H} mit domain H selbst. Ein Operator $T \in L(\mathcal{H}; \mathcal{H})$ besitzt einen *adjungierten Operator* T^* , falls ein Operator T^* in $L(\mathcal{H}; \mathcal{H})$ existiert mit

$$\langle Tu|v \rangle = \langle u|T^*v \rangle \text{ für alle } u, v \in H.$$

Existiert ein adjungierter Operator T^* , so ist er eindeutig bestimmt und es gilt $(T^*)^* = T$. Jeder stetige Operator T besitzt einen adjungierten Operator mit $|T| = |T^*|$, der ebenfalls stetig ist. Ein Operator T heißt *unitär*, wenn er invertierbar ist und $T^{-1} = T^*$ gilt. Unitäre Operatoren sind besonders ausgezeichnet. Es sind genau die bijektiven linearer Operationen, die die Norm, $|Tu| = |u|$, oder äquivalent dazu, das Skalarprodukt $\langle Tu|Tv \rangle = \langle u|v \rangle$, invariant lassen, oder, ebenfalls äquivalent dazu, die die Einheitskugel invariant lassen, d.h. für die $|Tu| = 1$ für $|u| = 1$ gilt.

Im Fall endlich dimensionaler Vektorräume sind alle lineare Operatoren stetig und als Matrizenmultiplikationen mit quadratischen Matrizen darstellbar. Es sei $A \in \mathbb{C}^{n,n}$ eine quadratische $n \times n$ -Matrix über \mathbb{C} mit den Elementen $z_{i,j}$ an der Stelle $A(i, j) = A_{i,j}$. E bezeichnet für eine aus dem Kontext bekannte Dimension stets die quadratische Einheitsmatrix dieser Dimension, d.h. es gilt $E_{i,j} = \delta_{i,j} := \delta_i(j)$. Folgende Begriffe sind zu A üblich:

- die *inverse* A^{-1} , die nicht immer existieren muss, ist die Matrix mit

$$AA^{-1} = A^{-1}A = E,$$

- die *konjugierte* \bar{A} besteht aus den Elementen

$$\bar{A}_{i,j} = \overline{z_{i,j}},$$

- die *transponierte* A^T , auch oft A' geschrieben, mit

$$A_{i,j}^T = A_{j,i},$$

- die *adjungierte* oder *begleitende* A^* mit

$$A^* = (\overline{A})^T.$$

A heißt

- *idempotent* oder eine *Projektion*, wenn gilt

$$AA = A,$$

- *symmetrisch*, wenn sie gleich ihrer Transponierten ist,
- *hermitesch* oder *selbstadjungiert*, wenn sie gleich ihrer Adjungierten ist,
- *orthogonal*, wenn gilt A nur reelle Werte enthält und es gilt

$$AA^T = E,$$

- *unitär*, wenn gilt

$$AA^* = E.$$

Unitäre Matrizen haben folgende Eigenschaften:

- sie sind invertierbar mit $A^{-1} = A^*$,
- die Norm ihrer Determinante ist 1,
- sie lassen das Skalarprodukt invariant, also

$$\langle Au | Av \rangle = \langle u | v \rangle,$$

- ihre Eigenwerte haben die Norm 1,
- das Produkt zweier unitärer Matrizen ist wieder unitär,
- die Spalten einer unitären Matrix bilden eine orthonormale Basis des \mathbb{C}^n .

Umgekehrt gilt auch, dass jede quadratische Matrix $A \in \mathbb{C}^{n,n}$, die das Skalarprodukt invariant ist, unitär sein muss. Unitäre Matrizen sind im endlich-dimensionalen Fall genau die unitären Operatoren. Jede Matrix $A \in \mathbb{C}^{n,n}$ definiert kanonisch einen linearen Operator von \mathbb{C}^n in \mathbb{C}^n durch $u \rightarrow Au$. Damit übertragen sich alle diese Begriffe auf Operatoren. So können wir Begriffe wie idempotente Transformation verstehen. Alle diese Begriffe lassen sich auch auf Operatoren auf unendlich-dimensionalen Räumen übertragen, wie wir es nur für den Begriff “unitär” gemacht haben.

6.10 Tensorprodukt

In der Quantenmechanik sind Tensorprodukte so essentiell, dass sie hier ihr eigenes Kapitel erhalten. Wie wollen hier klären, was das Tensorprodukt $\mathcal{V}_1 \otimes \dots \otimes \mathcal{V}_n$ von n Vektorräumen \mathcal{V}_i über einem gemeinsamen Grundkörper K ist. Zu beachten ist, dass verschiedene Definitionen des Tensorprodukts auf unterschiedlichen Abstraktionsniveaus gebräuchlich sind. Wir werden uns vom Abstrakten zum Konkreten vorarbeiten.

Eine Abbildung $f : V_1 \times \dots \times V_n \rightarrow V$ in einen weiteren Vektorraum \mathcal{V} über K heißt *multilinear*, falls f in jeder Komponente linear ist, das heißt, falls für alle i , $1 \leq i \leq n$ und alle $u_i \in V_i$ gilt

$$f_{u_1, \dots, u_{i-1}, u_{i+1}, \dots, u_n}(u) := f(u_1, \dots, u_{i-1}, u, u_{i+1}, \dots, u_n) \text{ ist linear.}$$

Ist dabei \mathcal{V} selbst der Grundkörper K , so heißt f *Multilinearform*, für $n = 2$ spricht man von *bilinearen Abbildungen* bzw von einer *Bilinearform* bei $V = K$. Analog ist eine *Linearform* ein $f \in L(\mathcal{V}_1; K)$. $L(\mathcal{V}_1, \dots, \mathcal{V}_n; V)$ ist selbst ein Vektorraum über K der Dimension $\dim(\mathcal{V}_1) \cdot \dots \cdot \dim(\mathcal{V}_n) \cdot \dim(V)$ mit der üblichen Addition von Funktionen und Multiplikation mit Skalaren. Eine Linearform ist also nichts anderes als ein Element des Dualraums eines Vektorraumes.

Schauen wir uns Bilinearität mal an: Es seien $f \in L(\mathcal{V}_1, \mathcal{V}_2; \mathcal{V})$, $u, u' \in V_1$, $v, v' \in V_2$ und $c_i, d_i \in K$, so gilt

$$\begin{aligned} f(c_1u + c_2u', d_1v + d_2v') &= c_1f(u, d_1v + d_2v') + c_2f(u', d_1v + d_2v') \\ &= c_1d_1f(u, v) + c_1d_2f(u, v') + c_2d_1f(u', v) \\ &\quad + c_2d_2f(u', v'). \end{aligned}$$

Gesucht ist ein Kalkül, der Untersuchungen von bilinearen Abbildung auf die Untersuchung von linearen zurückführt. Das *Tensorprodukt* $\mathcal{V}_1 \otimes \mathcal{V}_2$ von \mathcal{V}_1 und \mathcal{V}_2 wird der Vektorraum sein, so dass jede bilineare Abbildung in $L(\mathcal{V}_1, \mathcal{V}_2; \mathcal{V})$ einer linearen in $L(\mathcal{V}_1 \otimes \mathcal{V}_2; \mathcal{V})$ entspricht.

Beginnen wir mit $\mathcal{V} = K$. Es existiert eine kanonische Abbildung

$$\phi : V_1 \times V_2 \rightarrow L^*(V_1 \times V_2; K) \text{ mit}$$

$$(\phi(u, v))(h) := h(u, v)$$

für $h \in L(V_1 \times V_2; K)$. ϕ ist bilinear und der von $\phi(V_1 \times V_2)$ erzeugte Untervektorraum in $L^*(V_1 \times V_2; K)$ heißt das *Tensorprodukt* von \mathcal{V}_1 und \mathcal{V}_2 und wird mit $\mathcal{V}_1 \otimes \mathcal{V}_2$ bezeichnet und $\phi(u, v)$ mit $u \otimes v$. Anders gesagt,

der algebraische Dualraum des Tensorproduktes $\mathcal{V}_1 \otimes \mathcal{V}_2$ ist $L(\mathcal{V}_1, \mathcal{V}_2; K)$, also

$$\mathcal{V}_1 \otimes \mathcal{V}_2 = L^*(\mathcal{V}_1, \mathcal{V}_2; K).$$

Diese Eigenschaft überträgt sich von Bilinearformen auf bilineare Abbildungen:

Sei $f \in L(\mathcal{V}_1 \otimes \mathcal{V}_2; V)$ dann sei $\psi(f) := f \circ \phi$. Insbesondere gilt damit

$$\psi(f)(u, v) = f(\phi(x, y)) = f(x \otimes y).$$

ψ ist linear und bijektiv und vermittelt die Isomorphie

$$L(\mathcal{V}_1 \otimes \mathcal{V}_2; V) \approx L(\mathcal{V}_1, \mathcal{V}_2; \mathcal{V}).$$

Hier hat f als lineare Abbildung nur ein Argument, nämlich $u \otimes v$, $\psi(f)$ als bilineare Abbildung die beiden Argumente u und v .

Dieser Zusammenhang wird häufig kategoriell definiert und man findet folgende abstraktere *Universaldefinition* des Tensorproduktes: $\mathcal{V}_1 \otimes \mathcal{V}_2$ ist derjenige Vektorraum, zu dem eine bilineare Abbildung

$$\Phi : V_1 \times V_2 \rightarrow V_1 \otimes V_2$$

existiert, so dass man für jeden weiteren Vektorraum W über K jede **bilineare** Abbildung

$$B : V_1 \times V_2 \rightarrow W$$

zu einer eindeutig bestimmten **linearen** Abbildung

$$L : V_1 \otimes V_2 \rightarrow W$$

so fortsetzen kann, dass gilt:

$$B(u, v) = L(\Phi(u, v)).$$

Für $\Phi(u, v)$ schreibt man dann $u \otimes v$. Damit wird letzte Gleichung zu $B(u, v) = L(u \otimes v)$. B hat zwei Argumente, L nur noch ein Argument.

Jetzt alles noch mal etwas konkreter.

Es seien \mathcal{V}_1 und \mathcal{V}_2 nun endlich-dimensionale Vektorräume über einem gemeinsamen Grundkörper K mit den Basen $E = \{e_i | 1 \leq i \leq n\}$ und $F = \{f_j | 1 \leq j \leq m\}$. Das Tensorprodukt $\mathcal{V}_1 \otimes \mathcal{V}_2$ von \mathcal{V}_1 und \mathcal{V}_2 wird dann ein $n \cdot m$ -dimensionaler Vektorraum über K , dessen Basis eineindeutig $E \times F = \{(e_i, f_j) | 1 \leq i \leq n \wedge 1 \leq j \leq m\}$ zugeordnet werden kann. Das (e_i, f_j) zugeordnete Basiselement wird als $e_i \otimes f_j$, $|e_i\rangle \otimes |f_j\rangle$ oder als $|e_i f_j\rangle$ geschrieben. Zwei Elemente u aus V_1 und v aus V_2 lassen sich eindeutig als

$$u = \sum_{1 \leq i \leq n} x_i \cdot e_i \text{ und } v = \sum_{1 \leq j \leq m} y_j \cdot f_j,$$

mit Skalaren x_i, y_j darstellen. Das Tensorprodukt $u \otimes v$ von u und v wird damit gerade

$$u \otimes v := \sum_{1 \leq i \leq j} \sum_{1 \leq j \leq m} x_i y_j \cdot e_i \otimes f_j.$$

Damit gelten folgende Bilinearitätsregeln für $u, u_1, u_2 \in V_1, v, v_1, v_2 \in V_2, z \in \mathbb{C}$:

$$(u_1 + u_2) \otimes v = u_1 \otimes v + u_2 \otimes v, \quad (6.3)$$

$$u \otimes (v_1 + v_2) = u \otimes v_1 + u \otimes v_2, \quad (6.4)$$

$$\begin{aligned} (zu) \otimes v &= z \cdot (u \otimes v) \\ &= u \otimes (zv), \end{aligned} \quad (6.5)$$

und \otimes verhält sich wie ein Produkt. Allerdings gilt kein Kommutativgesetz.

Man kann nun $\mathcal{V}_1 \otimes \mathcal{V}_2$ aus dem kartesisches Produkt $\mathcal{V}_1 \times \mathcal{V}_2$ konstruieren, in dem man die Bilinearitätsregeln erzwingt. Dazu bildet man den Quotienten $(V_1 \times V_2)/\cong$ mit der durch

$$\begin{aligned} ((u_1 + u_2), v) &\cong (u_1, v) + (u_2, v), \\ (u, (v_1 + v_2)) &\cong (u, v_1) + (u, v_2), \\ (zu, v) &\cong (u, zv) \cong z(u, v) \end{aligned}$$

erzeugten Äquivalenzrelation.

Das Skalarprodukt als Abbildung $\otimes : (u, v) \rightarrow u \otimes v$ zweier Vektoren (und nicht Räume) betrachtet ist natürlich selbst bilinear.

Als dreifaches Tensorprodukt $\mathcal{V}_1 \otimes \mathcal{V}_2 \otimes \mathcal{V}_3$ definiert man

$$V_1 \otimes V_2 \otimes V_3 := (V_1 \otimes V_2) \otimes V_3 = V_1 \otimes (V_2 \otimes V_3),$$

indem man die Elemente $(u \otimes v) \otimes w$ und $u \otimes (v \otimes w)$ identifiziert. Für Basen $E_i = \{e_j^i | 1 \leq j \leq n_i\}$ von V_i wird das dem Tupel $(e_{i_1}^1, e_{i_2}^2, e_{i_3}^3)$ entsprechende Basiselement in $V_1 \otimes V_2 \otimes V_3$ entsprechend als

$$|e_{i_1}^1 e_{i_2}^2 e_{i_3}^3\rangle \text{ oder } |e_{i_1}^1\rangle \otimes |e_{i_2}^2\rangle \otimes |e_{i_3}^3\rangle$$

bezeichnet. Mit $v_i = \sum_{1 \leq j \leq n_i} z_j^i e_j^i$ gilt dann wieder

$$v_1 \otimes v_2 \otimes v_3 := \sum_{1 \leq j_1 \leq n_1} \sum_{1 \leq j_2 \leq n_2} \sum_{1 \leq j_3 \leq n_3} z_{j_1}^1 z_{j_2}^2 z_{j_3}^3 \cdot |e_{j_1}^1 e_{j_2}^2 e_{j_3}^3\rangle.$$

Für beliebiges n benutzt man den folgenden einfachen Zusammenhang:

$$L(\mathcal{V}_1, \dots, \mathcal{V}_n; \mathcal{V}) \approx L(\mathcal{V}_i; L(\mathcal{V}_1, \dots, \mathcal{V}_{i-1}, \mathcal{V}_{i+1}, \dots, \mathcal{V}_n; \mathcal{V}))$$

für jedes i . Für das Tensorprodukt $X = \mathcal{V}_1 \otimes \dots \otimes \mathcal{V}_n$ gilt

$$L(X; \mathcal{V}) \approx L(\mathcal{V}_1, \dots, \mathcal{V}_n; \mathcal{V}).$$

$\mathcal{V}_1 \otimes \dots \otimes \mathcal{V}_n$ besitzt die Dimension $\dim(\mathcal{V}_1) \cdot \dots \cdot \dim(\mathcal{V}_n)$.

Analog entsprechen die Basiselemente von $V_1 \otimes \dots \otimes V_n$ den n -Tupeln $(e_1^1, \dots, e_{i_n}^n)$ von Basisvektoren e_j^i der V_i und es ist

$$v_1 \otimes \dots \otimes v_n := \sum_{1 \leq j_i \leq n_i, 1 \leq i \leq n} z_{j_1}^1 \cdot \dots \cdot z_{j_n}^n \cdot |e_{j_1}^1 \dots e_{j_n}^n\rangle,$$

für $v_i = \sum_{1 \leq j \leq n_i} z_j^i e_j^i$, $1 \leq i \leq n$.

Sind die \mathcal{V}_i selbst Hilbert Räume mit einem Skalarprodukt $\langle | \rangle_i$, so definiert

$$\langle u_1 \otimes \dots \otimes u_n | v_1 \otimes \dots \otimes v_n \rangle := \langle u_1 | v_1 \rangle_1 \cdot \dots \cdot \langle u_n | v_n \rangle_n$$

ein Skalarprodukt auf $V_1 \otimes \dots \otimes V_n$.

Neben dem Tensorprodukt von Vektoren findet man auch ein tensorielles Produkt von linearen Abbildungen. Dazu seien \mathcal{W}_i weitere Vektorräume über den gleichen Grundkörper K und $f_i : V_i \rightarrow W_i$ seien lineare Abbildungen aus $L(\mathcal{V}_i; \mathcal{W}_i)$. Dann ist die Abbildung

$$f(u_1, \dots, u_n) := f_1(u_1) \otimes \dots \otimes f_n(u_n)$$

multilinear und liegt somit in $L(\mathcal{V}_1, \dots, \mathcal{V}_n; \mathcal{W}_1 \otimes \dots \otimes \mathcal{W}_n)$. Wegen der universellen Eigenschaft des Tensorproduktes existiert dann zu f eine eindeutig bestimmte Abbildung $\hat{f} : V_1 \otimes \dots \otimes V_n \rightarrow W_1 \otimes \dots \otimes W_n$ mit

$$f(u_1, \dots, u_n) = \hat{f}(u_1 \otimes \dots \otimes u_n).$$

\hat{f} heißt das *tensorielle Produkt* der f_i und wird als $f_1 \underline{\otimes} \dots \underline{\otimes} f_n$ bezeichnet. Da jedes f_i als Element des Vektorraumes $L(\mathcal{V}_i; \mathcal{W}_i)$ selbst ein Vektor ist, ist auch $f_1 \otimes \dots \otimes f_n$ erklärt. Der Unterschied ist nur, dass

$$f_1 \otimes \dots \otimes f_n \in L(\mathcal{V}_1; \mathcal{W}_1) \otimes \dots \otimes L(\mathcal{V}_n; \mathcal{W}_n)$$

gilt, aber

$$f_1 \underline{\otimes} \dots \underline{\otimes} f_n \in L(\mathcal{V}_1 \otimes \dots \otimes \mathcal{V}_n; \mathcal{W}_1 \otimes \dots \otimes \mathcal{W}_n).$$

Diese beiden Räume sind aber isomorph.

Stellen wir nun zwei lineare Abbildungen $f_i \in L(\mathcal{V}_i; \mathcal{W}_i)$ als Matrizen A_i dar, so wird $f_1 \underline{\otimes} f_2$ gerade durch das Tensorprodukt $A_1 \otimes A_2$ der beiden Matrizen dargestellt. Das Tensorprodukt zweier Matrizen heißt auch

Kroneckerprodukt und ist für beliebig dimensionale Matrizen definiert durch

$$A \otimes B := (a_{i,j} \cdot B)_{i,j} \text{ für } A = (a_{i,j})_{i,j}.$$

Ist A eine $m \times n$ -Matrix und B eine $p \times r$ -Matrix so ist das Kronecker-Produkt $C = A \otimes B$ also eine $mp \times nr$ -Matrix. Zum Beispiel

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \otimes \begin{pmatrix} 5 & 6 \\ 7 & 8 \\ 9 & 10 \end{pmatrix} = \begin{pmatrix} 1 \cdot \begin{pmatrix} 5 & 6 \\ 7 & 8 \\ 9 & 10 \end{pmatrix} & 2 \cdot \begin{pmatrix} 5 & 6 \\ 7 & 8 \\ 9 & 10 \end{pmatrix} \\ 3 \cdot \begin{pmatrix} 5 & 6 \\ 7 & 8 \\ 9 & 10 \end{pmatrix} & 4 \cdot \begin{pmatrix} 5 & 6 \\ 7 & 8 \\ 9 & 10 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 5 & 6 & 10 & 12 \\ 7 & 8 & 14 & 16 \\ 9 & 10 & 18 & 20 \\ 15 & 18 & 20 & 24 \\ 21 & 24 & 28 & 32 \\ 27 & 30 & 36 & 40 \end{pmatrix}$$

Literaturverzeichnis

- [1] R.M. Balzer. An 8-state minial time solution to the firing squad synchronization problem. *Information and Control*, 10:22–42, 1967.
- [2] E.R. Banks. Information processing and transmission in cellular automata. Technical Report MAC-81, MIT, Project MAC, 1971.
- [3] E.F. Codd. *Cellular Automata*. Academic Press, 1968.
- [4] F. Commoner. Deadlocks in Petri nets. Technical Report CA-7206-2311, Applied Data Research, Inc., 1972.
- [5] F. Commoner, A.W. Holt, S.Even, and A.Pnueli. Marked directed graphs. *JCSS*, 5:511–523, 1971.
- [6] J. Conway. Mathematical games, ed. by Martin Gardner. *Scientific America*, Oct., 1970.
- [7] E. Fredkin D. B. Miller. Two-state, reversible, universal cellular automata in three dimensions. *Proceedings of the 2nd conference on Computing frontiers, Ischia, Italy*, 2005.
- [8] R. Feynman. Simulating physics with computers. *J. Theoretical Physics*, 21(6/7), 1982.
- [9] U. Golze. *Endliche, periodische und rekursive zellulare Konfigurationen: Vorgängerberechnung und Garten-Eden-Probleme. Dissertation*. PhD thesis, TU Hannover, 1975.
- [10] J. Holland. *Adaption in Natural and Artificial Systems*. The MIT Press, Neuauflage, ISBN: 078-0-262-58111-0, 1992.
- [11] K. Culik II and I. Fris. The decidability of the equivalence problem for d0l-systems. *Information and Control*, 33:20–39, 1977.
- [12] K. Erk, L. Priese. *Theoretische Informatik*. Springer Verlag, 2000.

- [13] K. Zuse. *Rechnender Raum*. Verlag Friedrich Vieweg & Sohn, Braunschweig, 1967.
- [14] K. Zuse. *Ansätze einer Theorie des Netzautomaten*. Berichte der Deutsche Akademie der Naturforscher Leopoldina, 220, 1975.
- [15] J. Kari. Reversibility and surjectivity problems of cellular automata. *JCSS*, 48:149–182, 1994.
- [16] K. Lautenbach. Exakte Bedingungen der Lebendigkeit für eine Klasse von Petri-Netzen. *GMD, Bericht Nr. 92, (Dissertation)*, 1973.
- [17] A. Lindenmayer. Mathematical models for cellular interactions in development, part 1. *J. Theoretical Biology*, 18:280–299, 1968.
- [18] R.J. Lipton. The reachability problem requires exponential space. Technical Report 62, Yale University, 1976.
- [19] M. Hack. Analysis of production schemata by petri nets. Technical Report MAC TR-94, Project MAC, MIT, 1972.
- [20] M. Nielsen. Equivalence of L-systems. In *LSystems, LNCS 15*, pages 142–145, 1974.
- [21] A. Holliger P. Rosenstiehl, J.R. Fiksel. Intelligent graphs: Networks of finite automata capable of solving graph problems. In *Graph Theory and Computing, Read,R.C, ed.*, pages 219–265, 1972.
- [22] S. Patil. Limitations and capabilities of Dijkstra’s primitives for coordination among processes. Technical Report Computation Structure Group Memo 57, Project MAC, MIT, 1971.
- [23] A. Paz and A. Salomaa. Integral sequential word functions and growth equivalences of Lindenmayer systems. *Information and Control*, 23:313–343, 1973.
- [24] C.A. Petri. Kommunikation mit Automaten. Technical Report Schriften des IMM Nr. 2 (Dissertation), Universität Bonn, 1962.
- [25] Lutz Priese, Patrick Sturm, and Haojun Wang. Hierarchical cell structures for segmentation of voxel images. In *Image Analysis: 14th Scandinavian Conference, SCIA 2005, Joensuu, Finland, June 19-22, 2005.*, LNCS 3540, pages 6–16. Springer Verlag, 6 2005.
- [26] R. Vollmar. *Algorithmen in Zellularautomaten*. Teubner Studienbücher, 1797.

- [27] I. Rechenberg. Evolutionsstrategie. *problemata frommann-holzberg 15*, Friedrich Frommann Verlag, ISBN 3 7728 0373 3, 1973.
- [28] R.Feynman. *QED*. Piper, 1992.
- [29] G. Rozenberg. Theors of L systems: from the point of view from formal language theors. In *L-Systems, LNCS 15*, pages 1–23, 1974.
- [30] Y. Patt S. Amoroso, G. Cooper. Some clarification of the concept of a garden-of-eden configuration. *JCSS*, 10:77–82, 1975.
- [31] S. Wolfram. *A new kind of science*. Wolfram Media, Inc., 2002.
- [32] A. Salomaa. *Formal Languages*. Academic Press, Inc., New York and London, 1973.
- [33] P. Shor. Algorithms for quantum computation: Discrete log and factoring. In *Proceedings of the 35th Annual Symposium on the Foundations of Computer Science*, pages 20–22, 1994.
- [34] Legendi T. and Kotona E. A 5-state solution of the early bird problem in a one-dimendional cellular space. *Acta Cybernetica*, 1981.
- [35] T. Toffoli. Computation and construction universality of reversible cellular automata. *JCSS*, 15:213–231, 1977.
- [36] Ipke Wachsmuth. *Simultane zellulare Kalküle und lokal-synchrone Zellularautomaten*. PhD thesis, Universität Hannover, 1980.
- [37] T. Winograd. A simple algorithm for self-replication. Technical Report AI-memo 197, MIT, 1970 and 1967.
- [38] K. Zuse. Zusammenfassender Bericht über meine bisherigen Arbeiten auf dem Gebiet der zellularen Automaten. In *Beiträge zur Theorie der Polyautomaten*, U.Golze, R. Vollmar, ed., pages 117–121, 1977.