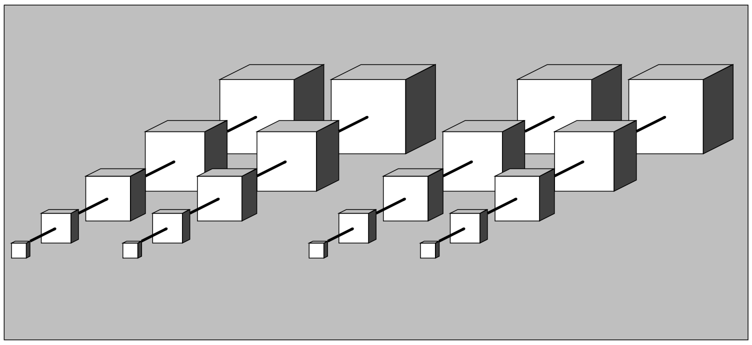
Arbeitsgruppe Informatik
Universität Giessen
Arndtstr. 2, D-35392 Giessen, Germany

Parallele Automaten

Martin Kutrib

Bericht 9401 Februar 1994

JUSTUS-LIEBIG-
UNIVERSITÄT
GIESSEN



Parallele Automaten

Martin Kutrib

Inhaltsverzeichnis

1 Vorwort und Einleitung

Im Jahre 1936 veröffentlichte der Mathematiker A. M. Turing seine berühmte Arbeit, in der er die Berechenbarkeit von Funktionen mittels mechanischer Prozeduren untersuchte. Er schlug ein formales Modell vor, das — gemäß genau festgelegter Regeln — in der Lage ist, Zeichenreihen gewissermaßen mechanisch zu manipulieren.

Mit einem auf diesem Modell basierenden Berechenbarkeitsbegriff unterschied er berechenbare und nichtberechenbare Funktionen. In derselben Arbeit zeigte Turing die Äquivalenz seines Berechenbarkeitsbegriffes mit dem des zuvor von A. Church (1936) vorgeschlagenen „effektiven Verfahrens“. Das formale Modell wird heute üblicherweise als Turingmaschine bezeichnet. Die große Bedeutung der Turingmaschine gründet sich zum einen wesentlich auf die Churchsche These, daß jede intuitiv berechenbare Funktion mit einem effektiven Verfahren — und damit auch mit einer Turingmaschine — berechnet werden kann. Die These wird unter anderem dadurch gestützt, daß sich bisher alle Berechenbarkeitsbegriffe als äquivalent zu dem von Turing und Church erwiesen haben.

Weitere Bedeutung erlangt die Turingmaschine dadurch, daß sie im Kern bereits die Fähigkeiten moderner von-Neumann-Universalrechner widerspiegelt. So ist es naheliegend, die berechenbaren Funktionen weiter zu klassifizieren. Eine Funktion etwa, deren Berechnung mehr Aufwand erfordert als die einer anderen, wird im allgemeinen auch auf modernen Rechenanlagen schwieriger zu berechnen sein. Bei geeigneter Präzisierung des Begriffes „Aufwand“ — z.B. durch Zeit oder Raum — lassen sich die berechenbaren Funktionen in sogenannte Komplexitätsklassen einteilen.

Der Nachweis, daß eine Funktion einer bestimmten Komplexitätsklasse angehört, wird durch die präzise Formalisierung des intuitiven Berechenbarkeitsbegriffes überhaupt erst möglich. Andererseits muß eine Einteilung in Komplexitätsklassen natürlich immer im Zusammenhang mit dem zugrundeliegenden Modell gesehen werden, da diese ja gerade durch das Modell festgelegt werden.

Eine weitere Problematik ergibt sich aus der Tatsache, daß immer mehrere auf dem Modell basierende Algorithmen existieren, die dieselbe Funktion berechnen. Es ist also zu unterscheiden, ob die Zugehörigkeit zu einer Komplexitätsklasse eine Eigenschaft des Algorithmus oder der durch ihn implementierten Funktion ist. Der Nachweis dieser Eigenschaft für Funktionen ist insofern nicht unproblematisch, da er für alle diese Funktion berechnenden Algorithmen geführt werden muß.

Auf der anderen Seite ergeben sich hieraus weitere Anwendungen von Turingmaschinen. Die Festlegung geeigneter Komplexitätsmaße vorausgesetzt, lassen sich Algorithmen mit formalen Methoden analysieren und bewerten. Auf diese Weise kann durch die Wahl eines geeigneten Algorithmus der Rechenaufwand auf realen Rechnern optimiert werden.

Ein weiterer Aspekt der Modellierung ist die Suche nach neuartigen Prinzipien der Rechnerarchitektur. Da die Klasse der berechenbaren Funktionen nach der Churchschen These beim Übergang von Turingmaschinen zu mächtigeren Modellen nicht erweitert werden kann, stellt sich hier die Frage, ob die Einteilung in Komplexitätsklassen evtl. günstiger ist als bei Turingmaschinen. Was dabei unter einer günstigen Einteilung verstanden werden soll, wird im allgemeinen individuell festzulegen sein.

Ein wesentlicher Faktor bei der Leistungsbewertung von Rechensystemen ist die Geschwindigkeit; also eine Einteilung der (auf dem System) berechenbaren Funktionen anhand des Zeitaufwandes, der für die Berechnung notwendig ist. Eine Möglichkeit, den Zeitaufwand zu reduzieren, liegt im Übergang von sequentiell arbeitenden Turingmaschinen zu Modellen, die die Fähigkeit zur Parallelverarbeitung besitzen. Wenn bei dieser Vorgehensweise auch nicht erwartet werden kann, sämtliche Funktionen schneller berechnen zu können, so erhofft man sich doch Zeitvorteile bei Problemen mit inhärenter Parallelität.

In der Vergangenheit ist diesbezüglich eine Vielzahl von Vorschlägen unterbreitet worden. J. von Neumann (1966) hat im Zusammenhang mit der Problematik selbstreproduzierender Systeme ein paralleles, berechnungsuniverselles Modell vorgeschlagen, das heute allgemein als Zellularraum bezeichnet wird (H. Müller (1978)); berechnungsuniversell in dem Sinne, daß jede durch Turingmaschinen berechenbare Funktion auch in einem Zellularraum berechnet werden kann. In der Folgezeit wurden verschiedene modifizierte und verallgemeinerte Varianten der Zellularräume untersucht, für die A. R. Smith III (1976) den Oberbegriff „Polyautomaten“ prägte.

Polyautomaten können als Modell für parallele Rechensysteme aufgefaßt werden, die über sehr viele einfache Verarbeitungselemente ohne gemeinsamen Speicher verfügen. Die Kommunikationsmöglichkeiten der Elemente sind dabei derart eingeschränkt, daß globale Ergebnisse nur aus parallelen, lokalen Entscheidungen gewonnen werden können. Derartige Systeme (bzw. die in solchen Systemen implementierbaren Algorithmen) sind von P. Rosenstiehl, J. R. Fiksel und A. Holliger (1972) als myopisch bezeichnet worden.

Im Zusammenhang mit der Modellierung neuer Rechnerarchitekturen stellt sich unter anderem die Frage nach der Realisierbarkeit, insbesondere im Hinblick auf die Anzahl der Verarbeitungselemente. Verfügte einer der ersten realisierten Parallelrechner — der ILLIAC IV, ein sogenannter Feldrechner — noch über nur 64 Prozessoren, so wurde mit dem Durchbruch der VLSI-Technologie der Bau von Rechenanlagen mit einigen zehntausend Prozessoren möglich (W. D. Hillis (1985)).
T. Toffoli und N. Margolus (1988) sowie K. Preston Jr. und M. J. B. Duff (1984) beschreiben mehrere Projekte, in denen Ergebnisse aus der Theorie der Polyautomaten unmittelbare Verwendung finden.

Für die Zukunft ist durch das Fortschreiten der Nanotechnik die Entwicklung neuartiger, massiv paralleler Rechner zu erwarten, wobei Halbleiter, die auf sogenannten Quantenpunkt-Gittern basieren, die Grundlage für Computer mit ungeahnter Leistungsfähigkeit bilden könnten (M. A. Reed (1993)).

Das vorliegende Skriptum widmet sich der Einführung in die Methoden, komplexe parallele Systeme auf der Grundlage formaler (Automaten-)Modelle zu untersuchen. Die Modelle werden hinsichtlich ihrer Eigenschaften und Fähigkeiten, formale Sprachen zu verarbeiten, betrachtet. Formale Sprachen bzw. die sie umfassenden Zeichenfolgen werden somit stellvertretend für z.B. Programmkode, Prozeßbeschreibungen oder anderen Eingabedaten gewählt; insbesondere im Hinblick auf deren Verarbeitung durch Automaten und/oder Ersetzungskalküle.

Um den Stoffumfang zu begrenzen, konnten hier aus der Vielzahl paralleler Automaten(modelle) nur einige exemplarisch ausgewählt werden. Der kundige Leser wird ferner bemerken, daß es sich ausschließlich um Modelle für Systeme ohne gemeinsamen Speicher handelt.

An dieser Stelle möchte ich mich bei all denjenigen bedanken, die zur Fertigstellung des Skriptums beigetragen haben. Insbesondere sind dies die Hörer der gleichnamigen Vorlesung und meine Kollegen Dr. Uwe Meyer und Dr. Andreas Mischnick für ihre Anregungen und Verbesserungsvorschläge, mein Chef Prof. Dr. Henner Kröger für die Motivation und den Freiraum und Karin Wagner für das Durchlesen vorläufiger Versionen.

2 Grundlagen

In diesem Kapitel werden zunächst grundlegende, im weiteren Verlauf benötigte Begriffe und Zusammenhänge aus der Theorie sequentieller Automaten rekapituliert. Es handelt sich dabei im wesentlichen um die Chomsky-Hierarchie formaler Sprachen und die damit verknüpfte Hierarchie sequentieller Automaten. Gleichzeitig bietet sich so die Gelegenheit, verschiedene Modelle formal zu definieren und damit eine einheitliche Bezeichnungsweise zu ermöglichen.

$\mathbb{N} := \{1, 2, 3, \ldots\}$ sei die Menge der natürlichen, $\mathbb{Z}$ die Menge der ganzen und $\mathbb{R}^+$ die Menge der positiven reellen Zahlen. Die Vereinigungen $\mathbb{N} \cup \{0\}$ und $\mathbb{R}^+ \cup \{0\}$ werden mit $\mathbb{N}_0$ und $\mathbb{R}_0^+$ bezeichnet.

Für eine Menge M bezeichnet $|M|$ deren Mächtigkeit, M^d das d-fache Kartesische Produkt und $\mathscr{P}(M)$ die Potenzmenge von M. Für das Kartesische Produkt (verschiedener) Mengen M und N wird auch die Schreibweise $M \times N$ benutzt.

Sind $M, N \subseteq \mathbb{Z}^d$ und $k \in \mathbb{N}_0$, dann wird $M + N := \{m + n | \, m \in M \;\wedge\; n \in N\}$ als *Summe* von M und N bezeichnet. kX wird folgendermaßen rekursiv definiert:

$$0X := \{(0, \ldots, 0)\}, \qquad (k+1)X := kX + X.$$

Es sei $t = (t_1, \ldots, t_n)$ ein n-Tupel. Mit $\pi_1(t) := t_1, \ldots, \pi_n(t) := t_n$ werden die *Projektionen* von t bezeichnet. Es sei $t = ((t_{1_1}, \ldots, t_{1_m}), \ldots, (t_{n_1}, \ldots t_{n_m}))$ ein n-Tupel von m-Tupeln, dann wird $\bar{\pi}_i(t) := (\pi_i((t_{1_1}, \ldots, t_{1_m})), \ldots, \pi_i((t_{n_1}, \ldots, t_{n_m})))$ für $i \in \{1, \ldots, m\}$ *erweiterte Projektion* genannt. Die Komposition von (erweiterten) Projektionen $\pi_i(\pi_j) = \pi_j \circ \pi_i$ wird durch die Schreibweise $\pi_{i,j}$ abgekürzt.

2.1 Formale Sprachen

Definition 2.1. Endliche, nichtleere Mengen heißen *Alphabete*.

Die noch zu präzisierenden Automatenmodelle werden im allgemeinen auf Wörtern operieren. Die dabei vorkommenden Alphabete und Zustandsmengen entsprechen häufig Kartesischen Produkten. Um die Zusammenhänge klarer darstellen zu können, wurden für die Konkatenation und die Produktbildung unterschiedliche Notationen gewählt. So bezeichnet $(A^2)^{\cdot 3}$ die Menge der Wörter mit Länge drei, deren Zeichen Paare mit Komponenten aus einer Menge A sind. Es sei $A := \{\mathtt{a}, \mathtt{b}\}$, dann ist $(\mathtt{a}, \mathtt{a})(\mathtt{b}, \mathtt{a})(\mathtt{b}, \mathtt{a})$ ein entsprechendes Wort. $(A^{\cdot 2})^3$ bezeichnet hingegen das dreifache Kartesische Produkt einer Menge, deren Elemente Wörter der Länge zwei über einem Alphabet A sind; zum Beispiel ist $(\mathtt{aa}, \mathtt{ba}, \mathtt{ba}) \in (A^{\cdot 2})^3$.

Falls die Komponenten eines Kartesischen Produkts eine besondere Bedeutung erlangen, werden auch die Bezeichnungen $(A^n)^m$ und $(A^m)^n$ unterschieden. Dies gilt in manchen Fällen auch bei heterogenen Konstruktionen, wobei etwa $A^n \times B$ von $A \times \cdots \times A \times B$ zu unterscheiden ist.

Definition 2.2.

a) Es sei A ein Alphabet. Die d-fache *Konkatenation* von A wird rekursiv definiert durch

$$A^{\cdot 1} := A, \qquad A^{\cdot(d+1)} := \{ab \mid a \in A^{\cdot d} \ \wedge\ b \in A\}.$$

b) Das neutrale Element der Konkatenation wird mit ε bezeichnet und *leeres Wort* genannt. Es gilt: $A^{\cdot 0} := \{\varepsilon\}$.

c) Die *Menge der Wörter* über A wird folgendermaßen festgelegt:

$$A^* := \bigcup_{d \in \mathbb{N}_0} A^{\cdot d}.$$

d) A^+ bezeichnet $A^* \setminus \{\varepsilon\}$.

Es sei w ein Wort über einem Alphabet A. Durch obige Definition wird dessen Länge $|w|$ implizit festgelegt. Für das leere Wort ε gilt $|\varepsilon| := 0$. Die Schreibweise $\mathtt{a}^{\cdot n}$ wird der Einfachheit halber abkürzend für $\{\mathtt{a}\}^{\cdot n}$ benutzt. Gelegentlich ist es bei dieser Vorgehensweise notwendig, Klammern zu benutzen. Es ist z.B. $(\mathtt{ab})^{\cdot n} = \overbrace{\mathtt{abab} \cdots \mathtt{ab}}^{n\text{-mal}}$ von $\mathtt{ab}^{\cdot n} = \mathtt{a}\overbrace{\mathtt{bb} \cdots \mathtt{b}}^{n\text{-mal}}$ zu unterscheiden.

Definition 2.3. Es sei A ein Alphabet. $L \subseteq A^*$ heißt *formale Sprache* über A.

Beispiel 2.1. $A := \{\mathtt{a}, \mathtt{b}\}$.

a) $\{\mathtt{a}^{\cdot n}\mathtt{b}^{\cdot n} | \, n \in \mathbb{N}_0\} \subseteq A^*$
b) $A^* \subseteq A^*$
c) $\{\mathtt{aaa}, \mathtt{abbb}, \varepsilon\} \subseteq A^*$

Später werden Familien formaler Sprachen daraufhin untersucht, ob sie bezüglich diverser Operationen abgeschlossen sind. Der Begriff der Abgeschlossenheit wird in der folgenden Definition formalisiert.

Definition 2.4.
a) Eine n-stellige Operation κ, $n > 0$, die aus n formalen Sprachen eine Sprachfamilie erzeugt, heißt *Sprachoperation*.
b) Eine Familie formaler Sprachen $\mathscr{L}$ heißt *abgeschlossen bzgl. κ*, falls gilt:

$$\forall L_1, \ldots, L_n \in \mathscr{L} : \kappa(L_1, \ldots, L_n) \subseteq \mathscr{L}.$$

Da häufig Sprachoperationen von Interesse sein werden, die als Ergebnis einelementige Familien liefern, werden wir in diesen Fällen das Ergebnis auch als Sprache auffassen und entsprechend die Sprechweise $\kappa(L_1, \ldots, L_n) = L \in \mathscr{L}$ benutzen.

2.2 Erzeugendensysteme

Beispiel 2.2. Eine formale Sprache L über $\{\mathtt{a}, \mathtt{b}\}$ wird folgendermaßen definiert:

a) $\varepsilon \in L$.
b) Falls $X \in L$ ist, dann ist auch $\mathtt{a}X\mathtt{b} \in L$.
c) Keine anderen Worte gehören zu L.

$L = \{\mathtt{a}^{\cdot n}\mathtt{b}^{\cdot n} | \; n \in \mathbb{N}_0\} \subseteq A^*$

Definition 2.5. Ein *Erzeugendensystem* ist ein Paar $RW = (V, F)$, wobei V ein Alphabet und F eine endliche Menge von Paaren von Wörtern über V ist. Die Elemente (P, Q) von F werden *Produktionen* genannt und mit $P \to Q$ bezeichnet.

Ein Wort P über V erzeugt ein Wort Q über V direkt, genau dann, wenn Wörter P', P_1, P'', Q_1 existieren, so daß $P = P'P_1P''$ und $Q = P'Q_1P''$ gilt und $P_1 \to Q_1$ eine Produktion ist. Diese Relation wird mit $\Rightarrow$ bezeichnet, ihre reflexive, transitive Hülle mit $\Rightarrow^*$.

Definition 2.6. Eine *Grammatik* ist ein 4-Tupel $G = (V_N, V_T, X_0, F)$, wobei gilt:

a) V_N und V_T sind disjunkte Alphabete, deren Elemente *nichtterminale* bzw. *terminale Symbole* genannt werden.
b) $X_0 \in V_N$ ist das *Axiom* der Grammatik.
c) F ist eine endliche Menge von Produktionen $P \to Q$, derart daß $P \in (V_N \cup V_T)^+$ mindestens ein nichtterminales Symbol enthält und $Q \in (V_N \cup V_T)^*$ ist.

Eine Grammatik induziert ein Erzeugendensystem $(V_N \cup V_T, F)$. Die von der Grammatik G erzeugte Sprache ist

$$L(G) = \{w | \; w \in V_T^* \;\wedge\; X_0 \Rightarrow^* w\}.$$

Beispiel 2.3.

a) $(\{X\}, \{\mathtt{a}, \mathtt{b}\}, X, \{X \to \varepsilon, X \to \mathtt{a}X\mathtt{b}\})$ erzeugt $\{\mathtt{a}^{\cdot n}\mathtt{b}^{\cdot n} | \; n \in \mathbb{N}_0\}$.
b) $(\{X_0, X, Y, Z\}, \{\mathtt{a}\}, X_0, \{X_0 \to YXY, YX \to YZ, ZX \to XXZ,$ $ZY \to XXY, X \to \mathtt{a}, Y \to \varepsilon\})$ erzeugt $\{\mathtt{a}^{\cdot(2^n)} | \; n \in \mathbb{N}_0\}$.

Definition 2.7. (Chomsky-Hierarchie)
Eine Grammatik und die von ihr erzeugte Sprache ist vom *Typ* i, falls sie die Einschränkung (i) erfüllt.
(0) Keine Einschränkung.
(1) Jede Produktion in F ist von der Form $Q_1 X Q_2 \to Q_1 P Q_2$, wobei $Q_1, Q_2 \in (V_N \cup V_T)^*, X \in V_N$ und $P \in (V_N \cup V_T)^+)$ gilt.
(2) Jede Produktion in F ist von der Form $X \to P$, wobei $X \in V_N$ und $P \in (V_N \cup V_T)^*$ gilt.
(3) Jede Produktion in F ist von einer der beiden Formen $X \to YP$ oder $X \to P$, wobei $X, Y \in V_N$ und $P \in V_T^*$ gilt.

Die Familie der Sprachen vom Typ i wird mit $\mathscr{L}_i$ bezeichnet. $\mathscr{L}_0$, $\mathscr{L}_1$, $\mathscr{L}_2$ bzw. $\mathscr{L}_3$ heißen auch *rekursiv aufzählbare, kontextsensitive, kontextfreie* bzw. *reguläre* Sprachen.

Satz 2.1. $\mathscr{L}_3 \subset \mathscr{L}_2 \subset \mathscr{L}_1 \subset \mathscr{L}_0$

2.3 Klassische Automatenhierarchie

2.3.1 Endliche Automaten

Definition 2.8. Ein *deterministischer endlicher Automat* (EA) ist ein 5-Tupel $(S, \Sigma, \delta, s_0, F)$, wobei gilt:
a) S ist eine endliche, nichtleere Menge von Zuständen.
b) Σ ist eine endliche, nichtleere Menge von Eingabesymbolen.
c) $s_0 \in S$ ist der Startzustand.
d) $F \subseteq S$ ist die Menge der Endzustände.
e) $\delta : S \times \Sigma \longrightarrow S$ ist die Überführungsfunktion.

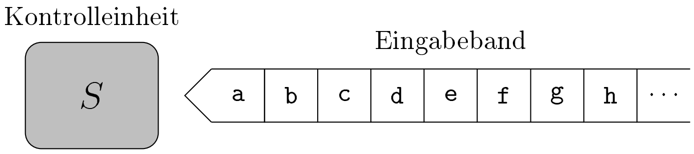


Abbildung 2.1. Endlicher Automat.

Um das Verhalten eines EA bei gegebener Eingabe zu beschreiben, muß die Überführungsfunktion erweitert werden.
$\forall s \in S, a \in \Sigma$ und $w \in \Sigma^*$:

$$\hat{\delta} : S \times \Sigma^* \longrightarrow S$$
$$\hat{\delta}(s, \varepsilon) = s \text{ und } \hat{\delta}(s, wa) = \delta(\hat{\delta}(s, w), a)$$

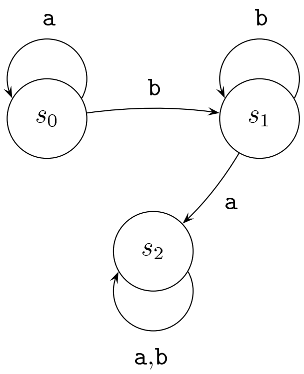
Definition 2.9. Es sei E ein EA. Die von E *akzeptierte Sprache* ist

$$L(E) := \{w \in \Sigma^* |\, \hat{\delta}(s_0, w) \in F\}.$$

Die Familie der von EAs akzeptierbaren Sprachen wird mit $\mathscr{L}(EA)$ bezeichnet.

Beispiel 2.4. $(\{s_0, s_1, s_2\}, \{\mathtt{a}, \mathtt{b}\}, \delta, s_0, \{s_0, s_1\})$ mit

δ	a	b
s_0	s_0	s_1
s_1	s_2	s_1
s_2	s_2	s_2



akzeptiert $\{\mathtt{a}^{\cdot n}\mathtt{b}^{\cdot m} |\, n, m \in \mathbb{N}_0\}$.

Satz 2.2. $\mathscr{L}(EA) = \mathscr{L}_3$

Ein *nichtdeterministischer endlicher Automat* (NEA) ist ein EA, dessen Überführungsfunktion folgendermaßen modifiziert wird:

$$\delta : S \times \Sigma \longrightarrow \mathscr{P}(S).$$

Entsprechend sei $\forall s \in S, a \in \Sigma$ und $w \in \Sigma^*$:

$\hat{\delta} : S \times \Sigma^* \longrightarrow \mathscr{P}(S)$
$\hat{\delta}(s, \varepsilon) = s$ und $\hat{\delta}(s, wa) = \{p |$ für ein $r \in \hat{\delta}(s, w)$ ist $p \in \delta(r, a)\}$.

Die von einem NEA N akzeptierte Sprache ist

$$L(N) := \{w \in \Sigma^* | \, \hat{\delta}(s_0, w) \cap F \neq \emptyset\}.$$

Die Familie der von NEAs akzeptierbaren Sprachen wird mit $\mathscr{L}(NEA)$ bezeichnet.

Satz 2.3. $\mathscr{L}(NEA) = \mathscr{L}(EA) = \mathscr{L}_3$

2.3.2 Kellerautomaten

Definition 2.10. Ein *(deterministischer) Kellerautomat* (PDA) ist gegeben durch das 7-Tupel $(S, \Sigma, \Gamma, \delta, s_0, g_0, F)$, wobei gilt:

a) S ist eine endliche, nichtleere *Menge von Zuständen*.
b) Σ ist eine endliche, nichtleere *Menge von Eingabesymbolen*.
c) Γ ist eine endliche, nichtleere *Menge von Kellersymbolen*.
d) $s_0 \in S$ ist der *Startzustand*.
e) $g_0 \in \Gamma$ ist das *Kellerendesymbol*.
f) $F \subseteq S$ ist die *Menge der Endzustände*.
g) $\delta : S \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \longrightarrow S \times \Gamma^*$ ist die (partielle) *Überführungsfunktion*, wobei für $q, q' \in S$ und $g \in \Gamma$ gilt:
   1) Falls $\delta(q, \varepsilon, g)$ definiert ist, dann ist $\delta(q, a, g)$ undefiniert für alle $a \in \Sigma$.
   2) Aus $\delta(q, a, g) = (q', \gamma)$ für $a \in \Sigma$ folgt $(\gamma \in (\Gamma \setminus \{g_0\})^* \wedge g \neq g_0) \vee (\gamma = \gamma' g_0$ mit $\gamma' \in (\Gamma \setminus \{g_0\})^* \wedge g = g_0)$.

Die Berechnung eines Kellerautomaten ist beendet, wenn eine Situation eintritt, für die die Zustandsüberführungsfunktion nicht definiert ist. Man sagt dann, der Kellerautomat hält.

Von Kellerautomaten wird verlangt, daß das Kellerendesymbol weder aus dem Keller entfernt noch mehrfach in ihn hineingeschrieben werden kann. Zur Erhaltung des Determinismus ist die Forderung notwendig, ε-Übergänge nicht alternativ zu anderen Zustandsübergängen zuzulassen.
Die beiden folgenden Definitionen klären, wie Kellerautomaten formale Sprachen verarbeiten können.

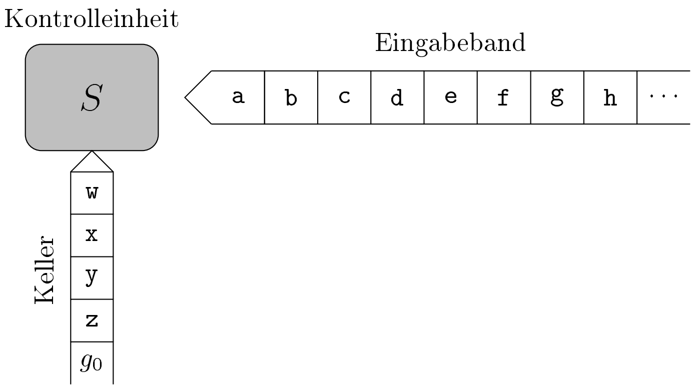


Abbildung 2.2. Kellerautomat.

Definition 2.11.

a) Die *Konfiguration* eines Kellerautomaten in einem Zeitpunkt t ist gegeben durch ein Tripel (q, w, γ), wobei $q \in S$ ein Zustand, $w \in \Sigma^*$ das noch zu lesende Eingabewort und $\gamma \in \Gamma^*$ der Kellerinhalt ist.

b) Der Übergang von einer Konfiguration zur Folgekonfiguration wird mit $\vdash$ bezeichnet.
$$(q, aw, g\gamma) \vdash (q', w, \beta\gamma) \iff \delta(q, a, g) = (q', \beta)$$

c) $\overset{*}{\vdash}$ bezeichnet die reflexive, transitive Hülle und $\overset{i}{\vdash}$ die i-malige Komposition von $\vdash$.

Definition 2.12. Es sei K ein Kellerautomat. Die von K akzeptierte Sprache ist $L(K) := \{w \in \Sigma^* \mid \exists i \in \mathbb{N} : (s_0, w, g_0) \stackrel{i}{\longmapsto} (s, \varepsilon, g\gamma)$ mit $s \in F, \gamma \in \Gamma^*, g \in \Gamma \wedge \delta(s, \varepsilon, g)$ ist undefiniert$\}$.

Die Familie der von PDAs akzeptierbaren Sprachen wird mit $\mathscr{L}(PDA)$ bezeichnet.

Beispiel 2.5. $L := \{\mathtt{a}^{\cdot n}\mathtt{b}^{\cdot n} \mid n \in \mathbb{N}_0\}$. Es gilt $L \notin \mathscr{L}(NEA) = \mathscr{L}_3$, aber $L \in \mathscr{L}(PDA)$.

$(\{s_0, s_1, s_2\}, \{\mathtt{a}, \mathtt{b}\}, \{g_0, \mathtt{a}, \mathtt{b}\}, \delta, s_0, g_0, \{s_2\})$ mit

δ:

s_0	a	b	ε
a	$(s_0, \mathtt{aa})$	(s_1, ε)	
b			
g_0	$(s_0, g_0\mathtt{a})$		

s_1	a	b	ε
a		(s_1, ε)	
b			
g_0		(s_2, g_0)	

Die Verallgemeinerung auf *nichtdeterministische Kellerautomaten* (NPDA) erfolgt wiederum über die Überführungsfunktion.

$$\delta : S \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \longrightarrow \text{endliche Teilmengen von } S \times \Sigma^*$$

Die Familie der von NPDAs akzeptierbaren Sprachen wird mit $\mathscr{L}(NPDA)$ bezeichnet.

Satz 2.4. $\mathscr{L}(PDA) \subset \mathscr{L}(NPDA)$

Beispiel 2.6. $L := \{w \mid w \in \{\mathtt{a}, \mathtt{b}\}^+ \wedge w = w^R\}$ ist die Sprache der Palindrome. Es gilt $L \in \mathscr{L}(NPDA)$, aber $L \notin \mathscr{L}(PDA)$. Weiterhin gilt: $L \notin \mathscr{L}(EA)$.

Satz 2.5. $\mathscr{L}(NPDA) = \mathscr{L}_2$

2.3.3 Linear beschränkte Automaten

Definition 2.13. Ein deterministischer *linear beschränkter Automat* (LBA) ist ein 8-Tupel $(S, \Sigma, \Pi, \delta, s_0, \texttt{\#}, \texttt{@}, F)$, wobei gilt:

a) S ist eine endliche, nichtleere Menge von Zuständen.
b) Σ ist eine endliche, nichtleere Menge von Eingabesymbolen.
c) Π ist eine endliche, nichtleere Menge von Bandsymbolen mit der Eigenschaft $\Pi \supseteq \Sigma$.
d) $s_0 \in S$ ist der Startzustand.
e) $\texttt{\#}$ und $\texttt{@}$ sind zwei verschiedene Eingabebegrenzungssymbole, für die gilt: $\texttt{\#} \notin \Pi$ und $\texttt{@} \notin \Pi$.
f) $F \subseteq S$ ist die Menge der Endzustände.
g) $\delta : S \times (\Pi \cup \{\texttt{\#}, \texttt{@}\}) \longrightarrow S \times \Pi \times \{0, 1, -1\}$ ist die Überführungsfunktion mit der Eigenschaft $\delta(s, \texttt{\#}) = (s', \texttt{\#}, 1)$ und $\delta(r, \texttt{@}) = (r', \texttt{@}, -1)$ für $r, r', s, s' \in S$.

Definition 2.14.

a) Die Konfiguration eines LBA in einem Zeitpunkt t ist ein Tripel (q, v, p), wobei $q \in S$ ein Zustand, $v \in \texttt{\#}\Pi^+\texttt{@}$ eine *Bandinschrift* und $p \in \{0, 1, \ldots, |v| - 1\}$ die *Nummer des aktuellen Feldes* ist.
b) Der Übergang von einer Konfiguration zur Folgekonfiguration wird wieder mit $\longmapsto$ bezeichnet.

$$(q, v_0 \cdots v_{n-1}, p) \longmapsto (q', v'_0 \cdots v'_{n-1}, p') \iff$$

$$\begin{aligned} q' &= \pi_1(\delta(q, v_p)) \\ v'_i &= \begin{cases} v_i & \text{falls } i \neq p \\ \pi_2(\delta(q, v_p)) & \text{falls } i = p \end{cases} \\ p' &= p + \pi_3(\delta(q, v_p)) \end{aligned}$$

c) $\overset{*}{\longmapsto}$ bezeichnet wieder die reflexive, transitive Hülle und $\overset{i}{\longmapsto}$ die i-malige Komposition von $\longmapsto$.

Definition 2.15. Es sei B ein linear beschränkter Automat. Die von B akzeptierte Sprache ist

$$
\begin{aligned}
L(B) := \{w \in \Sigma^* \mid \exists i \in \mathbb{N} :\ & (s_0, \texttt{\#}w\texttt{@}, 0) \overset{i}{\vdash} (s, \texttt{\#}w'\texttt{@}, p) \\
& \text{mit } w' \in \Pi^*, p \in \{0, \ldots, |w| + 1\}, s \in F \\
& \text{und } \delta(s, w'(p)) \text{ ist undefiniert}\}.
\end{aligned}
$$

Die Bezeichnung „linear beschränkt“ ist darauf zurückzuführen, daß bei einer Aufteilung der Felder des Bandes in mehrere Register der zur verfügbare Platz linear von der Länge der Eingabe abhängt.

Die Familie der von LBAs akzeptierbaren Sprachen wird mit $\mathscr{L}(LBA)$ bezeichnet.

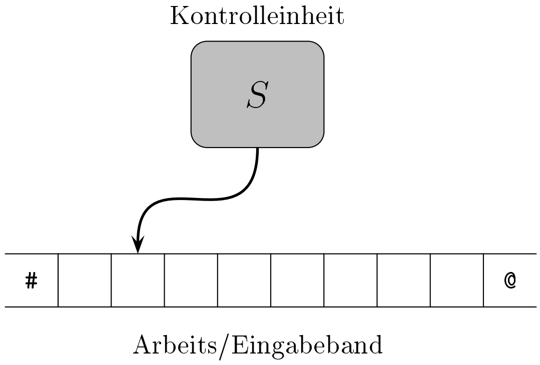


Abbildung 2.3. Linear beschränkter Automat.

Die Verallgemeinerung auf *nichtdeterministische linear beschränkte Automaten* (NLBA) erfolgt wiederum über die Überführungsfunktion und soll hier nicht durchgeführt werden. Der bisherigen Notation folgend, bezeichnet $\mathscr{L}(NLBA)$ die Familie der von NLBAs akzeptierbaren Sprachen.

Lemma 2.1. $\mathscr{L}(LBA) \subseteq \mathscr{L}(NLBA)$

Die Frage nach der Echtheit der Inklusion wird als LBA-Problem bezeichnet und ist wohl das berühmteste Problem der Automatentheorie.

Satz 2.6. $\mathscr{L}(NLBA) = \mathscr{L}_1$

Beispiel 2.7. $L := \{\mathtt{a}^{\cdot n}\mathtt{b}^{\cdot n}\mathtt{a}^{\cdot n} \mid n \in \mathbb{N}_0\}$. Es gilt $L \in \mathscr{L}(LBA)$, aber $L \notin \mathscr{L}(NPDA)$.

2.3.4 Turingmaschinen

Definition 2.16. Eine *deterministische Einband-Turingmaschine* ist ein 7-Tupel $(S, \Sigma, \Pi, \delta, s_0, \mathtt{b}, F)$, wobei gilt:
a) S ist eine endliche, nichtleere Menge von Zuständen.
b) Σ ist eine endliche, nichtleere Menge von Eingabesymbolen.
c) Π ist eine endliche, nichtleere Menge von Bandsymbolen mit der Eigenschaft $\Pi \supseteq \Sigma$.
d) $\mathtt{b} \in \Pi$ ist das Leerzeichen.
e) $s_0 \in S$ ist der Startzustand.
f) $F \subseteq S$ ist die Menge der Endzustände.
g) $\delta : S \times \Pi \longrightarrow S \times \Pi \times \{0, 1, -1\}$ ist die Überführungsfunktion.

Definition 2.17.
a) Die Konfiguration einer Turingmaschine in einem Zeitpunkt t ist ein Tripel (q, f_t, p), wobei $q \in S$ ein Zustand, $f_t : \mathbb{Z} \longrightarrow \Pi$ eine Abbildung, die jedem Feld des Bandes dessen Inhalt zuordnet, und p die Nummer des aktuellen Feldes ist.
b) Der Übergang von einer Konfiguration zur Folgekonfiguration wird mit $\longmapsto$ bezeichnet.

$$(q, f, p) \longmapsto (q', f', p') \iff$$

$$q' = \pi_1(\delta(q, f(p)))$$
$$f'(i) = \begin{cases} f(i) & \text{falls } i \neq p \\ \pi_2(\delta(q, f(p))) & \text{falls } i = p \end{cases}$$
$$p' = p + \pi_3(\delta(q, f(p)))$$

c) $\overset{*}{\vdash}$ bezeichnet die reflexive, transitive Hülle und $\overset{i}{\vdash}$ die i-malige Komposition von $\vdash$.

Definition 2.18. Es sei T eine Turingmaschine. Die von T akzeptierte Sprache ist

$$
\begin{aligned}
L(T) := \{w_1 w_2 \cdots w_n \in \Sigma^* \mid \exists i \in \mathbb{N} :\ & (s_0, f_0, 0) \overset{i}{\vdash} (s, f_i, p) \text{ mit} \\
& f_0(j) = w_j \text{ falls } j \in \{1, 2, \ldots, n\}, \mathtt{b} \text{ sonst.} \\
& s \in F, \text{ und } \ \delta(s, f_i(p)) \text{ ist undefiniert.}\}.
\end{aligned}
$$

Falls die Überführungsfunktion für eine Konfiguration nicht definiert ist, wird die Sprechweise „die Turingmaschine hält“ benutzt werden.

Die Familie der von TM akzeptierbaren Sprachen wird mit $\mathscr{L}(TM)$ bezeichnet.

Satz 2.7. $\mathscr{L}(TM) = \mathscr{L}_0$

These von Church/Turing. Jede intuitiv berechenbare Funktion ist mit einer Turingmaschine berechenbar.

Demnach würde eine Verallgemeinerung auf nichtdeterministische Turingmaschinen in diesem Sinne keine Steigerung der Erkennungsmächtigkeit mit sich bringen.

Zusammenfassung

$$
\begin{array}{ccccccccccccc}
\mathscr{L}_3 & & & \subset & & \mathscr{L}_2 & & \subset & & \mathscr{L}_1 & & \subset & \mathscr{L}_0 \\
= & & & & & = & & & & = & & & = \\
\mathscr{L}(EA) & = & \mathscr{L}(NEA) & \subset & \mathscr{L}(PDA) & \subset & \mathscr{L}(NPDA) & \subset & \mathscr{L}(LBA) & \subseteq & \mathscr{L}(NLBA) & \subset & \mathscr{L}(TM)
\end{array}
$$

Es stellt sich nun die Frage nach der Existenz von Sprachen, die nicht von Turingmaschinen erkannt werden können, also außerhalb der Familie $\mathscr{L}_0$ liegen.
Das bekannteste nichtberechenbare Problem ist wohl das Halteproblem, mit dessen Hilfe sich derartige formale Sprachen konstruieren lassen.

Zum Abschluß dieses Abschnittes wird für eine Sprache, der das nichtberechenbare Busy-Beaver-Problem zugrundeliegt, gezeigt, daß sie außerhalb der Familie $\mathscr{L}_0$ liegt, also von keiner Turingmaschine akzeptiert werden kann.

Definition 2.19. $BB : \mathbb{N} \longrightarrow \mathbb{N}$
$BB(n) := q$, wobei q die maximale Anzahl von $\mathtt{1}$en ist, die eine TM mit n Zuständen und dem Bandalphabet $\{\mathtt{1}, \mathtt{b}\}$ auf das leere Band schreiben kann, um anschließend zu halten.

Eine entsprechende formale Sprache ist dann z.B.

$$L_{BB} := \{\mathtt{a}^{\cdot n}\mathtt{b}^{\cdot BB(n)} \mid n \in \mathbb{N}\}.$$

Satz 2.8. $L_{BB} \notin \mathscr{L}_0$

Beweis. Offensichtlich gilt: $L_{BB} \in \mathscr{L}_0 \iff BB(n)$ ist berechenbar. Es genügt also zu zeigen, daß $BB(n)$ nicht berechenbar ist.

Angenommen, $BB(n)$ ist berechenbar, dann auch die Funktion $f(n) := BB(2n) + 1$. Dieses leiste eine TM T mit p Zuständen.
Aus T kann nun leicht eine TM T' mit $2p$ Zuständen konstruiert werden, die angesetzt auf das leere Band zunächst p Felder mit $\mathtt{1}$ beschriftet und anschließend T simuliert. Somit beschreibt T' mit $2p$ Zuständen angesetzt auf das leere Band $BB(2p) + 1$ Felder mit $\mathtt{1}$. Dies ist aber ein Widerspruch zur Maximalität von $BB(2p)$. $\square$

3 Zellularräume und -automaten

Auf der Suche nach Systemen, die zur Selbstreproduktion fähig sind, hat J. von Neumann (1966) das Modell des Zellularraums vorgeschlagen. Zellularräume und deren „endliche" Variante, die Zellularautomaten, sind in der Zwischenzeit hinsichtlich vielerlei Fragestellungen untersucht worden. Andererseits sind aber auch viele „recht einfach formulierbare" Probleme bis heute offen.

Die Terminologie der folgenden Einführung lehnt sich an diejenige von A. R. Smith III (1976) an. Die anschließenden Abschnitte über Standardisierung, Berechnungsuniversalität und das Firing Squad Synchronization Problem folgen entsprechenden Abschnitten in R. Vollmar (1979).

3.1 Einführung

Definition 3.1.

a) Es sei $d \in \mathbb{N}_0$ und $n \in \mathbb{N}$. Ein n-Tupel N von paarweise verschiedenen Elementen aus $\mathbb{Z}^d$ heißt *d-dimensionaler Nachbarschaftsindex* vom Grade n.
b) Die Menge der Komponenten von N heißt *Raster* und wird mit $\tilde{N}$ bezeichnet.
c) Zwei Elemente $i, j \in \mathbb{Z}^d$ heißen *benachbart* (bzgl. N), wenn es eine Komponente k von N gibt, so daß entweder $j = i + k$ oder $i = j + k$ gilt. Diese Relation wird mit $\vdash\!\dashv_N$ bezeichnet, ihre reflexive, transitive Hülle mit $\vdash\!\dashv_N^*$.

Definition 3.2. Sind $i \in \mathbb{Z}^d$ und $k \in \mathbb{N}_0$, dann werden mit Hilfe der Normen

$$|i| := \sum_{n=1}^{d} |i_n| \qquad \text{und} \qquad \|i\| := \max\{|i_n| \mid 1 \leq n \leq d\}$$

spezielle Raster folgendermaßen definiert:

$$
\begin{aligned}
H_k^d &:= \{i \mid k \geq |i|\} \\
\bar{H}_k^d &:= \{i \mid k \geq |i| \ \wedge \ \forall 1 \leq l \leq d : \ i_l \leq 0\} \\
M_k^d &:= \{i \mid k \geq \|i\|\} \\
\bar{M}_k^d &:= \{i \mid k \geq \|i\| \ \wedge \ \forall 1 \leq l \leq d : \ i_l \leq 0\}
\end{aligned}
$$

Die H_k-Raster werden auch *verallgemeinerte von-Neumann-Raster* und die M_k-Raster *verallgemeinerte Moore-Raster* genannt.

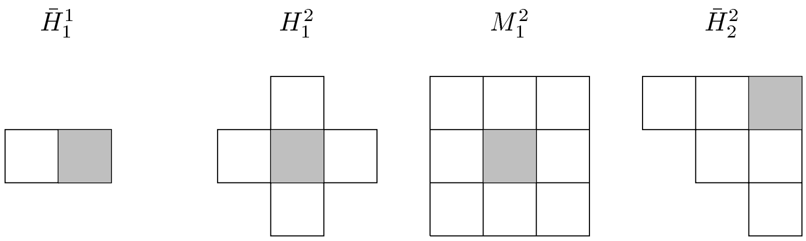


Abbildung 3.1. Schematische Rasterdarstellung mit schraffiertem Ursprung.

Falls die Reihenfolge der Elemente unwesentlich ist, können Nachbarschaftsindex und Raster synonym verwendet werden, wovon in der weiteren Bezeichnungsweise häufig Gebrauch gemacht wird. Benachbarte Automaten werden auch *Nachbarn* genannt.

Die Elemente des $\mathbb{Z}^d$ entsprechen den Gitterpunkten des d-dimensionalen Euklidischen Raumes, in denen anschaulich die deterministischen endlichen Automaten befestigt sind. Entsprechend werden sie auch als *Zellen* bezeichnet.

Definition 3.3. Ein 5-Tupel (S, d, N, σ, q_0) heißt *Zellularraum*, falls gilt:

a) S ist eine endliche, nichtleere *Zustandsmenge*.
b) $d \in \mathbb{N}_0$ ist die *Raumdimension*.
c) N ist d-dimensionaler Nachbarschaftsindex vom Grade n.
d) $\sigma : S^n \longrightarrow S$ ist *lokale Überführungsfunktion*.
e) $q_0 \in S$ ist der *Ruhezustand*, für den gilt: $\sigma(q_0, \ldots, q_0) = q_0$.

Definition 3.4. Eine Abbildung $c_t : \mathbb{Z}^d \longrightarrow S$ heißt *Konfiguration* des Zellularraums im Zeitpunkt $t \in \mathbb{N}_0$. Für $t = 0$ werden auch die Begriffe *Anfangs-* oder *Startkonfiguration* benutzt.

Eine Konfiguration beschreibt den Gesamtzustand des Raumes in einem bestimmten Zeitpunkt t. Sie ordnet jedem Element $i \in \mathbb{Z}^d$ ein Element aus der Zustandsmenge S zu. $c_t(i)$ heißt *Inhalt oder Zustand* des Automaten i in der Konfiguration c_t.

Der *Träger* einer Konfiguration umfaßt alle Zellen, die sich nicht im Ruhezustand befinden. Alle weiteren Betrachtungen werden ausschließlich von endlichen Trägern ausgehen, auch wenn dies nicht ausdrücklich erwähnt wird. In diesem Zusammenhang genügt es — wie weiter unten deutlicher wird —, vom Träger der Startkonfiguration Endlichkeit zu fordern, da sich Information nicht schlagartig im Raum ausbreiten kann.

Definition 3.5. Es seien A ein Alphabet, $N = (a_1, \ldots, a_n)$ ein d-dimensionaler Nachbarschaftsindex und C_d die Menge aller Konfigurationen $c : \mathbb{Z}^d \longrightarrow A$. Eine Abbildung $\mathcal{T} : C_d \longrightarrow C_d$ heißt *globale Überführungsfunktion* (oder *Transformation*), wenn sie durch eine lokale Überführungsfunktion σ folgendermaßen induziert wird:

$$\forall c \in C_d : \big(\mathcal{T}(c) = c' \iff \forall i \in \mathbb{Z}^d : c'(i) = \sigma(c(i+a_1), \ldots, c(i+a_n))\big)$$

Aus den Definitionen folgt, daß in einem Raum für eine Zustandsmenge S und einen Nachbarschaftsindex N eine bijektive Abbildung zwischen den Mengen der globalen und lokalen Überführungsfunktionen existiert. Dies wäre nicht der Fall, wenn man die Forderung nach paarweise verschiedenen Elementen des Nachbarschaftsindex fallen ließe.

Nach Anwendung von $\mathcal{T}$ legen die Inhalte der Zellen im allgemeinen eine neue Konfiguration fest. Dieselbe Zustandsbelegung ergibt sich durch die synchrone Anwendung der lokalen Überführungsfunktion auf alle Zellen des Raumes. Im folgenden werden demnach Konfigurationen zu diskreten Zeitpunkten betrachtet. Der Übergang von einem Zeitpunkt zum nächsten wird als *Zeitschritt* bezeichnet. Das Verhalten einer Zelle in einem Zeitschritt wird durch die Zustände der Nachbarn und die Überführungsfunktion bestimmt. Ein Automat kann dabei durchaus sein eigener Nachbar sein.

Im folgenden werden Zellularautomaten definiert. Sie arbeiten in einem festen, endlichen Teil des Raumes, der durch die sogenannte Retina festgelegt wird.

Definition 3.6. Es sei $M = (S, d, N, \sigma, q_0)$ ein Zellularraum. M heißt *Zellularautomat*, falls gilt:

a) $\exists \texttt{\#} \in S : \forall t \in \mathbb{N}_0, i \in \mathbb{Z}^d : c_{t+1}(i) = \texttt{\#} \iff c_t(i) = \texttt{\#}$, dabei heißt # *Grenzzustand*.
b) Ein endliches, nichtleeres Gebiet $R \subset \mathbb{Z}^d$, *Retina* genannt, hat folgende Eigenschaften:
   $\forall i \in R : c_0(i) \neq \texttt{\#}$
   $\forall k \in \{l \in \mathbb{Z}^d \setminus R | \, \exists i \in R : i \vdash\!\dashv_N l\} : c_0(k) = \texttt{\#}$
   $\forall k \notin \{l \in \mathbb{Z}^d \setminus R | \, \exists i \in R : i \vdash\!\dashv_N l\} \cup R : c_0(k) = q_0$
   $\forall i, j \in R : i \vdash\!\dashv^*_{H_1^d} j$

Die Retina wird von einer Schicht von Automaten umgeben, die sich im Grenzzustand befinden. Diejenigen Zellen der Retina, in deren Nachbarschaft eine Zelle im Grenzzustand liegt, bilden die Randschicht.

Es wird gefordert, daß sich alle Zellen außerhalb der Retina und der Grenzschicht im Ruhezustand befinden. Da die Grenzschicht zudem derart festgelegt wurde, daß die Zellen der Retina nicht von „außen“ beeinflußt werden können, können sich die Untersuchungen auf diese beschränken. Wie später noch verdeutlicht wird, läßt sich durch diese Vorgehensweise auch die gewünschte Eigenschaft erzielen, daß die Anzahl der benötigten aktiven Elemente — der Zellen — nur von der jeweiligen Eingabe abhängt. Die Forderung, alle Zellen seien bezüglich H_1 benachbart, stellt eine zusammenhängende Retina sicher.

In der obigen Definition wird für jeden Automaten die Existenz einer Retina gefordert. Da die Retina unmittelbar durch die Eingabe festgelegt werden wird, wird strenggenommen für jede Eingabe ein Automat benötigt. Alle Automaten, die sich nur in der Retina unterscheiden, lassen sich dann zu Klassen zusammenfassen. Unter Vernachlässigung dieser Differenzierung wird im weiteren aber die Sprechweise „Klasse von Zellularautomaten“ zu Zellularautomaten verkürzt. Eine diesbezüglich verallgemeinerte Definition — die der zellularen Algorithmen — wurde von B. Bleck und H. Kröger (1992) vorgeschlagen.

Ein im folgenden häufig benutztes Konzept zur Beschreibung von Algorithmen ist das der *Signale*:
Nimmt ein Automat nach einer bestimmten Zahl $k \in \mathbb{N}$ von Zeitschritten den Zustand s eines benachbarten Automaten an, verhält sich sein entsprechender Nachbar analog, usw., so läuft das Signal s mit der Geschwindigkeit $1/k$ in die entsprechende Richtung, bzw. das Signal wird mit der Geschwindigkeit $1/k$ ausgesandt. Offensichtlich kann bei Vorliegen eines H_1-Rasters die Signalgeschwindigkeit nicht größer als 1 werden, weshalb sie gelegentlich auch als Lichtgeschwindigkeit in Zellularräumen bezeichnet wird. (Statt Signal wird auch die Sprechweise *Welle* benutzt.)

Beispiel 3.1. (Game of life) Es sei $M = (S, d, N, \sigma, q_0)$ ein Zellularraum mit $S = \{\mathtt{0}, \mathtt{1}\}$, $d = 2$ und $\tilde{N} = M_1$. Die lokale Überführungsfunktion wird folgendermaßen festgelegt:

$$c_{t+1}(i) := \begin{cases} \mathtt{1} & \text{falls } \sum_{j \in M_1} c_t(i+j) = 3 \\ c_t(i) & \text{falls } \sum_{j \in M_1} c_t(i+j) = 4 \\ \mathtt{0} & \text{sonst} \end{cases}$$

In den folgenden Abbildungen sind Zellen im Zustand `1` grau hinterlegt, im Zustand `0` weiß dargestellt.

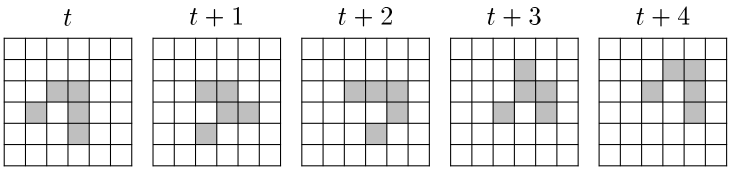


Abbildung 3.2. Ein Gleiter im Game of Life.

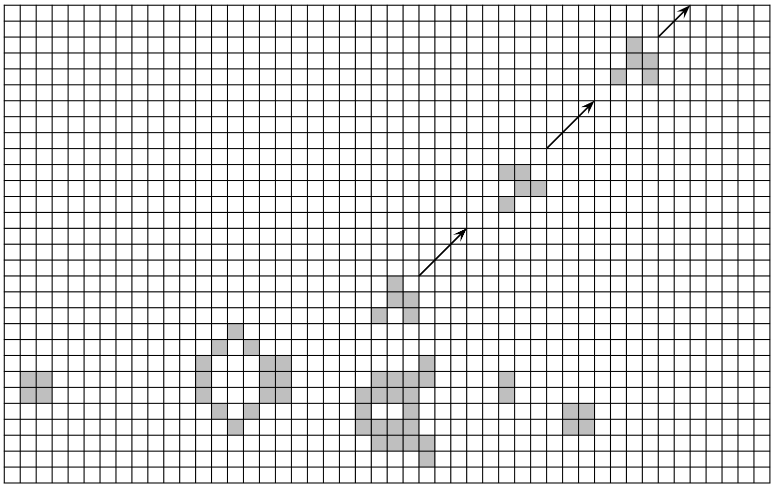


Abbildung 3.3. Eine Gleiterkanone im Game of Life.

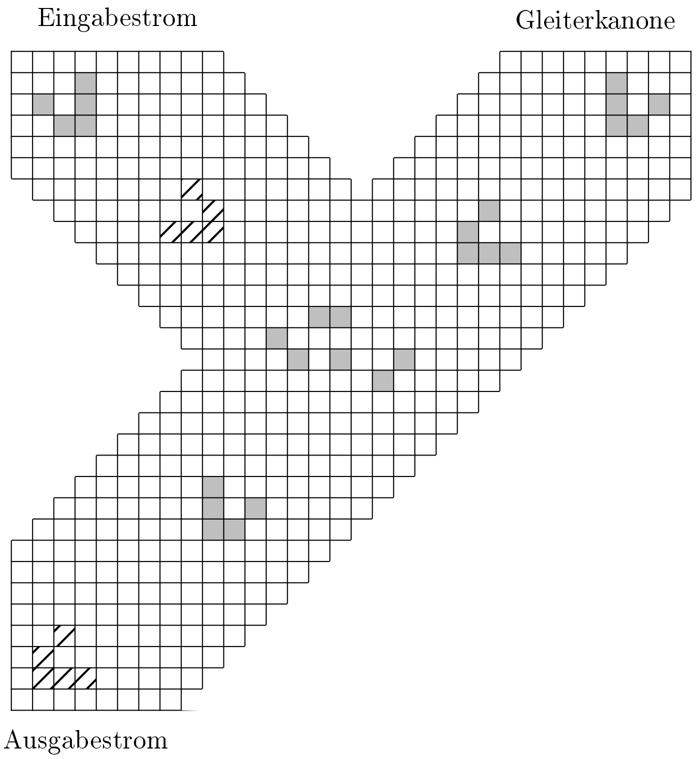


Abbildung 3.4. Ein NOT-Gatter im Game of Life.

3.2 Standardisierung von Zellularräumen

In diesem Abschnitt wird untersucht, wie Zellularräume auf gewisse Standardformen gebracht werden können. Dies umfaßt Möglichkeiten zur Reduzierung des Rasters, der Zeit bzw. der Zustandsmenge.

Da man verlangen wird, daß ein derart „reduzierter“ Zellularraum in gewisser Weise dasselbe leistet wie der ursprüngliche Automat, wird dieser Umstand durch den Simulationsbegriff präzisiert.

Definition 3.7. Es seien $M = (S, d, N, \sigma, q_0)$ und $\tilde{M} = (\tilde{S}, d, \tilde{N}, \tilde{\sigma}, \tilde{q_0})$ zwei Zellularräume. Deren Mengen von Konfigurationen seien mit C_d und $\tilde{C}_d$, die jeweiligen globalen Transformationen mit $\mathcal{T}$ und $\tilde{\mathcal{T}}$ bezeichnet. Es sei Φ_d die Menge aller globaler Transformationen für d-dimensionale Zellularräume. Für $k, \tilde{k} \in \mathbb{N}$ gilt: *$\tilde{M}$ simuliert M* in $\tilde{k}/k$-facher Zeit, genau dann, wenn es eine injektive, berechenbare Abbildung $G : C_d \longrightarrow \tilde{C}_d$ und eine berechenbare Abbildung $g : \Phi_d \longrightarrow \Phi_d$ gibt, so daß für alle $c \in C_d$ gilt:

$$\tilde{\mathcal{T}}^{\tilde{k}}(G(c)) = G(\mathcal{T}^k(c)),$$

wobei $\tilde{\mathcal{T}} = g(\mathcal{T})$ ist.

Im folgenden wird häufig das Konzept benutzt, Information aus mehreren Einzelautomaten zusammenzufassen. Diese Vorgehensweise läßt sich mit der Vorstellung verbinden, die einzelnen Automaten verfügen über eine endliche Zahl endlicher Register, deren Inhalte jeweils aus einer endlichen, nichtleeren Menge gewählt werden können. Die Zustandsmenge der Automaten entspricht dann dem Kartesischen Produkt der „Registermengen“.

Eine weitere Generalisierung dieser Technik sei kurz dargestellt:
Ist eine Zustandsmenge S konstruiert als das Kartesische Produkt $A \times B \times C$, dann kann ein gewisser Formalismus für die Fälle, in denen Registerinhalte undefiniert sind, vermieden werden, indem S als $A \cup B \cup C \cup A \times B \cup A \times C \cup B \times C \cup A \times B \times C$ angegeben wird. Es läßt sich damit auch die Vorstellung verbinden, jede Zelle besteht nur aus den gerade benötigten Registern, hat also keine statische Struktur.

Aufgrund der besseren Darstellbarkeit werden in den folgenden Konstruktionen ausschließlich Zellen mit statischer Registerzahl benutzt, obwohl sich die Sprechweise gelegentlich auf die erweiterten Möglichkeiten bezieht.

3.2.1 Rasterreduktion

Weil die Komplexität der Überführungsfunktion mit der Rastergröße wächst, und weil die Analyse und Entwicklung der Algorithmen dadurch wesentlich überschaubarer wird, erscheint es wünschenswert, sich auf relativ kleine Raster beschränken zu können.

Das folgende Verfahren stammt von A. R. Smith III (1971a). Für den Beweis ist der Begriff des minimalen Quaders um ein Raster nützlich. (vgl. R. Vollmar (1979))

Definition 3.8. Es sei $\tilde{N} = \{n_1, \ldots, n_q\}$ ein d-dimensionales Raster. Für $1 \leq i \leq d$ werden $u(\tilde{N}) = (u_1, \ldots, u_d)$ und $o(\tilde{N}) = (o_1, \ldots, o_d)$ definiert durch

$$\begin{aligned} u_i &:= |\min\{0, \min\{n_i |\, n \in \tilde{N}\}\}| \\ o_i &:= \max\{0, \max\{n_i |\, n \in \tilde{N}\}\}. \end{aligned}$$

Die *Gesamtausdehnung* $g(\tilde{N})$ von $\tilde{N}$, wird definiert durch

$$g(\tilde{N}) := u(\tilde{N}) + o(\tilde{N}).$$

Damit ergibt sich der *minimale Quader* für $\tilde{N}$ als

$$P(\tilde{N}) := \{(p_1, \ldots, p_d) |\; -u_i \leq p_i \leq o_i \text{ für } 1 \leq i \leq d\}.$$

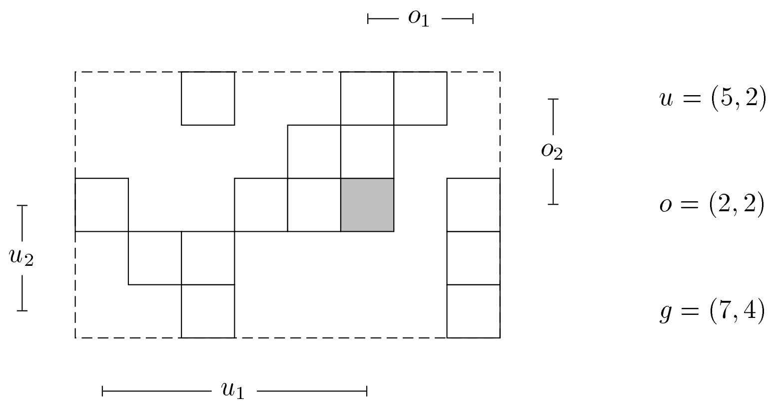


Abbildung 3.5. Ein Raster und der entsprechende minimale Quader.

Satz 3.1. Es sei $M = (S, 2, N, \sigma, q_0)$ ein Zellularraum. Dann existiert ein Zellularraum $M' = (S', 2, H_1^2, \sigma', q_0')$, der M in k-facher Zeit simuliert.

Beweis. Falls $\tilde{N} \subseteq H_1$ ist, kann $S = S'$ gewählt werden. σ' ergibt sich aus σ durch Permutieren der Argumente und/oder Weglassen zusätzlicher Komponenten.
In allen anderen Fällen besteht das Beweisprinzip darin, die zum Feststellen des richtigen Übergangsverhaltens notwendige Information in einer Reihe von Zwischenschritten innerhalb des H_1-Rasters zu Verfügung zu stellen.

Dazu wird in zwei Phasen sukzessive eine vertikale und horizontale Komprimierung der Information vorgenommen.
In der ersten Phase geschieht das Auffüllen in vertikaler Richtung so lange, bis die benötigte Information für jeden Automaten in einem Bereich der in Abbildung 3.6 dargestellten Form vorliegt.

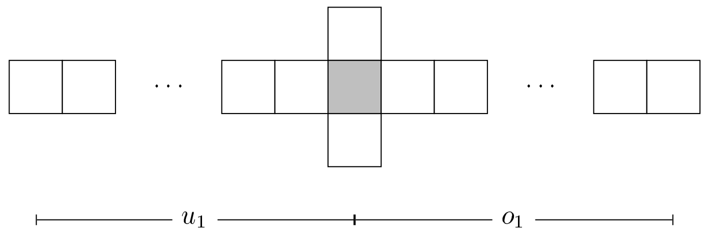


Abbildung 3.6. Informationskomprimierung am Ende der ersten Phase.

Jeder Automat muß daher über $1+o_2+u_2 = 1+g_2$ Register verfügen, um die Information aus den oberhalb und unterhalb gelegenen Automaten zusätzlich aufzunehmen. Ferner muß jeder Automat das Ende der vertikalen Kompressionsphase erkennen können. Da deren Dauer aber nur vom ursprünglichen Raster abhängt, kann jede Zelle über einen endlichen Zähler verfügen, der den jeweiligen Zeitpunkt anzeigt. Diese Phase ist nach $\max\{o_2, u_2\}$ Zeitschritten beendet.

In der zweiten Phase wird der analoge Prozeß zur Komprimierung der (vorkomprimierten) Information in horizontaler Richtung vorgenommen, und zwar so lange, bis die im minimalen Quader um jeden Automaten in M enthaltene Information in dem H_1-Raster des betreffenden Automaten vorliegt. Da jetzt nur auf eine Breite von drei Automaten komprimiert werden muß, sind in dieser Phase $1 + \max\{o_1-1, 0\}+\max\{u_1-1, 0\}$ Register erforderlich, von denen jedes wiederum die Information der obengenannten $1+g_2$ Register aufnehmen muß. Der Zähler kann entsprechend erweitert werden, so daß das Ende dieser Phase nach $\max\{\max\{o_1-1, u_1-1\}, 0\}$ Schritten erkannt werden kann. Anschließend wird die eigentliche Zustandsüberführung simuliert, da dann jedem Automaten in M' mindestens die gleiche Information zur Verfügung steht wie dem entsprechenden Automaten in M. Es sind also $\max\{o_2, u_2\} + \max\{\max\{o_1 - 1, u_1 - 1\}, 0\} + 1$ Schritte nötig, um einen Übergang von M in M' zu simulieren. □

Die obige Aussage läßt sich leicht auf beliebige Dimensionen über-

tragen.
Aus der Konstruktion folgt die Kardinalität der Zustandsmenge S':

$$|S'| = |S|(|S|+1)^{o_2+u_2}(|S|(|S|+1)^{o_2+u_2}+1)^{\max\{o_1-1,0\}+\max\{u_1-1,0\}}$$

3.2.2 Zeitreduktion

Bei den bisher zur Rasterreduktion eingeführten Verfahren wurde sowohl die Zustandsmenge vergrößert als auch die Simulation verzögert. In diesem Abschnitt wird die Möglichkeit einer beschleunigten Simulation nachgewiesen, wobei die Zustandsmenge i.a. wiederum vergrößert werden muß, das Raster jedoch nicht. Insgesamt ergibt sich damit die Möglichkeit, auf Kosten der Zustandsmenge gleichzeitig das Raster und die benötigte Zeit zu verringern.

Das Verfahren zur Beschleunigung setzt eine Abbildung vom simulierten auf den simulierenden Zellularraum voraus.

Der folgende Satz von S. N. Cole (1966) wurde von A. R. Smith III (1971a) auf Zellularräume übertragen. Er liefert eine hinreichende Bedingung für die Möglichkeit einer Beschleunigung. Die Idee ist, das Raster N' des simulierenden Zellularraums mittels eines Homomorphismus h aufzuspreizen. Damit läßt sich die Vorstellung verbinden, Zellen des simulierten Zellularraums werden durch das aufgespreizte Raster $h(N') = \{h(x) | \, x \in N'\}$ überlagert. Das Problem liegt dann darin, eine Menge K von Punkten des $\mathbb{Z}^d$ zu finden, so daß gilt: Wird die Menge der Zellen $h(x) + K$ als eine Zelle mit Nachbarschaft $h(N') + K$ aufgefaßt, so muß sie in einem Zeitschritt Zugriff auf soviel Information haben, wie die Zellen $h(x) + K$ des simulierten Zellularraums in k Zeitschritten.

Satz 3.2. Es seien $M = (S, d, N, \sigma, q_0)$ und $M' = (S', d, N', \sigma', q'_0)$ Zellularräume. h sei ein injektiver Homomorphismus von der additiven Gruppe $\mathbb{Z}^d$ nach $\mathbb{Z}^d$, und $K \subset \mathbb{Z}^d$ sei eine endliche Menge. Die Zustandsmenge eines Automaten x von M' werde als Kartesisches Produkt der Automaten $\{h(x)\} + K$ von M definiert. Eine hinreichende Bedingung für die Existenz einer Überführungsfunktion zur

Simulation von M durch M' in $1/k$-facher Zeit ist folgendermaßen gegeben:

$$h(N') + K \supseteq kN + K$$

Beweis. Es sei p eine Zelle in M. Innerhalb von k Übergängen hat p Zugriff zu der im Bereich kN enthaltenen Information, d.h., der Zustand von p zum Zeitpunkt $t+k$ ergibt sich eindeutig aus den Zuständen der Automaten q, wobei $q \in \{p\} + kN$, zum Zeitpunkt t. Insbesondere erhält man den Wert einer Komponenten $h(x)+a$ eines Automaten x aus M' zum Zeitpunkt $t+k$ (von M), wobei $a \in K$, aus den Zuständen der Automaten $\{h(x)\} + \{a\} + kN$ von M (im Zeitpunkt t). Der Zustand des Automaten x aus M' zum Zeitpunkt $t+k$ (in M) ergibt sich damit eindeutig aus den Zuständen der Automaten $\{h(x)\} + K + kN$ von M zum Zeitpunkt t.
Andererseits ist der nächste Zustand eines Automaten x aus M' durch die Zustände der Automaten $\{x\} + N'$ aus M' bestimmt. Der Zustand jedes dieser Automaten n' ist nach Voraussetzung das Kartesische Produkt der Zustände von $\{h(x+n')\} + K$ von M. Eine Beschleunigung um den Faktor k in M' kann also erreicht werden, wenn gilt:
$h(\{x\} + N') + K \supseteq \{h(x)\} + K + kN$. Da h als Homomorphismus vorausgesetzt war, ist $h(\{x\} + N') = \{h(x)\} + h(N')$. Da $h(x)$ nur den Ort festlegt, ergibt sich die Bedingung $h(N') + K \supseteq K + kN$. □

Beispiel 3.2. Es sei $M = (S, 2, M_1, \sigma, q_0)$ ein Zellularraum. Dann existiert ein Zellularraum $M' = (S', 2, H_1, \sigma', q_0')$, der M in 1/2-facher Zeit simuliert:

Ein injektiver Homomorphismus von $\mathbb{Z}^2$ nach $\mathbb{Z}^2$ ist $h((x_1, x_2)) := (2x_1 - 2x_2, 2x_1 + 2x_2)$. Wählt man $K = \bar{M}_3$, dann ist die Bedingung von Satz 3.2 erfüllt (vgl. Abbildung 3.7):

$$h(H_1) + \bar{M}_3 \supseteq 2M_1 + \bar{M}_3$$

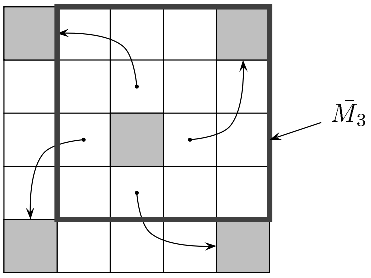


Abbildung 3.7. Raster zum Beispiel 3.2.

Satz 3.3. Für einen beliebigen Zellularraum $M = (S, d, N, \sigma, q_0)$ existiert ein Zellularraum $M' = (S', d, M_1, \sigma', q_0')$, der M in $1/k$-facher Zeit, $k \in \mathbb{N}$, simuliert.

Beweis. Es genügt, einen Homomorphismus h und eine Menge K festzulegen, so daß die Bedingungen von Satz 3.2 erfüllt sind.
Es sei also $P(kN)$ der minimale Quader für das Raster kN. Ferner seien $u(kN) = (u_1, \dots, u_d)$, $o(kN) = (o_1, \dots, o_d)$ und $a_i := \max\{u_i, o_i\}$ für $1 \leq i \leq d$. Dann werden h und K folgendermaßen gewählt:

$$h((x_1, \dots, x_d)) := (a_1 x_1, a_2 x_2, \dots, a_d x_d)$$
$$K := P(kN).$$

Mit diesen Festlegungen soll also gelten:

$$h(M_1) + P(kN) \supseteq kN + P(kN).$$

Im folgenden sei i immer aus der Menge $\{1, \dots, d\}$.

Aus den Definitionen folgt durch Einsetzen der Elemente von M_1:

$$(x_1, \dots, x_d) \in h(M_1) + K \iff \begin{cases} -2u_i \leq x_i \leq u_i + o_i & \text{falls } u_i \geq o_i \\ -u_i - o_i \leq x_i \leq 2o_i & \text{sonst} \end{cases}$$

Damit ist $h(M_1) + P(kN)$ eine Überdeckung von $\{x| -2u_i \leq x_i \leq 2o_i\} = 2P(kN)$. Weiterhin gilt offensichtlich $P(kN) \supseteq kN$, und damit ist $2P(kN) = P(kN) + P(kN)$ seinerseits eine Überdeckung der rechten Seite. $\square$

3.2.3 Zustandsreduktion

Nachdem gezeigt wurde, daß eine Rasterreduktion durch Erweiterung der Zustandsmenge und Zeitverzögerung möglich ist, die Zeitverzögerung aber ohne erneute Erweiterung des Rasters wieder eliminiert werden konnte, stellt sich nun die Frage, inwieweit die Zustandsmenge reduziert werden kann. Es erscheint intuitiv klar, daß eine Zustandsreduktion eine Vergrößerung des Rasters zur Folge hat.

Das folgende Beispiel von E. F. Codd (1968) zeigt, wie ein Zellularraum mit 8 Zuständen und H_1-Raster durch einen Zellularraum mit nur 2 Zuständen ohne Zeitverlust simuliert werden kann. Die Hauptschwierigkeit resultiert dabei aus der Homogenität der Räume: Es ist notwendig, jeweils diejenigen Automaten in dem simulierenden Zellularraum, die die (binäre) Kodierung des Zustandes des simulierten Zellularraumes enthalten, so zu kennzeichnen, daß sie in die Lage versetzt werden, sich selbst, d.h. genauer, ihren Stellenwert in der Kodierung zu identifizieren.

Beispiel 3.3. Jeder Zellularraum $M = (S, 2, H_1, \sigma, q_0)$ mit $|S| = 8$ kann durch einen Zellularraum $M' = (S', 2, N, \sigma', q'_0)$ mit $S' = \{0, 1\}$ ohne Zeitverlust simuliert werden.

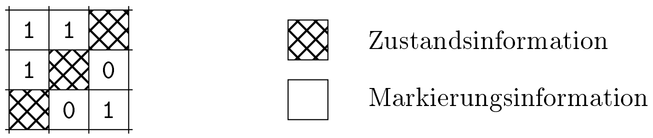


Abbildung 3.8. Schema einer Makrozelle.

Zur Vereinfachung der Beschreibung des Verfahrens denke man sich den Raum M' unterteilt in Quadrate der Kantenlänge 3. Jedes dieser Quadrate (bzw. die Gesamtheit der entsprechenden 9 Zellen) bildet jetzt eine Makrozelle, die genau einer Zelle des ursprünglichen Raumes entspricht. Der ursprüngliche Zustand wird binär kodiert in der Diagonalen der Makrozelle gespeichert. Abbildung 3.8 zeigt diese interne Einteilung einer Makrozelle.

Aufgrund der Anordnung der Markierungsinformation kann jede einzelne Zelle anhand ihrer Nachbarschaft feststellen, an welcher „Position“ in der Makrozelle sie liegt.

Die folgenden Abbildungen zeigen eine Konfiguration in M und die entsprechende Konfiguration in M'.

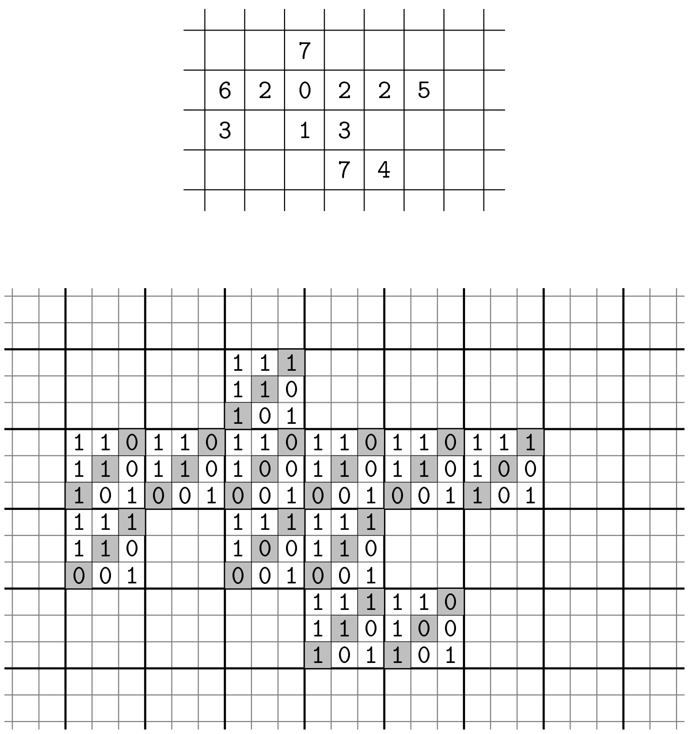


Abbildung 3.9. Konfigurationen zur Zustandsreduktion.

Um Zeitverlust bei der Simulation zu vermeiden, muß das Raster N so gewählt werden, daß jeder Automat der Makrozelle jeweils mindestens zur gesamten Information der fünf (nach dem H_1-Raster) zusammengehörigen Makrozellen Zugriff hat. Dies ist bei dem in der folgenden Abbildung aufgezeichneten Raster gewährleistet. Dabei sind die „Grenzen“ der Makrozellen fett gezeichnet. Die grau hinterlegte Zelle stellt den Ursprung der Nachbarschaft dar.

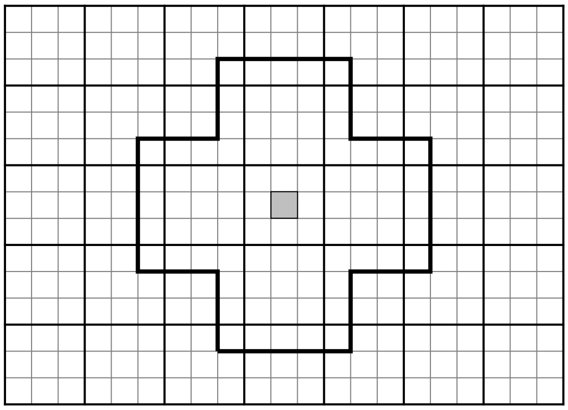


Abbildung 3.10. Raster eines Zwei-Zustands-Zellularraumes.

3.3 Berechnungsuniversalität

Wie bereits erwähnt, sollen verschiedene parallele Automatenmodelle hinsichtlich ihrer Fähigkeit, formale Sprachen verarbeiten zu können, untersucht werden. Deshalb besteht zunächst ein Interesse daran, prinzipielle Möglichkeiten von Zellularräumen auszuloten. In diesem Abschnitt wird gezeigt, daß bereits eindimensionale Zellularräume berechnungsuniversell im Sinne der Churchschen These sind.

Eine Nachweismöglichkeit besteht darin, die Simulation von Turingmaschinen durch Zellularräume zu betrachten (vgl. A. R. Smith III (1971b) und R. Vollmar (1979)). Falls diese Simulation für beliebige Turingmaschinen gelingt, dann natürlich auch für universelle. Da bekannt ist, daß universelle Turingmaschinen mit 6 Bandsymbolen und 6 Zuständen existieren, liefert diese Vorgehensweise auch Aussagen über die überhaupt notwendige Größe von Zellularräumen.

Satz 3.4. Es sei $T = (Q, \Sigma, \Pi, \delta, s_0, \mathtt{b}, F)$ eine Turingmaschine mit $|Q| = n$ und $|\Pi| = m$. Dann existiert ein Zellularraum $M = (S, 1, H_1, \sigma, q_0)$ mit $q_0 = \mathtt{b}$ und $|S| = m + 4n$, der T in 2facher Zeit simuliert.

Beweis. O.B.d.A. sei angenommen, daß $Q \cap \Pi = \emptyset$ gilt. Jedes Symbol der Bandinschrift der Turingmaschine wird in einem Automaten des Zellularraumes dargestellt. An der entsprechenden Stelle repräsentiert ein zusätzlicher, gewissermaßen zwischen zwei Feldern eingeschobener Automat den Kopf der Turingmaschine, der sich auf dem Feld, welches durch seinen rechten Nachbarn dargestellt wird, befinden soll (vgl. Abbildung 3.11). Daraus ergibt sich nun das Problem, daß im Falle einer Linksverschiebung des Kopfes die links vom Kopf gelegene Zelle dies aufgrund der H_1-Nachbarschaft nicht unmittelbar erkennen kann. Deshalb nimmt der den Kopf repräsentierende Automat in jeweils einem Zwischenschritt einen Zustand an, aus dem neben dem neu anzunehmenden Zustand (der Turingmaschine) die Bewegungsrichtung ersichtlich ist. Der Einfachheit halber wird diese Vorgehensweise auch für Rechtsverschiebungen benutzt.

Die Zustandsmenge wird folgendermaßen festgelegt:

$$S = \Pi \cup Q \cup (Q \times \{\mathtt{0}, \mathtt{1}, \mathtt{-1}\}).$$

Die lokale Überführungsfunktion σ wird in Abhängigkeit von δ bestimmt, wobei die Komponenten des H_1-Rasters gemäß $(-1, 0, 1)$ geordnet seien.

$$
\begin{aligned}
\delta(q,a) = (q',a',0) \Longrightarrow (\forall \tilde{a}, \hat{a} \in \Pi : \; & \\
& (\sigma(\tilde{a}, q, a) = (q', \mathtt{0}) \wedge \\
& \;\sigma(q, a, \hat{a}) = a' \wedge \\
& \;\sigma(\tilde{a}, (q', \mathtt{0}), \hat{a}) = q'))
\end{aligned}
$$

$$
\begin{aligned}
\delta(q,a) = (q',a',1) \Longrightarrow (\forall \tilde{a}, \hat{a}, \bar{a} \in \Pi : \; & \\
& (\sigma(\tilde{a}, q, a) = (q', \mathtt{1}) \wedge \\
& \;\sigma(q, a, \hat{a}) = a' \wedge \\
& \;\sigma(\tilde{a}, (q', \mathtt{1}), \hat{a}) = \hat{a} \wedge \\
& \;\sigma((q', \mathtt{1}), \hat{a}, \bar{a}) = q'))
\end{aligned}
$$

$$
\begin{aligned}
\delta(q,a) = (q',a',-1) \Longrightarrow (\forall \tilde{a}, \hat{a}, \bar{a}, \acute{a} \in \Pi : \; & \\
& (\sigma(\tilde{a}, q, a) = (q', \mathtt{-1}) \wedge \\
& \;\sigma(q, a, \hat{a}) = a' \wedge \\
& \;\sigma(\tilde{a}, (q', \mathtt{-1}), \bar{a}) = \tilde{a} \wedge \\
& \;\sigma((\acute{a}, \tilde{a}, (q', \mathtt{-1})) = q'))
\end{aligned}
$$

In allen anderen Fällen bleiben die Automaten in ihrem Zustand.

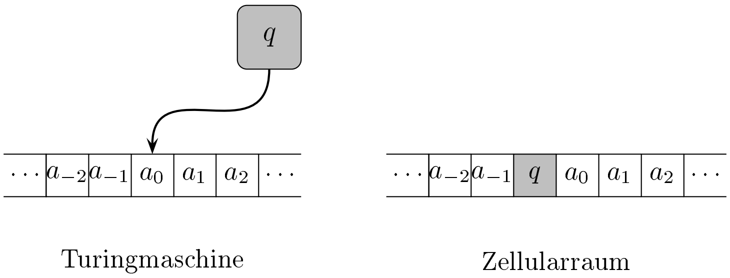


Abbildung 3.11. Konfigurationen von Turingmaschine und Zellularraum.

Falls als Nachbarschaftsindex $(-1, 0, 1, 2)$ gewählt wird, läßt sich eine Simulation ohne Zeitverlust erreichen, wobei $m+n$ Zustände benötigt werden.

Korollar 3.1. Es existiert ein eindimensionaler berechnungsuniverseller Zellularraum mit zwei Zuständen.

3.4 Das Firing Squad Synchronization Problem

Ein Synchronisationsproblem, das mittlerweile als *Firing Squad Synchronization Problem* (FSSP) bezeichnet wird, wurde um 1957 von J. Myhill gestellt. Es tauchte im Zusammenhang mit Möglichkeiten zum synchronen Arbeitsbeginn verschiedener Teile einer Maschine erstmalig auf. In der Zwischenzeit sind zeitoptimale Lösungen für das Problem selbst und eine Reihe von Modifikationen gefunden worden. Sie gelten als gute Beispiele für Algorithmen, die aufgrund lokaler Entscheidungen ein globales Gesamtverhalten aufweisen.

E. F. Moore (1964) formuliert das Problem folgendermaßen:

„Man betrachte eine endliche (aber beliebig lange) eindimensionale Kette von endlichen Automaten, die alle gleich sind, abgesehen von denen am Rand. Die Automaten werden Soldaten genannt, und einer der Endautomaten wird als General bezeichnet. Die Automaten arbeiten synchron, und der Zustand jedes Automaten zum Zeitpunkt $t+1$ hängt von seinem Zustand und den Zuständen seiner beiden Nachbarn zum Zeitpunkt t ab. Das Problem besteht darin, die Zustände und Übergänge der Soldaten derart festzulegen, daß der General sie veranlassen kann, zum genau gleichen Zeitpunkt in einen speziellen Endzustand (das ‚Feuern‘) überzugehen. Zu Beginn $(t = 0)$ seien alle Soldaten im Ruhezustand.“

An dieser Stelle wird nochmals auf die — vernachlässigte — Sprechweise „Klasse von Zellularautomaten“ hingewiesen. Es müssen eine Zustandsmenge und eine Überführungsfunktion gefunden werden, die das Problem für beliebig lange Ketten lösen. Die Ketten bilden dabei eine Klasse von Retinas. Insbesondere folgt, daß die einzelnen Automaten unabhängig von der Länge der Kette arbeiten müssen, da sie aufgrund ihrer endlichen Zustandsmenge nicht beliebig weit „zählen“ können.

Definition 3.9.
Das *Firing Squad Synchronization Problem* besteht darin, eine Klasse von eindimensionalen Zellularautomaten $(S, 1, H_1, \sigma, q_0)$ so anzugeben, daß für die von σ induzierte globale Transformation $\mathcal{T}$ und für $|R| = n$ mit $n > 1$ gilt:

1) $\exists t \in \mathbb{N} : \mathcal{T}^t(gq_0^{n-1}) = f^n$, wobei $f, g, q_0 \in S$.
2) $\forall t' < t : \mathcal{T}^{t'}(gq_0^{n-1}) \in (S \setminus \{f\})^n$.
3) Mit $H_1 = (0, -1, 1)$ ist $\sigma(q_0, q_0, \#) = q_0$.

Die minimale Lösungszeit liefert der folgende Satz.

Satz 3.5. Für das FSSP mit $|R| = n$ gilt:

$$t \geq 2n - 2.$$

Beweis. Im Gegensatz zur Behauptung sei angenommen, es liege ein zellularer Automat mit $|R| = n$ vor, so daß für ein $t_f < 2n - 2$ gilt:

$$\mathcal{T}^{t_f}(gq_0^{n-1}) = f^n.$$

Zunächst stellt man fest, daß der rechte Randautomat den Ruhezustand per Definition erst dann verlassen darf, wenn sein linker Nachbar sich nicht mehr im Ruhezustand befindet. Betrachtet man ferner ein Signal, welches zur „Aktivierung“ der einzelnen Automaten vom General nach rechts ausgesandt wird, so benötigt es aufgrund der H_1-Nachbarschaft mindestens $n - 1$ Zeitschritte bis zum rechten Rand. Also kann der General g zum Zeitpunkt t_f keine Rückmeldung vom n-ten Automaten empfangen haben, da hierzu $2n - 2$ Schritte notwendig sind. Er feuert somit unabhängig von einer derartigen Rückmeldung.

Liegt nun ein Zellularautomat mit $|R| = 2n$ vor (die Überführungsfunktion müßte immer noch das Gewünschte leisten), dann feuert der General wegen der deterministischen Arbeitsweise des Zellularautomaten und wegen des Gleichbleibens des linken Teils der Retina ebenfalls nach t_f Schritten. Zu diesem Zeitpunkt befindet sich aber der $2n$-te Automat noch im Ruhezustand, da er von einem Signal des Generals frühestens nach $2n - 1$ Schritten erreicht werden kann. □

Es ist bekannt, daß eine zeitoptimale Lösung mit sechs Zuständen existiert (J. Mazoyer (1987)), vier Zustände aber nicht ausreichen (R. M. Balzer (1967)).

Im folgenden wird zunächst eine nicht zeitoptimale Lösung von R. M. Balzer (1967) beschrieben, an der aber die prinzipielle Vorgehensweise sichtbar wird.

Algorithmus 3.1. Das Problem kann gelöst werden, indem die Reihe der Automaten in zwei, vier, acht usw. gleichlange Teile aufgespalten wird, so lange, bis alle Automaten Teilungspunkte markieren. In diesem Zeitschritt erfolgt das synchrone Feuern. Die Aufspaltung kann derart erfolgen, daß die Reihe zunächst in zwei gleichlange Teile aufgespalten und anschließend das Verfahren auf die entstandenen Teile angewandt wird usw.

Um die Reihe in zwei gleichlange Teile zu teilen, sendet der General gleichzeitig zwei Signale $S1$ und $S2$ nach rechts. $S1$ hat dabei die Geschwindigkeit 1, während $S2$ mit der Geschwindigkeit 1/3 läuft. Erreicht $S1$ den rechten Rand, so wird ein Signal $S3$ mit der Geschwindigkeit 1 nach links zurückgesandt. Durch die angegebenen Geschwindigkeiten der Signale bedingt, treffen sich die Signale $S2$ und $S3$ „in der Mitte“ der Retina. Die Mitte kann dabei durch einen oder durch zwei Automaten repräsentiert werden, je nachdem ob n ungerade oder gerade ist.

Der oder die Mittelautomaten fungieren ab jetzt als zusätzliche Generäle und senden jeweils $S1$- und $S2$-Signale nach rechts und links. Auf diese Weise werden simultan die Mittelautomaten der beiden Halbreihen bestimmt. Dieses Verfahren wird so lange fortgeführt, bis alle Automaten Generäle sind. Dann erfolgt synchron das Feuern.

Da die Zeiten zum Halbieren der Halbreihen durch

$$3n/2, 3n/4, 3n/8, \ldots$$

abgeschätzt werden können, ist das Verfahren nach spätestens $3n$ Zeitschritten beendet. $\square$

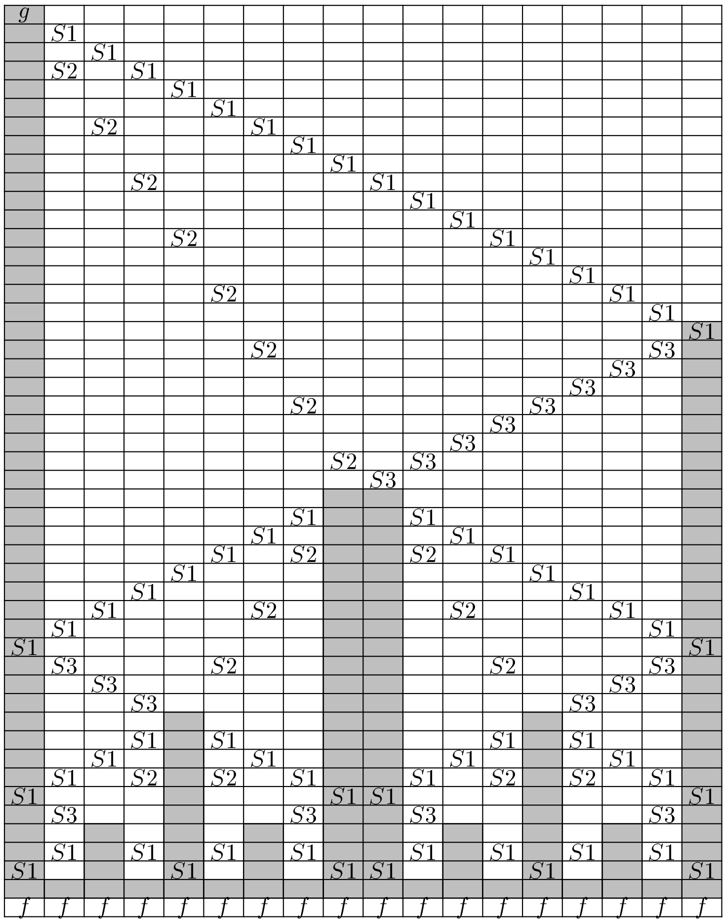


Abbildung 3.12. Synchronisation nach einem „langsamen“ Verfahren.

Im folgenden wird nun zunächst eine Vorgehensweise beschrieben, mit der sich ein zeitoptimales Verfahren ergibt, das jedoch nicht unabhängig von der Länge der Retina arbeitet, also keine Lösung darstellt. Anschließend wird eine Möglichkeit erläutert, diese Schwierigkeit zu beheben.

Algorithmus 3.2. Der Algorithmus 3.1 wird folgendermaßen modifiziert (vgl. Abbildung 3.13):
Nachdem das Signal $S1$ am rechten Rand eingetroffen ist, verhält sich der entsprechende Randautomat wie ein General und sendet zwei Signale $S3$ und $S4$ nach links. Das Signal $S4$ verhält sich, abgesehen von der Laufrichtung, wie $S2$. Die Mitte der Retina wird wieder durch das Zusammentreffen der Signale $S2$ und $S3$ bestimmt. Der oder die Mittelautomaten verhalten sich wie oben beschrieben und senden u.a. ein $S1$- und $S3$-Signal nach rechts.

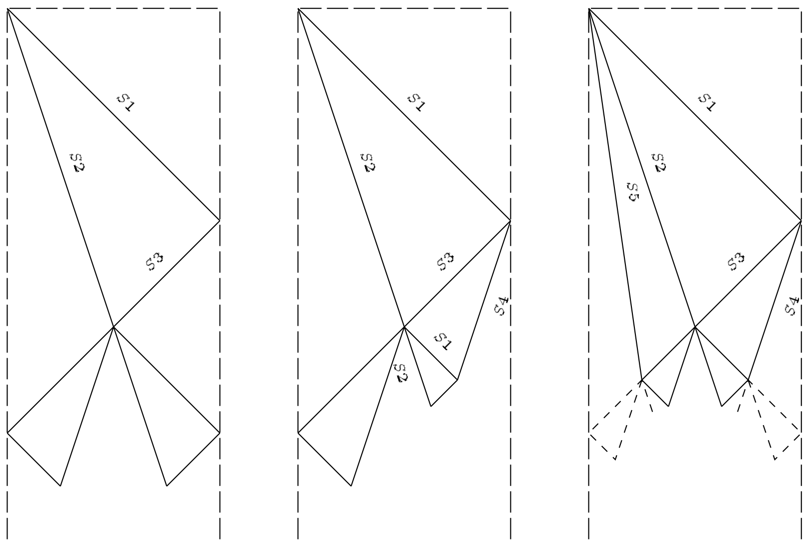


Abbildung 3.13. Schematische Darstellung eines „langsamen“ Verfahrens und einer möglichen Beschleunigung.

Das Zusammentreffen von $S1$ und $S4$ bestimmt den rechten Viertelpunkt der Retina nach $3n/2 + n/4$ Zeitschritten. Nach weiteren $n/8$ Zeitschritten wird der linke Achtelpunkt der rechten Hälfte gefunden. Falls weitere Teilungspunkte entsprechend gefunden werden können, so ergibt sich eine Gesamtlaufzeit von $2n$ Zeiteinheiten:

$$3n/2 + n/4 + n/8 + n/16 + \cdots = 2n$$

Da der General mitgezählt wurde, müssen nur $n - 1$ Automaten durchlaufen werden. Damit erhält man die optimale Zeit von $2(n-1)$ Zeitschritten.

Das bisherige Verfahren stellt aber noch keine Lösung dar, da in jedem Intervall nur einer der beiden Teilungspunkte in der geforderten Zeit gefunden wird. Den linken Viertelpunkt erhält man dadurch, daß der General im Zeitpunkt $t = 0$ ein zusätzliches Signal $S5$ mit der Geschwindigkeit $1/7$ nach rechts sendet. Nun besteht das Problem darin, den linken Achtelpunkt rechtzeitig zu bestimmen. Wiederum kann der General (und alle Zellen, die später zu Generälen „befördert" werden) ein zusätzliches Signal mit Geschwindigkeit $1/15$ nach rechts senden, für eine Lösung müssen aber weitere Wellen eingeführt werden. Mit den Geschwindigkeiten

$$\frac{1}{2^k - 1}, k \in \mathbb{N},$$

können alle linken $(1/2^k)$-tel Punkte in der angestrebten Zeit bestimmt werden. Genügend solcher Markierungssignale synchronisieren also die Automatenkette in Minimalzeit. □

Das eigentliche Problem wird durch dieses Verfahren nicht gelöst, weil die Anzahl der Signale von der Länge der Retina abhängt. Da sich die Zustandszahl aus der Anzahl der Signale ergibt, besteht ein Zusammenhang zwischen der Zustandszahl und der Länge der Retina. Die Unabhängigkeit dieser Größen sichert ein auf diesem Prinzip aufbauender Algorithmus von A. Waksman (1966).
Dabei werden die zusätzlich benötigten Wellen durch Triggersignale

bewegt. In Abbildung 3.14 sind diese Signale durch ∘ dargestellt. Die Triggerwellen selbst werden von den nach rechts bzw. links laufenden Signalen $S1$ bzw. $S3$ alle zwei Zeitschritte in die jeweils entgegengesetzte Richtung gesandt. Jede getriggerte Welle absorbiert jedes zweite auf sie treffende Triggersignal. Auf diese Weise ergibt sich das gewünschte Verhalten und damit eine zeitoptimale Lösung.

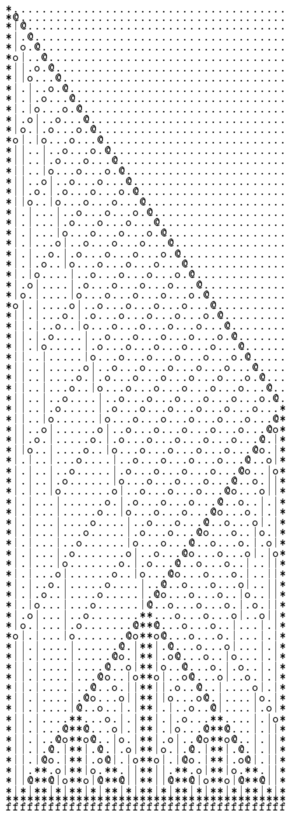


Abbildung 3.14. Schema einer zeitoptimalen Synchronisation nach Waksman.

Es soll nun eine Verallgemeinerung des FSSP in eindimensionalen Zellularautomaten untersucht werden, bei der die Forderung fallengelassen wird, der General befinde sich am Rand der Retina (F. R. Moore und G. C. Langdon (1968)).

Definition 3.10. Das *Firing Squad Synchronization Problem mit dem General an beliebiger Stelle der Retina* besteht darin, eine Klasse von eindimensionalen Zellularautomaten $(S, 1, H_1, \sigma, q_0)$ so anzugeben, daß für die von σ induzierte globale Transformation $\mathcal{T}$ und für $|R| = n$ mit $n > 1$ und $i + j = n - 1$ gilt:

1) $\exists t \in \mathbb{N} : \mathcal{T}^t(q_0^i g q_0^j) = f^n$, wobei $f, g, q_0 \in S$.

2) $\forall t' < t : \mathcal{T}^{t'}(q_0^i g q_0^j) \in (S \setminus \{f\})^n$.

3) Mit $H_1 = (0, -1, 1)$ ist $\sigma(q_0, q_0, \#) = q_0$ und $\sigma(q_0, \#, q_0) = q_0$.

Satz 3.6. Für das FSSP mit dem General an beliebiger Stelle und $|R| = n$ gilt:

$$t \geq 2n - 2 - \min\{i, j\}.$$

Beweis. Der Beweis über die minimale Zeit für das ursprüngliche Problem kann fast wörtlich übernommen werden. □

Algorithmus 3.3. O.B.d.A. sei $\min\{i, j\} = i$. Das Verfahren beruht auf demjenigen von Waksman. Die Lösungsidee besteht darin, die Signale so zu rekonstruieren, wie sie beim FSSP mit dem General am linken Rand vorkommen würden. In Abbildung 3.15 ist eine Beispielsynchronisation dargestellt. Die gestrichelten Linien zeigen die zu rekonstruierenden Signale.

Der General sendet nach beiden Seiten Signale mit Geschwindigkeit 1 aus, die die Triggersignale und damit auch die getriggerten Signale mit den Geschwindigkeiten $1/(2^k - 1), k \in \mathbb{N}$, gemäß dem Verfahren von Waksman erzeugen. Das Signal, welches von dem Endautomaten, der näher beim General liegt, reflektiert wird, trifft als erstes Signal auf die Position des Generals (Punkt A) und bestimmt somit den dem General am nächsten liegenden Rand. In diesem Punkt muß die

Geschwindigkeit des Signals von 1 auf 1/3 reduziert werden. Ebenso müssen alle weiteren Signale mit den Geschwindigkeiten $1/(2^k - 1)$, $k \in \mathbb{N}$, auf der kürzeren Seite der Retina entlang der gedachten Linie AB (Geschwindigkeit 1) auf $1/(2^{k+1} - 1)$ verlangsamt werden. Dies wird dadurch erreicht, daß die Welle der Geschwindigkeit 1, die alle zwei Zeiteinheiten die Triggersignale erzeugt, im Punkt A nicht verlangsamt wird, sondern ungestört weiterläuft und von der ersten Welle des gegenüberliegenden Endautomaten gelöscht wird (Punkt C). Zusätzlich wird im Punkt A eine Welle der Geschwindigkeit 1/3 erzeugt, die den Mittelpunkt festlegt, also die Funktion der ersten zu verlangsamenden Welle erfüllt. Da dieses Signal nur jedes zweite Triggersignal durchläßt, werden automatisch alle Markierungssignale in der gewünschten Weise verlangsamt.

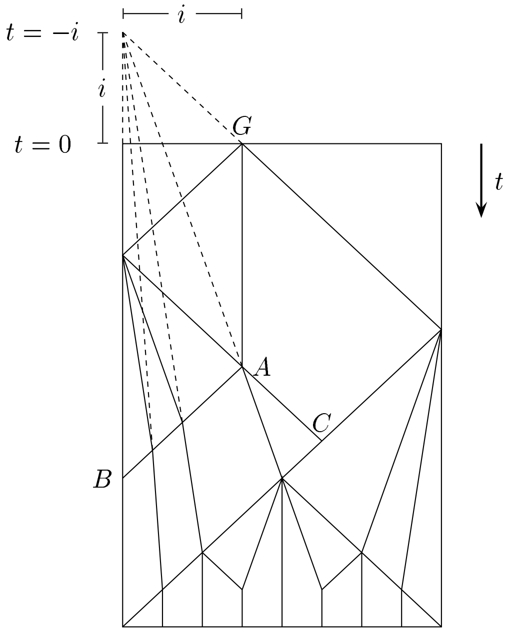


Abbildung 3.15. Schema für den General an beliebiger Stelle.

Nach dem Zeitpunkt B stimmt die Lösung mit der von Waksman überein. Gegenüber der Lösung des ursprünglichen Problems spart man die Zeit, die das Signal $S1$ für den Weg vom näheren Endautomaten zur Position des Generals benötigt, also $\min\{i,j\}$ Zeiteinheiten. □

Zum Abschluß des Abschnittes soll der obige Algorithmus zur Synchronisation von rechteckigen Retinas benutzt werden (I. Shinahr (1974)).

Es seien eine zweidimensionale Retina mit m Zeilen und n Spalten und ein H_1^2-Raster gegeben. Der General befinde sich in in der linken oberen Ecke $(1,1)$. Die Definition 3.9 sei entsprechend auf zwei Dimensionen erweitert.

Satz 3.7. Für die zur Synchronisation einer $m \times n$-Retina benötigte Zeit t gilt:

$$t \geq m + n + \max\{m,n\} - 3.$$

Beweis. Im Gegensatz zur Behauptung wird angenommen, daß eine $m \times n$-Retina existiert, so daß alle Automaten nach t_f Zeitschritten mit

$$t_f < m + n + \max\{m,n\} - 3$$

den Feuerzustand annehmen.
O.B.d.A. kann $n \geq m$ vorausgesetzt werden. Wegen $t_f < 2n + m - 3$ kann der linke untere Automat $(m,1)$ noch kein Signal, das durch den General initiiert wurde, von einem Automaten der rechten Spalte n empfangen haben: Ein Signal benötigt für den Weg vom Automaten $(1,1)$ zum Automaten (i,n) mindestens $n-1+i-1$ Zeiteinheiten. Dazu addieren sich für die Strecke von (i,n) nach $(m,1)$ wenigstens $m+n-i-1$ weitere Zeiteinheiten. Die Summe beträgt $m+2n-3$ Zeiteinheiten. Der Automat $(m,1)$ hat also unbeeinflußt von irgendeinem Automaten aus der rechten Spalte gefeuert. Vergrößert man das Feld an der rechten Seite um $n \cdot m$ gleiche Automaten, die in n weiteren Spalten angeordnet liegen, so feuert der Automat $(m,1)$ ebenfalls zur Zeit t_f, da die Überführungsfunktion deterministisch ist und sich nichts an der Automatenkonstruktion und den Automaten,

die den Automaten $(m, 1)$ beeinflussen, geändert hat. Das neue Feld besteht aus $2n \cdot m$ Automaten. Wegen $t_f < 2n + m - 3$ kann aber noch kein Signal des Generals zur Zeit t_f den rechten unteren Automaten $(m, 2n)$ erreicht haben, d.h., er befindet sich noch im Ruhezustand.

□

Der folgende Algorithmus liefert die Lösung in minimaler Zeit.

Algorithmus 3.4. General ist der Automat $(1, 1)$. Er initiiert die Synchronisation in der ersten Spalte und der ersten Reihe und sendet gleichzeitig ein Signal mit der Geschwindigkeit 1/3 im 45°-Winkel nach unten. Dieses Signal „befördert" die Automaten auf der Winkelhalbierenden zu Generälen, die dann ihrerseits in den Teilspalten und Teilreihen, deren obere bzw. linke Ecke sie bilden, die Synchronisation einleiten: Der Automat (i, i), wobei $1 \leq i \leq \min\{m, n\}$, geht zum Zeitpunkt $t_i = 3(i-1)$ in den Generalszustand über. Der Automat (i, i), die Automaten unterhalb von (i, i) und die Automaten rechts von (i, i) bilden eine Kette der Länge $n_i = (n-i) + (m-i) + 1$. $l_i = \min\{n-i, m-i\}$ ist der minimale Abstand vom General zu einem der beiden Enden der entsprechenden Kette.

Für jedes i, wobei $1 \leq i \leq \min\{m, n\}$, erreichen die Automaten (i, j) mit $i \leq j \leq n$ und die Automaten (j, i) mit $i \leq j \leq m$ den Feuerzustand nach

$$\begin{aligned} t &= t_i + 2n_i - 2 - l_i \\ &= 3(i-1) + 2(n-i+m-i+1) - 2 - \min\{n-i, m-i\} \\ &= m + n + \max\{m, n\} - 3 \end{aligned}$$

Zeiteinheiten. □

3.5 Sprachverarbeitung

Um formale Sprachen mit Zellularautomaten bearbeiten bzw. erkennen zu können, muß zunächst festgelegt werden, wie Eingabe und

Ausgabe erfolgen sollen. Die Sprachfamilien, die durch Zellularräume und -automaten ohne weitere Einschränkungen akzeptiert werden können, sind recht leicht zu bestimmen. Einer der Gründe für die Einführung von parallelen Automaten war ein erhoffter Zeitgewinn gegenüber den sequentiellen. Es ist nun naheliegend, gewisse Ressourcen einzuschränken und z.B. Sprachfamilien zu betrachten, deren Elemente in einer bestimmten Zeit akzeptiert werden können.

3.5.1 Grundlegendes

Im folgenden bezeichnet $\mathcal{T}$ immer die durch σ induzierte globale Transformation. Für ein Wort, das aus unendlich vielen Zeichen $\mathtt{x}$ besteht, schreiben wir $\mathtt{x}^\omega$.

Definition 3.11. Es seien eine formale Sprache $L \subseteq A^+$ und eine Funktion $t : \mathbb{N} \longrightarrow \mathbb{N}$ gegeben. L wird durch einen Zellularautomat $M = (S, 1, N, \sigma, q_0)$ mit $A \subseteq S$ und $F \subseteq S$ genau dann mit der Zeitkomplexität t akzeptiert, wenn

$$L = \{w \in A^+ \mid \mathcal{T}^{t(|w|)}(\mathtt{\#}^\omega w \mathtt{\#}^\omega) \in (\mathtt{\#}^\omega S^{|w|-1} F \mathtt{\#}^\omega)\}.$$

Das Verfahren zum Akzeptieren von Sprachen in Zellularräumen ergibt sich aus der obigen Definition unmittelbar durch die Festlegung, daß # den Ruhezustand bezeichnet. (In Zellularräumen wird die Existenz eines Grenzzustandes nicht gefordert.)

Definition 3.12.

a) Die Familie der von Zellularräumen mit H_1-Raster und Zeitkomplexität t akzeptierbaren Sprachen wird mit $\mathscr{L}_t(\mathrm{CS})$ bezeichnet.
b) Die Familie der von Zellularautomaten mit H_1-Raster und Zeitkomplexität t akzeptierbaren Sprachen wird mit $\mathscr{L}_t(\mathrm{CA})$ bezeichnet.
c) Die Familie der von Zellularautomaten mit $\bar{H}_1$-Raster und Zeitkomplexität t akzeptierbaren Sprachen wird mit $\mathscr{L}_t(\mathrm{OCA})$ bezeichnet.

Falls ein Automat beliebig viel Zeit für seine Berechnungen zur Verfügung hat, wird der Index bei der entsprechenden Sprachfamilie weggelassen. In diesem Fall müssen aber zusätzliche Festlegungen bzgl. des Akzeptierens getroffen werden. Es ist ja durchaus möglich, daß der rechte Randautomat zunächst einen akzeptierenden Zustand annimmt, diesen wieder verläßt usw. Um an dieser Stelle weiterführende Details zu vermeiden, soll angenommen werden, daß sich in diesen Fällen (und nur in diesen) akzeptierende Zustände erhalten. D.h., für eine Zelle i gilt in jedem Zeitpunkt t: $c_t(i) \in F \Rightarrow c_{t+1}(i) \in F$.

Die beiden folgenden Zeitkomplexitäten werden von besonderem Interesse sein und sollen daher speziell bezeichnet werden.

Definition 3.13.
a) Die Zeitkomplexität $rt : \mathbb{N} \longrightarrow \mathbb{N}$ mit $rt(n) = n$ wird mit *Realzeit* bezeichnet.
b) Für alle $z \in \mathbb{N}$ wird die Zeitkomplexität $lt : \mathbb{N} \longrightarrow \mathbb{N}$ mit $lt(n) = z \cdot n$ mit *Linearzeit* bezeichnet.

Satz 3.8. $\mathscr{L}(\mathrm{CS}) = \mathscr{L}_0$

Beweis. „$\Longleftarrow$“ folgt unmittelbar aus der Berechnungsuniversalität der Zellularräume und der Feststellung $\mathscr{L}(\mathrm{TM}) = \mathscr{L}_0$.
„$\Longrightarrow$“ ergibt sich, da in der Anfangskonfiguration des Zellularraumes immer nur endlich viele Zellen nicht im Ruhezustand sind. Ein globaler Übergang kann so von der TM in einer Folge von Übergängen, in denen sich der Kopf einmal über den nichtleeren Teil des Bandes bewegt, simuliert werden. □

Satz 3.9. $\mathscr{L}(\mathrm{CA}) = \mathscr{L}(\mathrm{LBA})$

Beweis. Zellularautomaten gehen aus Zellularräumen hervor, indem die Größe des zur Verfügung stehenden Raumes auf die Länge der Eingabe beschränkt wird. Analog gehen LBAs aus TMs hervor, indem die Länge des zur Verfügung stehenden Bandabschnittes durch die Länge der Eingabe beschränkt wird. Die Vorgehensweise entspricht also der des vorangegangenen Satzes. □

Beispiel 3.4. $\{v\mathtt{1}w \mid v, w \in \{\mathtt{0}, \mathtt{1}\}^+, |v| = |w|\} \in \mathscr{L}_{rt}(\mathrm{OCA})$

Ein entsprechender OCA arbeitet z.B. nach dem folgenden Prinzip: Jede Zelle verfügt über zwei Register. In einem wird die Eingabe gespeichert und in jedem zweiten Zeitschritt um eine Zelle nach rechts verschoben. Das andere Register dient zur Realisierung von Signalen. Im ersten Zeitschritt wird vom linken Randautomat ein Signal mit der Geschwindigkeit 1 nach rechts gesandt. Im Zeitpunkt i befindet sich das Signal in der Zelle i, deren „wanderndes“ Eingaberegister den ursprünglichen Eingabezustand der Zelle $\lceil i/2 \rceil$ enthält. So schaltet jede Zelle, die das Signal i in einem ungeraden Zeitschritt empfängt und im Eingaberegister eine $\mathtt{1}$ enthält, in einen Endzustand.

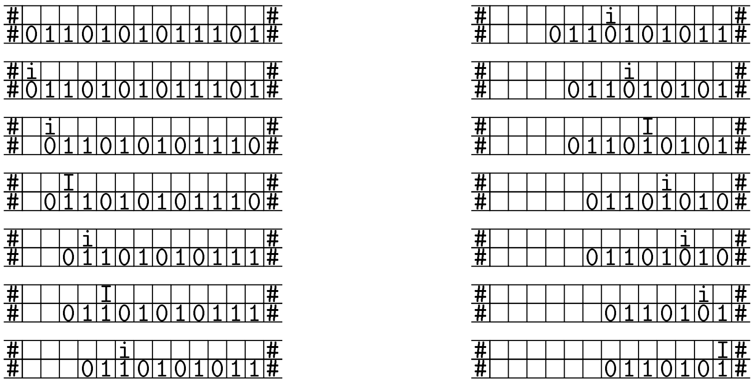


Abbildung 3.16. Schema eines Raum-Zeit-Diagramms zum Beispiel 3.4.

Beispiel 3.5. $\{\mathtt{a}^{\cdot n}\mathtt{b}^{\cdot n} \mid n \in \mathbb{N}\} \in \mathscr{L}_{rt}(\mathrm{OCA})$

(Vgl. Abbildung 3.17.) Im ersten Zeitschritt erkennt die linke Randzelle ihre Position und schaltet in einen ausgezeichneten Zustand $\mathtt{r}$. Anschließend werden alle Zelleninhalte $\mathtt{a}$ und $\mathtt{r}$ in jedem Zeitschritt um eine Zelle nach rechts verschoben. Trifft ein Zustand $\mathtt{a}$ auf eine

Zelle im Zustand `b`, wechselt die Zelle in einen Zustand `c` und das `a` wird somit gelöscht. Trifft der Zustand `r` auf eine Zelle im Zustand `b`, so schaltet diese in einen Endzustand. Auf diese Weise werden Wörter erkannt bzw. abgelehnt, die die gewünschte Form $\mathtt{a}^+\mathtt{b}^+$ aufweisen. Für andere denkbare Eingaben läßt sich der Algorithmus leicht dahingehend verfeinern, daß in einem zusätzlichen Register u.U. Fehlersignale nach rechts gesandt werden, die die Zellen daran hindern, einen Endzustand anzunehmen.

#	a	a	a	a	a	a	b	b	b	b	b	b	#
#	r	a	a	a	a	a	b	b	b	b	b	b	#
#		r	a	a	a	a	c	b	b	b	b	b	#
#			r	a	a	a	a	b	b	b	b	b	#
#				r	a	a	a	c	b	b	b	b	#
#					r	a	a	a	b	b	b	b	#
#						r	a	a	c	b	b	b	#
#							r	a	a	b	b	b	#
#								r	a	c	b	b	#
#									r	a	b	b	#
#										r	c	b	#
#											r	b	#
#												R	#

Abbildung 3.17. Raum-Zeit-Diagramm zum Beispiel 3.5.

Nachdem sich die Sprachfamilien, die durch Zellularautomaten und -räume ohne Beschränkung der zur Verfügung stehenden Zeit festgelegt sind, unmittelbar in die Chomsky-Hierarchie einordnen lassen, wird im folgenden gezeigt, daß dies bei den Realzeit-Sprachfamilien leider nicht der Fall ist.

Es ist für das weitere Vorgehen günstig, noch eine Sprechweise zu vereinbaren: Falls ein eindimensionaler Zellularautomat (oder -raum) vorliegt, dessen Zellen mehrere Register haben, dann wird die Gesamtheit aller ersten Register auch als erste *Spur* bezeichnet. Analog für alle anderen Register.

Satz 3.10. $\mathscr{L}_3 \subset \mathscr{L}_{rt}(\text{OCA})$

Beweis. Es seien $L \in \mathscr{L}_3$ und E ein endlicher Automat, der L akzeptiert. Aus der Definition endlicher Automaten folgt unmittelbar, daß sie in jedem Zeitschritt genau ein Eingabesymbol verbrauchen, also grundsätzlich in Realzeit arbeiten. Zunächst wird ein Realzeit-OCA M konstruiert, der das Verhalten von E simuliert.

Die Zellen von M verfügen über zwei Register. Da jeder Einzelautomat selbst endlicher Automat ist, kann im ersten Register jeweils die Kontrolleinheit von E simuliert werden. Die zweite Spur dient nun dazu, jeder Kontrolleinheit die Eingabe zuzuführen. Auf ihr sei in der Anfangskonfiguration das zu verarbeitende Eingabewort gespeichert. In jedem weiteren Zeitschritt wird ihr Inhalt um eine Zelle nach rechts verschoben. Auf diese Weise erhält die Kontrolleinheit der rechten Randzelle sukzessive die gesamte Eingabe und kann somit E vollständig simulieren. Dabei ist ein Zustand s von M aus der Menge der Endzustände F von M, falls die erste Komponente von s aus der Menge der Endzustände von E ist.

Somit ist gezeigt, daß $\mathscr{L}_3 \subseteq \mathscr{L}_{rt}(\text{OCA})$ ist. Die Echtheit der Inklusion folgt aus Beispiel 3.5, da $\{\mathtt{a}^{\cdot n}\mathtt{b}^{\cdot n} \mid n \in \mathbb{N}\} \notin \mathscr{L}_3$ ist. $\square$

Korollar 3.2. $\mathscr{L}_3 \subset \mathscr{L}_{rt}(\text{CA})$

Nachdem in vorangegangenen Abschnitten unter anderem Möglichkeiten zur Reduktion von Rastern ohne Zeitverlust behandelt wurden, stellt sich nun die Frage, ob Realzeit-CAs überhaupt mächtiger sind als Realzeit-OCAs. Sicherlich gilt: $\mathscr{L}_{rt}(\text{OCA}) \subseteq \mathscr{L}_{rt}(\text{CA})$. Um die Echtheit der Inklusion nachzuweisen, benötigen wir zunächst folgenden Satz (S. R. Seidel (1979)).

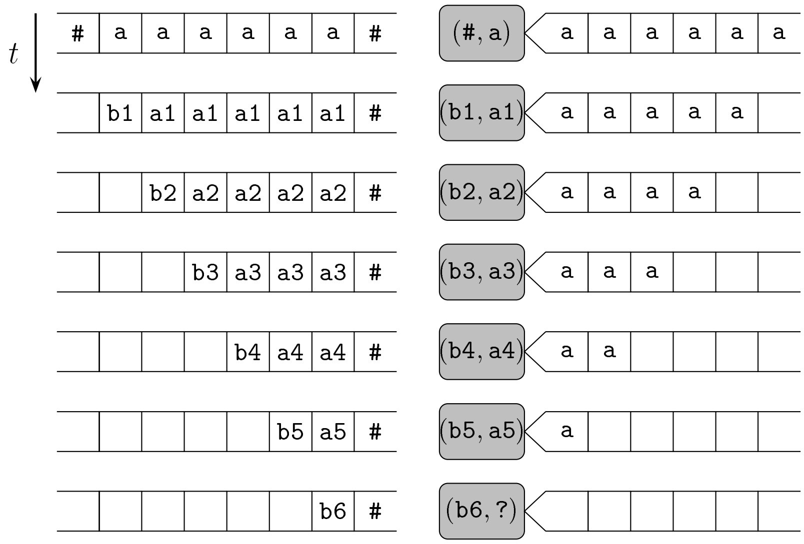
Satz 3.11. Es sei M ein OCA und L eine formale Sprache über einem einelementigen Alphabet. Falls L von M in Realzeit akzeptiert wird, dann ist $L \in \mathscr{L}_3$.

Beweis. Es seien $A = \{\texttt{a}\}$ ein Alphabet, $L \subseteq A^+$ und $M = (S, 1, \bar{H}_1, \sigma, q_0)$ ein OCA, der L mit Endzuständen aus F in Realzeit akzeptiert. Der Nachweis der Behauptung erfolgt durch die Konstruktion eines endlichen Automaten $E = (Q, \Sigma, \delta, s_0, F')$, der L akzeptiert.

$$
\begin{aligned}
&Q := S \times S, \qquad \Sigma := A, \qquad s_0 := (\texttt{\#}, \texttt{a}),\\
&F' := \{(q_1, q_2) \in Q \mid q_1 \in F\},\\
&\forall (q_1, q_2) \in Q:\ \delta((q_1, q_2), \texttt{a}) := (\sigma(q_1, q_2), \sigma(q_2, q_2))
\end{aligned}
$$

Der Nachweis, daß der endliche Automat E das Gewünschte leistet, sei hier anhand zweier sich entsprechender Verhaltensfolgen gegeben:

OCA: Endlicher Automat:



□

Aus dem folgenden Beispiel ergibt sich die Echtheit der Inklusion $\mathscr{L}_{rt}(\text{OCA}) \subset \mathscr{L}_{rt}(\text{CA})$, da die Sprache $\{\mathtt{a}^{\cdot(2^n)} | \, n \in \mathbb{N}_0\}$ nicht regulär und damit nicht in der Familie $\mathscr{L}_{rt}(\text{OCA})$ ist.

Beispiel 3.6. $L = \{\mathtt{a}^{\cdot(2^n)} | \, n \in \mathbb{N}_0\} \in \mathscr{L}_{rt}(\text{CA})$

Der Algorithmus zur Lösung des Problems basiert auf einer Modifikation des ursprünglichen FSSP:
Wenn man davon ausgeht, daß in der Anfangskonfiguration an beiden Enden der Retina ein General vorhanden ist, dann läßt sich die minimale Synchronisationszeit halbieren, d.h., eine Synchronisation in Realzeit ist möglich. Dazu genügt es, die Generäle zu veranlassen, jeweils einen zeitoptimalen Algorithmus in ihrer Hälfte der Retina zu starten. Die Mitte der Retina wird dabei durch diejenigen Einzelautomaten festgelegt, in denen sich die zuerst ausgesandten $S1$-Signale treffen.

Zum Akzeptieren von L wird nun die Eigenschaft des FSSP ausgenutzt, die Retina bzw. einzelne Segmente der Retina laufend zu halbieren. (Die eigentliche Synchronisation ist hier unwesentlich.) Ein Wort w gehört genau dann zur Sprache L, wenn alle Segmente, die während der fortlaufenden Halbierung entstehen, eine gerade Länge haben. (Mit Ausnahme des „letzten“ Segments, das natürlich jeweils nur aus einer einzelnen Zelle besteht.) Ob ein Segment eine gerade Länge hat, läßt sich anhand der Tatsache feststellen, daß dessen „Mitte“ aus zwei Zellen besteht.

Es genügt nun, im Zeitpunkt 0 am linken Rand ein Signal mit Geschwindigkeit 1 nach rechts auszusenden, das auf seinem Weg überprüft, ob bei jeder Teilung zwei Zellen Mittelautomaten (Beförderung zu Generälen) werden. Trifft dieses Signal am rechten Rand ein, kann die Randzelle entsprechend einen Endzustand annehmen oder nicht.

Es sei noch bemerkt, daß bei der Teilung durch das FSSP eigentlich nur jeweils vom rechten Teilungspunkt Gebrauch gemacht wird, da sich das Signal mit Geschwindigkeit 1 nach rechts bewegt. Das FSSP ließe sich also in diesem Fall noch wesentlich vereinfachen. (Dann wäre die Bezeichnung FSSP allerdings nicht mehr zutreffend.)

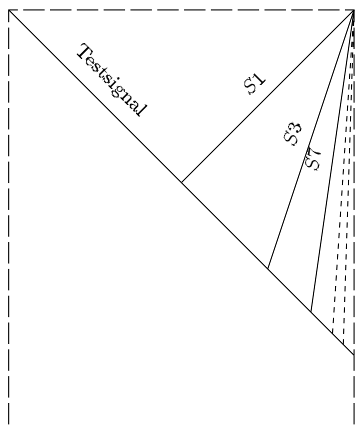


Abbildung 3.18. Schematisches Raum-Zeit-Diagramm zum Beispiel 3.6.

3.5.2 Unidirektional versus bidirektional

Nach dem oben Gesagten kann eine Sprache aus der Familie $\mathscr{L}_{rt}(\mathrm{CA})$ im allgemeinen nicht in Realzeit von OCAs akzeptiert werden.

Es soll nun gezeigt werden, daß bei der Sprachverarbeitung eine additive Konstante der Zeitkomplexität vernachlässigt werden kann, sofern die minimale Bearbeitungszeit — im allgemeinen Realzeit — nicht unterschritten wird (C. Choffrut und K. Culik II (1984)).

Satz 3.12. Es sei $t : \mathbb{N} \longrightarrow \mathbb{N}$ eine Funktion, so daß für alle $n \in \mathbb{N}$ gilt: $t(n) \geq n - 1$. Dann ist $\mathscr{L}_{t(n)}(\mathrm{CA}) = \mathscr{L}_{t(n)+1}(\mathrm{CA})$.

Beweis.
Für jedes $L \in \mathscr{L}_{t(n)}(\mathrm{CA})$ gilt sicherlich auch $L \in \mathscr{L}_{t(n)+1}(\mathrm{CA})$. Es genügt also, die umgekehrte Richtung zu zeigen.

Es sei M ein CA, der eine Sprache L mit Zeitkomplexität $t(n) + 1$ akzeptiert. Die Konstruktion eines CA M', der L mit Zeitkomplexität $t(n)$ akzeptiert, erfolgt derart, daß für die Konfigurationen gilt:

$$c'_0(i) = c_0(i),\ 1 \le i \le n,$$
$$c'_{t+1}(i) = ((c_0(i-t-1), c_{t+1}(i), c_0(i+t+1)), G),\ 1 \le i \le n,\ t \ge 0,$$

wobei $G : S^2 \longrightarrow S$ — die Zustandsmenge von M wird wie üblich mit S bezeichnet — eine Abbildung ist, die für jedes mögliche Paar (x, y) den Zustand $c_{t+2}(i)$ liefert, unter der Voraussetzung, daß $c_0(i-t-2) = x$ und $c_0(i+t+2) = y$ gilt. Eine endliche Abbildung kann in den einzelnen Zellen z.B. als endliche Menge von Tripeln realisiert werden. Auf diese Weise bleibt die Zustandsmenge des entsprechenden Registers stets endlich. Zur Vereinfachung sei noch zusätzlich angenommen:
$\forall i \in \mathbb{N}\backslash\{1, \ldots, n\} : c_0(i) = \#$. Es sei ausdrücklich darauf hingewiesen, daß diese Annahme prinzipiell nicht notwendig ist.

Somit ist die Zustandsmenge von M' festgelegt durch

$$S' := S \cup \{((a, b, c), G) |\ a, b, c \in S\ \wedge\ G \subseteq (S \times S) \times S\}.$$

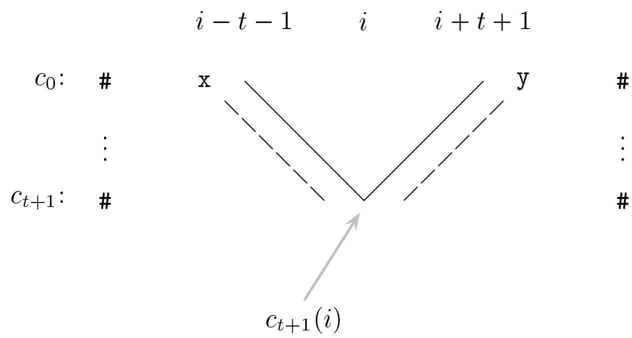


Abbildung 3.19. Schema des Informationsgehalts eines Zustandes von M'.

Das Verfahren der Simulation geht von dem Gedanken aus, daß die einzelnen Zellen von M' diejenigen von M simulieren, dabei aber gleichzeitig Information über deren jeweils mögliche Folgezustände speichern. Legt man nun die Menge der Endzustände derart fest, daß sich eine Zelle von M' in einem Endzustand befindet, falls die entsprechende Zelle von M im nächsten Zeitschritt unter der Voraussetzung, die ganze Eingabe verarbeitet zu haben, in einen Endzustand schalten würde, so leistet die Simulation das Gewünschte. Dabei läßt sich die Voraussetzung durch G ausdrücken:

$$F' := \{((a,b,c),G) |\, a,b,c \in S \,\wedge\, G(\texttt{\#},\texttt{\#}) \in F\}.$$

Mit $H_1 = (-1,0,1)$ und $a,b,c,u_i,v_i,w_i \in S$ für $1 \leq i \leq 3$ wird die Überführungsfunktion $\sigma' : S'^3 \longrightarrow S'$ festgelgt durch

1) $\sigma'(r,\texttt{\#},s) = \texttt{\#}$ für $r,s \in S'$
2) $\sigma'(a,b,c) = ((a,\sigma(a,b,c),c),G)$ mit $a,b,c \in S$, wobei für $G : S^2 \longrightarrow S$ gilt:
   - falls $a = \texttt{\#}$, dann $G(\texttt{\#},x) = \sigma(\texttt{\#},\sigma(a,b,c),\sigma(b,c,x))$ für alle $x \in S$
   - falls $c = \texttt{\#}$, dann $G(x,\texttt{\#}) = \sigma(\sigma(x,a,b),\sigma(a,b,c),\texttt{\#})$ für alle $x \in S$
   - sonst $G(x,y) = \sigma(\sigma(x,a,b),\sigma(a,b,c),\sigma(b,c,y))$ für alle $x,y \in S$
3) $\sigma'(((u_1,u_2,u_3),G),((v_1,v_2,v_3),H),((w_1,w_2,w_3),I)) = ((u_1,\sigma(u_2,v_2,w_2),w_3),K)$, wobei $K(x,y) = \sigma(G(x,v_3),\sigma(u_2,v_2,w_2),I(v_1,y))$ für alle $x,y \in S$.

□

Satz 3.13. Es sei $t : \mathbb{N} \longrightarrow \mathbb{N}$ eine Funktion, so daß für alle $n \in \mathbb{N}$ gilt: $t(n) \geq n-1$. Dann ist $\mathscr{L}_{t(n)}(\text{OCA}) = \mathscr{L}_{t(n)+1}(\text{OCA})$.

Beweis. Analog zum Beweis des voherigen Satzes. □

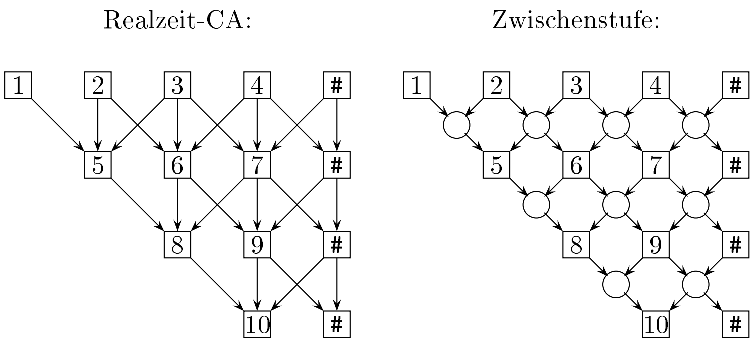
Mit dem obigen Satz kann jetzt eine obere Schranke für die Zeitkomplexität angegeben werden, mit der OCAs Sprachen aus der Familie $\mathscr{L}_{rt}(\text{CA})$ akzeptieren können. Allerdings muß zu diesem Zweck ein klein wenig von der Definition von OCAs abgewichen werden; und zwar wird angenommen, das Ergebnis der Berechnung wird durch den

Zustand der *linken* Randzelle dargestellt. Um dies zu ermöglichen, genügt es, das Raster $(0,1)$ anstelle von $(-1,0)$ zu wählen. Es soll mit $\bar{H}_1^R$ bezeichnet werden. Der Informationsfluß findet also von rechts nach links statt, bleibt aber unidirektional. Diese Annahme ist keineswegs trivial, da bisher unbekannt ist, ob der folgende Satz auch mit einem $\bar{H}_1$-Raster gilt. Wie später noch erläutert wird, liegt dem Problem eine ungeklärte Abschlußeigenschaft der Familie $\mathscr{L}_{2n}(\mathrm{OCA})$ zugrunde. Der Beweis macht von der Technik der *Transformation von Raum-Zeit-Diagrammen* Gebrauch, die auch im weiteren öfter benutzt wird. Detailiertere Ausführungen finden sich in H. Umeo, K. Morita und K. Sugata (1982), C. Choffrut und K. Culik II (1984).

Satz 3.14. Es sei ein $\bar{H}_1^R$-Raster für OCAs angenommen, dann gilt:

$$\mathscr{L}_{2n}(\mathrm{OCA}) = \mathscr{L}_{rt}(\mathrm{CA})$$

Beweis.
„$\Longleftarrow$“ Zum Nachweis $\mathscr{L}_{2n}(\mathrm{OCA}) \supseteq \mathscr{L}_{rt}(\mathrm{CA})$ betrachten wir für jede Sprache L aus der zweiten Familie einen CA, der L mit Zeitkomplexität $n-1$ akzeptiert. (Satz 3.12 sichert jeweils die Existenz.) Dessen Raum-Zeit-Diagramm wird nun schrittweise transformiert:



(2n − 2)-Zeit-OCA:

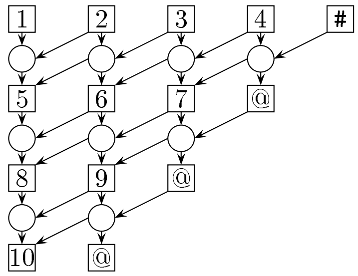


Abbildung 3.20. Schrittweise Transformation des Raum-Zeit-Diagrammes eines CA.

Formal läßt sich der OCA $(S', \sigma', \bar{H}_1^R, q_0')$ aus dem CA (S, σ, H_1, q_0) folgendermaßen konstruieren:

Es seien $\bar{S} := \{\bar{s}|\ s \in S \setminus \{\#\}\}$ und @, $\bar{@}$ zwei verschiedene Symbole, die nicht aus $S \cup \bar{S}$ sind.

$S' := S \cup \{@\} \cup (\bar{S} \cup \{\bar{@}\})^2$
$\bar{H}_1^R := (0, 1)$

$\forall s' \in S' :\ \sigma'(\#, s') := \#$
$\forall a, b \in (S \setminus \{\#\}) \cup \{@\} :$
$\quad \sigma'(a, b) := (\bar{a}, \bar{b})$ und $\sigma'(a, \#) := (\bar{a}, \bar{@})$
$\forall \bar{a}, \bar{b}, \bar{c} \in \bar{S} :$
$\quad \sigma'((\bar{a}, \bar{b}), (\bar{b}, \bar{c})) := \sigma(a, b, c)$ und $\sigma'((\bar{a}, \bar{b}), (\bar{b}, \bar{@})) := \sigma(a, b, \#)$
$\sigma'(x, y) = @$ sonst

„$\Longrightarrow$“ Zum Nachweis $\mathscr{L}_{2n}(\text{OCA}) \subseteq \mathscr{L}_{rt}(\text{CA})$ betrachten wir für jede Sprache L aus der ersten Familie einen OCA, der L mit Zeitkomplexität $2n - 2$ akzeptiert. (Satz 3.13 sichert wieder die Existenz.)
Wird das Raum-Zeit-Diagramm etwas anders aufgeschrieben, so besteht die eigentliche Idee darin, das entstehende Dreieck „in der Mitte“ zu falten. Betrachtet man jede zweite Reihe vom Ergebnis der Faltung, so erhält man das Raum-Zeit-Diagramm eines $(n - 1)$-Zeit CA. Ein zusätzlicher Zeitschritt wird dabei für die Initialisierung benötigt.

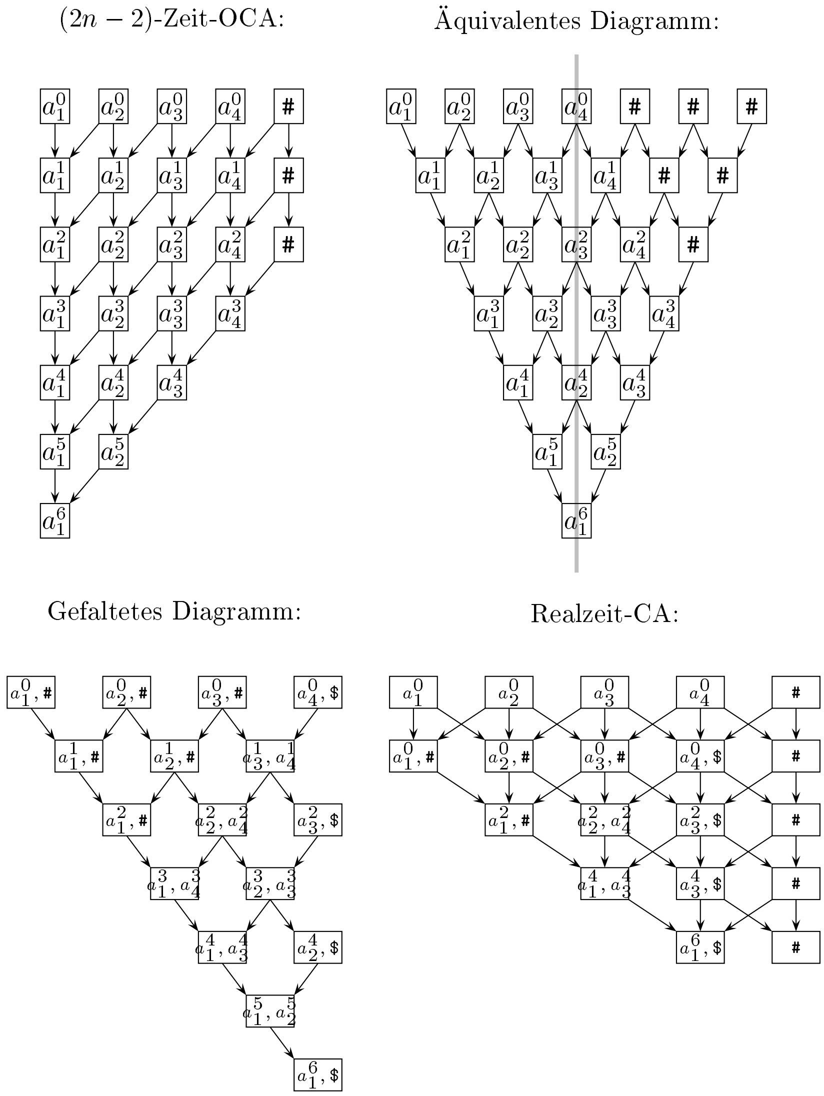


Abbildung 3.21. Transformation und Faltung des Raum-Zeit-Diagrammes eines OCA.

□

Zum Abschluß dieses Abschnittes seien von den vielen, grundlegenden, ungelösten Problemen in diesem Bereich der Automatentheorie (oder anders ausgedrückt: Man weiß relativ wenig) zwei besonders hervorgehoben:

$$\mathscr{L}(\mathrm{CA}) = \mathscr{L}(\mathrm{OCA})?$$

$$\{ww \mid w \in \{0,1\}^+\} \in \mathscr{L}_{rt}(\mathrm{OCA})?$$

3.5.3 Abschlußeigenschaften

Da die Untersuchungen zur Sprachverarbeitung im wesentlichen hinsichtlich der Zeitkomplexität durchgeführt werden, ist es oftmals von Interesse zu wissen, inwieweit Operationen auf den Komplexitätsklassen angewandt werden können, deren Ergebnisse wieder in der Komplexitätsklasse liegen. Diese Abschlußeigenschaften bilden häufig die Grundlage zu Untersuchungen auf einer höheren Ebene, stellen also ein gewisses „Handwerkszeug“ dar.

Die folgenden Sprachoperationen basieren auf mengentheoretischen Operationen. Sie werden häufig auch elementare Sprachoperationen genannt.

Definition 3.14. Es seien A ein Alphabet und $L \subseteq A^*$ eine formale Sprache. Die Sprachoperation

$$COM(L) := \Big\{\{w \in A^+ \mid w \notin L\}\Big\}$$

erzeugt das *Komplement* von L.

Gelegentlich wird auch die Schreibweise $\overline{L}$ für $COM(L)$ benutzt.

Satz 3.15. $\mathscr{L}_{rt}(\mathrm{OCA})$ ist abgeschlossen bzgl. COM.

Beweis. Es ist zu zeigen, daß für jede Sprache $L \in \mathscr{L}_{rt}(\mathrm{OCA})$ ihr Komplement wieder in $\mathscr{L}_{rt}(\mathrm{OCA})$ liegt. Dies geschieht durch die Konstruktion eines Realzeit-OCA für $\overline{L}$ aus demjenigen für L. Hierzu genügt die Festlegung $F' := S \setminus F$. Es wird also lediglich die Menge der Endzustände derart festgelegt, daß alle Wörter, die vorher nicht akzeptiert wurden, jetzt akzeptiert werden. □

Satz 3.16. $\mathscr{L}(\mathrm{CA})$ und $\mathscr{L}_{rt}(\mathrm{CA})$ sind abgeschlossen bzgl. COM.

Beweis. Der Beweis für $\mathscr{L}_{rt}(\mathrm{CA})$ entspricht demjenigen von Satz 3.15.

Da bei $\mathscr{L}(\mathrm{CA})$ nicht unbedingt ein fester Zeitpunkt existiert, in dem das Ergebnis vorliegt, wurde verlangt, daß sich Endzustände erhalten. Dies bedeutet, daß eine Zelle, die einen Zustand aus der Menge F angenommen hat, in allen weiteren Zeitpunkten ebenfalls wieder Endzustände annehmen wird.

t

#	0	0	0	0	0	0	#
#	1	0	0	0	0	0	#
#	2	0	0	0	0	0	#
#	0	1	0	0	0	0	#
#	1	1	0	0	0	0	#
#	2	1	0	0	0	0	#
#	0	2	0	0	0	0	#
#	1	2	0	0	0	0	#
#	2	2	0	0	0	0	#
#	0	0	1	0	0	0	#
#	1	0	1	0	0	0	#
#	2	0	1	0	0	0	#
#	0	1	1	0	0	0	#

Abbildung 3.22. Realisierung eines tertiären Zählers.

Die Abgeschlossenheit läßt sich nun mit folgender Konstruktion nachweisen: Ein CA mit der Zustandsmenge S kann bei Eingabe eines Wortes w mit der Länge n maximal $|S|^n$ verschiedene Konfigurationen annehmen. Aufgrund der deterministischen Überführungsfunktion wird sein Verhalten anschließend zyklisch. Verfügt der CA für das Komplement nun über eine zweite Spur, auf der ein Zähler im $|S|$-adischen Zahlensystem realisiert wird, kann er den Zeitpunkt $|S|^n$ daran erkennen, daß in der rechten Randzelle ein Überlauf stattfindet, und einen Endzustand annehmen, falls die Eingabe bis dahin nicht akzeptiert wurde bzw. umgekehrt. □

Definition 3.15. Es seien $\mathscr{L}$ eine Sprachfamilie und $L_1, L_2 \in \mathscr{L}$. Die Sprachoperationen

$$\begin{aligned} UNI(L_1, L_2) &:= \{L_1 \cup L_2\} \\ INT(L_1, L_2) &:= \{L_1 \cap L_2\} \\ DIF(L_1, L_2) &:= \{L_1 \setminus L_2\} \end{aligned}$$

werden respektive *Vereinigung*, *Durchschnitt* und *Mengendifferenz* genannt.

Satz 3.17. $\mathscr{L}_{rt}$(OCA) ist abgeschlossen bzgl. UNI.

Beweis. Die Beweisidee folgt den vorangegangenen Sätzen. Ein entsprechender OCA simuliert auf zwei Spuren OCAs für die beiden Sprachen. Die Menge der Endzustände wird dann so festgelegt, daß der Automat die Eingabe akzeptiert, falls sie auf einer der beiden Spuren akzeptiert wird. □

Auch dieser Satz gilt wieder für CAs:

Satz 3.18. $\mathscr{L}$(CA) und $\mathscr{L}_{rt}$(CA) sind abgeschlossen bzgl. UNI.

Satz 3.19. $\mathscr{L}$(CA), $\mathscr{L}_{rt}$(CA) und $\mathscr{L}_{rt}$(CA) sind jeweils abgeschlossen bzgl. INT und DIF.

Beweis. Es seien L_1 und L_2 beide jeweils aus einer der Sprachfamilien.

Die Abgeschlossenheit bzgl. DIF ergibt sich aus

$$DIF(L_1, L_2) = COM(UNI(COM(L_1), L_2)),$$

diejenige bzgl. INT aus

$$INT(L_1, L_2) = COM(UNI(COM(L_1), COM(L_2))).$$

□

Definition 3.16.

a) Es seien A ein Alphabet und $w = w_1 \cdots w_n \in L \subseteq A^+$; dann heißt $m(w)$,

$$\begin{aligned} m : A^+ &\longrightarrow A^+ \\ w &\mapsto w_n w_{n-1} \cdots w_1, \end{aligned}$$

Spiegelbild von w.

b) $MIR(L) := \Big\{\{m(w) | \; w \in L\}\Big\}$ formalisiert die entsprechende Sprachoperation.

Gelegentlich wird auch die Schreibweise w^R bzw. L^R für $m(w)$ und $MIR(L)$ benutzt.

Die letzte Abschlußeigenschaft in diesem Abschnitt gewinnt an Bedeutung, da sie für CAs nicht geklärt ist.

Satz 3.20. $\mathscr{L}_{rt}$(OCA) ist abgeschlossen bzgl. MIR.

Beweis. Nach Satz 3.13 existiert zu jedem Realzeit-OCA M stets ein $(n-1)$-Zeit-OCA M', der dieselbe Sprache L akzeptiert. Betrachtet man nun das Raum-Zeit-Diagramm von M', so stellt man fest, daß das Ergebnis der Berechnung unabhängig von Zellen im Grenzzustand ist. Zur Konstruktion eines OCA M'', der $MIR(L)$ akzeptiert, genügt es also, die Argumente der Überführungsfunktion zu vertauschen. Dabei ist zu beachten, daß die „neue“ Funktion für ein Argument, das den Grenzzustand beinhaltet, entweder nicht definiert wird oder einen beliebigen Zustand liefert. (Die Berechnung ist ja wieder unabhängig von derartigen Werten.)

$t \downarrow$

#	a_1^0	a_2^0	a_3^0	a_4^0	a_5^0	#
#		a_2^1	a_3^1	a_4^1	a_5^1	#
#			a_3^2	a_4^2	a_5^2	#
#				a_4^3	a_5^3	#
#					a_5^4	#

#	a_5^0	a_4^0	a_3^0	a_2^0	a_1^0	#
#		a_5^1	a_4^1	a_3^1	a_2^1	#
#			a_5^2	a_4^2	a_3^2	#
#				a_5^3	a_4^3	#
#					a_5^4	#

#	a_1^0	a_2^0	a_3^0	a_4^0	a_5^0	#
#	a_2^1	a_3^1	a_4^1	a_5^1		#
#	a_3^2	a_4^2	a_5^2			#
#	a_4^3	a_5^3				#
#	a_5^4					#

Abbildung 3.23. Raum-Zeit-Diagramme zum Satz 3.20.

Formal läßt sich $M'' = (S'', 1, \bar{H}_1, \sigma'', q_0'')$ aus $M' = (S', 1, \bar{H}_1, \sigma', q_0')$ folgendermaßen konstruieren:

$$
\begin{array}{l}
S'' = S', \qquad \bar{H}_1 = (-1, 0), \qquad q_0'' = q_0' \\
\forall a \in S' : \ \sigma''(a, \#) = \# \\
\forall a \in S' : \ \sigma''(\#, a) = a \\
\forall a, b \in S' \setminus \{\#\} : \ \sigma''(a, b) = \sigma'(b, a)
\end{array}
$$

□

Die Abgeschlossenheit von Realzeit-OCAs bzgl. MIR kann auf zweierlei Arten interpretiert werden: Zum einen läßt sich ein OCA finden, der das Spiegelbild der Sprache akzeptiert, d.h., jedes Wort wird vor der Eingabe gespiegelt. Zum anderen läßt sich ein OCA finden, der die gleiche (ungespiegelte) Sprache am linken Rand akzeptiert.

Für den Beweis ist die Tatsache wesentlich, daß die Berechnung unabhängig von Zellen im Grenzzustand ist. Andererseits müßte bei der Definition der neuen Überführungsfunktion auf Argumente, die den Grenzzustand enthalten, derart Rücksicht genommen werden, daß keine relevante Information verlorengeht. Dies bedeutete aber für einzelne Zellen die Möglichkeit, den Grenzzustand zu verlassen. Damit wäre dann der benötigte Raum nicht mehr auf den Eingabebereich eingeschränkt.

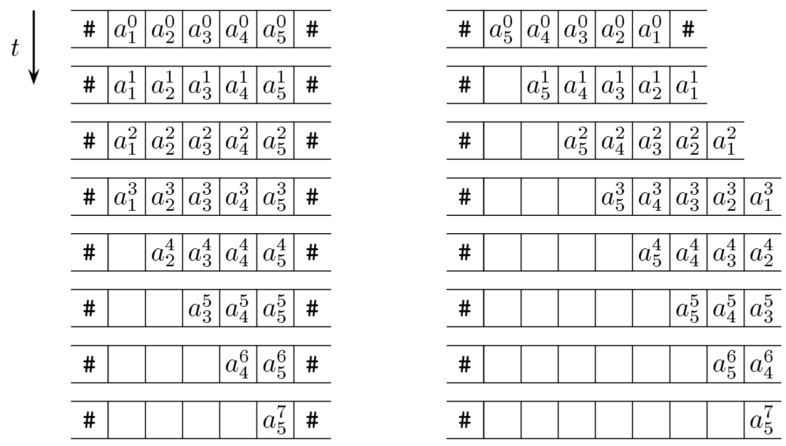


Abbildung 3.24. Raum-Zeit-Diagramm einer unmöglichen Berechnung.

Das Problem, ob

$$\mathscr{L}_{lt}(\text{OCA}) \text{ bzgl. } MIR \text{ abgeschlossen}$$

ist, ist bis heute ungelöst. Daraus ergibt sich auch die Notwendigkeit, in Satz 3.13 zu fordern, daß der $2n$-Zeit-OCA seine Eingabe am linken Rand akzeptiert. Sollte das Problem negativ geklärt werden, so hätte man zwischen zwei verschiedenen OCA-Varianten zu unterscheiden, da sie verschiedene Sprachfamilien zu akzeptieren vermögen.
Nach Satz 3.13 sind $2n$-Zeit-OCAs, die am linken Rand akzeptieren, mit Realzeit-CAs, die am rechten Rand akzeptieren, äquivalent. So-

mit ist das Problem identisch mit der Frage, ob

$$\mathscr{L}_{rt}(\mathrm{CA}) \text{ bzgl. } MIR \text{ abgeschlossen}$$

ist. Da heute jedoch noch nicht einmal bekannt ist, ob die Inklusion $\mathscr{L}_{rt}(\mathrm{CA}) \subseteq \mathscr{L}_{lt}(\mathrm{CA})$ echt ist, gewinnt das Problem an Bedeutung. Eine negative Lösung hätte dann auch eine echte Inklusion zur Folge: Eine Sprache $L \in \mathscr{L}_{rt}(\mathrm{CA})$, deren Spiegelbild nicht in $\mathscr{L}_{rt}(\mathrm{CA})$ liegt, könnte von einem CA in $2n$-Zeit akzeptiert werden. Der CA würde seine Eingabe in n Zeitschritten zunächst spiegeln und dann den ursprünglichen CA simulieren.

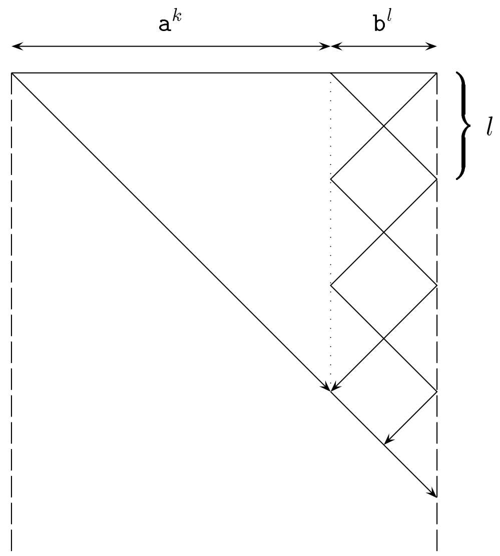


Abbildung 3.25. Schema eines Raum-Zeit-Diagrammes für $L = \{\mathtt{a}^{\cdot k}\mathtt{b}^{\cdot l} \mid k, l \in \mathbb{N} \ \wedge \ l \text{ teilt } k\}$.

Die Sprache $L = \{\mathtt{a}^{\cdot k}\mathtt{b}^{\cdot l} \mid k, l \in \mathbb{N} \ \wedge \ l \text{ teilt } k\}$ war längere Zeit ein Kandidat für den Nachweis der Nichtabgeschlossenheit. Ein CA, der L akzeptiert, könnte z.B. nach folgendem Prinzip arbeiten: Im b-Bereich der Eingabe werden zwei Signale mit Geschwindigkeit 1 realisiert, die an den Bereichsrändern jeweils reflektiert werden. Am linken Rand der Eingabe wird im Zeitpunkt 0 ein Signal mit Geschwindigkeit 1 nach rechts ausgesandt, das genau dann am Übergang von a-Bereich und b-Bereich auf eines der beiden anderen Signale trifft, wenn die Eingabe zur Sprache gehört.

Es wurde vermutet, daß $MIR(L)$ nicht in $\mathscr{L}_{rt}(\mathrm{CA})$ liegt. O. H. Ibarra und T. Jiang (1988) haben diese Vermutung jedoch widerlegt und mit der Sprache

$$\{x^{\cdot n}\texttt{\#}x \mid n \geq 1, x \in \{\texttt{0},\texttt{1}\}^{+}\}$$

einen „komplizierteren" neuen Kandidaten vorgeschlagen.

4 Iterative Arrays

Bei den Zellularräumen wurde bisher davon ausgegangen, die Eingabe liege im Zeitpunkt 0 bereits in den einzelnen Zellen vor. Eine anschauliche Erklärungsmöglichkeit ergibt sich aus der Vorstellung, das zu verarbeitende Muster werde in einem Zeitschritt parallel in den Automaten eingegeben, wobei jede Zelle über eine Verbindung zur Außenwelt verfügt. In der Realität wird es aber häufig vorkommen, daß die zu verarbeitenden Muster sequentiell an das Verarbeitungsgerät herangeführt werden.
Es ist nun naheliegend, Zellularräume unter diesem Aspekt zu betrachten. S. N. Cole (1966) hat in diesem Zusammenhang die iterativen Arrays vorgeschlagen. Das Modell geht davon aus, daß sich in der Anfangskonfiguration alle Zellen im Ruhezustand befinden, wobei eine von ihnen im Laufe der Berechnung sequentiell die Eingabe über eine zusätzliche Verbindung von außen erhält. Hierbei wird dann allerdings die Homogenität des Raumes (an einer Stelle) verletzt.

4.1 Definitionen

Definition 4.1. Ein 7-Tupel $(S, \Sigma, d, N, \sigma, q_0, \texttt{@})$ heißt *iterativer Array*, falls gilt:

a) S ist eine endliche, nichtleere Zustandsmenge.
b) Σ ist eine endliche, nichtleere Menge von Eingabesymbolen mit der Eigenschaft $S \cap \Sigma = \emptyset$.
c) $d \in \mathbb{N}_0$ ist die Raumdimension.
d) N ist d-dimensionaler Nachbarschaftsindex vom Grade n.
e) $\sigma : S^n \cup (S^n \times (\Sigma \cup \{\texttt{@}\})) \longrightarrow S$ ist *(lokale) iterative Überführungsfunktion*.
f) $q_0 \in S$ ist der Ruhezustand, für den gilt: $\sigma(q_0, \ldots, q_0) = q_0$.
g) $\texttt{@} \notin \Sigma \cup S$ ist das Eingabeendesymbol.

Ein iterativer Array entspricht also einem Zellularraum, mit dem

Unterschied, daß die Eingabe-Zelle (dies wird im weiteren immer die Zelle im Ursprung des $\mathbb{Z}^d$ sein) ihren Zustand außer von den Zuständen ihrer Nachbarn auch von der Eingabe aus der Außenwelt abhängig machen kann. Das Symbol @ wird als Füllsymbol benutzt, das am Ende des jeweiligen Eingabewortes als externe Dauereingabe anliegt.

Bevor festgelegt werden kann, wie formale Sprachen in iterativen Arrays verarbeitet werden, muß zunächst geklärt werden, wie ein Konfigurationsübergang bewerkstelligt wird.

Definition 4.2. Ein Paar (w, c_t) heißt Konfiguration eines iterativen Array oder *iterative Konfiguration* im Zeitpunkt $t \in \mathbb{N}_0$, falls $w \in \Sigma^*$ das noch zu lesende Eingabewort und $c_t : \mathbb{Z}^d \longrightarrow S$ eine Konfiguration im Sinne von Zellularräumen ist.

Eine Konfiguration beschreibt wieder den Gesamtzustand des Raumes in einem bestimmten Zeitpunkt t, wobei zusätzlich die restliche Eingabe berücksichtigt wird. Falls extern nur noch das Füllsymbol anliegt, sagen wir, die restliche Eingabe sei das leere Wort ε.

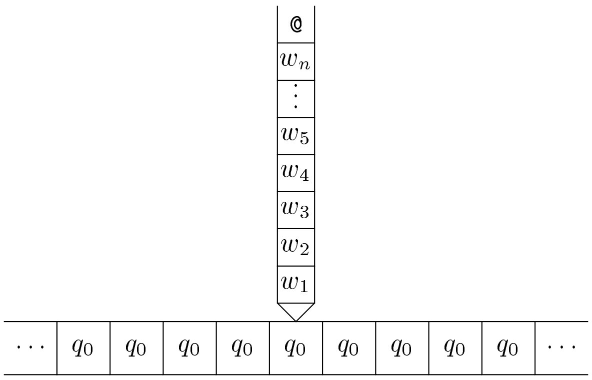


Abbildung 4.1. Die Anfangskonfiguration eines iterativen Arrays.

Definition 4.3.
Es seien S und Σ disjunkte Alphabete, $N = (a_1, \ldots, a_n)$ ein d-dimensionaler Nachbarschaftsindex und C_I die Menge aller iterativen Konfigurationen der Art $(w = w_1 \cdots w_n, c : \mathbb{Z}^d \longrightarrow S), w \in \Sigma^*$. Eine Abbildung $\mathcal{T}_I : C_I \longrightarrow C_I$ heißt *iterative Transformation*, wenn sie durch eine (lokale) iterative Überführungsfunktion σ folgendermaßen induziert wird:

$$\forall (w,c) \in C_I : \left(\mathcal{T}_I((w,c)) = (w',c') \iff \right.$$

$$\left. \begin{array}{c} w' = \begin{cases} w_2 \cdots w_n & \text{falls } w \neq \varepsilon \\ \varepsilon & \text{sonst} \end{cases} \\ \forall i \in \mathbb{Z}^d : c'(i) = \begin{cases} \sigma(c(i+a_1), \ldots, c(i+a_n)) & \text{falls } i \neq 0 \\ \sigma(c(i+a_1), \ldots, c(i+a_n), w_1) & \text{falls } i = 0 \\ & \wedge\, w \neq \varepsilon \\ \sigma(c(i+a_1), \ldots, c(i+a_n), @) & \text{falls } i = 0 \\ & \wedge\, w = \varepsilon \end{cases} \end{array} \right)$$

Damit kann ein Konfigurationsübergang naheliegend durch eine Anwendung der iterativen Transformation beschrieben werden.

Im folgenden bezeichnet $\mathcal{T}_I$ immer die durch σ induzierte iterative Transformation.

Definition 4.4. Es seien eine formale Sprache $L \subseteq A^+$ und eine Funktion $t : \mathbb{N} \longrightarrow \mathbb{N}$ gegeben. L wird durch einen iterativen Array $M = (S, A, 1, N, \sigma, q_0, @)$ mit einer Menge von Endzuständen $F \subseteq S$ genau dann mit der Zeitkomplexität t akzeptiert, wenn

$$L = \{w \in A^+ | \mathcal{T}_I^{t(|w|)}((w, c_0(i) = q_0)) = (\varepsilon, c_{t(|w|)}) \text{ mit } c_{t(|w|)}(0) \in F\}.$$

Das Ergebnis der Berechnung wird durch den Zustand der Zelle 0 repräsentiert. Sie ist damit sowohl bzgl. der Eingabe als auch bzgl. der Ausgabe von besonderer Bedeutung. Zu beachten ist ferner, daß der zur Verfügung stehende Raum nicht beschränkt ist.

Definition 4.5. Die Familie der von iterativen Arrays mit H_1-Raster und Zeitkomplexität t akzeptierbaren Sprachen wird mit $\mathscr{L}_t(\mathrm{IA})$ bezeichnet.

Aufgrund des Eingabe/Ausgabe-Verhaltens ergeben sich für iterative Arrays Raum-Zeit-Diagramme, die nur „relevante“ Information darstellen, folgender Art:

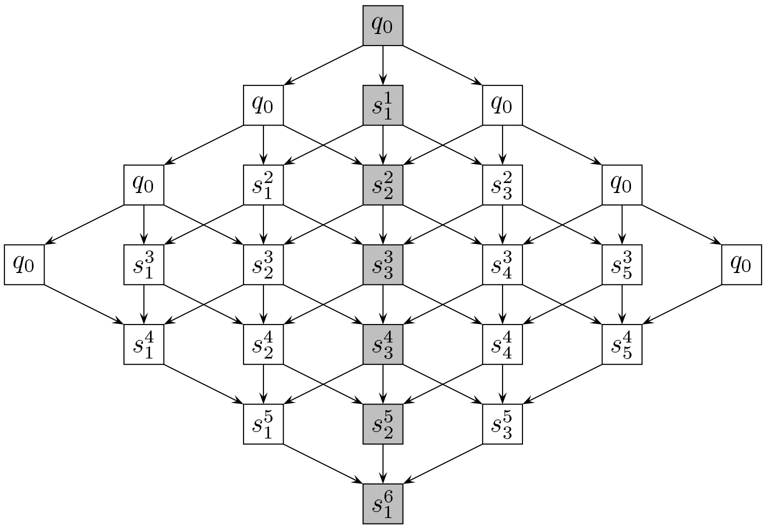


Abbildung 4.2. Raum-Zeit-Diagramm eines iterativen Arrays.

4.2 Vergleich mit Zellularräumen

Da der zur Verfügung stehende Raum bei iterativen Arrays nicht beschränkt ist, liegt die Vermutung nahe, daß es sich um ein berechnungsuniverselles Modell handelt. Dies folgt aus Satz 4.1 und dem (bereits nachgewiesenen) Zusammenhang $\mathscr{L}(\mathrm{CS}) = \mathscr{L}_0$.

Satz 4.1. $\mathscr{L}(\mathrm{IA}) = \mathscr{L}(\mathrm{CS})$

Beweis.
„$\Longrightarrow$“ folgt einerseits aus der Berechnungsuniversalität der Zellularräume, kann andererseits aber auch folgendermaßen nachgewiesen werden:
Ein Zellularraum kann einen IA simulieren, indem er die (bereits vollständig vorliegende) Eingabe auf einer Spur in jedem Zeitschritt um eine Zelle nach rechts verschiebt, während auf einer zweiten Spur die Arbeitsweise der Zellen des IA simuliert werden. Der Einzelautomat am rechten Rand der Anfangskonfiguration entspricht dabei der Ein/Ausgabe-Zelle.

„$\Longleftarrow$“ Die Simulation eines Zellularraumes durch einen IA erfolgt in zwei Phasen. Zuerst wird die (endliche) Eingabe sukzessive eingelesen und auf entsprechend viele zusammenhängende einzelne Zellen verteilt. In der zweiten Phase, die z.B. nach einer Synchronisation mittels FSSP gleichzeitig gestartet werden kann, arbeitet der IA unabhängig von der Außenwelt wie der Zellularraum. □

4.3 Unvergleichbarkeit der Familien $\mathscr{L}_{rt}$(IA) und $\mathscr{L}_{rt}$(OCA)

Ziel dieses Abschnittes ist der Nachweis, daß die Familie der Realzeit-IA Sprachen nicht mit der Familie der Realzeit-OCA Sprachen vergleichbar ist. Zunächst wird anhand eines Beispiels eine Sprache vorgestellt, die in der ersten, nicht aber in der zweiten Familie liegt (C. Choffrut und K. Culik II (1984)).

Eine Teilfamilie der kontextfreien Sprachen wird anschließend dazu benutzt, eine Sprache aus der zweiten Familie zu charakterisieren, die von keinem iterativen Array (beliebiger Dimension und mit beliebiger Nachbarschaft) in Realzeit akzeptiert werden kann.

Beispiel 4.1. $\{\mathsf{a}^{\cdot(2^n)} \mid n \in \mathbb{N}\} \in \mathscr{L}_{rt}(\mathrm{IA})$

Die Ein/Ausgabe-Zelle liest in jedem Zeitschritt ein Eingabesymbol. Im ersten Zeitschritt sendet sie ein Signal mit der Geschwindigkeit

1/3 nach rechts. Anschließend sendet sie ein zweites Signal mit der Geschwindigkeit 1 in dieselbe Richtung, welches zwischen dem ersten (langsamen) Signal und der Ein/Ausgabe-Zelle pendelt. Der Zeitraum zwischen zwei Eintreffen in der Ein/Ausgabe-Zelle wird auf diese Weise jeweils verdoppelt.
Eine Eingabe wird genau dann akzeptiert, wenn sich das (schnelle) Signal in dem Zeitpunkt in der Ein/Ausgabe-Zelle befindet, in dem diese das letzte Eingabesymbol gelesen hat. Ein entsprechender IA benutzt somit nur einen Halbraum.

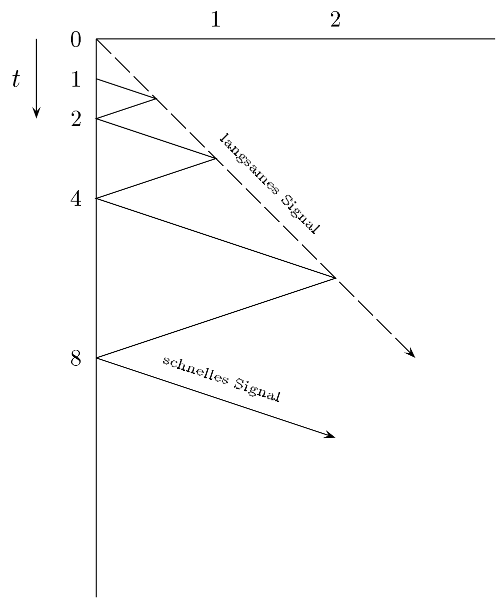


Abbildung 4.3. Schema des Raum-Zeit-Diagrammes zum Beispiel 4.1.

Daß sich die Zeiträume jeweils verdoppeln, kann per vollständiger Induktion nachgewiesen werden, wobei es ausreicht zu zeigen, daß sich beide Signale jeweils in den Zellen $2^n, n \in \mathbb{N}$, treffen:

Die Überführungsfunktion (s.u.) wird so festgelegt, daß beide Signale im Zeitpunkt 3 in der Zelle 1 sind.
Es sei jetzt angenommen, die Signale befinden sich gemeinsam in der Zelle i. Wenn die Zelle des nächsten Treffens mit j bezeichnet wird, benötigt das schnelle Signal $i + j$ weitere Zeitschritte, um die Zelle j zu erreichen (es muß ja zunächst zurück). In dieser Zeit bewegt sich das langsame Signal $\frac{(i+j)}{3}$ Zellen weiter. Es gilt also

$$j = i + \frac{(i+j)}{3} \text{ und somit } j = 2i.$$

Formal läßt sich der IA festlegen durch

$S := \{(i,j) \mid i \in \{-1,0,1\}, j \in \{0,1,2,3\}\} \cup \{q_0\}$,

wobei die erste Komponente für das schnelle Signal steht (1, falls es von links nach rechts läuft, -1 anderenfalls) und die zweite für das langsame Signal (1,2 und 3 realisieren die Geschwindigkeit 1/3).

$\Sigma := \{\mathtt{a}\}$, $\quad H_1 = (-1,0,1)$, $\quad F := \{(1,j) \mid j \in \{0,1,2,3\}\}$.

Für alle im folgenden nicht aufgeführten Fälle ändert sich der Zustand nicht.

Initialisierung:
$\sigma((q_0,q_0,q_0),\mathtt{a}) := (1,2)$
$\sigma((q_0,(1,2),q_0),\mathtt{a}) := (1,3)$
$\sigma((q_0,(1,3),q_0),\mathtt{a}) := (0,0)$

Das schnelle Signal wird von der Ein/Ausgabe-Zelle reflektiert:
$\sigma((q_0,(0,0),(-1,j)),\mathtt{a}) := (1,0)$, falls $j \in \{0,1,2,3\}$
$\sigma((q_0,(1,0),x),\mathtt{a}) := (0,0)$, falls $x \in S$

Das schnelle Signal bewegt sich nach rechts:
$\sigma((1,0),(0,0),x) := (1,0)$, falls $x \in S$
$\sigma(x,(1,0),y) := (0,0)$, falls $x,y \in S$

Eingabe

$t = 0$:	$\cdots$	q_0	q_0	q_0	$\cdots$					
$t = 1$:	$\cdots$	q_0	1,2	q_0	$\cdots$					
$t = 2$:	$\cdots$	q_0	1,3	q_0	$\cdots$					
	$\cdots$	q_0	0,0	-1,1	q_0	$\cdots$				
$t = 4$:	$\cdots$	q_0	1,0	0,2	q_0	$\cdots$				
	$\cdots$	q_0	0,0	1,3	q_0	$\cdots$				
	$\cdots$	q_0	0,0	0,0	-1,1	q_0	$\cdots$			
	$\cdots$	q_0	0,0	-1,0	0,2	q_0	$\cdots$			
$t = 8$:	$\cdots$	q_0	1,0	0,0	0,3	q_0	$\cdots$			
	$\cdots$	q_0	0,0	1,0	0,0	0,1	q_0	$\cdots$		
	$\cdots$	q_0	0,0	0,0	1,0	0,2	q_0	$\cdots$		
	$\cdots$	q_0	0,0	0,0	0,0	1,3	q_0	$\cdots$		
	$\cdots$	q_0	0,0	0,0	0,0	0,0	-1,1	q_0	$\cdots$	
	$\cdots$	q_0	0,0	0,0	0,0	-1,0	0,2	q_0	$\cdots$	
	$\cdots$	q_0	0,0	0,0	-1,0	0,0	0,3	q_0	$\cdots$	
	$\cdots$	q_0	0,0	-1,0	0,0	0,0	0,0	0,1	q_0	$\cdots$
$t = 16$:	$\cdots$	q_0	1,0	0,0	0,0	0,0	0,0	0,2	q_0	$\cdots$

Abbildung 4.4. Anfang eines Raum-Zeit-Diagrammes zum Beispiel 4.1.

Das langsame Signal bewegt sich nach rechts:
$\sigma((1,3), q_0, q_0) := (-1, 1)$
$\sigma((i,3), q_0, q_0) := (0, 1)$, falls $i \neq 1$
$\sigma(x, (i,3), q_0) := (0, 0)$, falls $i \in \{-1, 0, 1\}$ und $x \neq (1, 0)$
$\sigma(x, (i,j), q_0) := (0, j+1)$, falls $j \in \{1, 2\}$, $i \in \{-1, 0, 1\}$ und $x \neq (1, 0)$

Das schnelle Signal bewegt sich nach links:
$\sigma(x, (0,0), (-1, j)) := (-1, 0)$, falls $x \in S \setminus \{q_0\}$, $j \in \{0, 1, 2, 3\}$
$\sigma(x, (-1, 0), y) := (0, 0)$, falls $x, y \in S$

Das schnelle Signal wird vom langsamen reflektiert:
$\sigma((1, 0), (0, j), q_0) := (1, j + 1)$, $j \in \{1, 2\}$
$\sigma((1, 0), (0, 3), q_0) := (1, 0)$
$\sigma(x, (-1, j), q_0) := (0, j + 1)$, falls $j \in \{1, 2\}$ und $x \in S$
$\sigma(x, (-1, 3), q_0) := (0, 0)$, falls $x \in S$

Da $L = \{\mathtt{a}^{\cdot(2^n)} \mid n \in \mathbb{N}\}$ keine reguläre Sprache ist (L ist kontextsensitiv), folgt mit Satz 3.11:

Korollar 4.1. $\exists L \in \mathscr{L}_{rt}(\mathrm{IA}) : L \notin \mathscr{L}_{rt}(\mathrm{OCA})$

Im weiteren werden die linearen kontextfreien Sprachen benötigt. Durch sie läßt sich eine Teilfamilie von $\mathscr{L}_{rt}$(OCA) durch Grammatiken charakterisieren (A. R. Smith III (1970)). An der Definition läßt sich unmittelbar ablesen, daß die linearen kontextfreien Sprachen in den kontextfreien enthalten sind. (Die Inklusion ist sogar echt.)

Definition 4.6. Eine Grammatik $G = (V_N, V_T, X_0, F)$ heißt *linear kontextfrei*, falls jede Produktion in F von einer der Formen $X \longrightarrow a$, $X \longrightarrow Ya$ oder $X \longrightarrow aY$ ist, wobei $X, Y \in V_N$ und $a \in V_T$ gilt.

Die von linear kontextfreien Grammatiken erzeugten Sprachen sollen entsprechend wieder *linear kontextfreie Sprachen* heißen und mit $\mathscr{L}_{2,lin}$ bezeichnet werden.

Satz 4.2. $\mathscr{L}_{2,lin} \subseteq \mathscr{L}_{rt}(\mathrm{OCA})$

Beweis.
Es seien $G = (V_N, V_T, X_0, F)$ eine linear kontextfreie Grammatik, $X, Y \in V_N$ und $w = w_1 w_2 \cdots w_n \in L(G)$, also $w_i \in V_T$ für $1 \leq i \leq n$.

Es gilt:
Wenn $X \Rightarrow^* w_i \cdots w_j, \quad 1 \leq i < j \leq n,$
dann $\exists Y : ((Y \Rightarrow^* w_i \cdots w_{j-1} \ \wedge \ X \Rightarrow Y w_j) \ \vee$
$(Y \Rightarrow^* w_{i+1} \cdots w_j \ \wedge \ X \Rightarrow w_i Y)).$

Darauf aufbauend lassen sich folgende Mengen festlegen:
$N(i,i) := \{X \in V_N \mid (X \to w_i) \in F\} \quad 1 \leq i \leq n$
$N(i,j) := N_1(i,j) \cup N_2(i,j), \quad 1 \leq i < j \leq n$, mit
$N_1(i,j) := \{X \in V_N \mid (X \to Y w_j) \in F \ \wedge \ Y \in N(i, j-1)\}$
$N_2(i,j) := \{X \in V_N \mid (X \to w_i Y) \in F \ \wedge \ Y \in N(i+1, j)\}$

Das Wort $w_1 \cdots w_n$ gehört also genau dann zur Sprache $L(G)$, wenn $X_0 \in N(1,n)$ gilt.

Ein OCA $M = (S, 1, \bar{H}_1, \sigma, q_0)$ für $L(G)$ analysiert seine Eingabe $w_1 \cdots w_n$, indem jede Zelle nacheinander gewisse Mengen $N(i,j)$ berechnet.

Im ersten Zeitschritt berechnen alle Zellen i den Zustand $(N(i,i), (w_i, w_i))$. Daraus ergibt sich auch die mindestens notwendige Zustandsmenge $S = \mathscr{P}(V_N) \times V_T^2$.
Im Zeitschritt k berechnen alle Zellen i $(N(i-k+1, i), (w_{i-k+1}, w_i))$, falls $k \leq i$; anderenfalls ändert sich ihr Zustand nicht.
Ein derartiges Verhalten ist möglich, weil jede Zelle in ihrer Nachbarschaft über die Information $(N(i-k+1, i-1), (w_{i-k+1}, w_{i-1}))$ (im linken Nachbarn) und $(N(i-k+2, i), (w_{i-k+2}, w_i))$ (eigener Zustand) verfügt.
Also berechnet die rechte Randzelle im Zeitschritt $t = n$ auch $N(1,n)$. Legt man die Menge der Endzustände noch entsprechend fest, so leistet das Verfahren das Gewünschte. □

Ohne Nachweis sei an dieser Stelle noch bemerkt, daß die Inklusion des Satzes echt ist, da z.B. die kontextsensitive, nicht kontextfreie Sprache $L = \{\mathtt{a}^{\cdot n}\mathtt{b}^{\cdot n}\mathtt{c}^{\cdot n} \mid n \in \mathbb{N}\}$ zur Familie $\mathscr{L}_{rt}$(OCA) gehört.

Beispiel 4.2.
Die Sprache $L = \{vv' \mid v, v' \in \{\mathtt{a}, \mathtt{b}, \mathtt{0}, \mathtt{1}\}^+ \ \wedge \ |v'| \geq 2 \ \wedge \ v' = v'^R\}$ ist linear kontextfrei.

Eine entsprechende Grammatik ist $G = (V_N, V_T, X_0, F)$ mit

$V_N := \{X_0, Y, Y_a, Y_b, Y_0, Y_1\}$
$V_T := \{\mathtt{a}, \mathtt{b}, \mathtt{0}, \mathtt{1}\}$
$F := \{X_0 \to \mathtt{a}X_0,\ X_0 \to \mathtt{b}X_0,\ X_0 \to \mathtt{1}X_0,\ X_0 \to \mathtt{0}X_0,$
$X_0 \to Y_a\mathtt{a},\ X_0 \to Y_b\mathtt{b},\ X_0 \to Y_1\mathtt{1},\ X_0 \to Y_0\mathtt{0},$
$Y_a \to \mathtt{a}Y,\ Y_b \to \mathtt{b}Y,\ Y_1 \to \mathtt{1}Y,\ Y_0 \to \mathtt{0}Y,$
$Y_a \to \mathtt{a},\ Y_b \to \mathtt{b},\ Y_1 \to \mathtt{1},\ Y_0 \to \mathtt{0},$
$Y \to Y_a\mathtt{a},\ Y \to Y_b\mathtt{b},\ Y \to Y_1\mathtt{1},\ Y \to Y_0\mathtt{0},$
$Y \to \mathtt{a},\ Y \to \mathtt{b},\ Y \to \mathtt{1},\ Y \to \mathtt{0}\}$

Das Wort $\mathtt{a10b10abb11bba0} \in L(G)$ läßt sich folgendermaßen erzeugen:

$X_0 \Rightarrow \mathtt{a}X_0 \Rightarrow \mathtt{a1}X_0 \Rightarrow \mathtt{a10}X_0 \Rightarrow \mathtt{a10b}X_0 \Rightarrow \mathtt{a10b1}X_0$
$\Rightarrow \mathtt{a10b1}Y_0\mathtt{0} \Rightarrow \mathtt{a10b10}Y\mathtt{0} \Rightarrow \mathtt{a10b10}Y_a\mathtt{a0} \Rightarrow \mathtt{a10b10a}Y\mathtt{a0}$
$\Rightarrow \mathtt{a10b10a}Y_b\mathtt{ba0} \Rightarrow \mathtt{a10b10ab}Y\mathtt{ba0} \Rightarrow \mathtt{a10b10ab}Y_b\mathtt{bba0}$
$\Rightarrow \mathtt{a10b10abb}Y\mathtt{bba0} \Rightarrow \mathtt{a10b10abb}Y_1\mathtt{1bba0} \Rightarrow \mathtt{a10b10abb11bba0}$

Ein nach obigem Verfahren konstruierter OCA durchliefe bei Eingabe des Wortes folgende Konfigurationen:

#	a	1	0	b	1	0	a	b	b	1	1	b	b	a	0	#
#	a	1	0	b	1	0	a	b	b	1	1	b	b	a	0	#
#	Y, Y_a	Y, Y_1	Y, Y_0	Y, Y_b	Y, Y_1	Y, Y_0	Y, Y_a	Y, Y_b	Y, Y_b	Y, Y_1	Y, Y_1	Y, Y_b	Y, Y_b	Y, Y_a	Y, Y_0	#

#		1	0	b	1	0	a	b	b	1	1	b	b	a	0	#
#		a	1	0	b	1	0	a	b	b	1	1	b	b	a	#
#		$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	X_0, Y, Y_b	$\emptyset$	X_0, Y, Y_1	$\emptyset$	X_0, Y, Y_b	$\emptyset$	$\emptyset$	#

#			0	b	1	0	a	b	b	1	1	b	b	a	0	#
#			a	1	0	b	1	0	a	b	b	1	1	b	b	#
#			$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	X_0, Y_a	$\emptyset$	X_0, Y_b	$\emptyset$	X_0, Y_1	$\emptyset$	$\emptyset$	#

#				b	1	0	a	b	b	1	1	b	b	a	0	#
#				a	1	0	b	1	0	a	b	b	1	1	b	#
#				$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	X_0	$\emptyset$	X_0	Y	X_0	$\emptyset$	$\emptyset$	#

#					1	0	a	b	b	1	1	b	b	a	0	#
#					a	1	0	b	1	0	a	b	b	1	1	#
#					$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	X_0	$\emptyset$	X_0	Y_b	X_0	$\emptyset$	$\emptyset$	#

#						0	a	b	b	1	1	b	b	a	0	#
#						a	1	0	b	1	0	a	b	b	1	#
#						$\emptyset$	$\emptyset$	$\emptyset$	X_0	$\emptyset$	X_0	$\emptyset$	X_0, Y	$\emptyset$	$\emptyset$	#

#							a	b	b	1	1	b	b	a	0	#
#							a	1	0	b	1	0	a	b	b	#
#							$\emptyset$	$\emptyset$	X_0	$\emptyset$	X_0	$\emptyset$	X_0, Y_a	$\emptyset$	$\emptyset$	#

#								b	b	1	1	b	b	a	0	#
#								a	1	0	b	1	0	a	b	#
#								$\emptyset$	X_0	$\emptyset$	X_0	$\emptyset$	X_0	X_0, Y	$\emptyset$	#

#									b	1	1	b	b	a	0	#
#									a	1	0	b	1	0	a	#
#									X_0	$\emptyset$	X_0	$\emptyset$	X_0	X_0, Y_0	$\emptyset$	#

#										1	1	b	b	a	0	#
#										a	1	0	b	1	0	#
#										$\emptyset$	X_0	$\emptyset$	X_0	X_0	X_0, Y	#

#											1	b	b	a	0	#
#											a	1	0	b	1	#
#											X_0	$\emptyset$	X_0	X_0	X_0, Y_1	#

#												b	b	a	0	#
#												a	1	0	b	#
#												$\emptyset$	X_0	X_0	X_0	#

#													b	a	0	#
#													a	1	0	#
#													X_0	X_0	X_0	#

#														a	0	#
#														a	1	#
#														X_0	X_0	#

#															0	#
#															a	#
#															X_0	#

Abbildung 4.5. Raum-Zeit-Diagramm eines OCA zum Beispiel 4.2.

Der folgende Satz von S. N. Cole (1966) ist insofern bemerkenswert, da weder die Dimension noch die Nachbarschaft des iterativen Arrays beschränkt werden. Als Folgerung ergibt sich dann die Unvergleichbarkeit von $\mathscr{L}_{rt}(\mathrm{OCA})$ und $\mathscr{L}_{rt}(\mathrm{IA})$. Der Beweis umfaßt ein Beispiel und den Nachweis zweier Behauptungen.

Satz 4.3. Es existiert eine kontextfreie Sprache, die von keinem iterativen Array in Realzeit akzeptiert wird.

Beweis. Die Sprache

$$L := \{vv' \mid v, v' \in \{\mathtt{a}, \mathtt{b}, \mathtt{0}, \mathtt{1}\}^* \ \wedge\ |v'| \geq 2 \ \wedge\ v' = v'^R\}$$

ist linear kontextfrei, also auch kontextfrei.

Es sei $W_n := \{\mathtt{0}w\mathtt{1} \mid w \in \{\mathtt{a}, \mathtt{b}\}^* \ \wedge\ |w| = n\}$. Dann wird für alle Untermengen $U = \{u_1, \ldots, u_m\}$ von W_n ein Wort u festgelegt durch

$$u := \begin{cases} \varepsilon & \text{falls } U = \emptyset \\ u'_m & \text{sonst} \end{cases},$$

wobei u'_m rekursiv definiert wird:

$$u'_0 := \varepsilon, \qquad u'_{i+1} := u^R_{i+1} u'^R_i u'_i \text{ für } i \in \{0, \ldots, m-1\}.$$

Da jedes Wort aus einer Menge U als Teilwort in u enthalten ist und die Teilwörter durch die Symbole $\mathtt{1}$ und $\mathtt{0}$ untereinander abgegrenzt sind, sind für verschiedene Untermengen U und V von W_n die Wörter u und v stets verschieden.

Beispiel 4.3. Für $n = 3$ ist

$$U := \{\overbrace{\mathtt{0aaa1}}^{u_1}, \overbrace{\mathtt{0abb1}}^{u_2}, \overbrace{\mathtt{0bab1}}^{u_3}\}$$

Teilmenge von W_n. Das Wort $u = u'_3$ wird dann festgelegt durch

$$
\begin{aligned}
u'_0 &= \varepsilon \\
u'_1 &= \underbrace{\mathtt{1aaa0}}_{u_1^R} \\
u'_2 &= \underbrace{\mathtt{1bba0}}_{u_2^R}\underbrace{\underbrace{\mathtt{0aaa1}}_{u_1}}_{u_1'^R}\underbrace{\underbrace{\mathtt{1aaa0}}_{u_1^R}}_{u'_1} \\
u'_3 &= \underbrace{\mathtt{1bab0}}_{u_3^R}\underbrace{\underbrace{\mathtt{0aaa1}}_{u_1}\underbrace{\mathtt{1aaa0}}_{u_1^R}\underbrace{\mathtt{0abb1}}_{u_2}}_{u_2'^R}\underbrace{\underbrace{\mathtt{1bba0}}_{u_2^R}\underbrace{\mathtt{0aaa1}}_{u_1}\underbrace{\mathtt{1aaa0}}_{u_1^R}}_{u'_2}
\end{aligned}
$$

Behauptung 4.1. $\forall u_i \in U : uu_i \in L$

Beweis der Behauptung. Bei der Konstruktion der u'_{j+1} wird u'_j jeweils am Ende angefügt. Also endet z.B. auch u'_{j+2} mit u'_j. Folglich existiert für alle $u'_j, j \in \{1, \ldots, m\}$, ein Wort $x \in \{\mathtt{a}, \mathtt{b}, \mathtt{0}, \mathtt{1}\}^*$, so daß sich u darstellen läßt als $u = xu'_j$. Somit gilt: $uu_i = xu'_i u_i = xu_i^R u'^R_{i-1} u'_{i-1} u_i \in L$. □

Behauptung 4.2. $\forall \bar{u} \in W_n \setminus U : u\bar{u} \notin L$

Beweis der Behauptung. Für $U = \emptyset$ ist $u = \varepsilon$ und die Behauptung folgt, da $\bar{u}$ mit $\mathtt{1}$ endet und $\mathtt{1}$ sonst nicht in $\bar{u}$ vorkommt.

Es wird angenommen, $u\bar{u}$ ist aus der Sprache L. Das Wort u kann in Teilwörter der Länge $n+2$ zerlegt werden, wobei jedes Teilwort mit $\mathtt{1}$ oder $\mathtt{0}$ beginnt und mit $\mathtt{0}$ oder $\mathtt{1}$ endet. Nirgendwo sonst in u können die Zeichen $\mathtt{1}$ und $\mathtt{0}$ auftreten. Da $\bar{u}$ selbst nicht Palindrom sein kann, muß eines dieser Teilwörter $\bar{u}^R$ sein. Wegen der Asymmetrie und der Konstruktion von u folgt dann $\bar{u} \in U$. Dies ist ein Widerspruch zur Voraussetzung $\bar{u} \in W_n \setminus U$. □

Es sei nun angenommen, L könnte von einem eindimensionalen iterativen Array mit H_1-Nachbarschaft und Zustandsmenge S in Realzeit akzeptiert werden.

Ein $n \in \mathbb{N}$ kann so gewählt werden, daß $2^{2^n} > |S|^{(2n+5)}$ gilt. Betrachtet man die entsprechende Menge W_n, so enthält sie 2^n Elemente. Also existieren $|\mathscr{P}(W_n)| = 2^{2^n}$ verschiedene Teilmengen U mit ebensovielen Wörtern u.

Nachdem ein iterativer Array obiger Struktur ein Eingabewort bis auf die letzten l Zeichen eingelesen hat, kann die Zelle 0 die Entscheidung, ob die Eingabe akzeptiert wird oder nicht, nur von der Information abhängig machen, die ihr in l weiteren Schritten zugänglich ist. Bei einer H_1-Nachbarschaft ist dies gerade die Information in den Zellen i mit $-l \leq i \leq l$. Hierfür gibt es weniger als $|S|^{(2l+1)}$ Möglichkeiten.

Für $l = n+2$ ergeben sich also weniger als $|S|^{(2n+5)}$ Möglichkeiten. Demnach gibt es mindestens zwei verschiedene Wörter $\tilde{u}$ und $\hat{u}$, die die Untermengen $\tilde{U}$ und $\hat{U}$ charakterisieren, für die diese Information übereinstimmt. O.B.d.A. kann angenommen werden, daß ein $u_i \in \tilde{U} \setminus \hat{U}$ existiert. Also ist nach der ersten Behauptung $\tilde{u}u_i \in L$. Da der Automat aber auch $\hat{u}u_i$ akzeptieren würde, ergibt sich ein Widerspruch mit der zweiten Behauptung.

Betrachtet wird nun ein iterativer Array mit beliebiger Nachbarschaft N und Dimension d. Dann existiert ein minimales k, so daß gilt: $N \subseteq M_k$, wobei M_k das verallgemeinerte Moore-Raster ist (vgl. Definition 3.2). Das heißt, die in l Schritten verfügbare Information befindet sich in höchstens $(2lk+1)^d$ Zellen. Dafür gibt es $|S|^{(2lk+1)^d}$ Möglichkeiten. Die weitere Schlußfolge ist analog zu der oben Gezeigten. $\square$

Korollar 4.2. $\exists L \in \mathscr{L}_{rt}(\mathrm{OCA}) : L \notin \mathscr{L}_{rt}(\mathrm{IA})$

Korollar 4.3. Die Familien $\mathscr{L}_{rt}(\mathrm{OCA})$ und $\mathscr{L}_{rt}(\mathrm{IA})$ sind unvergleichbar.

4.4 Iterative Arrays versus bidirektionale Zellularautomaten

Der letzte Abschnitt dieses Kapitels klärt den Zusammenhang zwischen Realzeit-IAs und Realzeit-CAs. Hierbei wird ein „Umweg“ über OCAs beschritten, und es gelten wieder die Einschränkungen bzgl. des Akzeptierens am rechten und linken Rand, die beim Satz 3.14 notwendig waren.

Satz 4.4. Es sei ein $\bar{H}_1$-Raster für OCAs angenommen. Dann gilt:

$$\mathscr{L}_{2n}(\mathrm{OCA}) \supset \mathscr{L}_{rt}(\mathrm{IA})$$

Beweis. Nach Satz 3.14 wird eine Sprache L genau dann von einem Realzeit-CA akzeptiert, wenn sie von einem OCA mit $\bar{H}_1^R$-Raster in $2n$ Zeitschritten akzeptiert werden kann. Außerdem kann L genau dann von einem OCA mit $\bar{H}_1^R$-Raster (in $2n$ Zeitschritten) akzeptiert werden, wenn L^R $(MIR(L))$ von einem OCA mit $\bar{H}_1$-Raster (in $2n$ Zeitschritten) akzeptiert wird.

Zum Nachweis des Satzes müßte zunächst gezeigt werden, daß jede Sprache L aus $\mathscr{L}_{rt}(\mathrm{IA})$ auch in $\mathscr{L}_{2n}(\mathrm{OCA})$ ist. Nach dem eben Gesagten ist dies genau dann der Fall, wenn L^R von einem OCA mit $\bar{H}_1^R$-Raster in $2n$ Zeitschritten akzeptiert wird. Dies wiederum gilt genau dann, wenn L^R aus der Familie $\mathscr{L}_{rt}(\mathrm{CA})$ ist.

Es sei also ein Realzeit IA gegeben. Ein CA, der das Spiegelbild der Sprache des IA akzeptiert, arbeitet mit drei Spuren. Die Ein/Ausgabe-Zelle ist diejenige am rechten Rand. Auf der ersten Spur wird die Eingabe in jedem Zeitschritt um eine Zelle nach rechts verschoben. Auf diese Weise kann die rechte Randzelle die Ein/Ausgabe-Zelle simulieren. Auf den beiden anderen Spuren wird der rechte bzw. linke Halbraum des IA simuliert. Da der IA in Realzeit arbeitet, ist der Platz ausreichend.

Die Echtheit der Inklusion folgt sofort mit Korollar 4.2. □

Mit den einschränkenden Bemerkungen von Satz 3.14 folgt dann:

Korollar 4.4. $\mathscr{L}_{rt}(\mathrm{CA}) \supset \mathscr{L}_{rt}(\mathrm{IA})$

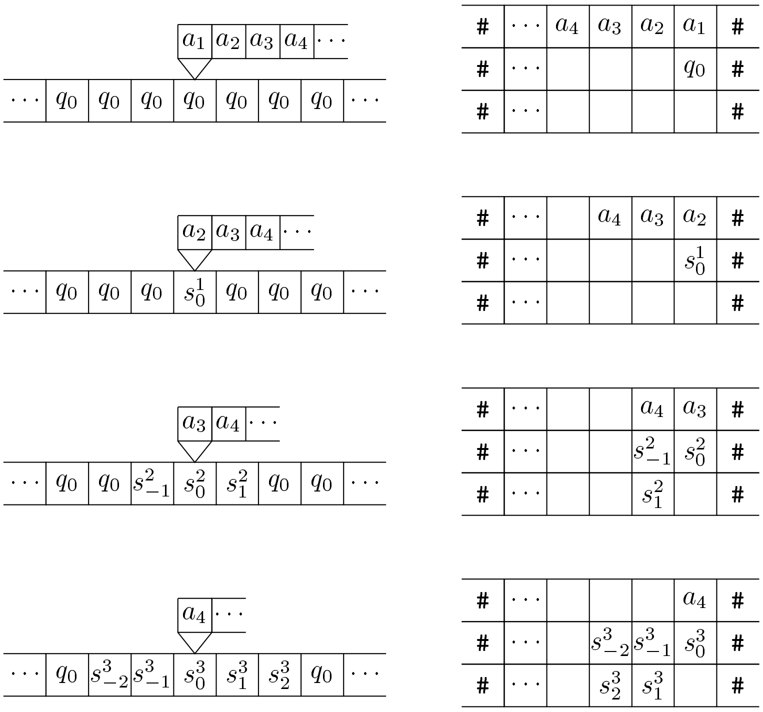


Abbildung 4.6. Simulation eines Realzeit-IA
durch einen Realzeit-CA.

5 Keller-Zellularautomaten

Betrachtet man die endlichen Automaten als aktive Verarbeitungselemente, so scheint die Voraussetzung, beliebig viele von ihnen zur Verfügung zu haben, für technische Realisierungen nicht geeignet zu sein. Eine diesbezügliche Einschränkung der Zellularräume führt zu dem Modell der Zellularautomaten. Hierin wird die Anzahl der aktiven Elemente von der jeweiligen Eingabe abhängig gemacht. Auf diese Weise sind zwar im allgemeinen immer noch beliebig viele endliche Automaten notwendig, aber deren Anzahl kann im Laufe einer Berechnung nicht mehr anwachsen. Demgegenüber steht der Verlust der Berechnungsuniversalität. Ein Vergleich mit Turingmaschinen — sie verfügen über ein einziges aktives Element — zeigt, daß die Berechnungsuniversalität unter Hinzunahme von beliebig vielen passiven Elementen — etwa Speicherstellen — wieder erreicht werden kann. Im Hinblick auf technische Realisierungen scheint die Annahme beliebig großer Speicher weniger problematisch zu sein als diejenige, über beliebig viele Prozessoren verfügen zu können. Da aber gerade der Wunsch nach höherer Leistungsfähigkeit zur Konstruktion von Parallelrechnern führte, ist es naheliegend, Systeme zu betrachten, die eine große Anzahl aktiver Verarbeitungselemente mit beliebig vielen passiven Elementen vereinen.

In der Chomsky-Hierarchie formaler Sprachen und der damit verknüpften Hierarchie sequentieller Automaten ist der deterministische Kellerautomat nach den endlichen Automaten das nächste mächtigere Modell. Entsprechend gehen die Keller-Zellularautomaten aus den Zellularautomaten hervor, indem die endlichen Automaten durch deterministische Kellerautomaten ersetzt werden. Zum einen wird durch diese Vorgehensweise erreicht, daß auch bei einer Einschränkung der Anzahl der aktiven Elemente die Berechnungsuniversalität nicht verlorengeht. Zum anderen werden die passiven Speicherelemente durch die Modellierung als Kellerspeicher gegenüber wahlfreiem Zugriff vereinfacht.

5.1 Das Modell

Definition 5.1. Ein 7-Tupel $(S, \Gamma, d, N, \sigma, q_0, g_0)$ heißt *Keller-Zellularraum*, falls gilt:

a) S ist eine endliche, nichtleere *Zustandsmenge*.
b) Γ ist eine endliche, nichtleere Menge von *Kellersymbolen*.
c) $d \in \mathbb{N}_0$ ist die *Raumdimension*.
d) N ist d-dimensionaler Nachbarschaftsindex vom Grade n.
e) $\sigma : S^n \times \Gamma \longrightarrow S \times \Gamma^*$ ist *lokale Überführungsfunktion*, wobei für $g \in \Gamma$, $\tilde{q} \in S^n$ und $q' \in S$ gilt: $\sigma(\tilde{q}, g) = (q', \gamma) \Longrightarrow$ $(\gamma \in (\Gamma \setminus \{g_0\})^* \ \wedge \ g \neq g_0)$
$$\vee\ (\gamma = \gamma' g_0 \text{ mit } \gamma' \in (\Gamma \setminus \{g_0\})^* \ \wedge \ g = g_0).$$
f) $q_0 \in S$ ist ein *Ruhezustand*, so daß gilt:
$$\forall g \in \Gamma : \ \sigma(q_0, \ldots, q_0, g) = (q_0, g).$$
g) $g_0 \in \Gamma$ ist das *Kellerendesymbol*.

Aus der Definition geht hervor, daß die Zustandsüberführung eines einzelnen Kellerautomaten von den Zuständen seiner Nachbarn und seinem obersten Kellereintrag abhängt. Der Zugriff auf den eigenen Keller ist dabei unabhängig vom benutzten Raster.
Die geforderte Einschränkung der Überführungsfunktion stellt sicher, daß ein einzelner Automat im Ruhezustand diesen nur aufgrund seines Kellerinhaltes nicht verlassen kann. Ferner kann der Kellerinhalt nicht verändert werden, falls sich der Automat und seine Nachbarn im Ruhezustand befinden, und es wird verhindert, daß das Kellerendesymbol aus dem Keller entfernt bzw. mehrfach hineingeschrieben wird.

Definition 5.2. Eine Abbildung $c_t : \mathbb{Z}^d \longrightarrow S \times \Gamma^*$ heißt *Konfiguration* des Keller-Zellularraums im Zeitpunkt $t \in \mathbb{N}_0$.

Eine Konfiguration legt wieder für jede Zelle im Raum deren aktuellen Zustand fest. Zusätzlich wird der Kellerinhalt berücksichtigt. Dieser wird als ein Wort über dem Kelleralphabet aufgefaßt, wobei das Wort von links nach rechts den Kellerinhalt von oben nach unten repräsentiert.

Definition 5.3. Es seien A ein Alphabet, $N = (a_1, \ldots, a_n)$ ein d-dimensionaler Nachbarschaftsindex und C_K die Menge aller Konfigurationen der Art $c(i) = (s, \gamma = g_{i_1} \cdots g_{i_{n_i}} g_0), s \in S, \gamma \in \Gamma^*$. Eine Abbildung $\mathcal{T}_K : C_K \longrightarrow C_K$ heißt *globale Transformation*, wenn sie durch eine lokale Überführungsfunktion σ folgendermaßen induziert wird:

$$\forall c \in C_K : \Bigg(\mathcal{T}(c) = c' \iff \Big(\forall i \in \mathbb{Z}^d : \quad c'(i) =$$
$$\big(\pi_1(\sigma((\pi_1(c(i+a_1)), \ldots, \pi_1(c(i+a_n))), g_{i_1})),$$
$$\pi_2(\sigma((\pi_1(c(i+a_1)), \ldots, \pi(c(i+a_n))), g_{i_1}))g_{i_2} \cdots g_{i_{n_i}} g_0)\big)\Big)\Bigg)$$

Es wird also festgelegt, daß das oberste Kellerelement beim Anwenden der Überführungsfunktion aus dem Keller entfernt wird. Dies stellt insoweit keine Einschränkung dar, als jede Zelle das oberste Kellerelement durch ein Wort über dem Kelleralphabet Γ ersetzen kann.

Definition 5.4. Es sei $M = (S, \Gamma, d, N, \sigma, q_0, g_0)$ ein Keller-Zellularraum. M heißt *Keller-Zellularautomat*, falls gilt:

a) $\exists \# \in S : \forall t \in \mathbb{N}_0, i \in \mathbb{Z}^d : c_{t+1}(i) = \# \iff c_t(i) = \#$, dabei heißt $\#$ *Grenzzustand*.

b) Ein endliches, nichtleeres Gebiet $R \subset \mathbb{Z}^d$, *Retina* genannt, hat folgende Eigenschaften:
$\forall i \in R : \pi_1(c_0(i)) \neq \#$
$\forall k \in \{l \in \mathbb{Z}^d \setminus R | \, \exists i \in R : i \vdash_N l\} : \pi_1(c_0(k)) = \#$
$\forall k \notin \{l \in \mathbb{Z}^d \setminus R | \, \exists i \in R : i \vdash_N l\} \cup R : \pi_1(c_0(k)) = q_0$
$\forall i, j \in R : i \vdash^*_{H_1^d} j$

Im folgenden Beispiel wird ein eindimensionaler Keller-Zellularautomat mit drei Spuren und H_1-Raster angegeben, der den Inhalt seiner zweiten Spur spiegelt. In der Anfangskonfiguration sind alle Keller bis auf das Kellerendesymbol leer. Die Inhalte der ersten und dritten Register sind für alle Zellen identisch.

Beispiel 5.1.
Das Verfahren setzt eine Eingabe voraus, die mindestens zwei Zeichen enthält. Die prinzipielle Vorgehensweise ist folgende (vgl. Abbildung 5.1): Die rechte Randzelle identifiziert sich anhand ihres rechten Nachbarn selbst. Im Verlauf der Berechnung kopiert sie in jedem Zeitschritt den Inhalt ihres zweiten Registers in das erste.

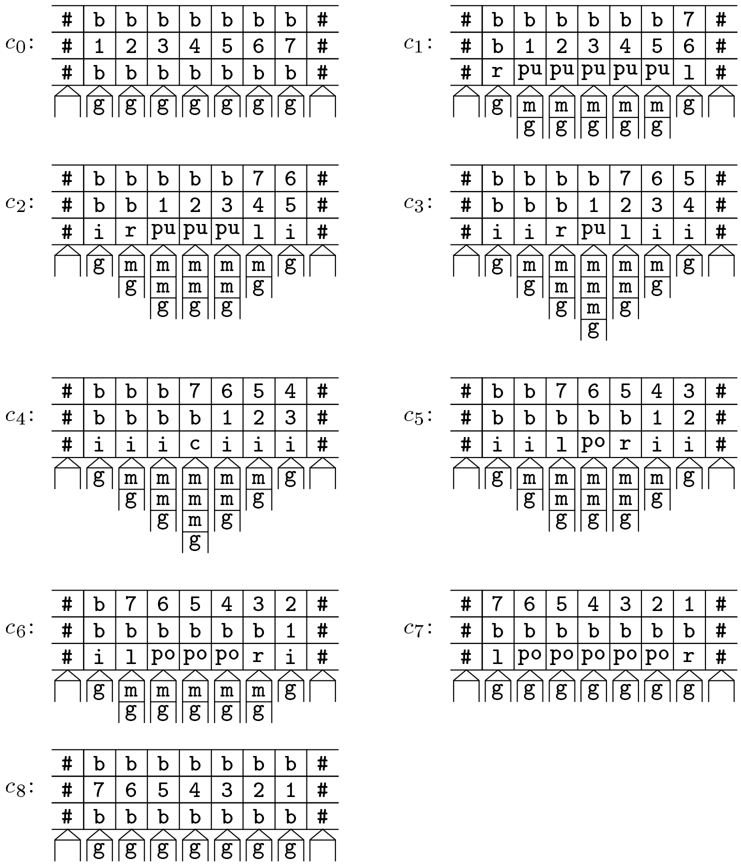


Abbildung 5.1. Beispiel eines Keller-Zellularautomaten, der seine Eingabe spiegelt.

Die Inhalte der zweiten Spur werden in jedem Zeitschritt um eine Zelle nach rechts verschoben, diejenigen der ersten Spur nach links. Auf diese Weise liegt die Eingabe nach einer Zeit, die ihrer Länge entspricht, gespiegelt auf der ersten Spur vor. Alle Zellen können dann in einem weiteren Zeitschritt synchron die Inhalte ihrer ersten Register in die zweiten kopieren. Voraussetzung hierfür ist, daß alle Zellen gleichzeitig erkennen können, wann die Spiegelung abgeschlossen ist.

Um dies zu ermöglichen, werden auf der dritten Spur im ersten Zeitschritt am rechten und linken Rand zwei Signale erzeugt, die zum jeweils gegenüberliegenden Rand gesandt werden. Alle Zellen beginnen im ersten Zeitschritt damit, ihren Keller jeweils mit einem zusätzlichen Symbol zu beschriften. Dieser Prozeß wird beim Empfang des ersten Signals beendet. Beim Empfang des zweiten Signals wird in jedem weiteren Zeitschritt ein Symbol aus dem Keller entfernt. Auf diese Weise werden die Keller aller Zellen gleichzeitig leer, und zwar in genau dem Zeitpunkt, in dem die Spiegelung abgeschlossen ist.

Der Keller-Zellularautomat $M = (S, \Gamma, 1, H_1^1, \sigma, q_0, g_0)$ wird formal wie folgt konstruiert:

$S := \{\mathtt{\#}, \mathtt{1}, \mathtt{2}, \mathtt{3}, \mathtt{4}, \mathtt{5}, \mathtt{6}, \mathtt{7}, \mathtt{b}\} \times \{\mathtt{\#}, \mathtt{1}, \mathtt{2}, \mathtt{3}, \mathtt{4}, \mathtt{5}, \mathtt{6}, \mathtt{7}, \mathtt{b}\} \times \{\mathtt{\#}, \mathtt{i}, \mathtt{pu}, \mathtt{po}, \mathtt{r}, \mathtt{l}, \mathtt{c}, \mathtt{b}\}$

$\Gamma := \{\mathtt{m}, \mathtt{g}\}$

$q_0 := (\mathtt{b}, \mathtt{b}, \mathtt{b})$

$g_0 := \mathtt{g}$

Grenzzustand: $(\mathtt{\#}, \mathtt{\#}, \mathtt{\#})$

$H_1^1 = (-1, 0, 1)$

σ wird wie folgt festgelegt:

$\sigma((o_1, o_2, o_3), (p_1, p_2, p_3), (q_1, q_2, q_3), h) := ((p_1', p_2', p_3'), \gamma)$,

wobei gilt:

$$\gamma := \begin{cases} \mathtt{m}h & \text{falls } (p_3 = \mathtt{pu} \,\wedge\, o_3 \neq \mathtt{r} \,\wedge\, q_3 \neq \mathtt{l}) \\ & \qquad \vee\, (o_3 = \mathtt{b} \,\wedge\, p_3 = \mathtt{b} \,\wedge\, q_3 = \mathtt{b}) \\ \varepsilon & \text{falls } (p_3 = \mathtt{po} \,\wedge\, h \neq \mathtt{g}) \,\vee\, (p_3 = \mathtt{l} \,\wedge\, o_3 = \mathtt{i}) \\ & \qquad \vee\, (p_3 = \mathtt{r} \,\wedge\, q_3 = \mathtt{i}) \,\vee\, (p_3 = \mathtt{c}) \\ h & \text{sonst} \end{cases}$$

$$p'_1 := \begin{cases} \mathtt{b} & \text{falls } (p_3 = \mathtt{po} \ \wedge \ h = \mathtt{g}) \ \vee \ (p_3 = \mathtt{l} \ \wedge \ o_3 = \mathtt{\#}) \\ & \qquad \vee \ (p_3 = \mathtt{r} \ \wedge \ q_3 = \mathtt{\#}) \\ p_2 & \text{falls } q_3 = \mathtt{\#} \ \wedge \ p_3 \neq \mathtt{r} \\ q_1 & \text{sonst} \end{cases}$$

$$p'_2 := \begin{cases} p_1 & \text{falls } (p_3 = \mathtt{po} \ \wedge \ h = \mathtt{g}) \ \vee \ (p_3 = \mathtt{r} \ \wedge \ q_3 = \mathtt{\#}) \\ & \qquad \vee \ (o_1 = \mathtt{\#}) \\ o_2 & \text{sonst} \end{cases}$$

$$p'_3 := \begin{cases} \mathtt{r} & \text{falls } (o_3 = \mathtt{\#} \ \wedge \ p_3 = \mathtt{b}) \ \vee \ (o_3 = \mathtt{r} \ \wedge \ q_3 \neq \mathtt{l}) \ \vee \ (o_3 = \mathtt{c}) \\ \mathtt{l} & \text{falls } (q_3 = \mathtt{\#} \ \wedge \ p_3 = \mathtt{b}) \ \vee \ (q_3 = \mathtt{l} \ \wedge \ o_3 \neq \mathtt{r}) \ \vee \ (q_3 = \mathtt{c}) \\ \mathtt{c} & \text{falls } o_3 = \mathtt{r} \ \wedge \ q_3 = \mathtt{l} \\ \mathtt{pu} & \text{falls } o_3 = \mathtt{b} \ \wedge \ p_3 = \mathtt{b} \ \wedge \ q_3 = \mathtt{b} \\ \mathtt{i} & \text{falls } (p_3 = \mathtt{r} \ \wedge \ q_3 = \mathtt{pu}) \ \vee \ (p_3 = \mathtt{l} \ \wedge \ o_3 = \mathtt{pu}) \\ \mathtt{po} & \text{falls } (p_3 = \mathtt{l} \ \wedge \ o_3 = \mathtt{i}) \ \vee \ (p_3 = \mathtt{r} \ \wedge \ q_3 = \mathtt{i}) \ \vee \ (p_3 = \mathtt{c}) \\ \mathtt{b} & \text{falls } (p_3 = \mathtt{po} \ \wedge \ h = \mathtt{g}) \ \vee \ (p_3 = \mathtt{l} \ \wedge \ o_3 = \mathtt{\#}) \\ & \qquad \vee \ (p_3 = \mathtt{r} \ \wedge \ q_3 = \mathtt{\#}) \\ p_3 & \text{sonst} \end{cases}$$

Definition 5.5.
Es seien die Abbildungen $k : \mathbb{N} \longrightarrow \mathbb{N}$ und $t : \mathbb{N} \longrightarrow \mathbb{N}$ und eine formale Sprache $L \subseteq A^+$ gegeben. L wird durch einen Keller-Zellularautomaten $M = (S, \Gamma, 1, N, \sigma, q_0, g_0)$ mit einer Menge von Endzuständen $F \subseteq S$ genau dann mit Kellerkomplexität k und Zeitkomplexität t akzeptiert, wenn

$$L = \Big\{ w \in A^+ | \ \mathcal{T}_K^{t(|w|)}(c_0) = c_{t(|w|)}, \text{ wobei } \forall i \in \mathbb{Z}, 0 \leq \hat{t} \leq t(|w|) \text{ gilt:}$$

$$\pi_1(c_0(i)) = \begin{cases} w_i & \text{falls } 1 \leq i \leq |w| \\ \mathtt{\#} & \text{sonst} \end{cases}, \qquad \pi_2(c_0(i)) = g_0,$$

$$\pi_1(c_{t(|w|)}(|w|)) \in F, \qquad |\pi_2(c_{\hat{t}}(i))| \leq k(|w|) \Big\}.$$

Definition 5.6. Die Familie der von Keller-Zellularautomaten mit H_1-Raster, Zeitkomplexität t und Kellerkomplexität k akzeptierbaren Sprachen wird mit ${}_{k}\mathscr{L}_t$(PDCA) bezeichnet. Für unidirektionalen Informationsfluß ($\bar{H}_1$-Raster) wird entsprechend die Bezeichnung OPDCA benutzt.

5.2 Kellernormalisierung und -kompression

Dieser Abschnitt widmet sich der Frage, inwieweit die Kelleroperationen einzelner Zellen normalisiert werden können. Zunächst wird gezeigt, daß sich das Kelleralphabet verringern läßt, wobei allerdings sowohl die Zeit als auch die Kellerkomplexität zunehmen. Anschließend wird die Voraussetzung dafür geschaffen, die Kellertiefe um einen beliebigen Faktor zu komprimieren. Den Abschluß des Abschnittes bildet dann der Nachweis, daß die Überführungsfunktion bzgl. der Kelleroperation ohne Zeitverlust eingeschränkt werden kann, was im weiteren als Normalisierung bezeichnet wird.

5.2.1 Reduktion des Kelleralphabets

Aus einem gegebenen Keller-Zellularautomaten kann ein anderer mit reduziertem Kelleralphabet konstruiert werden, indem die ursprünglichen Kellersymbole über dem neuen Alphabet kodiert werden. Vor jedem eigentlichen Konfigurationsübergang muß der Kode eines Kellersymbols in mehreren Zeitschritten aus dem Keller gelesen und interpretiert werden. In dieser Phase hängt der Folgezustand der Zellen von ihrem eigenen Zustand ab. Daraus resultiert die Forderung $0^d \in \tilde{N}$. Der Beweis des nächsten Satzes umfaßt die Konstruktion des entsprechenden Automaten und den Beweis einer Behauptung, aus der dann die Korrektheit der Konstruktion folgt.

Satz 5.1. Es sei $M = (S, \Gamma, d, N, \sigma, q_0, g_0)$ ein Keller-Zellularautomat mit der Eigenschaft $0^d \in \tilde{N}$. Dann existiert für alle $3 \leq n < |\Gamma|$ ein Kelleralphabet Γ' mit der Mächtigkeit n und ein Keller-Zellularautomat $M' = (S', \Gamma', d, N, \sigma', q'_0, g'_0)$, so daß gilt:

$$\forall i \in \mathbb{Z}^d : \forall t \in \mathbb{N}_0 : \pi_1(c_t(i)) = \pi_{1,1}(c'_{l \cdot t}(i)),$$

wobei $l = \lceil \log_{n-1} |\Gamma| \rceil$ ist.

Beweis. Es sei $|\Gamma| = m$. Ohne Beschränkung der Allgemeinheit kann angenommen werden, daß die Elemente aus Γ' von h_0 bis h_{n-1} und diejenigen aus Γ von g_0 bis g_{m-1} durchnumeriert sind. Die sich daraus ergebende Ordnung stellt sicher, daß $\max(\Gamma') := h_{n-1}$ definiert ist.

Faßt man die Elemente aus $\Gamma' \setminus \{h_{n-1}\}$ als Ziffern des $(n-1)$-adischen Zahlensystems auf, so können die Elemente aus Γ über diesem Zahlensystem eindeutig kodiert werden, wobei die Länge aller Kodierungen — durch linksseitiges Auffüllen mit h_0 — gerade l ist. Für $g \in \Gamma$ und $\gamma \in \Gamma^*$ bezeichnen g_{code} und γ_{code} die entsprechenden Kodes von g und γ. Ein Wort $\gamma = g_1 \cdots g_i \in \Gamma^{\cdot i}$ werde kodiert als $g_{1_{code}} \cdots g_{i_{code}}$.

Der Konstruktion liegt folgende Idee zugrunde: Statt der ursprünglichen Kellersymbole wird deren Kode im Keller abgelegt. Da jede Zelle in jedem Zeitschritt ein Wort in den Keller schreiben kann, ist die Schreiboperation ohne Zeitverlust möglich. Das Lesen hingegen muß in einer Sequenz von Konfigurationsübergängen erfolgen. Zu diesem Zweck verfügen die Zellen über ein Register, welches den Kode eines ursprünglichen Kellersymbols aufnehmen kann und welches in einer Folge von Zeitschritten mit den einzelnen Symbolen des Kodes sukzessive gefüllt wird. Anschließend erfolgt dann die eigentliche Simulation der Zustandsüberführung.

Im folgenden bezeichnet R immer die Retina.

Beispiel 5.2. Es seien $\Gamma = \{g_0, \ldots, g_7\}$ und $\Gamma' = \{h_0, h_1, h_2\}$. Dann gilt für die Bezeichnungen des Beweises: $|\Gamma| = 8$ und $|\Gamma'| = 3$, also $l = 3$. Mit $H_1 = (-1, 0, 1)$ ist $r = 2$ und h_2 entspricht g'_0. Mit binärer Kodierung gilt $g_{6_{code}} = h_1 h_1 h_0$.

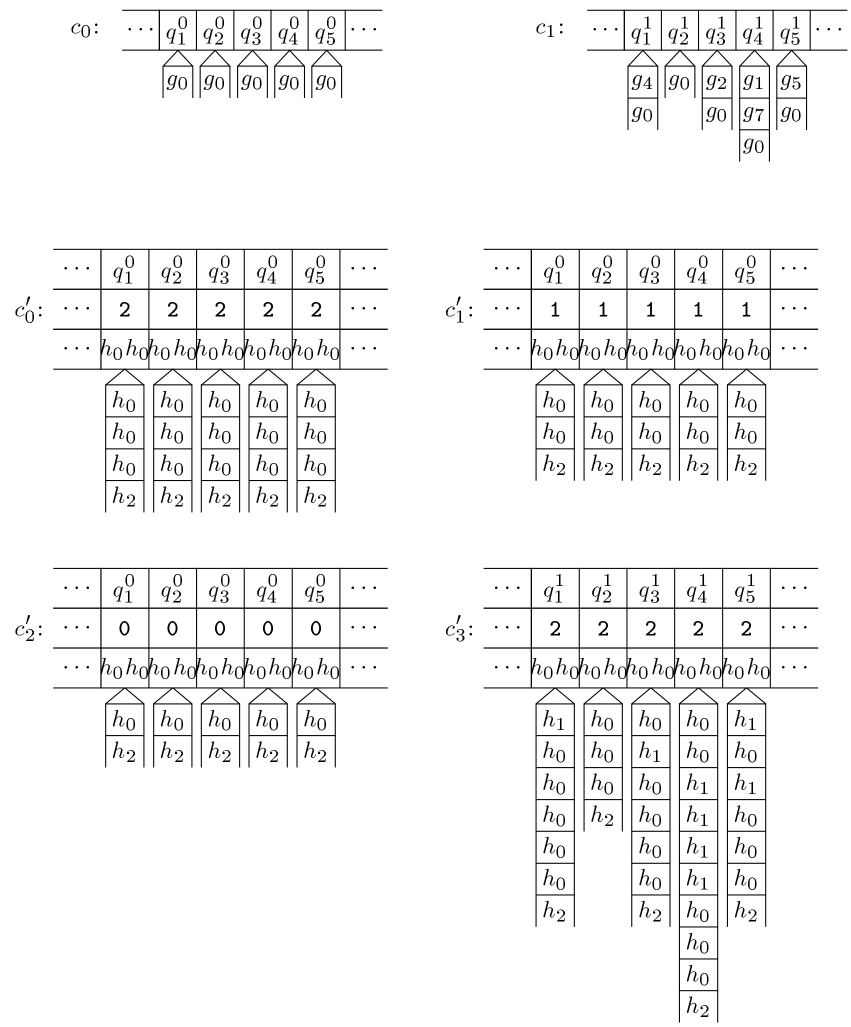


Abbildung 5.2. Konfigurationsübergänge von Keller-Zellularautomaten mit ursprünglichem und reduziertem Kelleralphabet.

Konstruktion von M':

$S' := S \times \{\mathtt{0}, \cdots, \mathtt{l-1}\} \times (\Gamma' \setminus \{h_{n-1}\})^{\cdot(l-1)}$
$q'_0 := (q_0, \mathtt{l-1}, h_0^{\cdot(l-1)})$
$g'_0 := h_{n-1}$

Die Anfangskonfiguration c'_0 von M' wird festgelegt durch:

$$\forall i \in \mathbb{Z}^d, q \in S: \; c_0(i) = (q, g_0) \Longrightarrow c'_0(i) = ((q, \mathtt{l-1}, h_0^{\cdot(l-1)}), g_{0_{code}} g'_0)$$

Ist $\pi_r(N) = 0^d$, dann wird σ' wie folgt festgelegt:

$\forall e \in \Gamma' \setminus \{h_{n-1}\}$:
$$\sigma'(((q_0, \;), \ldots, (q_0, \mathtt{l-1}, h_{r_1} \cdots h_{r_{l-1}}), \ldots, (q_0, \;)), e) = ((q_0, \mathtt{l-1}, h_{r_1} \cdots h_{r_{l-1}}), e)$$

Anderenfalls:
$\forall q_r \in S$ und $e, h_{r_1}, \ldots, h_{r_{l-1}} \in \Gamma' \setminus \{h_{n-1}\}$:

$$\sigma'(((), \ldots, (q_r, \mathtt{l-1}, h_{r_1} \cdots h_{r_{l-1}}), \ldots, ()), e) = ((q_r, \mathtt{l-2}, e h_{r_2} \cdots h_{r_{l-1}}), \varepsilon)$$
$$\vdots$$
$$\sigma'(((), \ldots, (q_r, \mathtt{1}, h_{r_1}, \ldots, h_{r_{l-1}}), \ldots, ()), e) = ((q_r, \mathtt{0}, h_{r_1} \cdots h_{r_{l-2}} e), \varepsilon),$$

wobei „()“ für beliebige Zustände aus S' steht. Insoweit ist σ' also unabhängig von der Konfiguration der Nachbarschaft (mit Ausnahme des Nachbarn 0^d und des Ruhezustands). Weiterhin wird festgelegt:
$\forall q_1, \ldots, q_n, q' \in S, \; g \in \Gamma, \gamma' \in \Gamma^*$:

$$\Big(\sigma((q_1, \ldots, q_r, \ldots, q_n), g) = (q', \gamma') \iff \sigma'(((q_1, \;, \;), \ldots, (q_r, \mathtt{0}, h_{r_1} \cdots h_{r_{l-1}}), \ldots, (q_n, \;, \;)), h_{r_l}) = ((q', \mathtt{l-1}, h_0^{\cdot(l-1)}), \gamma'_{code}), \text{ falls } g_{code} = h_{r_1} \cdots h_{r_l} \text{ ist} \Big).$$

Behauptung 5.1.
Es seien $c_t(i) = (q, g_1 \cdots g_n)$ und $c'_{l \cdot t}(i) = ((q', v, h_1 \cdots h_{l-1}), e_1 \cdots e_k)$ die Konfigurationen einer Zelle i im Zeitpunkt t unter M und im Zeitpunkt $l \cdot t$ unter M'. Es gilt:

$$\forall i \in R : \forall t \in \mathbb{N}_0 : \big(q = q' \wedge e_1 \cdots e_k = g_{1_{code}} \cdots g_{n_{code}} g'_0 \wedge v = \mathtt{1\text{-}1}\big)$$

Beweis der Behauptung.

$t = 0:$
Für die Anfangskonfiguration gilt die Behauptung aufgrund der Definition von c'_0.

$t \to t+1:$
Es seien $q_1, \ldots, q_r, \ldots, q_n \in S$ die Zustände der Nachbarn von Zelle i im Zeitpunkt t.

„$\Longrightarrow$“
Es sei $c_t(i) = (q_r, g_1 \cdots g_m)$ und $c_{t+1}(i) = (q'_r, \gamma g_2 \cdots g_m)$, also

$$\sigma((q_1, \ldots, q_r, \ldots, q_n), g_1) = (q'_r, \gamma).$$

Nach Induktionsannahme gilt:
$c'_{l \cdot t}(i) = ((q_r, \mathtt{1\text{-}1}, h_1 \cdots h_{l-1}), e_1 \cdots e_l \cdots e_k)$ und
$e_1 \cdots e_k = (g_1 \cdots g_m)_{code}$
$\Rightarrow e_1 \cdots e_l = g_{1_{code}}$.
Wegen der Konstruktion von σ' folgt
$c_{l \cdot t+(l-1)}(i) = ((q_r, \mathtt{0}, e_1 \cdots e_{l-1}), e_l \cdots e_k)$
$\Rightarrow c'_{l(t+1)}(i) = ((q'_r, \mathtt{1\text{-}1}, h_0^{\cdot(l-1)}), \gamma_{code} e_{l+1} \cdots e_k)$ Also gilt „ $\Longrightarrow$ “.

„$\Longleftarrow$“
Es seien

$$\begin{aligned} c'_{l \cdot t}(i) &= ((q_r, \mathtt{1\text{-}1}, h_1 \cdots h_{l-1}), e_1 \cdots e_l \cdots e_k) \\ &\vdots \\ c'_{l \cdot t+(l-1)}(i) &= ((q_r, \mathtt{0}, e_1 \cdots e_{l-1}), e_l \cdots e_k) \\ c'_{l(t+1)}(i) &= ((q'_r, \mathtt{1\text{-}1}, h_0^{\cdot(l-1)}), \gamma_{code} e_{l+1} \cdots e_k) \end{aligned}$$

l aufeinanderfolgende Konfigurationen einer Zelle i unter M'. Dann

folgt
$\sigma'((q_1,\ ,\ ),\ldots,(q_r,\mathtt{0},e_1\cdots e_{l-1}),\ldots,(q_n,\ ,\ )),e_l) =$
$$((q_r',\mathtt{1\text{-}1},h_0^{\cdot(l-1)}),\gamma_{code})$$
$\Rightarrow \sigma((q_1,\ldots,q_r,\ldots,q_n),g_1) = (q_r',\gamma)$ für $g_{1_{code}} = e_1\cdots e_l$.
Da nach Induktionsannahme $c_t(i) = (q_r, g_1\cdots g_n)$ und $(g_1\cdots g_n)_{code} = e_1\cdots e_k$ gilt, folgt $c_{t+1}(i) = (q_r', \gamma g_2\cdots g_n)$.

Also gilt „ $\Longleftarrow$ “ und damit die Behauptung. Aus der Behauptung und der Konstruktion von σ' bzgl. des Ruhezustands folgt der Satz. $\square$

Die (in einer Komponente willkürliche) Komposition der Projektionen $\pi_{1,1}$ legt nur fest, in welchem Register das Ergebnis der Berechnung abzulesen ist.

Die lokale Transformation σ' ist in der Konstruktion von Satz 5.1 nicht total. Zwar können Situationen, für die σ' nicht definiert ist, nicht auftreten, es soll aber stillschweigend angenommen werden, daß σ' in diesen Fällen den Zustand und den Kellerinhalt nicht verändert.

Aus dem obigen Satz ergibt sich die Möglichkeit, jedes Kelleralphabet auf drei Elemente zu reduzieren. Wegen der per Definition geforderten Existenz eines Kellerendesymbols sind zwei Elemente im allgemeinen nicht möglich. Speziell für diese Konstruktion gilt aber, daß M' das Symbol g_0' niemals lesen wird, da vorher $g_{0_{code}}$ im Keller notiert ist, welches ebenfalls per Definition nicht aus dem Keller entfernt werden kann.

Korollar 5.1. Es sei M ein Keller-Zellularautomat mit Kelleralphabet Γ und L die Sprache, die von M mit Zeitkomplexität $t(n)$ akzeptiert wird. Es existiert ein Keller-Zellularautomat M' mit drei Kellersymbolen, der L mit Zeitkomplexität $t(n)\log_2|\Gamma|$ akzeptiert.

5.2.2 Reduktion der Kellertiefe

Zunächst wird festgelegt, was unter der Kellertiefe formal verstanden werden soll.

Definition 5.7. Es sei M ein Keller-Zellularautomat. Die *Kellertiefe* im Zeitpunkt t wird für alle $i \in \mathbb{Z}^d$ definiert als

$$\begin{aligned} k : \mathbb{Z}^d \times \mathbb{N}_0 &\longrightarrow \mathbb{N} \\ (i,t) &\mapsto \Big|\pi_2(c_t(i))\Big|. \end{aligned}$$

Außerdem werden die Zeitpunkte wichtig sein, in denen eine Zelle eine bestimmte Kellertiefe aufweist:

Definition 5.8. Es sei M ein Keller-Zellularautomat. Die Zeitpunkte, in denen eine Zelle i die Kellertiefe u aufweist, werden formalisiert durch

$$\begin{aligned} k^{-1} : \mathbb{Z}^d \times \mathbb{N} &\longrightarrow \mathscr{P}(\mathbb{N}_0) \\ (i,u) &\mapsto \{t|\ k(i,t) = u\}. \end{aligned}$$

Der folgende Satz stellt sicher, daß die Kellertiefe um eine additive Konstante verringert werden kann, ohne das Kelleralphabet zu modifizieren. Die entsprechende Anzahl an Kellersymbolen wird dabei lediglich in einem zusätzlichen Register abgelegt.

Satz 5.2. Es sei $M = (S, \Gamma, d, N, \sigma, q_0, g_0)$ ein Keller-Zellularautomat mit der Eigenschaft $0^d \in \tilde{N}$. Dann existiert für alle $z \in \mathbb{N}$ ein Keller-Zellularautomat $M' = (S', \Gamma, d, N, \sigma', q_0', g_0)$, so daß gilt:
$\forall i \in \mathbb{Z}^d : \forall t \in \mathbb{N}_0 :$

$$\Big(\pi_1(c_t(i)) = \pi_{1,1}(c_t'(i)) \ \wedge\ k'(i,t) = \max\{1, k(i,t) - z\}\Big).$$

Beweis. Konstruktion von M':

$$S' := S \times \bigcup_{0 \le i \le z} \Gamma^{\cdot i}$$

$$q_0' := (q_0, \varepsilon)$$

Die Anfangskonfiguration c'_0 von M' wird festgelegt durch

$$\forall i \in \mathbb{Z}^d, q \in S: \ \big(c_0(i) = (q, g_0) \iff c'_0(i) = ((q, \varepsilon), g_0)\big)$$

Sind $\pi_r(N) = 0^d, q_1, \ldots, q_r, \ldots, q_n \in S$ und $h_{r_m} \cdots h_{r_1} g \in \bigcup_{1 \leq i \leq z+1} \Gamma^{\cdot i}$,
dann wird σ' folgendermaßen festgelegt:
$\sigma'\left(((q_1, \ ), \ldots, (q_r, h_{r_m} \cdots h_{r_1}), \ldots, (q_n, \ )), g\right) := ((q, u), v)$, wobei gilt:

$$q := \begin{cases} \pi_1\left(\sigma((q_1, \ldots, q_n), h_{r_m})\right) & \text{falls } m \geq 1 \\ \pi_1\left(\sigma((q_1, \ldots, q_n), g)\right) & \text{sonst} \end{cases}$$

Mit

$$\gamma = g_l \cdots g_2 g_1 := \begin{cases} \pi_2(\sigma((q_1, \ldots, q_n), h_{r_m})) & \text{falls } m \geq 1 \\ \pi_2(\sigma((q_1, \ldots, q_n), g)) & \text{sonst} \end{cases}$$

ist $\quad (u, v) :=$

$$\begin{cases} (g_l \cdots g_2, g_1) & \text{falls } m = 0 \ \wedge \ g_1 = g_0 \\ & \qquad \wedge\, l \leq z \\ (g_l \cdots g_1, \varepsilon) & \text{falls } m = 0 \ \wedge \ g_1 \neq g_0 \\ & \qquad \wedge\, l \leq z \\ (g_l \cdots g_{j+1}, g_j \cdots g_1) & \text{falls } m = 0 \ \wedge \ l - j = z \\ & \qquad \wedge\, 1 \leq j \leq l-1 \\ (g_l \cdots g_{j+1}, g_j \cdots g_1 h_{r_{m-1}} \cdots h_{r_1} g) & \text{falls } m > 0 \ \wedge \ z = l - j \\ & \qquad \wedge\, 1 \leq j \leq l-1 \\ (g_l \cdots g_1 h_{r_{m-1}} \cdots h_{r_{j+1}}, h_{r_j} \cdots h_{r_1} g) & \text{falls } m > 0 \\ & \qquad \wedge\, z = l + m - 1 - j \\ & \qquad \wedge\, 0 \leq j \leq m-1 \\ (g_l \cdots g_1 h_{r_{m-1}} \cdots h_{r_1} g, \varepsilon) & \text{falls } m > 0 \ \wedge \ z \geq l + m \\ & \qquad \wedge\, g \neq g_0 \\ (g_l \cdots g_1 h_{r_{m-1}} \cdots h_{r_1}, g) & \text{falls } m > 0 \ \wedge \ z \geq l + m \\ & \qquad \wedge\, g = g_0 \end{cases}$$

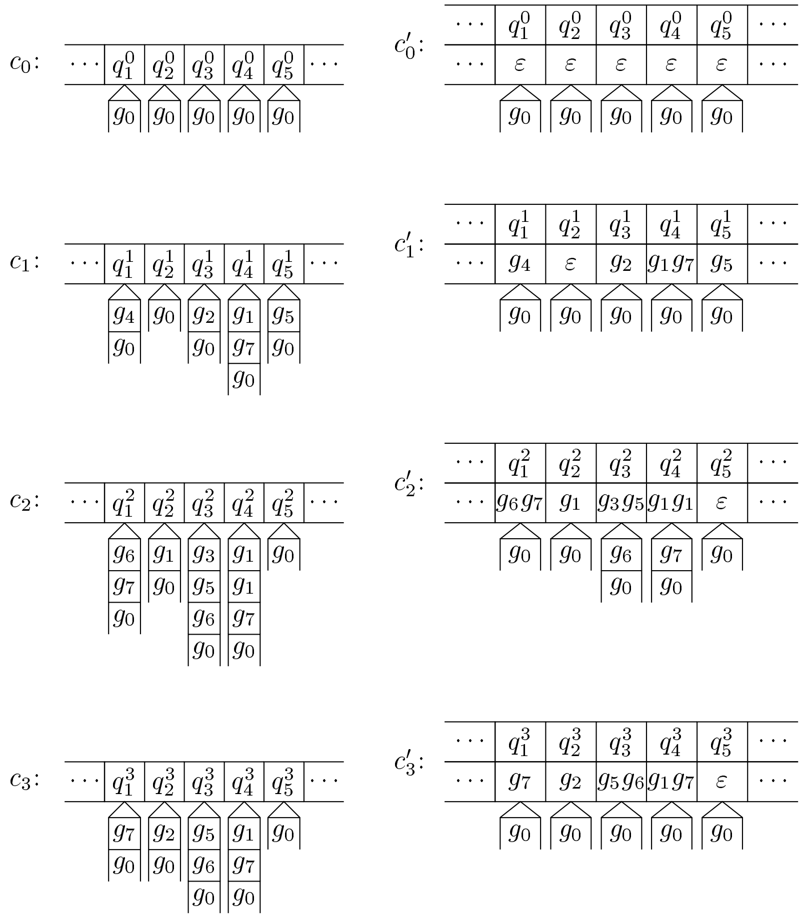


Abbildung 5.3. Konfigurationsübergänge eines Keller-Zellularautomaten mit konstant reduzierter Kellertiefe ($z = 2$).

Die Behauptung $\pi_1(c_t(i)) = \pi_{1,1}(c'_t(i))$ für alle $i \in \mathbb{Z}^d$ und alle $t \in \mathbb{N}_0$ folgt direkt aus der Konstruktion. Sie gilt offensichtlich für $t = 0$. Für $t > 0$ wurden die ersten Komponenten der Folgezustände der Zellen von M' gerade als Folgezustände der entsprechenden Zellen von M definiert.
Weiterhin gilt: $\forall i \in \mathbb{Z}^d : \forall t \in \mathbb{N}_0 : k(i,t) = k'(i,t) + |\pi_{2,1}(c'_t(i))|$.
Es bleibt zu zeigen, daß $|\pi_{2,1}(c'_t(i))|$ in gewissem Sinne maximal ist, d.h., aus $|\pi_{2,1}(c'_t(i))| < z$ folgt $\pi_2(c'_t(i)) = g_0$. Dies gilt offensichtlich für die Anfangskonfiguration. Betrachtet man die Definitionsgleichungen für (u, v) bei der Konstruktion von σ', so ist im ersten und letzten Fall $\pi_2(c'_t(i)) = g_0$. Ausgehend davon, daß die zu zeigende Behauptung im Zeitpunkt t gültig war, läßt sich aus der zweiten und vorletzten Definitionsgleichung $m = z$ ablesen. Anderenfalls wäre wegen $g_1 \neq g_0$ bzw. $g \neq g_0$ die Behauptung falsch gewesen. Somit folgt auch für diese Fälle die Behauptung. In allen anderen Fällen ist $|\pi_2(c'_t(i))| = z$ direkt aus der Definitionsgleichung ablesbar. □

Korollar 5.2. Es sei POLY $\in$ {OPDCA, PDCA} und k eine Kellerkomplexität.

$$\forall z \in \mathbb{N} : {}_k\mathscr{L}(\mathrm{POLY}) = {}_{k-z}\mathscr{L}(\mathrm{POLY})$$

Nachdem nachgewiesen wurde, daß das Kelleralphabet bei Zunahme der Kellerkomplexität bis auf drei Elemente reduziert werden kann, stellt der folgende Satz eine Möglichkeit zur Verfügung, die Kellerkomplexität um einen konstanten Faktor zu verringern, wobei das Kelleralphabet vergrößert wird. Die bei der Reduktion des Kelleralphabets verlorene Zeit kann dabei allerdings nicht zurückgewonnen werden. Andererseits wird durch die Kellerkompression die Zeitkomplexität auch nicht erhöht.

Satz 5.3. Es sei $M = (S, \Gamma, d, N, \sigma, q_0, g_0)$ ein Keller-Zellularautomat. Dann existiert für alle $z \in \mathbb{N}$ ein Keller-Zellularautomat $M' = (S, \Gamma', d, N, \sigma', q_0, g'_0)$, so daß gilt:

$$\forall i \in \mathbb{Z}^d : \forall t \in \mathbb{N}_0 : \left(\pi_1(c_t(i)) = \pi_1(c'_t(i)) \ \wedge \ k'(i,t) = \left\lceil \frac{k(i,t)}{z} \right\rceil + 1\right)$$

Beweis. Durch Konstruktion von M' aus M.

Es sei $g'_0 \notin \Gamma$ ein beliebiges Symbol.

$$\Gamma' := g'_0 \cup \bigcup_{0 \le i \le z} \Gamma^{\cdot i}.$$

Die Anfangskonfiguration c'_0 von M' wird festgelegt durch

$$\forall i \in \mathbb{Z}^d, q \in S : \Big(c_0(i) = (q, g_0) \iff c'_0(i) = (q, (g_0)(g'_0)) \Big)$$

σ' wird folgendermaßen festgelegt:

Ist für $\tilde{q} \in S^{|\tilde{N}|}, q \in S$ und $g, g_1, \ldots, g_m \in \Gamma : \sigma(\tilde{q}, g) = (q, g_m \cdots g_1)$,
dann gilt für alle $g' = (g, g'_e, \ldots, g'_1) \in \Gamma'$:
$\sigma'(\tilde{q}, g') = (q, (g_{az-e+b}, \ldots, g_{az-e+1})(g_{az-e}, \ldots, g_{(a-1)z-e+1}) \cdots$
$\cdots (g_{2z-e}, \ldots, g_{z-e+1})(g_{z-e}, \ldots, g_1, g'_e, \ldots, g'_1)),$
wobei $m + e = az + b$ für $a, b \in \mathbb{N}_0, b < z$ sei.

Der Nachweis, daß M' die gewünschte Eigenschaft hat, kann mit vollständiger Induktion über t erbracht werden. Die Idee, die der Konstruktion zugrundeliegt, ist diejenige, z Kellersymbole von M zusammenzufassen. Dies geschieht naheliegenderweise durch Wortbildung, wobei die Elemente von Γ' Wörter höchstens der Länge z über Γ sind. Bei der Festlegung der Überführungsfunktion σ' wird dabei nur das erste Symbol des obersten Kellereintrages unterschieden. Wesentlich ist, daß die neuen obersten Kellereinträge nur aus Wörtern der Länge z — eventuell mit Ausnahme des obersten Wortes — bestehen. $\square$

Wie bei der Konstruktion zum Satz 5.1 gilt auch hier die Bemerkung zur Festlegung von σ'. Die Konstante 1 bei der Kellertiefe ist wieder auf das zusätzlich eingeführte Symbol q'_0 zurückzuführen.

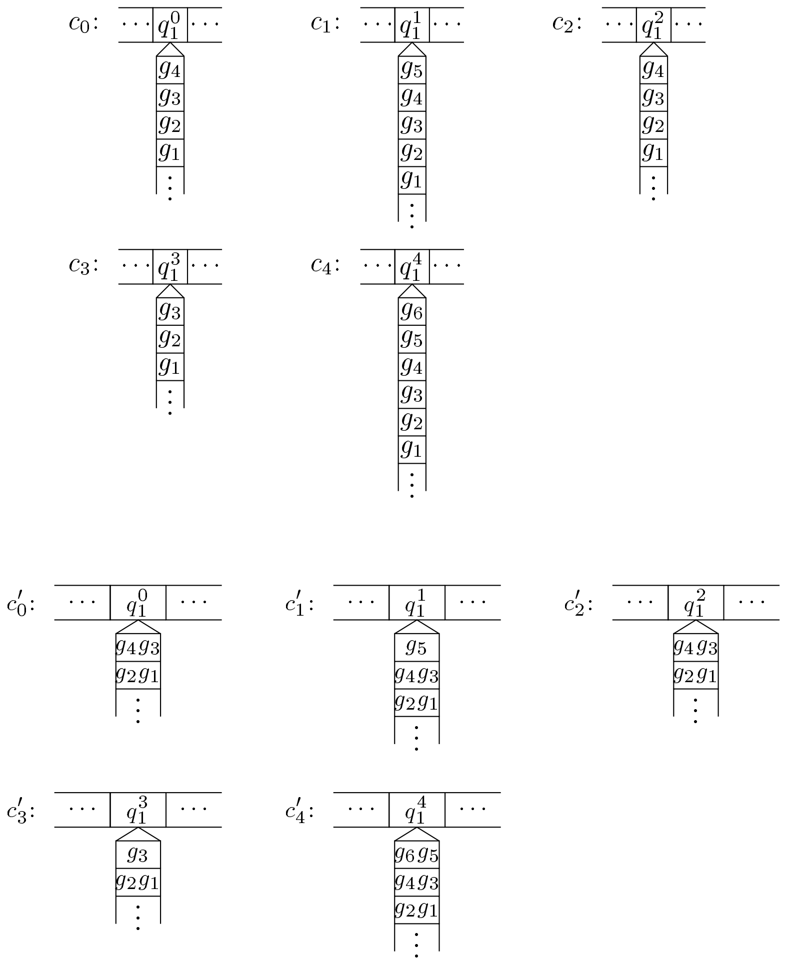


Abbildung 5.4. Konfigurationsübergänge eines Keller-Zellularautomaten mit linear reduzierter Kellertiefe ($z = 2$).

Der folgende Satz faßt die beiden vorangegangenen zusammen und stellt die Verbindung zu den Komplexitätsklassen her.

Satz 5.4. Es sei POLY $\in \{\text{PDCA}, \text{OPDCA}\}$ und k eine Kellerkomplexität.

$$\forall z \in \mathbb{N} : {}_{k}\mathscr{L}(\text{POLY}) = {}_{\frac{1}{z}k}\mathscr{L}(\text{POLY})$$

Beweis. Es sei für $L \in {}_{k}\mathscr{L}(\text{POLY})$ M ein entsprechender Akzeptor. Nach Satz 5.3 existiert dann ein Akzeptor M' mit z-fach komprimierter Kellertiefe. Also gilt für alle Zellen i in jedem Zeitpunkt t: $k'(i,t) = \left\lceil \frac{k(i,t)}{z} \right\rceil + 1 \leq \frac{k(i,t)}{z} + 2$. Demnach ist $L \in {}_{\frac{1}{z}k+2}\mathscr{L}(\text{POLY})$, und mit Korollar 5.2 folgt der Satz. □

5.2.3 Kellernormalisierung

In diesem Unterabschnitt wird das Konzept der Kellernormalisierung für Keller-Zellularautomaten eingeführt und gezeigt, daß man sich im weiteren ohne Beschränkung der Allgemeinheit auf derartige Keller-Zellularautomaten beschränken kann.

Definition 5.9.
Ein Keller-Zellularautomat $M = (S, \Gamma, d, N, \sigma, q_0, g_0)$ heißt *kellernormalisiert*, falls für alle $\gamma \in \Gamma^*, g \in \Gamma$ und $q_1, \ldots, q_n, q' \in S$ gilt:

$$\sigma((q_1, \ldots, q_n), g) = (q', \gamma) \Longrightarrow |\gamma| \leq 2$$

Satz 5.5. Es sei $M = (S, \Gamma, d, N, \sigma, q_0, g_0)$ ein Keller-Zellularautomat. Dann existiert ein kellernormalisierter Keller-Zellularautomat $M' = (S, \Gamma', d, N, \sigma', q_0, g_0)$, so daß gilt:

$$\forall i \in \mathbb{Z}^d : \forall t \in \mathbb{N}_0 : \pi_1(c_t(i)) = \pi_1(c'_t(i))$$

Beweis.
Wählt man $l := \max\{|\gamma| \mid \gamma = \pi_2(\sigma((q_1, \ldots, q_n), g))$ für $q_1, \ldots, q_n \in S$, $g \in \Gamma\}$ und komprimiert die Kellertiefe von M l-fach, so folgt die

Behauptung unmittelbar aus der Konstruktion des Keller-Zellularautomaten mit komprimierter Kellertiefe. □

Korollar 5.3. Es sei M ein Keller-Zellularautomat und L die Sprache, die von ihm mit Zeitkomplexität $t(n)$ akzeptiert wird. Es existiert ein kellernormalisierter Keller-Zellularautomat M', der L mit Zeitkomplexität $t(n)$ akzeptiert.

5.3 (Nicht-)Akzeptanz spezieller Sprachfamilien

Es ist bekannt, daß eine Sprache über einem einelementigen Alphabet genau dann kontextfrei ist, wenn sie regulär ist. Bezüglich derartiger Sprachen führt der Übergang von endlichen Automaten zu Kellerautomaten also nicht zur Vergrößerung der erkennbaren Sprachfamilie. Satz 3.11 hat gezeigt, daß dies auch für Realzeit OCAs gilt. Aber selbst das parallele Modell der unidirektionalen Keller-Zellularautomaten hat hinsichtlich der Realzeiterkennbarkeit solcher Sprachen nur die Fähigkeiten endlicher Automaten. Aus dem folgenden Nachweis dieser Behauptung ergibt sich zugleich ein Beispiel dafür, daß Parallelität nicht in jedem Fall die gewünschten Vorteile mit sich bringt, obwohl im allgemeinen wieder die echte Inklusion $\mathscr{L}_3 \subset \mathscr{L}_{rt}(\mathrm{OPDCA})$ gilt.

Der folgende Satz wird hier der Vollständigkeit halber aufgeführt. Zum Teil recht unterschiedliche Beweise finden sich z.B. bei S. Ginsburg und H. G. Rice (1962), A. Salomaa (1973) und P. Gvozdjak (1993).

Satz 5.6. Eine Sprache über einem Zeichen ist genau dann kontextfrei, wenn sie regulär ist.

Satz 5.7. Es sei M ein OPDCA und L eine formale Sprache über einem einelementigen Alphabet A. Falls L von M in Realzeit akzeptiert wird, dann ist $L \in \mathscr{L}_3$.

Beweis. Es seien $A := \{\mathtt{a}\}$ ein Alphabet, $L \subseteq A^+$ und $M = (S, \Gamma, 1, \bar{H}_1, \sigma, q_0, g_0)$ ein OPDCA, der L mit Endzuständen aus F in Realzeit akzeptiert.
Zunächst wird ein Kellerautomat $B = (S', A, \Gamma, \delta, s_0, g_0, F')$ konstruiert, der L erkennt. Nach dem oben Gesagten ist L dann kontextfrei und mit Satz 5.6 regulär.

$S' := (S \cup \{\mathtt{1}\})^2$ für ein Symbol $\mathtt{1} \notin S$
$s_0 := (\mathtt{1}, \mathtt{1})$
$F' := \{s \in S' |\, \pi_2(s) \in F\}$

δ wird folgendermaßen festgelegt:
$\delta(s_0, \mathtt{a}, g_0) := ((\mathtt{a}, \pi_1(\sigma((\#, \mathtt{a}), g_0))), g_0)$
Es sei $si := \sigma((s, s), g)$. $\quad \forall s, \tilde{s} \in S, g \in \Gamma :$

$$\delta((s, \tilde{s}), \mathtt{a}, g) := ((\pi_1(si), \quad \pi_1(\sigma((\tilde{s}, \pi_1(si)), \pi_2(si)))), \quad \pi_2(si))$$

Behauptung 5.2. $\quad w \in L(M) \iff w \in L(B)$

Beweis.
„$\Longrightarrow$“
Es sei $w = \mathtt{a}^{\cdot n} \in L(M)$ für ein $n \in \mathbb{N}$. Bis zum Zeitpunkt n durchläuft M folgende Konfigurationsfolge:

$$\begin{array}{rl} t = 0 : & \#(\mathtt{a}, g_0)^{\cdot n}\# \\ 0 < t < n : & \#(r_{t_1}, \gamma_{t_1}) \cdots (r_{t_t}, \gamma_{t_t})(s_t, \beta_t)^{\cdot (n-t)}\# \\ t = n : & \#(r_{n_1}, \gamma_{n_1}) \cdots (r_{n_n}, \gamma_{n_n})\# \end{array}$$

Für $0 \leq t \leq n$ können (r_{t_t}, γ_{t_t}) und (s_t, β_t) präzisiert werden durch

$$(r_{t_t}, \gamma_{t_t}) = \begin{cases} \sigma((\#, \mathtt{a}), g_0) & \text{für } t = 1 \\ \sigma((r_{t-1_{t-1}}, s_{t-1}), \beta_{t-1}) & \text{sonst} \end{cases}$$

$$(s_t, \beta_t) = \begin{cases} \sigma((\mathtt{a}, \mathtt{a}), g_0) & \text{für } t = 1 \\ \sigma((s_{t-1}, s_{t-1}), \beta_{t-1}) & \text{sonst} \end{cases}$$

Da w nach Voraussetzung zur Sprache $L(M)$ gehört, muß gelten: $r_{n_n} \in F$.

Wird nun der Automat B auf w angesetzt, ergibt sich aus der Konstruktion von δ die Berechnungsfolge:

$$((\mathtt{1},\mathtt{1}),\mathtt{a}^{\cdot n},g_0) \longmapsto ((\mathtt{a},r_{1_1}),\mathtt{a}^{\cdot(n-1)},g_0) \longmapsto ((s_1,r_{2_2}),\mathtt{a}^{\cdot(n-2)},\beta_1) \overset{n-2}{\longmapsto} ((s_{n-1},r_{n_n}),\varepsilon,\beta_{n-1})$$

Nach n Schritten hält B mit leerem Eingabeband, da die Konstruktion von δ nur Zustandsübergänge bei Eingabe von $\mathtt{a}$ zuläßt. Aus $\pi_2(s_{n-1},r_{n_n}) = r_{n_n} \in F$ folgt $(s_{n-1},r_{n_n}) \in F'$. Also gilt $w \in L(B)$.

„$\Longleftarrow$“
Es sei $w = \mathtt{a}^{\cdot n} \in L(B)$. B angesetzt auf w liefere die Berechnungsfolge:

$((\mathtt{1},\mathtt{1}),\mathtt{a}^{\cdot n},g_0) \longmapsto ((s_1,s_1'),\mathtt{a}^{\cdot(n-1)},\gamma_1) \longmapsto ((s_2,s_2'),\mathtt{a}^{\cdot(n-2)},\gamma_2) \overset{n-2}{\longmapsto}$
$((s_n,s_n'),\varepsilon,\gamma_n)$, wobei $s_n' \in F'$ ist.
Wegen $\delta((\mathtt{1},\mathtt{1}),\mathtt{a},g_0) = ((\mathtt{a},\pi_1(\sigma((\#,\mathtt{a}),g_0))),g_0)$ gilt:
$s_1 = \mathtt{a}$, $s_1' = \pi_1(\sigma((\#,\mathtt{a}),g_0))$ und $\gamma_1 = g_0$.
Aufgrund der Konstruktion von δ folgt aus
$\delta(((s_t,s_t'),\mathtt{a},\gamma_t)) = ((s_{t+1},s_{t+1}'),\gamma_{t+1})$ für $t > 1$:
$s_{t+1} = \pi_1(\sigma((s_t,s_t),\gamma_t))$, $s_{t+1}' = \pi_1(\sigma((s_t',s_{t+1}),\gamma_{t+1}))$ und
$\gamma_{t+1} = \pi_2(\sigma((s_t,s_t),\gamma_t))$.

Daraus ergibt sich
$s_2 = \pi_1(\sigma((\mathtt{a},\mathtt{a}),g_0))$, $s_2' = \pi_1(\sigma((s_1',s_2),\gamma_2))$ und
$\gamma_2 = \pi_2(\sigma((\mathtt{a},\mathtt{a}),g_0))$.
Also durchläuft M die Konfigurationsfolge:

$$\begin{array}{rl}
t=0: & \#(\mathtt{a},g_0)^{\cdot n}\# \\
t=1: & \#(s_1',\ )(s_2,\gamma_2)^{\cdot(n-1)}\# \\
t=2: & \#(\ ,\ )(s_2',\ )(s_3,\gamma_3)^{\cdot(n-2)}\# \\
 & \vdots \\
t=n-1: & \#(\ ,\ )\cdots(s_{n-1}',\ )(s_n,\gamma_n)\# \\
t=n: & \#(\ ,\ )\cdots(s_n',\ )\#
\end{array}$$

Da s'_n nach Voraussetzung aus F' ist, akzeptiert M das Wort w in Realzeit. Aus „$\Longleftarrow$“ und „$\Longrightarrow$“ folgt die Behauptung. □

Aus dem Satz folgt unmittelbar, daß nichtreguläre Sprachen über einem einelementigen Alphabet nicht von einem OPDCA in Realzeit akzeptiert werden können.

Für den Beweis des nächsten Satzes sind zunächst einige Vorbetrachtungen notwendig. Die interessierende Frage ist, wie sich ein beliebiger, nie haltender Kellerautomat verhält, der eine konstante Eingabe erhält. Aus Satz 5.6 läßt sich folgern, daß der Automat ein zyklisches Verhalten aufweisen muß.

Diese Frage ist gleichbedeutend mit der Frage nach dem Verhalten gewisser Zellen eines OPDCA bei Eingabe eines hinreichend langen Wortes über einem einelementigen Alphabet. Insbesondere ist dabei der Kellerinhalt der Zellen von Interesse.

Bei den weiteren Betrachtungen wird immer davon ausgegangen, die Automaten seien kellernormalisiert, was nach Satz 5.5 keine Einschränkung der Allgemeinheit bedeutet.

Bei der Analyse des Verhaltens lassen sich zwei Fälle unterscheiden.
1) Während der gesamten Berechnung ist die Kellertiefe durch eine Konstante $z \in \mathbb{N}$ beschränkt. In diesem Fall wird der einzelne Automat nach spätestens $|S| \cdot |\Gamma|^z$ Zeitschritten ein zyklisches Verhalten aufweisen.
2) Die Kellertiefe ist nicht beschränkt, sondern wird im Laufe einer beliebig langen Berechnung beliebig groß. In diesem Fall gilt folgende

Behauptung 5.3. Für einen einzelnen Automaten i, dessen Kellertiefe im Laufe einer Berechnung bei beliebig langer konstanter Eingabe nicht beschränkt ist, gilt:

$$\forall u \in \mathbb{N} : |k^{-1}(i,u)| < \infty$$

Beweis. Angenommen, es existiert eine Kellertiefe u, die die Zelle i unendlich oft aufweist. Es seien die entsprechenden Zeitpunkte chronologisch geordnet und durchnumeriert: $t_{u_1}, t_{u_2}, \ldots, t_{u_r}, t_{u_{r+1}}, \ldots$ Spätestens im Zeitpunkt t_{u_r} mit $u_r = |\Gamma|^u \cdot |S|$ gilt:

$$\exists t \in \{t_{u_1}, \ldots, t_{u_{r-1}}\} : c_t(i) = c_{t_{u_r}}(i).$$

D.h., der Automat durchläuft eine Konfiguration erneut. Aus diesem zyklischen Verhalten folgt, daß die Kellertiefe durch $\max\{k(i,t) | \, t \leq t_{u_r}\}$ beschränkt ist. Dies ist ein Widerspruch zur Voraussetzung. $\square$

Es kann also davon ausgegangen werden, daß eine Zelle i die Kellertiefe u im Zeitpunkt $\max(k^{-1}(i,u))$ „das letzte Mal" aufweist. Um ein zyklisches Gesamtverhalten nachzuweisen, wird weiterhin eine Abschätzung für $\max(k^{-1}(i,u+1)) - \max(k^{-1}(i,u))$ benötigt.

Behauptung 5.4. $\forall \max(k^{-1}(i,u)) < t \leq \max(k^{-1}(i,u+1)) : k(i,t) \leq u + |S| \cdot |\Gamma|$

Beweis. Angenommen, es existiert ein Zeitpunkt t aus dem Intervall der Behauptung, so daß gilt: $k(i,t) > u + |S| \cdot |\Gamma|$. Daraus folgt für alle Kellertiefen u' mit $u \leq u' \leq u + |S| \cdot |\Gamma|$: Es existiert ein maximales $t' \leq t$ mit $|\pi_2(c_{t'}(i))| = u' \;\wedge\; |\pi_2(c_{t'+1}(i))| > u'$.

Die Mächtigkeit des Wertebereichs der Überführungsfunktion bei konstanter Eingabe ist aber gerade $|S| \cdot |\Gamma|$. Also sind für mindestens zwei der Zeitpunkte t' zugleich das oberste Kellersymbol und der Zustand identisch. Da die t' bezüglich der Kellertiefe maximal gewählt wurden, wird der Automat in einen Zyklus geraten und die Kellertiefe $u+1$ nicht mehr erreichen. Dies ist ein Widerspruch zur Voraussetzung $t \leq \max(k^{-1}(i,u+1))$. $\square$

Aus der letzten Behauptung läßt sich nun die etwas grobe, aber für das Folgende ausreichende Abschätzung ableiten:

$$\max(k^{-1}(i,u+1)) - \max(k^{-1}(i,u)) \leq |S| \cdot |\Gamma|^{|S| \cdot |\Gamma|}$$

Insgesamt wird der Kellerinhalt zyklisch zunehmen, da eine Zelle höchstens $|S| \cdot |\Gamma|$ verschiedene Möglichkeiten hat, eine Kellertiefe „das letzte Mal" zu erreichen. Das heißt, ausgehend von einer Kellertiefe

u, wird spätestens, nachdem die Kellertiefe $u + |S| \cdot |\Gamma|$ „das letzte Mal" erreicht wurde, die Beschriftung des Kellers zyklisch.

Es läßt sich also feststellen, daß das Verhalten eines den Voraussetzungen entsprechenden Automaten Zyklen entspricht, deren Länge höchstens $|S| \cdot |\Gamma| \cdot |S| \cdot |\Gamma|^{|S| \cdot |\Gamma|}$ Zeitschritte beträgt. Insbesondere hängt die Länge nur von der Wahl der Alphabete S und Γ ab.

Satz 5.8. Es sei $f : \mathbb{N} \longrightarrow \mathbb{N}$ eine Abbildung mit der Eigenschaft

$$\lim_{n \to \infty} \frac{n^{(n+1)^2}}{f(n)} = 0,$$

dann ist $L = \{\mathtt{b}^{\cdot n}\mathtt{a}^{\cdot f(n)} | \, n \in \mathbb{N}\} \notin \mathscr{L}_{rt}(\mathrm{OPDCA})$.

Beweis. Angenommen, es existiert ein OPDCA $M = (S, \Gamma, 1, \bar{H}_1, \sigma, q_0, g_0)$, der L in Realzeit erkennt. O.B.d.A. kann wieder davon ausgegangen werden, M sei kellernormalisiert. Abbildung 5.5 zeigt schematisch die Anfangskonfiguration und eine Konfiguration im Zeitpunkt $t = m + 1 < n + f(n)$. Die stark umrandeten Bereiche beinhalten die für die weitere Berechnung relevante Information.

Der rechte Bereich — dargestellt durch $f(n) - m - 1$ Zellen im Zustand $\mathtt{a}$ — repräsentiert einzelne Automaten, die sich genau so verhalten müssen, wie zuvor für Automaten mit konstanter Eingabe gezeigt wurde. Das heißt, eine genügend lange Eingabe vorausgesetzt, werden sie nach spätestens $z(|S|, |\Gamma|)$ Zeitschritten Zyklen mit höchstens dieser Länge durchlaufen. z ist dabei eine nur von $|S|$ und $|\Gamma|$ abhängige Konstante.

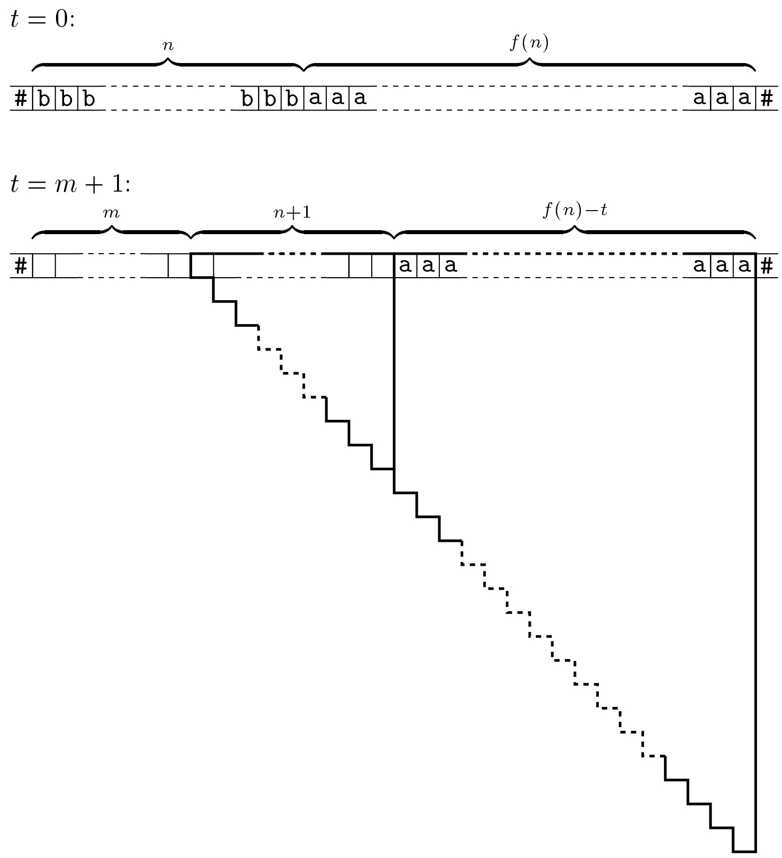


Abbildung 5.5. Konfigurationen zum Beweis von Satz 5.8.

Mit der so festgelegten Darstellungsweise läßt sich die Vorstellung verbinden, der mittlere Bereich — bestehend aus $n + 1$ Zellen — wandere in jedem Zeitschritt um eine Zelle nach rechts. Eine grobe Abschätzung ergibt, daß er — bzw. die in ihm enthaltene relevante Information — $|S|^{n+1} \cdot |\Gamma|^{n^2}$ verschiedene Gesamtzustände annehmen kann. Faßt man die Gesamtzustände als Zustände einer „Superzelle" auf, so ergibt sich, daß diese bei ihrer Wanderung nach rechts auf die sich zyklisch verhaltenden Zellen des rechten Bereichs trifft. Folglich wird auch diese „Superzelle“ nach höchstens $z(|S|, |\Gamma|) \cdot |S|^{n+1} \cdot |\Gamma|^{n^2}$ Zeitschritten ein zyklisches Verhalten mit höchstens dieser Länge aufweisen. Diese sei mit $z'(n, |S|, |\Gamma|)$ bezeichnet.

Aus

$$\begin{aligned}
& z \cdot |S|^{n+1} \cdot |\Gamma|^{n^2} \\
& \leq z \cdot \max\{|S|, |\Gamma|\}^{n^2+n+1} \\
& = z \cdot \max\{|S|, |\Gamma|\}^{(n+1)^2-n} \\
& \leq z \cdot \max\{|S|, |\Gamma|\}^{(n+1)^2} \\
& \leq (z \cdot \max\{|S|, |\Gamma|\})^{(n+1)^2}
\end{aligned}$$

und der Voraussetzung

$$\lim_{n \to \infty} \frac{n^{(n+1)^2}}{f(n)} = 0$$

folgt

$$\forall |S|, |\Gamma| \in \mathbb{N} : \exists n \in \mathbb{N} : \forall i \in \mathbb{N} : f(n+i) > (z \cdot \max\{|S|, |\Gamma|\})^{(n+i+1)^2}.$$

Insbesondere gilt mit den entsprechenden Werten

$$f(n+i) > z'(n+i, |S|, |\Gamma|).$$

Es lassen sich also unendlich viele Wörter $w \in L$ finden, bei deren Erkennung diese Zyklen mindestens einmal vollständig durchlaufen würden. Das hieße aber, daß neben einem solchen Wort $\mathtt{b}^{\cdot n}\mathtt{a}^{\cdot f(n)} \in L$ auch $\mathtt{b}^{\cdot n}\mathtt{a}^{\cdot f(n)}\mathtt{a}^{\cdot z'(n,|S|,|\Gamma|)}$ erkannt würde. Dies ist ein Widerspruch zur Annahme. $\square$

Eigenschaften lassen sich gelegentlich anhand konkreter Sprachen nachweisen. Die beiden folgenden Sätze werden im weiteren öfter benötigt werden.

Satz 5.9. $L = \{(\mathtt{ab})^{\cdot(n^2)} | \, n \in \mathbb{N}\} \in \mathscr{L}_{rt}(\mathrm{OPDCA})$

Beweis. Die Gültigkeit des Satzes wird durch die Konstruktion eines entsprechenden OPDCA $M = (S, \Gamma, 1, \bar{H}_1, \sigma, q_0, g_0)$ und dessen Korrektheit nachgewiesen.

Der Algorithmus beruht darauf, daß die Differenz zwischen zwei aufeinanderfolgenden Quadratzahlen jeweils um zwei wächst $((x+1)^2 - x^2 = (x^2 - (x-1)^2) + 2$ für $x \geq 1)$.

Jeweils zwei benachbarte Zellen mit dem Eingabesymbol a und b schichten ihre Kellerinhalte fortlaufend um, wobei bei jeder Umschichtung ein zusätzliches Symbol in den Keller geschrieben wird. (Bei zwei Umschichtungen also gerade zwei zusätzliche Symbole.) Das Eingabewort wird genau dann akzeptiert, wenn ein zu Beginn der Berechnung am linken Rand gestartetes Signal auf eine Zelle mit der ursprünglichen Eingabe b am rechten Rand trifft und der Keller deren Nachbarzelle in diesem Zeitpunkt leer ist.

In Abbildung 5.6 ist die Konfigurationsfolge für die Eingabe $(\mathtt{ab})^8$ dargestellt.

Konstruktion des OPDCA:

$S := \{\mathtt{\#}, \mathtt{a}, \mathtt{b}, \mathtt{s}, \mathtt{r}\} \times \{\mathtt{\#}, \mathtt{T}, \mathtt{pu}, \mathtt{po}, \mathtt{e}, \mathtt{-}\}$
$\Gamma := \{\mathtt{m}, \mathtt{g}\}$
$q_0 := (\mathtt{a}, \mathtt{T})$
$g_0 = \mathtt{g}$,
Grenzzustand: $(\mathtt{\#}, \mathtt{\#})$
$F = \{s \in S | \, \pi_2(s) = \mathtt{T}\}$
$\bar{H}_1 = (-1, 0)$

Die Anfangskonfiguration c_0 von M bei Eingabe des Wortes $w_1 \cdots w_n \in \{\mathtt{a}, \mathtt{b}\}^+$ wird für alle i aus der Retina festgelegt durch:
$c_0(i) := ((w_i, \mathtt{e}), \mathtt{g})$

σ wird wie folgt festgelegt:
$\sigma((p_1, p_2), (q_1, q_2), h) = ((q'_1, q'_2), \gamma)$, wobei gilt:

$$q_1' := \begin{cases} \mathtt{r} & \text{falls } (p_1 = \mathtt{\#} \ \wedge \ q_1 = \mathtt{a}) \ \vee \ (p_1 = \mathtt{s} \ \wedge \ q_1 = \mathtt{a}) \\ \mathtt{s} & \text{falls } p_1 = \mathtt{r} \ \wedge \ q_1 = \mathtt{b} \\ q_1 & \text{sonst} \end{cases}$$

$$q_2' := \begin{cases} \mathtt{-} & \text{falls } p_1 = \mathtt{\#} \ \vee \ (p_1 = \mathtt{b} \ \wedge \ p_2 = \mathtt{e}) \\ & \qquad \vee \ (q_2 = \mathtt{po} \ \wedge \ h = \mathtt{g}) \\ \mathtt{T} & \text{falls } (p_1 = \mathtt{r} \ \wedge \ p_2 = \mathtt{-} \ \wedge \ q_1 = \mathtt{b}) \ \vee \ q_2 = \mathtt{T} \\ \mathtt{pu} & \text{falls } (p_1 = \mathtt{a} \ \wedge \ p_2 = \mathtt{e}) \ \vee \ q_2 = \mathtt{-} \\ \mathtt{po} & \text{falls } p_2 = \mathtt{-} \ \wedge \ (p_1 \neq \mathtt{r} \ \vee \ q_1 \neq \mathtt{b}) \\ q_2 & \text{sonst} \end{cases}$$

$$\gamma := \begin{cases} \mathtt{m}h & \text{falls } (q_1 = \mathtt{b} \ \wedge \ p_2 = \mathtt{-}) \ \vee \ (q_2 = \mathtt{pu} \ \wedge \ p_2 = \mathtt{po}) \\ \varepsilon & \text{falls } (q_2 = \mathtt{po} \ \wedge \ h = \mathtt{m}) \\ & \qquad \vee \ (p_2 = \mathtt{-} \ \wedge \ q_1 = \mathtt{a} \ \wedge \ q_2 = \mathtt{pu}) \\ h & \text{sonst} \end{cases}$$

Zum Nachweis der Korrektheit der Konstruktion werden nun mehrere Behauptungen gezeigt.

Behauptung 5.5. $w \in L(M) \Longrightarrow w \in \{(\mathtt{ab})^{\cdot n} | \, n \in \mathbb{N}\}$

Beweis. Es sei $w \in L(M)$ mit $|w| = n$, dann gilt:
$\pi_{2,1}(c_t(n)) = \mathtt{T} \ \wedge \ \pi_{2,1}(c_{t-1}(n)) \neq \mathtt{T}$
$\Rightarrow \pi_{1,1}(c_{t-1}(n-1)) = \mathtt{r} \ \wedge \ \pi_{1,1}(c_{t-1}(n)) = \mathtt{b} \Rightarrow \pi_{1,1}(c_{t-2}(n-1)) \neq \mathtt{r}$
$\Rightarrow \pi_{1,1}(c_{t-2}(n-2)) = \mathtt{s} \ \wedge \ \pi_{1,1}(c_{t-2}(n-1)) = \mathtt{a}$
$\Rightarrow \pi_{1,1}(c_{t-3}(n-2)) \neq \mathtt{s}$

$\vdots$

$\Rightarrow \pi_{1,1}(c_{t-(n-1)}(1)) = \mathtt{r} \wedge \pi_{1,1}(c_{t-(n-1)}(2)) = \mathtt{b} \Rightarrow \pi_{1,1}(c_{t-n}(1)) \neq \mathtt{r}$
$\Rightarrow \pi_{1,1}(c_{t-n}(0)) = \mathtt{\#} \ \wedge \ \pi_{1,1}(c_{t-n}(1)) = \mathtt{a}$

Weiterhin folgt $\pi_{2,1}(c_t(n)) = \mathtt{T} \Rightarrow \forall i \in \mathbb{N}_0 : \ \pi_{2,1}(c_{t+i}(n)) = \mathtt{T}$. Es sei l eine Zelle der Retina. Wegen $\pi_{1,1}(c_t(l)) = \mathtt{b} \Rightarrow \forall 0 \leq i \leq t : \pi_{1,1}(c_i(l)) = \mathtt{b}$ und $\pi_{1,1}(c_t(l)) = \mathtt{a} \Rightarrow \forall 0 \leq i \leq t : \ \pi_{1,1}(c_i(l)) = \mathtt{a}$ folgt $w \in \{(\mathtt{ab})^{\cdot n} | \, n \in \mathbb{N}\}$. □

Behauptung 5.6. Es sei $w \in L(M)$, dann wird w von M in Realzeit erkannt.

Beweis. Aus dem Beweis der vorangegangenen Behauptung entnimmt man $\pi_{1,1}(c_{t-n}(1)) \neq \mathtt{r}$. Aus $\sigma((\mathtt{\#},\mathtt{\#}),(\mathtt{a},\mathtt{e}),\mathtt{g}) = ((\mathtt{r},\mathtt{-}),\mathtt{g})$ folgt $\pi_{1,1}(c_1(1)) = \mathtt{r}$ und somit $t-n=0 \Rightarrow t=n$. Also wird w in Realzeit erkannt. $\square$

Behauptung 5.7. Es sei $w \in \{(\mathtt{ab})^{\cdot n} \mid n \in \mathbb{N}\}$. Dann gilt:
$\forall i \in \mathbb{N}$ mit $1 \leq i^2 \leq \frac{n}{2}$:

$$\Bigg(t = 2i^2 - 1 \iff \Bigg(c_t(t) = ((\mathtt{r},\mathtt{-}),\mathtt{g}) \;\wedge\; \forall j \in \mathbb{N} \text{ mit } t < j \leq n: \quad c_t(j) = \begin{cases} ((\mathtt{b},\mathtt{pu}),\mathtt{m}^{\cdot(2i-2)}\mathtt{g}) & \text{falls } j \text{ gerade} \\ ((\mathtt{a},\mathtt{-}),\mathtt{g}) & \text{falls } j \text{ ungerade} \end{cases}\Bigg)\Bigg)$$

Beweis.
„$\Longrightarrow$“ Induktion über i:

$i = 1$:
$w \in \{(\mathtt{ab})^{\cdot n} \mid n \in \mathbb{N}\} \Rightarrow \forall j$ mit $1 \leq j \leq n$:

$$c_0(j) = \begin{cases} ((\mathtt{b},\mathtt{e}),\mathtt{g}) & \text{falls } j \text{ gerade} \\ ((\mathtt{a},\mathtt{e}),\mathtt{g}) & \text{falls } j \text{ ungerade} \end{cases}$$

$\Rightarrow c_1(1) = ((\mathtt{r},\mathtt{-}),\mathtt{g}) \;\wedge\; \forall j$ mit $1 < j \leq n$:

$$c_1(j) = \begin{cases} ((\mathtt{b},\mathtt{pu}),\mathtt{g}) & \text{falls } j \text{ gerade} \\ ((\mathtt{a},\mathtt{-}),\mathtt{g}) & \text{falls } j \text{ ungerade} \end{cases}$$

$i \longrightarrow i+1$:
Es sei also $2(i+1)^2 - 1 \leq n$. Offensichtlich gilt für eine Zelle l der Retina:
$\big(\pi_{1,1}(c_0(l)) = \mathtt{b} \Rightarrow \forall 0 \leq t < l : \pi_{1,1}(c_t(l)) = \mathtt{b}\big)$ und ebenso
$\big(\pi_{1,1}(c_0(l)) = \mathtt{a} \Rightarrow \forall 0 \leq t < l : \pi_{1,1}(c_t(l)) = \mathtt{a}\big)$.

Nach Induktionsannahme gilt für $t = 2i^2 - 1$:

$c_t(t) = ((\mathtt{r},\mathtt{-}),\mathtt{g}) \;\wedge\; \forall j$ mit $t < j \leq n$:

$$c_t(j) = \begin{cases} ((\mathtt{b},\mathtt{pu}),\mathtt{m}^{\cdot(2i-2)}\mathtt{g}) & \text{falls } j \text{ gerade} \\ ((\mathtt{a},\mathtt{-}),\mathtt{g}) & \text{falls } j \text{ ungerade} \end{cases}$$

$\Rightarrow c_{t+1}(t+1) = ((\mathtt{s},\mathtt{T}),\mathtt{m}^{\cdot(2i-1)}\mathtt{g}) \ \wedge \ \forall j \text{ mit } t+1 < j \leq n:$

$$c_{t+1}(j) = \begin{cases} ((\mathtt{b},\mathtt{po}),\mathtt{m}^{\cdot(2i-1)}\mathtt{g}) & \text{falls } j \text{ gerade} \\ ((\mathtt{a},\mathtt{pu}),\mathtt{g}) & \text{falls } j \text{ ungerade} \end{cases}$$

$\Rightarrow c_{t+2}(t+2) = ((\mathtt{r},\mathtt{pu}),\mathtt{mg}) \ \wedge \ \forall j \text{ mit } t+2 < j \leq n:$

$$c_{t+2}(j) = \begin{cases} ((\mathtt{b},\mathtt{po}),\mathtt{m}^{\cdot(2i-2)}\mathtt{g}) & \text{falls } j \text{ gerade} \\ ((\mathtt{a},\mathtt{pu}),\mathtt{mg}) & \text{falls } j \text{ ungerade} \end{cases}$$

$\overset{2i-2}{\Longrightarrow} c_{t+2i}(t+2i) = ((\mathtt{r},\mathtt{pu}),\mathtt{m}^{\cdot(2i-1)}\mathtt{g}) \ \wedge \ \forall j \text{ mit } t+2i < j \leq n:$

$$c_{t+2i}(j) = \begin{cases} ((\mathtt{b},\mathtt{po}),\mathtt{g}) & \text{falls } j \text{ gerade} \\ ((\mathtt{a},\mathtt{pu}),\mathtt{m}^{\cdot(2i-1)}\mathtt{g}) & \text{falls } j \text{ ungerade} \end{cases}$$

$\Rightarrow c_{t+2i+1}(t+2i+1) = ((\mathtt{s},\mathtt{-}),\mathtt{g}) \ \wedge \ \forall j \text{ mit } t+2i+1 < j \leq n:$

$$c_{t+2i+1}(j) = \begin{cases} ((\mathtt{b},\mathtt{-}),\mathtt{g}) & \text{falls } j \text{ gerade} \\ ((\mathtt{a},\mathtt{pu}),\mathtt{m}^{\cdot(2i)}\mathtt{g}) & \text{falls } j \text{ ungerade} \end{cases}$$

$\Rightarrow c_{t+2i+2}(t+2i+2) = ((\mathtt{r},\mathtt{po}),\mathtt{m}^{\cdot(2i-1)}\mathtt{g}) \wedge \forall j \text{ mit } t+2i+2 < j \leq n:$

$$c_{t+2i+2}(j) = \begin{cases} ((\mathtt{b},\mathtt{pu}),\mathtt{g}) & \text{falls } j \text{ gerade} \\ ((\mathtt{a},\mathtt{po}),\mathtt{m}^{\cdot(2i-1)}\mathtt{g}) & \text{falls } j \text{ ungerade} \end{cases}$$

$\Rightarrow c_{t+2i+3}(t+2i+3) = ((\mathtt{s},\mathtt{pu}),\mathtt{mg}) \ \wedge \ \forall j \text{ mit } t+2i+3 < j \leq n:$

$$c_{t+2i+3}(j) = \begin{cases} ((\mathtt{b},\mathtt{pu}),\mathtt{mg}) & \text{falls } j \text{ gerade} \\ ((\mathtt{a},\mathtt{po}),\mathtt{m}^{\cdot(2i-2)}\mathtt{g}) & \text{falls } j \text{ ungerade} \end{cases}$$

$\overset{2i-2}{\Longrightarrow} c_{t+4i+1}(t+4i+1) = ((\mathtt{s},\mathtt{pu}),\mathtt{m}^{\cdot(2i-1)}\mathtt{g}) \ \wedge \ \forall j \text{ mit } t+4i+1 < j \leq n:$

$$c_{t+4i+1}(j) = \begin{cases} ((\mathtt{b},\mathtt{pu}),\mathtt{m}^{\cdot(2i-1)}\mathtt{g}) & \text{falls } j \text{ gerade} \\ ((\mathtt{a},\mathtt{po}),\mathtt{g}) & \text{falls } j \text{ ungerade} \end{cases}$$

$\Rightarrow c_{t+4i+2}(t+4i+2) = ((\mathtt{r},\mathtt{-}),\mathtt{g}) \ \wedge \ \forall j \text{ mit } t+4i+2 < j \leq n:$

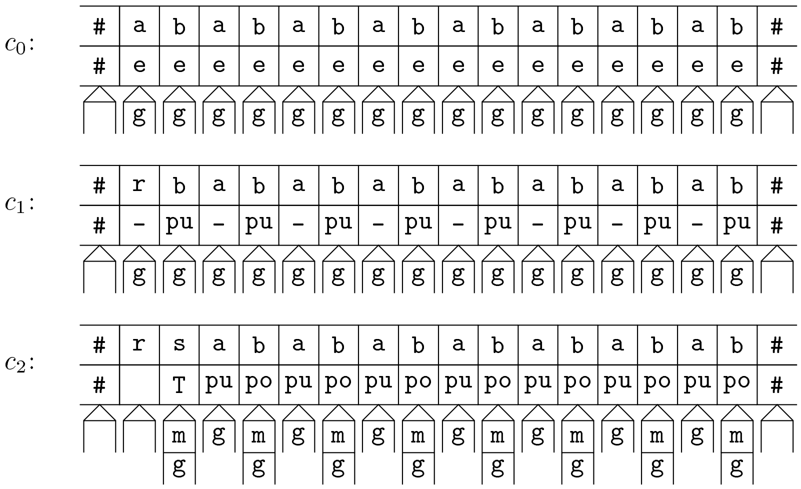
$$c_{t+4i+2}(j) = \begin{cases} ((\mathtt{b},\mathtt{pu}),\mathtt{m}^{\cdot(2i)}\mathtt{g}) & \text{falls } j \text{ gerade} \\ ((\mathtt{a},\mathtt{-}),\mathtt{g}) & \text{falls } j \text{ ungerade} \end{cases}$$

Aus $t+4i+2 = 2(i+1)^2-1$ folgt „$\Longrightarrow$“. Aus obiger Ableitung folgt

$\forall i \in \mathbb{N}$ mit $1 \leq i^2 \leq \frac{n}{2}$: $\big(t \neq 2i^2 - 1 \Rightarrow \pi_1(c_t(t)) \neq (\texttt{r}, \texttt{-})\big)$ und somit „$\Longleftarrow$“. $\square$

Beweis des Satzes.
$w \in L \Rightarrow |w| = 2i^2$, für $i \in \mathbb{N}$. Weiterhin gilt nach Behauptung 5.7:
$c_{2i^2-1}(2i^2 - 1) = ((\texttt{r}, \texttt{-}), \texttt{g})$ und $c_{2i^2-1}(2i^2) = ((\texttt{b}, \texttt{pu}), \texttt{m}^{\cdot(2i-2)}\texttt{g})$.
Also $c_{2i^2}(2i^2) = ((\texttt{s}, \texttt{T}), \texttt{m}^{\cdot(2i-1)}\texttt{g})$.
Damit ist $\pi_1(c_{2i^2}(2i^2)) \in F$ und mit $\pi_{2,1}(c_t(n)) = \texttt{T} \Rightarrow \forall j \in \mathbb{N}_0$: $\pi_{2,1}(c_{t+1}(n)) = \texttt{T}$ folgt, daß M das Wort w erkennt. Nach Behauptung 5.6 wird w in Realzeit erkannt.

Es sei $w \notin L$. Falls $w \notin \{(\texttt{ab})^{\cdot n} | \; n \in \mathbb{N}\}$, dann wird w nach Behauptung 5.5 nicht erkannt. Anderenfalls gelte $|w| = n$ mit $n \neq 2i^2$ für ein $i \in \mathbb{N}$. Falls w erkannt wird, dann in Realzeit. Also sind $\pi_{2,1}(c_n(n)) = \texttt{T}$ und $\pi_{2,1}(c_{n-1}(n)) \neq \texttt{T}$. Dann muß gelten $\pi_1(c_{n-1}(n-1)) = (\texttt{r}, \texttt{-})$, und aus dem Beweis der Behauptung 5.7 folgt $n - 1 = 2i^2 - 1$, also $n = 2i^2$. Dies ist ein Widerspruch zur Annahme. $\square$



c_3:

#	r	s	r	b	a	b	a	b	a	b	a	b	a	b	a	b	#
#			pu	po	pu	po	pu	po	pu	po	pu	po	pu	po	pu	po	#
			m	g	m	g	m	g	m	g	m	g	m	g	m	g	
			g		g		g		g		g		g		g		

c_4:

#	r	s	r	s	a	b	a	b	a	b	a	b	a	b	a	b	#
#				–	pu	–	pu	–	pu	–	pu	–	pu	–	pu	–	#
				g	m	g	m	g	m	g	m	g	m	g	m	g	
					m		m		m		m		m		m		
					g		g		g		g		g		g		

c_5:

#	r	s	r	s	r	b	a	b	a	b	a	b	a	b	a	b	#
#					po	pu	po	pu	po	pu	po	pu	po	pu	po	pu	#
					m	g	m	g	m	g	m	g	m	g	m	g	
					g		g		g		g		g		g		

c_6:

#	r	s	r	s	r	s	a	b	a	b	a	b	a	b	a	b	#
#						pu	po	pu	po	pu	po	pu	po	pu	po	pu	#
						m	g	m	g	m	g	m	g	m	g	m	
						g		g		g		g		g		g	

c_7:

#	r	s	r	s	r	s	r	b	a	b	a	b	a	b	a	b	#
#							–	pu	–	pu	–	pu	–	pu	–	pu	#
							g	m	g	m	g	m	g	m	g	m	
								m		m		m		m		m	
								g		g		g		g		g	

c_8:

#	r	s	r	s	r	s	r	s	a	b	a	b	a	b	a	b	#
#								T	pu	po	pu	po	pu	po	pu	po	#
								m	g	m	g	m	g	m	g	m	
								m		m		m		m		m	
								m		m		m		m		m	
								g		g		g		g		g	

c_9:

#	r	s	r	s	r	s	r	s	r	b	a	b	a	b	a	b	#
#									pu	po	pu	po	pu	po	pu	po	#
									m	m	m	m	m	m	m	m	
									g	m	g	m	g	m	g	m	
										g		g		g		g	

c_{10}:

#	r	s	r	s	r	s	r	s	r	s	a	b	a	b	a	b	#
#										po	pu	po	pu	po	pu	po	#
										m	m	m	m	m	m	m	
										g	m	g	m	g	m	g	
											g		g		g		

c_{11}:

#	r	s	r	s	r	s	r	s	r	s	r	b	a	b	a	b	#
#											pu	po	pu	po	pu	po	#
											m	g	m	g	m	g	
											m		m		m		
											m		m		m		
											g		g		g		

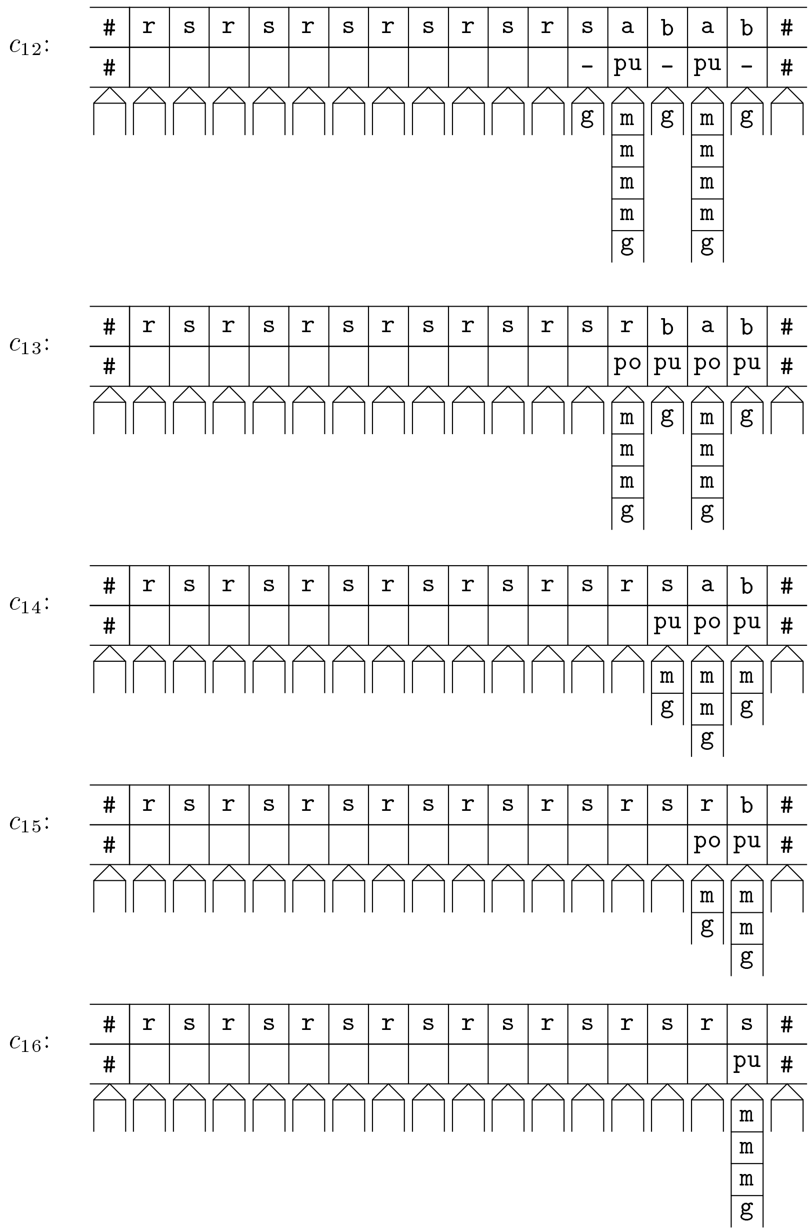


Abbildung 5.6. Beispiel einer Konfigurationsfolge des Akzeptors für $\{(\mathtt{ab})^{\cdot(n^2)} \mid n \in \mathbb{N}\}$.

Satz 5.10. $L = \{\mathtt{a}^{\cdot(2^{2^n})}\mathtt{b}^{\cdot n} |\, n \in \mathbb{N}\} \in \mathscr{L}_{rt}(\text{OPDCA})$

Beweis. Konstruktion eines OPDCA M, der L in Realzeit akzeptiert.

$$S := \{\mathtt{\#}, \mathtt{e}, \mathtt{1}, \mathtt{0}, \mathtt{*}, \mathtt{!}, \mathtt{T}, \mathtt{F}\} \times \{\mathtt{\#}, \mathtt{e}, \mathtt{pu}, \mathtt{po}\} \times \{\mathtt{\#}, \mathtt{a}, \mathtt{b}, \mathtt{r}\} \times \{\mathtt{\#}, \mathtt{e}, \mathtt{0}, \mathtt{1}, \mathtt{2}, \mathtt{3}, \mathtt{4}\}$$

$\Gamma := \{\mathtt{m}, \mathtt{g}\}$
$q_0 := (\mathtt{T}, \mathtt{e}, \mathtt{r}, \mathtt{e})$
$g_0 := \mathtt{g}$
Grenzzustand : $(\mathtt{\#}, \mathtt{\#}, \mathtt{\#}, \mathtt{\#})$
$F := \{s \in S |\, \pi_1(s) = \mathtt{T}\}$
$\bar{H}_1 = (-1, 0)$

Es sei $w = w_1 \cdots w_n$ ein Eingabewort. Die Anfangskonfiguration wird für alle i aus der Retina festgelegt durch
$c_0(i) := ((\mathtt{e}, \mathtt{e}, w_i, \mathtt{e}), \mathtt{g})$.

σ wird wie folgt festgelegt:
$\sigma(((p_1, p_2, p_3, p_4), (q_1, q_2, q_3, q_4)), h) = ((q_1', q_2', q_3', q_4'), \gamma)$, wobei gilt:

$$q_1' = \begin{cases} \mathtt{F} & \text{falls } p_3 = \mathtt{b} \wedge q_3 = \mathtt{a} \\ & \qquad \vee p_1 = \mathtt{F} \vee q_1 = \mathtt{F} \\ & \qquad \vee p_3 = \mathtt{\#} \wedge q_3 = \mathtt{b} \\ \mathtt{1} & \text{falls } q_1 = \mathtt{e} \wedge q_3 = \mathtt{b} \wedge p_3 = \mathtt{a} \\ \mathtt{0} & \text{falls } q_1 \in \{\mathtt{*}, \mathtt{!}\} \wedge p_1 \neq \mathtt{F} \\ \mathtt{!} & \text{falls } q_1 = \mathtt{1} \wedge p_1 \neq \mathtt{F} \wedge \\ & \qquad (q_4 = \mathtt{3} \vee q_2 = \mathtt{po} \wedge p_1 = \mathtt{*} \wedge h = \mathtt{g}) \\ \mathtt{*} & \text{falls } q_1 = \mathtt{0} \wedge p_1 \neq \mathtt{F} \wedge \\ & \qquad (q_4 = \mathtt{3} \vee q_2 = \mathtt{po} \wedge p_1 = \mathtt{*} \wedge h = \mathtt{g}) \\ \mathtt{T} & \text{falls } q_1 \neq \mathtt{F} \wedge p_1 \neq \mathtt{F} \wedge \\ & \qquad (q_1 = \mathtt{!} \wedge p_3 = \mathtt{r} \wedge q_3 = \mathtt{a} \vee q_1 = \mathtt{T}) \\ q_1 & \text{sonst} \end{cases}$$

$$q_2' := \begin{cases} \mathtt{pu} & \text{falls } q_2 = \mathtt{e} \;\wedge\; q_3 = \mathtt{b} \\ & \qquad \vee\, p_1 \in \{\mathtt{*}, \mathtt{!}\} \;\wedge\; q_2 = \mathtt{po} \;\wedge\; h = \mathtt{g} \\ \mathtt{po} & \text{falls } q_2 = \mathtt{pu} \;\wedge\; p_1 \in \{\mathtt{*}, \mathtt{!}\} \\ q_2 & \text{sonst} \end{cases}$$

$$q_3' := \begin{cases} \mathtt{r} & \text{falls } q_3 = \mathtt{r} \;\vee\; p_3 = \mathtt{\#} \\ p_3 & \text{sonst} \end{cases}$$

$$q_4' := \begin{cases} \mathtt{0} & \text{falls } q_4 = \mathtt{e} \;\wedge\; (q_3 = \mathtt{a} \;\vee\; q_3 = \mathtt{b} \;\wedge\; p_3 = \mathtt{b}) \\ \mathtt{1} & \text{falls } q_4 = \mathtt{e} \;\wedge\; q_3 = \mathtt{b} \;\wedge\; p_3 = \mathtt{a} \\ & \qquad \vee\, q_4 = \mathtt{4} \\ \mathtt{2} & \text{falls } q_4 = \mathtt{1} \\ \mathtt{3} & \text{falls } q_4 = \mathtt{2} \\ \mathtt{4} & \text{falls } q_4 = \mathtt{3} \\ q_4 & \text{sonst} \end{cases}$$

$$\gamma := \begin{cases} \mathtt{m}h & \text{falls } q_1 \in \{\mathtt{0}, \mathtt{1}\} \;\wedge\; q_2 = \mathtt{pu} \;\wedge\; p_1 \notin \{\mathtt{*}, \mathtt{!}\} \;\wedge\; q_4 = \mathtt{0} \\ \varepsilon & \text{falls } q_2 = \mathtt{po} \;\wedge\; p_1 \in \{\mathtt{*}, \mathtt{!}\} \;\wedge\; h \neq \mathtt{g} \\ h & \text{sonst} \end{cases}$$

Auf der ersten Spur wird genau dann ein F erzeugt, wenn auf der dritten Spur, auf der das Eingabewort notiert ist, ein Teilwort ba auftritt. Das Symbol F wandert anschließend an den rechten Rand. Außerdem ist es selbsterhaltend und dominiert alle anderen Symbole auf der ersten Spur. Es können also nur Eingaben, deren Struktur $\mathtt{a}^+\mathtt{b}^*$ genügt, akzeptiert werden.

Auf der dritten Spur wird das Eingabewort sukzessive nach rechts verschoben, wobei am linken Rand mit dem Symbol r aufgefüllt wird. Die Information am rechten Rand geht verloren.

Für die weitere Analyse (Abbildung 5.7) wird angenommen, daß die Zelle mit der Eingabe b, deren linker Nachbar in der Anfangskonfiguration ein a im dritten Register notiert hat, die Nummer 1 erhält und alle weiter rechts liegenden in aufsteigender Reihenfolge numeriert werden.
Zelle 1 kann sich selbst identifizieren. Das vierte Register wird ausschließlich von ihr benötigt, wobei sie darin zyklisch die Symbole 1, 2, 3 und 4 erzeugt. Alle anderen Zellen generieren im ersten Zeitschritt im vierten Register das selbsterhaltende Symbol 0.

In der Anfangskonfiguration enthält das erste Register aller Zellen $i \geq 1$ das Symbol `e`. Im Zeitpunkt i wird das `e` durch `1` ersetzt — dies wird Aktivierung der Zelle genannt. Weiterhin erzeugen die Zellen im ersten Zeitschritt ein `pu` auf der zweiten Spur.

Nach diesen Vorbemerkungen kann jetzt gezeigt werden:

Behauptung 5.8. Es sei eine hinreichend lange Eingabe vorausgesetzt, dann gilt:

$$\forall n, i \in \mathbb{N} : \left(t = i - 1 + n2^{2^i} \iff \pi_{1,1}(c_t(i)) = \begin{cases} \texttt{!} & \text{falls } n = 1 \\ \texttt{*} & \text{falls } n > 1 \end{cases} \right)$$

Beweis der Behauptung.
$i = 1$:
Die Zelle 1 generiert im Zeitpunkt 1 das Symbol `1` auf der ersten und vierten Spur. Anschließend durchläuft ihr viertes Register fortlaufend den Zyklus `2`, `3`, `4` und `1`. Im Zeitschritt 4 wird ein `!` auf der ersten Spur erzeugt, im Zeitschritt `5` das Symbol `0`. Im weiteren Verlauf wird im ersten Register genau dann ein `*` erzeugt, wenn im vierten das Symbol `4` generiert wird. Aus dem zyklischen Verhalten folgt die Behauptung.

$i \longrightarrow i + 1$:
Im Zeitpunkt $i + 1$ der Aktivierung von Zelle $i + 1$ gilt: $c_{i+1}(i + 1) = ((\texttt{1}, \texttt{pu}, \texttt{a}, \texttt{0}), \texttt{g})$. Anschließend wird in jedem Zeitschritt ein zusätzliches Symbol `m` in den Keller geschrieben, bis die benachbarte Zelle i auf der ersten Spur `!` oder `*` erzeugt. Dies geschieht nach Induktionsannahme im Zeitpunkt $i - 1 + 2^{2^i}$ zum ersten Mal. Im Keller der Zelle $i + 1$ befinden sich also $i - 1 + 2^{2^i} - (i + 1) = 2^{2^i} - 2$ Symbole `m`.
Im nächsten Zeitschritt ($t = i + 2^{2^i}$) erzeugt die Zelle das Symbol `po` auf der zweiten Spur, ohne den Kellerinhalt zu verändern. Im weiteren Verlauf wird immer dann ein Symbol aus dem Keller entfernt, wenn die Zelle i das Symbol `*` auf der ersten Spur notiert hat — also alle 2^{2^i} Zeitschritte. Dies gelingt $2^{2^i} - 2$ mal, bis der Keller leer ist. Anschließend wartet Zelle $i + 1$ aber noch weitere 2^{2^i} Zeitschritte mit leerem Keller, bis ihr linker Nachbar erneut ein `*` generiert hat. Dies

geschieht im Zeitpunkt $i-1+2^{2^i}+(2^{2^i}-2)2^{2^i}+2^{2^i} = i-1+2^{2^{(i+1)}}$. Der Versuch, ein Symbol aus dem leeren Keller zu entfernen, veranlaßt Zelle $i+1$ anschließend, das Symbol `!` auf der ersten Spur zu erzeugen; also im Zeitpunkt $i+2^{2^{(i+1)}}$. Daraus folgt die Behauptung für $n=1$. Für die Konfiguration in diesem Zeitpunkt gilt:

$$c_{i+2^{2^{(i+1)}}}(i+1) = ((\texttt{!}, \texttt{pu}, \texttt{a}, \texttt{0}), \texttt{g}) \text{ und damit}$$

$$c_{i+1+2^{2^{(i+1)}}}(i+1) = ((\texttt{0}, \texttt{pu}, \texttt{a}, \texttt{0}), \texttt{g}).$$

Die Konfiguration unterscheidet sich von derjenigen im Zeitpunkt $i+1$ nur durch das Symbol `0` auf der ersten Spur. Dies bewirkt aber gerade, daß gegebenenfalls statt des `!` ein `*` generiert wird. Also wird sich das gesamte Verhalten wiederholen. □

Es bleibt zu zeigen, wie der Automat eine Eingabe akzeptiert. Es wird ein `T` auf der ersten Spur nur dann generiert, wenn eine der Zellen das Symbol `!` auf der ersten Spur notiert hat und gleichzeitig vom letzten Eingabesymbol `a` auf der dritten Spur passiert wurde. Es sei $\texttt{a}^{\cdot m}\texttt{b}^{\cdot n}$ die Eingabe. Das Symbol `!` wird von einer Zelle i nur einmal im Zeitpunkt $i-1+2^{2^i}$ erzeugt. Das letzte `a` benötigt $m+i-1$ Zeitschritte bis zur Zelle i. Also wird die Eingabe nur dann akzeptiert, wenn $m=2^{2^i}$ gilt und i die rechte Randzelle ist. □

Das Verfahren zur Erkennung der Sprache $\{(\texttt{ab})^{\cdot(n^2)} | n \in \mathbb{N}\}$ ging davon aus, daß jeweils zwei benachbarte Zellen sich verlängernde Zyklen durchlaufen, die mit dem Leerwerden eines Kellers enden. Die Zellen waren dabei anhand ihres Kellerinhaltes in der Lage, die Länge des Folgezyklus selbst zu berechnen. Dies war notwendig, da die Struktur der Eingabe ein gleichartiges Verhalten aller Zellenpaare `ab` vorgab. Hier liegt der wesentliche Unterschied zum Erkennungsverfahren für die Sprache $\{\texttt{a}^{\cdot(2^{2^n})}\texttt{b}^{\cdot n} | n \in \mathbb{N}\}$, deren Wortstruktur es den Zellen mit Eingabe `b` ermöglicht, Zyklen zu durchlaufen, deren Länge jeweils dem Quadrat der Länge des Zyklus des Nachbarn entspricht.

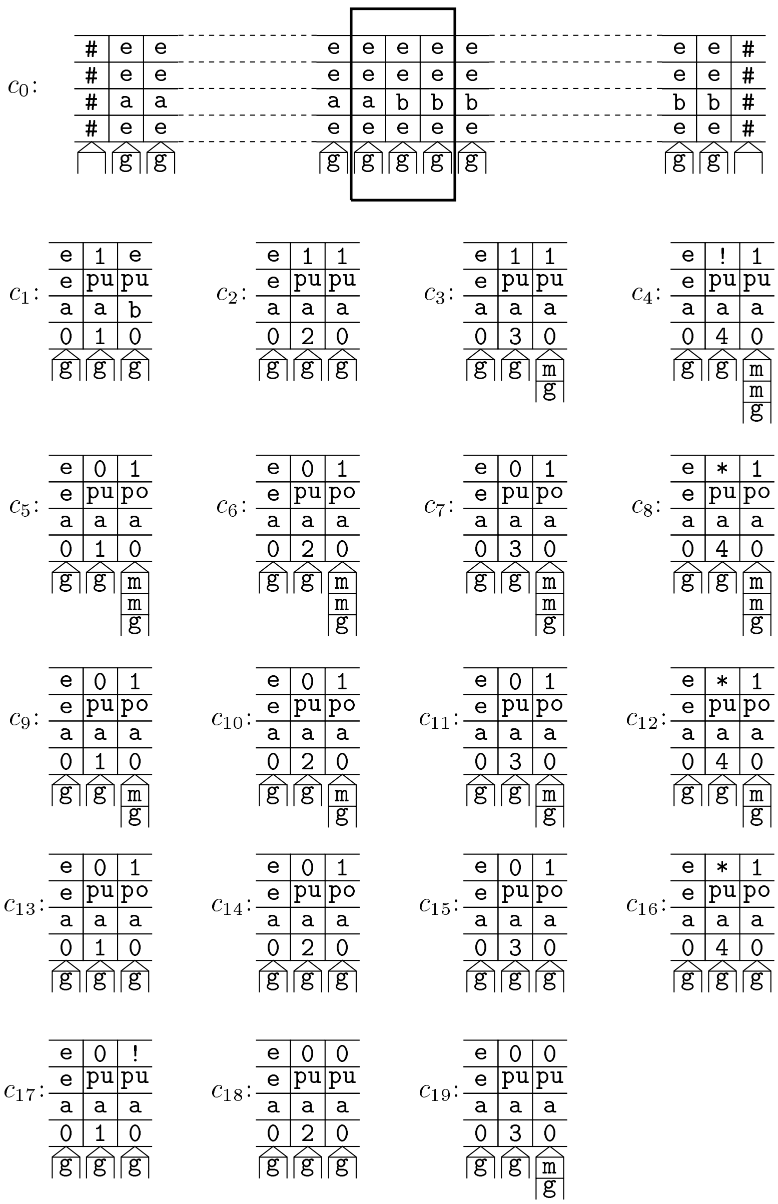


Abbildung 5.7. Ausschnitt aus einer Konfigurationsfolge zum Satz 5.10.

5.4 Abschlußeigenschaften

Die im folgenden hergeleiteten Abschlußeigenschaften belegen, daß die durch Keller-Zellularautomaten definierten Sprachfamilien sich von den durch Zellularautomaten definierten unterscheiden. Inwieweit die Sprachfamilien miteinander vergleichbar sind, wird anschließend untersucht werden.

In Satz 3.20 wurde die Abgeschlossenheit der Familie $\mathscr{L}_{rt}(\mathrm{OCA})$ bzgl. Spiegelung nachgewiesen. Der nächste Satz zeigt, daß dies für die Familie $\mathscr{L}_{rt}(\mathrm{OPDCA})$ nicht gilt.

Satz 5.11. $\mathscr{L}_{rt}(\mathrm{OPDCA})$ ist nicht abgeschlossen bzgl. Spiegelung.

Beweis. Nach Satz 5.10 ist $L = \{\mathtt{a}^{\cdot(2^{2^n})}\mathtt{b}^{\cdot n} | n \in \mathbb{N}\} \in \mathscr{L}_{rt}(\mathrm{OPDCA})$. Der Nachweis, daß $MIR(L) = \{\mathtt{b}^{\cdot n}\mathtt{a}^{\cdot(2^{2^n})} | \; n \in \mathbb{N}\} \notin \mathscr{L}_{rt}(\mathrm{OPDCA})$ ist, erfolgt mit Satz 5.8. Mit $n^{(n+1)^2} = 2^{(n+1)^2 \log_2 n}$ folgt sofort

$$\lim_{n\to\infty} \frac{2^{(n+1)^2 \log_2 n}}{2^{2^n}} = 0$$

und damit $MIR(L) \notin \mathscr{L}_{rt}(\mathrm{OPDCA})$. $\square$

Eine in der Theorie der formalen Sprachen bedeutende Sprachoperation ist die Bildung homomorpher Bilder. Es läßt sich z.B. jede rekursiv aufzählbare, nicht rekursive Sprache mittels eines Homomorphismus aus einer kontextsensitiven Sprache gewinnen (A. Salomaa (1973)). Daraus resultiert unter anderem die Nichtabgeschlossenheit kontextsensitiver Sprachen unter beliebigen Homomorphismen. Es ist jedoch bekannt, daß die Sprachfamilie bzgl. ε-freien Homomorphismen abgeschlossen ist, was für die Familie $\mathscr{L}_{rt}(\mathrm{OPDCA})$ nicht gilt.

Definition 5.10.

a) Es sei A ein Alphabet. Für alle $a \in A$ sei $\xi(a)$ eine Sprache über einem Alphabet X_a. Mit $X := \bigcup_{a \in A} X_a$ heißt $\xi : A^* \longrightarrow \mathscr{P}(X^*)$ *Substitution*, falls gilt:

$$\xi(\varepsilon) = \varepsilon \ \wedge \ \xi(uv) = \xi(u)\xi(v) \text{ für alle } u, v \in A^*.$$

b) Die Erweiterung von ξ auf Sprachen ist definiert durch

$$\bar{\xi}(L) := \bigcup_{w \in L} \xi(w).$$

Definition 5.11. Es sei $\mathscr{L}$ eine Sprachfamilie und $L \in \mathscr{L}$ eine Sprache über dem Alphabet A. Die Sprachoperation SUB wird folgendermaßen definiert: $SUB(L) := \{\bar{\xi}(L) |\ \xi$ ist Substitution derart, daß $\forall a \in A : \ \xi(a) \in \mathscr{L}$ ist$\}$.

Definition 5.12.

a) Es seien A ein Alphabet und $\xi : A^* \longrightarrow \mathscr{P}(X^*)$ eine Substitution, für die gilt:
$\forall a \in A : \ |\xi(a)| = 1$. Dann heißt

$$\begin{aligned} h : A^* &\longrightarrow X^* \\ w &\mapsto \xi(w) \end{aligned}$$

Homomorphismus.

b)

$$\begin{aligned} h^{-1} : X^* &\longrightarrow \mathscr{P}(A^*) \\ v &\mapsto \{w |\ h(w) = v\} \end{aligned}$$

heißt *inverser Homomorphismus*.

c) Ein Homomorphismus heißt ε-frei, falls gilt: $h^{-1}(\varepsilon) = \{\varepsilon\}$.

d) Die Erweiterung von h auf Sprachen ist definiert durch
$\bar{h}(L) := \{h(w) |\ w \in L\}$.

e) $\bar{h}^{-1}(L) := \{w |\ h(w) \in L\}$

Definition 5.13. Es sei L eine formale Sprache.
$HOM(L) := \{L' \mid L' = \bar{h}(L)$ für einen Homomorphismus $h\}$ und $HOM^{-1}(L) := \{L' | L' = \bar{h}^{-1}(L)$ für einen inversen Homomorphismus $h^{-1}\}$ sind Sprachoperationen. Für ε-freie Homomorphismen wird die entsprechende Sprachoperation mit HOM_ε bezeichnet.

Satz 5.12. $\mathscr{L}_{rt}$(OPDCA) ist nicht abgeschlossen bzgl. HOM_ε.

Beweis. Nach Satz 5.9 ist $L = \{(\mathtt{ab})^{\cdot(n^2)} \mid n \in \mathbb{N}\} \in \mathscr{L}_{rt}(\text{OPDCA})$.
Ein ε-freier Homomorphismus h sei folgendermaßen definiert:
$h(\varepsilon) := \varepsilon$, $h(\mathtt{a}) = h(\mathtt{b}) := \mathtt{a}$. Dann ist $L' = \{\mathtt{a}^{\cdot(2n^2)} \mid n \in \mathbb{N}\}$ aus $HOM_\varepsilon(L)$.
Da L' nicht regulär ist, folgt mit Satz 5.7 $L' \notin \mathscr{L}_{rt}(\text{OPDCA})$. Also gilt: $HOM_\varepsilon(L) \not\subseteq \mathscr{L}_{rt}(\text{OPDCA})$. □

Korollar 5.4. $\mathscr{L}_{rt}$(OPDCA) ist nicht abgeschlossen bzgl. HOM und SUB.

Satz 5.13. $\mathscr{L}_{rt}$(OPDCA) ist nicht abgeschlossen bzgl. HOM^{-1}.

Beweis. Ein Homomorphismus h sei definiert als $h(\varepsilon) := \varepsilon$ und $h(\mathtt{a}) := \mathtt{ab}$. Die Sprache $L = \{(\mathtt{ab})^{\cdot(n^2)} \mid n \in \mathbb{N}\}$ gehört zur Familie $\mathscr{L}_{rt}$(OPDCA), die Sprache $\bar{h}^{-1}(L) = \{\mathtt{a}^{\cdot(n^2)} \mid n \in \mathbb{N}\}$ aufgrund von Satz 5.7 nicht.
Da $\bar{h}^{-1}(L) \in HOM^{-1}(L)$ ist, gilt: $HOM^{-1} \not\subseteq \mathscr{L}_{rt}(\text{OPDCA})$. □

Im folgenden werden weitere elementare Abschlußeigenschaften gezeigt.

Satz 5.14. Jede der Sprachfamilien $\mathscr{L}_{rt}$(OPDCA), $\mathscr{L}_{lt}$(OPDCA), $\mathscr{L}_{rt}$(PDCA) und $\mathscr{L}_{lt}$(PDCA) ist abgeschlossen bzgl. COM.

Beweis. Die Abgeschlossenheit kann für jede der Familien wieder mit der sogenannten Mehrspurtechnik gezeigt werden:

Es sei L eine Sprache, die in Linearzeit von einem Keller-Zellularautomaten M akzeptiert wird. Dann existiert eine Konstante $z \in \mathbb{N}$, so daß für alle $w \in L$ der Automat M nach $z \cdot |w|$ Zeitschritten seine Berechnung beendet hat. Ein Keller-Zellularautomat M', der

$COM(L)$ akzeptiert, arbeitet mit einer zweiten Spur, auf der am Anfang der Berechnung am linken Rand ein Signal generiert wird. Dieses wird nach jeweils z Zeitschritten um eine Zelle nach rechts verschoben. Erreicht das Signal nach $z \cdot |w|$ Zeitschritten den rechten Rand, wird w akzeptiert, falls die Simulation von M auf der ersten Spur w nicht akzeptiert hat. Für Realzeit-Sprachen wird der entsprechende Nachweis mit $z = 1$ erbracht. □

Der Beweis ist insofern nicht konstruktiv, da er nur die Existenz der Konstanten z erfordert. Nach W. T. Beyer (1969) und S. R. Kosaraju (1974) ist das Problem, die Konstante für zweidimensionale Zellularautomaten zu finden, nicht algorithmisch lösbar.

Die Abgeschlossenheit bzgl. MIR gilt nach A. R. Smith III (1972) auch für die Familie $\mathscr{L}_{lt}$(CA).

Satz 5.15. $\mathscr{L}_{lt}$(PDCA) ist abgeschlossen bzgl. MIR.

Beweis. Nach Beispiel 5.1 kann ein PDCA seine Eingabe w in $|w|$ Zeitschritten spiegeln. Für $L \in \mathscr{L}_{lt}$(PDCA) simuliert der Akzeptor von L^R den Akzeptor von L, nachdem die Eingabe gespiegelt wurde. □

Die bei Zellularautomaten benutzte Technik, auf mehreren Spuren verschiedene Akzeptoren parallel zu simulieren und damit z.B. die Abgeschlossenheit von Realzeitsprachen bzgl. Vereinigung nachzuweisen, ist nicht ohne weiteres auf Keller-Zellularautomaten übertragbar. Während durch Aufspaltung in mehrere Register mehrere endliche Automaten durch einen simuliert werden können, ist es nicht möglich, mehrere Keller in einem parallel zu simulieren. Daraus resultiert z.B. die Nichtabgeschlossenheit der deterministischen kontextfreien Sprachen bzgl. Vereinigung. Andererseits wäre es sonst z.B. möglich, Turingmaschinen durch einzelne Kellerautomaten zu simulieren.

Für die Familie $\mathscr{L}_{lt}$(PDCA) lassen sich die folgenden Abschlußeigenschaften durch Komposition mehrerer Algorithmen nachweisen.

Satz 5.16. $\mathscr{L}_{lt}$(PDCA) ist abgeschlossen bzgl. UNI, INT und DIF.

Beweis.
Abgeschlossenheit bzgl. UNI:
Es seien $L_1, L_2 \in \mathscr{L}_{lt}$(PDCA), und M_1 und M_2 zwei PDCAs. M_1 akzeptiere jede Eingabe $w \in L_1$ in $z_1 \cdot |w|$ Zeitschritten und M_2 jede Eingabe $w \in L_2$ in $z_2 \cdot |w|$ Zeitschritten. Ein PDCA M', der $L_1 \cup L_2$ akzeptiert, arbeitet mit vier Spuren. Auf einer Spur wird zunächst die Eingabe w konserviert. Auf der zweiten Spur wird wieder ein Signal von links nach rechts gesandt, welches nach $z_1 \cdot |w|$ Zeitschritten den rechten Rand erreicht. Während dieser Zeit simuliert M' den PDCA M_1 auf der dritten Spur und in den Kellern. Falls die Simulation akzeptierend endet, akzeptiert M'. Anderenfalls wird ein FSSP auf der ersten Spur gestartet, welches alle Zellen nach $2|w| - 2$ Zeitschritten synchronisiert. Anschließend simuliert M' den PDCA M_2 auf der ersten Spur und in den Kellern. M' akzeptiert, falls die Simulation akzeptierend endet.
Insgesamt benötigt M' also höchstens $(z_1 + z_2 + 2)|w| - 2$ Zeitschritte.

Abgeschlossenheit bzgl. INT und DIF:
Es seien L_1 und L_2 zwei Sprachen aus $\mathscr{L}_{lt}$(PDCA).

Wegen $L_1 \cap L_2 = (\overline{\overline{L_1} \cup \overline{L_2}})$ und der Abgeschlossenheit bzgl. COM und UNI ist $\mathscr{L}_{lt}$(PDCA) abgeschlossen bzgl. INT.

Wegen $L_1 \setminus L_2 = \overline{L_2} \cap L_1$ und der Abgeschlossenheit bzgl. COM und INT ist $\mathscr{L}_{lt}$(PDCA) abgeschlossen bzgl. DIF. □

Die Methode der Komposition durch Einschub eines FSSP wurde hier der Einfachheit halber gewählt. B. Bleck und H. Kröger (1992) und U. Becker (1990) betrachten allgemeinere, zeitoptimierte Methoden, zellulare Algorithmen hintereinanderzuschalten.

Für den Nachweis weiterer Abschlußeigenschaften werden einige Ergebnisse aus der Theorie formaler Sprachen benötigt. Sie werden im folgenden der Vollständigkeit halber ohne Beweis referiert.

Definition 5.14. Eine *Dyck-Sprache* ist eine formale Sprache über einem Alphabet $\{a_1, a_1', a_2, a_2', \ldots, a_n, a_n'\}$, die durch eine kontextfreie Grammatik mit einem nichtterminalen Zeichen X und den Produktionen $X \longrightarrow \varepsilon, X \longrightarrow XX$ und $X \longrightarrow a_i X a_i'$ für $1 \leq i \leq n$ erzeugt wird.

C. R. Dyer (1980) hat gezeigt, daß die Dyck-Sprachen von OCAs in Realzeit erkannt werden können.

Satz 5.17. Es sei D eine Dyck-Sprache. $D \in \mathscr{L}_{rt}(\text{OCA})$.

Korollar 5.5. Es sei D eine Dyck-Sprache. $D \in \mathscr{L}_{rt}(\text{OPDCA})$, $D \in \mathscr{L}_{rt}(\text{CA})$ und $D \in \mathscr{L}_{rt}(\text{PDCA})$.

Der folgende Satz stammt von N. Chomsky (1962).

Satz 5.18. Es sei $L \in \mathscr{L}_2$ eine formale Sprache. Dann existieren eine reguläre Sprache R, eine Dyck-Sprache D und ein Homomorphismus h, so daß $L = h(D \cap R)$ ist.

Die homomorphe Charakterisierung der rekursiv aufzählbaren Sprachen wurde von S. Ginsburg, S. A. Greibach und M. A. Harrison (1967) veröffentlicht.

Satz 5.19. Es sei $L \in \mathscr{L}_0$. Dann existieren zwei kontextfreie Sprachen L_1 und L_2 sowie ein Homomorphismus h, so daß $L = h(L_1 \cap L_2)$ ist.

Mit diesen Vorbemerkungen können nun die beiden folgenden Sätze bewiesen werden.

Satz 5.20. $\mathscr{L}_{lt}(\text{PDCA})$ ist nicht abgeschlossen bzgl. HOM.

Beweis. Angenommen, $\mathscr{L}_{lt}(\text{PDCA})$ wäre abgeschlossen bzgl. HOM. Da für eine beliebige Dyck-Sprache $D \in \mathscr{L}_{lt}(\text{PDCA})$ gilt und $\mathscr{L}_{lt}(\text{PDCA})$ bzgl. Durchschnitt mit regulären Sprachen abgeschlossen ist, folgt mit Satz 5.18 $\mathscr{L}_2 \subseteq \mathscr{L}_{lt}(\text{PDCA})$. Aufgrund der Abgeschlossenheit bzgl. INT folgt dann mit Satz 5.19 $\mathscr{L}_0 = \mathscr{L}_{lt}(\text{PDCA})$. Hieraus ergibt sich der Widerspruch, daß $\mathscr{L}_{lt}(\text{PDCA})$ bzgl. COM abgeschlossen ist, $\mathscr{L}_0$ bekanntlich aber nicht. $\square$

Korollar 5.6. $\mathscr{L}_{lt}(\text{PDCA})$ ist nicht abgeschlossen bzgl. SUB.

Mit derselben Argumentation läßt sich für $\mathscr{L}_{lt}(\text{OPDCA})$ und $\mathscr{L}_{rt}(\text{PDCA})$ zeigen:

Satz 5.21. Die Familien $\mathscr{L}_{lt}(\text{OPDCA})$ und $\mathscr{L}_{rt}(\text{PDCA})$ sind jeweils nicht gleichzeitig abgeschlossen bzgl. HOM und INT.

5.5 Vergleich mit Zellularräumen

Es stellt sich die Frage, ob der größere Hardwareaufwand von Keller-Zellularautomaten gegenüber den klassischen Zellularautomaten gerechtfertigt ist. Im Rahmen der hier gewählten Betrachtungsweise heißt das, ob die jeweiligen Komplexitätsklassen mächtiger sind.

Im ersten Unterabschnitt wird neben anderen Beziehungen zwischen den beiden parallelen Modellen gezeigt, daß dies für unidirektionale Realzeiterkenner der Fall ist. Anhand der Erkennbarkeit einer Sprache, deren kompliziertere Variante als Kandidat für die Nichterkennbarkeit durch Realzeit-CAs gilt, läßt sich das Problem für bidirektionale Verbindungsstrukturen zwar nicht lösen, lassen sich aber u.U. Lösungswege finden (vgl. Ende des Abschnittes 3.5.3).

5.5.1 Zellularautomaten

Die beiden folgenden Sätze beschreiben naheliegende Beziehungen zwischen den Sprachfamilien.

Satz 5.22. Für jede Abbildung $t : \mathbb{N} \longrightarrow \mathbb{N}$ gilt:

$$\begin{aligned} \mathscr{L}_{t(n)}(\text{CA}) &\subseteq \mathscr{L}_{t(n)}(\text{PDCA}) \\ \text{und } \mathscr{L}_{t(n)}(\text{OCA}) &\subseteq \mathscr{L}_{t(n)}(\text{OPDCA}) \end{aligned}$$

Beweis. Zellularautomaten können als Spezialfälle von Keller-Zellularautomaten aufgefaßt werden, wobei die zweite Komponente der Überführungsfunktion die Identität ist. $\square$

Satz 5.23. $\mathscr{L}(\text{PDCA}) = \mathscr{L}(\text{CS}) \supset \mathscr{L}(\text{CA})$

Beweis. Es genügt zu zeigen, daß PDCAs beliebige Turingmaschinen simulieren können, also berechnungsuniversell sind. Der Satz folgt dann mit den Sätzen 3.4 und 3.9.

Es sei zunächst angenommen, die Länge der Eingabe ist größer als eins. Die beiden rechten Zellen der Retina können mit ihren Kellern das Verhalten einer Turingmaschine simulieren. Alle von der Turingmaschine jemals besuchten Felder des Bandes (bzw. deren Inschriften), die rechts des aktuellen Feldes liegen, befinden sich im Keller der rechten Zelle, alle links vom aktuellen Feld im Keller der linken Zelle.

Der Inhalt des aktuellen Feldes wird in ein Register der Zelle übernommen. Bewegt die Turingmaschine den Schreiblesekopf auf dem Band nach rechts, wird ein Symbol aus dem Keller der rechten Zelle entfernt und ein Symbol in den Keller der linken Zelle geschrieben. Alle anderen Zellen des PDCA benötigen ihre Keller nicht. Zur Initialisierung der Simulation muß lediglich das Eingabewort in einer Folge von Verschiebeschritten sukzessive in den Keller der rechten Randzelle geschrieben werden.

Die Anzahl der möglichen Eingaben der Länge eins ist endlich. Sie können daher als Sonderfälle vom PDCA erkannt werden. □

Das Ziel dieses Abschnittes ist es, die Ungültigkeit der Umkehrung von Satz 5.22 nachzuweisen. Setzt man jedoch zunächst voraus, daß der maximal zur Verfügung stehende Kellerspeicher durch eine Konstante beschränkt ist, läßt sich die folgende, eingeschränkte Umkehrung zeigen.

Satz 5.24. Ist $z \in \mathbb{N}$ eine beliebige Konstante und $t : \mathbb{N} \longrightarrow \mathbb{N}$ eine Abbildung, dann gilt:

$$\begin{aligned} {}_z\mathscr{L}_{t(n)}(\text{PDCA}) &\subseteq \mathscr{L}_{t(n)}(\text{CA}) \\ \text{und } {}_z\mathscr{L}_{t(n)}(\text{OPDCA}) &\subseteq \mathscr{L}_{t(n)}(\text{OCA}) \end{aligned}$$

Beweis. Es seien M ein PDCA (OPDCA) und $L = L(M)$ die von ihm mit Kellerkomplexität $k(n) = z$, $z \in \mathbb{N}$, akzeptierte Sprache. Ein CA (OCA), der L akzeptiert, verfügt über $z + 1$ Spuren, wobei jede Zelle im ersten Register die Zustandsüberführung des PDCA (OPDCA) und in den verbleibenden z Registern einen Keller bis zur Kellertiefe z simuliert. $\square$

Der folgende Satz stellt ein häufig benutztes Hilfsmittel zur Verfügung. Er resultiert aus der grundsätzlichen Möglichkeit von Keller-Zellularautomaten, auf einer zusätzlichen Spur Zellularautomaten parallel zu „eigenen" Berechnungen zu simulieren.

Satz 5.25. Es seien $t : \mathbb{N} \longrightarrow \mathbb{N}$ eine Abbildung, $L_1 \in \mathscr{L}_{t(n)}(\text{PDCA})$ und $L_2 \in \mathscr{L}_{t(n)}(\text{CA})$, dann gilt:

a) $L_1 \cap L_2 \in \mathscr{L}_{t(n)}(\text{PDCA})$
b) $L_1 \cup L_2 \in \mathscr{L}_{t(n)}(\text{PDCA})$
c) $L_1 \setminus L_2 \in \mathscr{L}_{t(n)}(\text{PDCA})$
d) $L_2 \setminus L_1 \in \mathscr{L}_{t(n)}(\text{PDCA})$

Beweis. Ein entsprechender Akzeptor simuliert in seinen Kellern und auf einer Spur den Akzeptor für L_1 und auf einer weiteren Spur den Akzeptor für L_2. Ist $M = (S, \Gamma, 1, H_1, \sigma, q_0, g_0)$ der PDCA und $M' = (S', 1, H_1, \sigma', q_0')$ der CA, dann wird der Akzeptor $\hat{M} = (\hat{S}, \hat{\Gamma}, 1, H_1, \hat{\sigma}, \hat{q}_0, \hat{g}_0)$ wie folgt konstruiert:

$$\hat{S} := S \times S'$$
$$\hat{\Gamma} := \Gamma$$
$$\hat{q}_0 := (q_0, q_0')$$
$$\hat{g}_0 := g_0$$

$$\forall \tilde{q} \in (S \times S')^3, g \in \Gamma :$$
$$\hat{\sigma}(\tilde{q}, g) := \big((\pi_1(\sigma(\bar{\pi}_1(\tilde{q}), g)), \sigma'(\bar{\pi}_2(\tilde{q}))), \pi_2(\sigma(\bar{\pi}_1(\tilde{q}), g))\big)$$

F wird entsprechend den vier Behauptungen festgelegt.

$$\begin{array}{ll} \text{a)} & \hat{F} := F \times F' \\ \text{b)} & \hat{F} := (F \times S') \cup (S \times F') \\ \text{c)} & \hat{F} := F \times (S' \setminus F') \\ \text{d)} & \hat{F} := (S \setminus F) \times F' \end{array}$$

Da die Simulationen völlig unabhängig voneinander durchgeführt werden, folgen die Behauptungen. □

Korollar 5.7. Satz 5.25 gilt auch, wenn PDCA durch OPDCA und CA durch OCA ersetzt werden.

Satz 5.26. $\mathscr{L}_{rt}(\text{OCA}) \subset \mathscr{L}_{rt}(\text{OPDCA})$

Beweis. Nach Satz 5.22 genügt es, die Echtheit der Inklusion zu zeigen. In Satz 3.20 wurde nachgewiesen, daß die Familie $\mathscr{L}_{rt}(\text{OCA})$ bzgl. MIR abgeschlossen ist. Da dies für $\mathscr{L}_{rt}(\text{OPDCA})$ nach Satz 5.11 nicht gilt, existiert eine Sprache $L \in \mathscr{L}_{rt}(\text{OPDCA})$, deren Spiegelbild nicht zur Familie $\mathscr{L}_{rt}(\text{OPDCA})$ gehört. Angenommen, L wäre aus $\mathscr{L}_{rt}(\text{OCA})$, dann auch L^R. Somit wäre L^R auch aus $\mathscr{L}_{rt}(\text{OPDCA})$. □

Korollar 5.8. $L = \{\mathtt{a}^{\cdot(2^{2^n})}\mathtt{b}^{\cdot n} \mid n \in \mathbb{N}\} \notin \mathscr{L}_{rt}(\text{OCA})$

5.5.2 Eine besondere Sprache

Das bis heute ungelöste Problem der Abgeschlossenheit der Familie $\mathscr{L}_{rt}(\text{CA})$ bzgl. MIR wurde von A.R. Smith III (1972) formuliert. Die Verallgemeinerung der am Ende von Abschnitt 3.5.3 genannten Sprache ist ein Kandidat für den Nachweis der Nichtabgeschlossenheit.

Der nächste Satz könnte ein Schritt zum Nachweis sein, daß die Familie $\mathscr{L}_{rt}(\text{CA})$ eine echte Teilmenge der Familie $\mathscr{L}_{rt}(\text{PDCA})$ ist.

Satz 5.27. $L = \{\mathtt{b}^{\cdot n}\mathtt{a}^{\cdot m} \mid n, m \in \mathbb{N} \ \wedge \ n \text{ teilt } m\} \in \mathscr{L}_{rt}(\mathrm{PDCA})$

Beweis. Zunächst wird die Konstruktion eines PDCA für den Fall durchgeführt, daß n gerade ist und $m > n$ gilt. Nachdem die Korrektheit dieser Konstruktion nachgewiesen ist, wird der PDCA geeignet erweitert.

$S := \{\mathtt{\#}, \mathtt{e}, \mathtt{1}, \mathtt{0}\} \times \{\mathtt{pu}, \mathtt{cu}, \mathtt{po}, \mathtt{co}, \mathtt{\#}, \mathtt{T}\} \times \{\mathtt{\#}, \mathtt{r}, \mathtt{a}, \mathtt{b}\} \times \{\mathtt{\#}, \mathtt{e}, \mathtt{m}, \mathtt{-}\}$
$\Gamma := \{\mathtt{g}, \mathtt{m}, \mathtt{-}\}$
$q_0 := (\mathtt{e}, \mathtt{T}, \mathtt{r}, \mathtt{-})$
$g_0 := \mathtt{g}$
Grenzzustand: $(\mathtt{\#}, \mathtt{\#}, \mathtt{\#}, \mathtt{\#})$
$F := \{s \in S \mid \pi_2(s) = \mathtt{T}\}$
$H_1 = (-1, 0, 1)$

Es sei $w = w_1 \cdots w_n$ ein Eingabewort. Die Anfangskonfiguration wird für alle i aus der Retina festgelegt durch

$$c_0(i) := ((\mathtt{e}, \mathtt{pu}, w_i, \mathtt{-}), \mathtt{g})$$

σ wird wie folgt festgelegt:

$$\sigma(((o_1, o_2, o_3, o_4), (p_1, p_2, p_3, p_4), (q_1, q_2, q_3, q_4)), h) := ((p'_1, p'_2, p'_3, p'_4), \gamma),$$

wobei gilt:

$$p'_1 := \begin{cases} \mathtt{1} & \text{falls } p_1 = \mathtt{e} \ \wedge \ q_1 = \mathtt{\#} \\ & \qquad \vee\, p_1 = \mathtt{e} \ \wedge \ q_1 = \mathtt{0} \\ \mathtt{0} & \text{falls } p_1 = \mathtt{e} \ \wedge \ q_1 = \mathtt{1} \\ p_1 & \text{sonst} \end{cases}$$

$$p_2' := \begin{cases} \texttt{po} & \text{falls } p_1 = \texttt{1} \ \wedge \ o_4 = \texttt{-} \\ & \qquad \vee\, p_1 = \texttt{0} \ \wedge \ q_4 = - \ \wedge \ o_3 = \texttt{a} \\ \texttt{co} & \text{falls } p_1 = \texttt{0} \ \wedge \ q_4 = \texttt{-} \ \wedge \ o_3 = \texttt{b} \ \wedge \ p_3 = \texttt{a} \\ \texttt{pu} & \text{falls } p_1 = \texttt{1} \ \wedge \ p_4 = \texttt{-} \\ & \qquad \vee\, p_1 = \texttt{0} \ \wedge \ p_4 = \texttt{-} \ \wedge \ o_3 = \texttt{a} \\ \texttt{cu} & \text{falls } p_1 = \texttt{0} \ \wedge \ p_4 = \texttt{-} \ \wedge \ o_3 = \texttt{b} \ \wedge \ p_3 = \texttt{a} \\ \texttt{T} & \text{falls } p_2 = \texttt{T} \ \vee \ o_2 = \texttt{T} \\ & \qquad \vee\, p_1 = \texttt{0} \ \wedge \ p_3 = \texttt{b} \ \wedge \ o_3 = \texttt{r} \\ & \qquad\quad \wedge\, (p_2 = \texttt{co} \ \wedge \ p_4 = \texttt{-} \ \vee \ p_2 = \texttt{cu} \ \wedge \ q_4 = \texttt{-}) \\ p_2 & \text{sonst} \end{cases}$$

$$p_3' := \begin{cases} \texttt{r} & \text{falls } o_3 = \texttt{\#} \ \vee \ p_3 = \texttt{r} \\ o_3 & \text{sonst} \end{cases}$$

$$p_4' := \begin{cases} \texttt{m} & \text{falls } p_1 = \texttt{e} \\ & \qquad \vee\, p_2 \in \{\texttt{pu}, \texttt{cu}\} \ \wedge \ (p_1 \neq \texttt{1} \ \vee \ o_1 \neq \texttt{e}) \\ & \qquad \vee\, p_2 \in \{\texttt{co}, \texttt{po}\} \ \wedge \ h \notin \{\texttt{g}, \texttt{-}\} \\ & \qquad \vee\, p_2 = \texttt{po} \ \wedge \ p_4 = \texttt{-} \\ \texttt{-} & \text{falls } p_1 = \texttt{1} \ \wedge \ o_1 = \texttt{e} \\ & \qquad \vee\, p_2 \in \{\texttt{po}, \texttt{co}\} \ \wedge \ h \in \{\texttt{g}, \texttt{-}\} \ \wedge \ p_4 \neq \texttt{-} \\ p_4 & \text{sonst} \end{cases}$$

$$\gamma := \begin{cases} \texttt{m}h & \text{falls } p_4 = \texttt{m} \wedge \\ & \quad (p_1 = \texttt{0} \ \wedge \ p_2 \in \{\texttt{pu}, \texttt{cu}\} \ \wedge \ q_4 \neq \texttt{-} \\ & \quad \vee\, p_1 = \texttt{1} \ \wedge \ p_2 = \texttt{pu} \ \wedge \ o_4 \neq \texttt{-} \ \wedge \ o_1 \neq \texttt{e} \\ & \quad \vee\, p_1 = \texttt{e}) \\ \varepsilon & \text{falls } h \notin \{\texttt{g}, \texttt{-}\} \wedge \\ & \quad p_1 = \texttt{0} \ \wedge \ p_2 \in \{\texttt{pu}, \texttt{cu}\} \ \wedge \ q_4 = \texttt{-} \\ & \quad \vee\, p_1 = \texttt{1} \ \wedge \ p_2 = \texttt{pu} \ \wedge \ o_4 = \texttt{-} \\ & \quad \vee\, p_2 \in \{\texttt{po}, \texttt{co}\} \\ \texttt{-}h & \text{falls } p_1 = \texttt{1} \ \wedge \ o_1 = \texttt{e} \\ h & \text{sonst} \end{cases}$$

Bei der Analyse der Konstruktion von σ stellt man fest, daß ein auf der zweiten Spur generiertes Symbol `T` selbsterhaltend ist und an den rechten Rand der Retina wandert. Aufgrund der Festlegung von F wird der Automat die Eingabe also genau dann akzeptieren, wenn im Laufe der Berechnung eine der Zellen in einen Zustand schaltet, dessen zweite Komponente `T` ist.

Die dritte Spur dient nur dazu, die auf ihr notierte Eingabe in jedem Zeitschritt um eine Zelle nach rechts zu verschieben, wobei am rechten Rand Information verlorengeht und am linken Rand jeweils mit dem Zeichen `r` aufgefüllt wird.

Zur vereinfachten Darstellung wird ausnahmsweise angenommen, die Zellen der Retina sind, beginnend bei 1, von rechts nach links aufsteigend durchnumeriert. Die Bezeichnung „Zelle i“ hat also insofern eine von der üblichen abweichende Bedeutung, daß „Zelle $i+1$“ den linken Nachbarn von Zelle i bezeichnet.

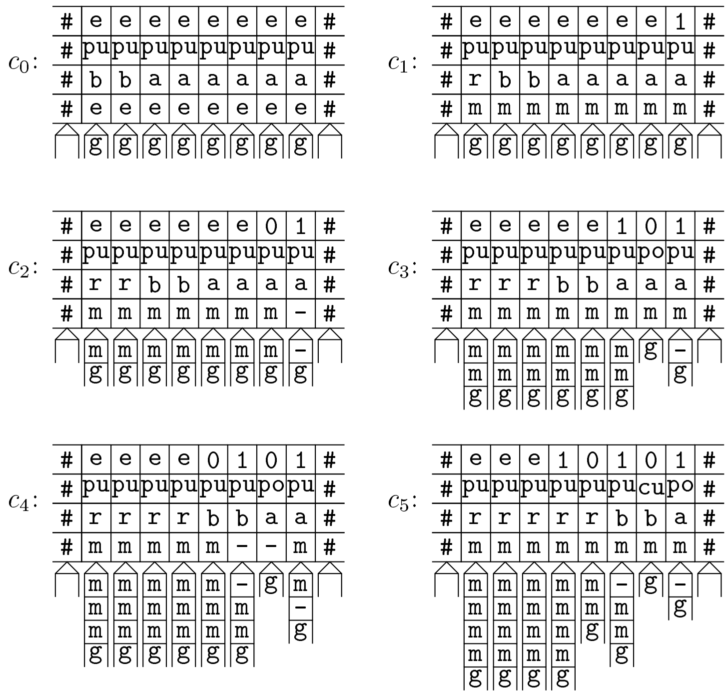
Auf der ersten Spur wird im Zeitpunkt 1 am rechten Rand ein Signal erzeugt, welches alle durchlaufenen Zellen abwechselnd mit `1` und `0` markiert. Auf diese Weise wird eine Unterteilung der „Positionen“ in gerade oder ungerade erreicht. Im weiteren werden auch die Bezeichnungen Typ0- und Typ1-Zelle benutzt. Im Laufe der Berechnung bilden sich dadurch Paare, die sich gegenseitig beeinflussen. Die hier angestrebte Interaktion zweier Zellen eines Paares macht es notwendig, daß jede Zelle den obersten Kellereintrag ihres Nachbarn kennt. Da dies per Definition von Keller-Zellularräumen nicht vorgesehen ist, wird das vierte Register zur Speicherung des obersten Kellersymbols genutzt. Dieses wird dann im allgemeinen nicht noch einmal im „wirklichen“ Keller notiert. Da das Kellerendesymbol weder aus dem Keller entfernt noch mehrfach in ihn hineingeschrieben werden kann, wird ein Kellersymbol `-` eingeführt, welches alternativ das relevante Kellerende anzeigt.

Nach diesen Vorbemerkungen kann jetzt das Gesamtverhalten einer Berechnung betrachtet werden (vgl. Abbildung 5.8).

In der Anfangskonfiguration befindet sich im zweiten Register aller Zellen das Zeichen `pu`, daher wird der Kellerinhalt in jedem weiteren Zeitschritt um ein Symbol `m` erweitert. Dieses Verhalten dauert an, bis die Zelle mit `1` oder `0` auf der ersten Spur markiert wird. Jetzt sind zwei Fälle zu unterscheiden:

– Eine Typ1-Zelle schreibt unmittelbar nach ihrer Markierung das Symbol `-` in den Keller, welches nach dem oben Gesagten den bisherigen Kellerinhalt unzugänglich macht. Man könnte auch sagen, der Keller ist leer.

– Eine Typ0-Zelle — sie wird einen Zeitschritt nach der zugehörigen Typ1-Zelle markiert — beendet unmittelbar den bisherigen Prozeß der Kellererweiterung. Ist i die Nummer der Zelle, dann beträgt die Kellertiefe im Zeitpunkt ihrer Markierung genau $i+1$, wobei $\pi_2(c_i(i)) = \mathtt{m}^{\cdot i}\mathtt{g}$ gilt.



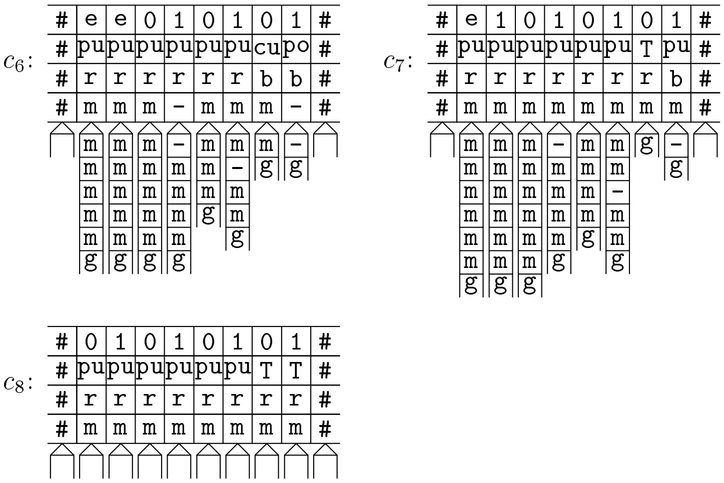


Abbildung 5.8. Beispiel zum Satz 5.27 für gerade Teiler.

In weiteren i Zeitschritten wird der Kellerinhalt der Typ0-Zelle — ohne das Kellerendesymbol — sukzessive in den Keller der „Partnerzelle“ vom Typ 1 transportiert. Dies ist, da i gerade ist, die rechte Nachbarzelle. Anschließend wird der Kellerinhalt wieder zurücktransportiert und das Verfahren beginnt erneut. Entsprechend ist nach jeweils i Zeitschritten einer der beiden Keller leer.

Zur Realisierung dieses Verfahrens werden die Symbole `pu` (push) und `po` (pop) im zweiten Register unterschieden, anhand derer jede Zelle die momentane Richtung des Transportes erkennen kann. Hier liegt auch der Grund dafür, daß jede Zelle den obersten Kellereintrag ihres Nachbarn kennen muß. Ansonsten wäre eine Umkehrung der Transportrichtung nur mit einem Zwischenschritt möglich.

Dieses Verfahren wird fortgesetzt, bis eine Typ0-Zelle zum ersten Mal das Zeichen `b` im dritten Register ihrer linken Nachbarzelle erkennt. Bei Eingabe des Wortes $\mathtt{b}^{\cdot n}\mathtt{a}^{\cdot m}$ ist dies nach $m - i$ Zeitschritten der Fall. Falls der Keller der Typ0- oder der Keller der Typ1-Zelle in diesem Zeitpunkt leer ist, ist i ein möglicher Teiler von m. Die Keller

werden zu den Zeitpunkten $z \cdot i$, $z \in \mathbb{N}$, leer, also muß gelten: $m - i = z \cdot i \Rightarrow m = (z+1)i$. (Es wude ja zunächst $m > n$ angenommen.)
Daher erzeugt die Typ0-Zelle ein Symbol `cu` statt `pu` bzw. ein Symbol `co` statt `po` im zweiten Register. Anschließend wird der Kellerinhalt in i weiteren Schritten nochmals ausgetauscht. Erkennt die Typ0-Zelle nun im dritten Register ihres linken Nachbarn das Symbol `r`, während in ihrem eigenen das letzte `b` notiert ist, generiert sie das Symbol `T` im zweiten Register und die Eingabe wird akzeptiert, weil nach gerade i Zeitschritten alle `b`s die Zelle passiert haben, also $n = i$ gilt.

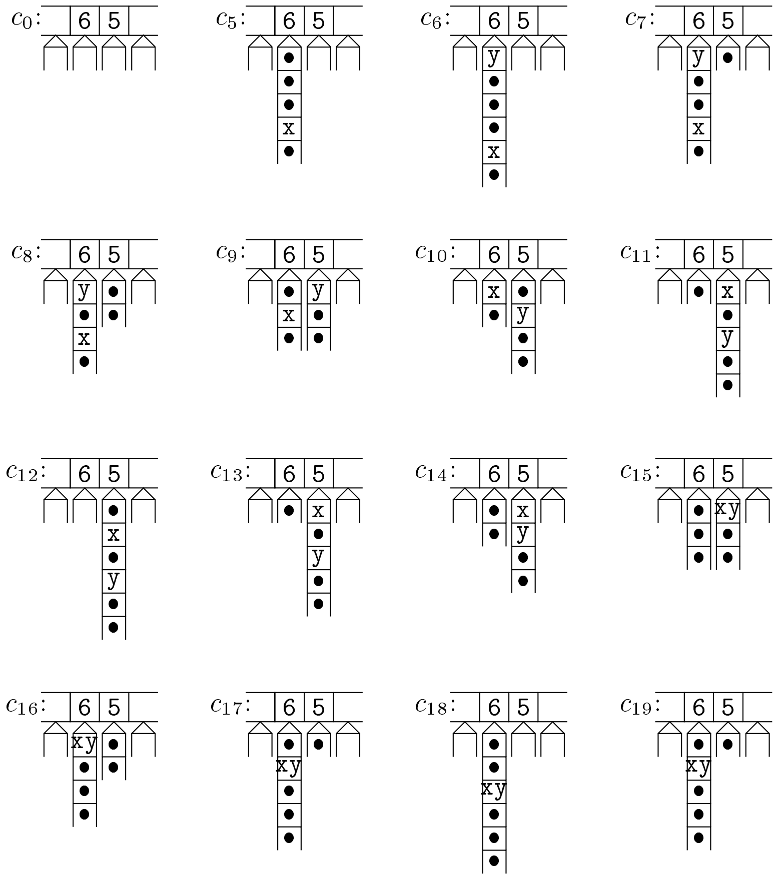
Der Gesamtzeitbedarf errechnet sich aus den Zeiten bis zum Empfang des ersten `b`, weiteren i Zeitschritten bis zum Empfang des letzten `b` und nochmals i Zeitschritten, bis das Symbol `T` den rechten Rand erreicht hat. Also $m - i + i + i = m + i$ Zeitschritte. Für $i = n$ folgt daraus Realzeit.

Für den Nachweis, daß $L \in \mathscr{L}_{rt}(\text{PDCA})$ gilt, muß die Konstruktion erweitert werden. Der zunächst ausgeschlossene Fall $n = m$ kann durch Sonderbehandlung auf einer zusätzlichen Spur erkannt werden. Da $\{\mathtt{b}^{\cdot n}\mathtt{a}^{\cdot n} | \; n \in \mathbb{N}\} \in L_{rt}(\text{CA})$ ist, ist dies mit Satz 5.25 a) ohne weiteres möglich.

Um auch ungerade Teiler zuzulassen, genügt es zu zeigen, daß ein Zellenpaar i und $i-1$ zusätzlich in die Lage versetzt werden kann, die Zeitpunkte $z(i-1), z \in \mathbb{N}$, zu erkennen. Das weitere Verfahren entspricht dann dem oben Beschriebenen.

Das Kelleralphabet und $\pi_4(S)$ werden um die Zeichen `x`, `y` und `xy` erweitert. Zusätzlich wird jede Zelle dahingehend modifiziert, daß sie beim initialen Kellerfüllen als zweites Symbol ein `x` anstelle des `m` in den Keller schreibt. Außerdem notiert jede Typ0-Zelle im Zeitpunkt ihrer Markierung ein Symbol `y` anstelle des `m` als oberstes Kellerzeichen (vgl. Abbildung 5.9).

Beim anschließenden Transport der Kellerinhalte müssen diese Marken `x` und `y` jetzt jeweils so verschoben werden, daß in jedem Zeitpunkt $z(i-1), z \in \mathbb{N}$, gerade ein Symbol `x` oder `y` von einer Zelle zur anderen übertragen wird. Dies läßt sich folgendermaßen erreichen: Bei einem Transport von links nach rechts soll das Übertragen des Symbols `x` den Zeitpunkt anzeigen, beim Transport in die andere Richtung entsprechend `y`. Das Prinzip wird für `y` beschrieben; für die Marke `x` ist es identisch.



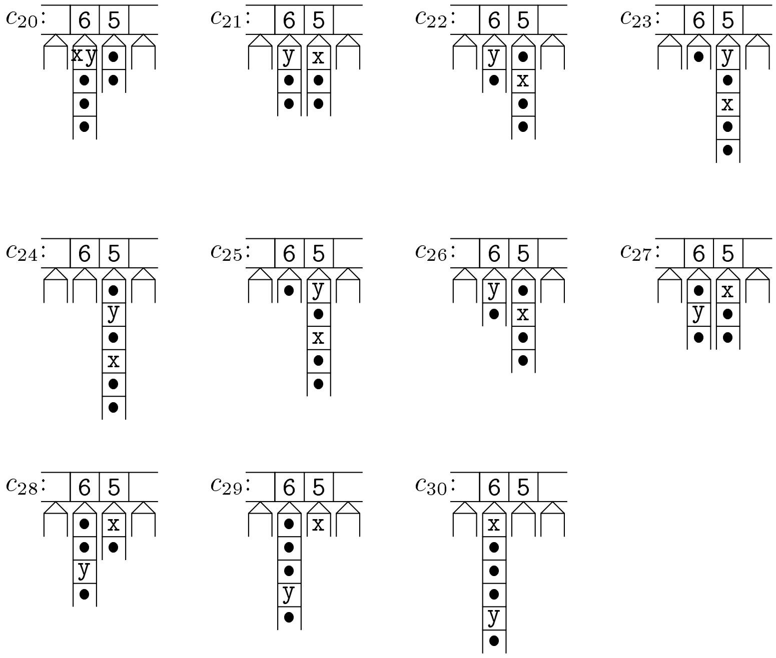


Abbildung 5.9. Schematische Darstellung der Kelleroperationen zur Erkennung ungerader Teiler.

Wird das Symbol `y` von rechts nach links übertragen, zeigt es einen gewünschten Zeitpunkt an. Bei einem Transport von links nach rechts wird die Übertragung um zwei Zeitschritte verzögert, das heißt, zunächst wird zweimal statt `y` ein `m` in den Keller geschrieben und nach der Verzögerung anstatt eines `m` wieder `y`.

Eine Analyse dieser Methode ergibt unter Vernachlässigung von `x`:

$t = i$: Im Keller der Zelle i befindet sich das Wort $\mathtt{ym}^{\cdot(i-1)}\mathtt{g}$.
$t = 2i$: Nachdem eine Verzögerung von `y` stattgefunden hat, befindet sich im Keller der Zelle $i-1$ das Wort $\mathtt{m}^{\cdot(i-3)}\mathtt{ymmg}$. Also wird `y` nach weiteren $i-3$ Zeitschritten oben im Keller zum Transport anstehen. Dies markiert den Zeitpunkt $2i + i - 3 = 3(i-1)$.

$t = 3i$: Im Keller der Zelle i befindet sich das Wort $\texttt{mmym}^{\cdot(i-3)}\texttt{g}$.
$t = 4i$: Wiederum hat eine Verzögerung stattgefunden. Im Keller der Zelle $i-1$ steht das Wort $\texttt{m}^{\cdot(i-5)}\texttt{ym}^{\cdot 4}\texttt{g}$. Also steht $\texttt{y}$ nach weiteren $i-5$ Zeitschritten zum Transport an. Dies markiert den Zeitpunkt $4i+i-5 = 5(i-1)$. Da der Transport der Marke $\texttt{y}$ immer weiter verzögert wird, kann dieser Vorgang nur endlich oft wiederholt werden.

$t = (i-1)i$:
Im Keller der Zelle i befindet sich das Wort $\texttt{m}^{\cdot(i-2)}\texttt{ym}^{\cdot(i-(i-1))}\texttt{g}$. Diese Situation unterscheidet sich von derjenigen im Zeitpunkt i nur dadurch, daß die Symbole $\texttt{x}$ und $\texttt{y}$ vertauscht sind. (Dies wurde für $\texttt{y}$ gezeigt. Der Nachweis für $\texttt{x}$ ist identisch.) Die Zellen können nun das Verfahren von neuem beginnen, wobei die Bedeutung der Symbole $\texttt{x}$ und $\texttt{y}$ vertauscht wird.

Es bleibt noch zu zeigen, daß die Zellen den Zeitpunkt $(i-1)i$ erkennen können. Nun ist $(i-1)i$ ein gemeinsames Vielfaches von i und $(i-1)$, und die Zellen wurden ja gerade so konstruiert, daß sie Vielfache von i und $(i-1)$ erkennen können. Da für alle $i-1 \in \mathbb{N}$ das kleinste gemeinsame Vielfache von $i-1$ und i deren Produkt ist, können die Zellen diesen Zeitpunkt eindeutig erkennen. Damit leistet die Konstruktion das Gewünschte. $\square$

Literaturverzeichnis

Balzer, R. M. *An 8-state minimal time solution to the firing squad synchronization problem.* Information and Control 10 (1967), 22–42.

Becker, U. *Methoden zur Konstruktion und Komposition zeitlich verzerrter zellularer Algorithmen.* Diplomarbeit, Universität Gießen, 1990.

Beyer, W. T. *Recognition of topological invariants by iterative arrays.* Bericht MAC TR-66, Massachusetts Institute of Technology, 1969.

Bleck, B. und Kröger, H. *Cellular algorithms.* In Evans, D. (Hrsg.), *Advances in Parallel Computing Volumne 2.* JAI Press, London, 1992, S. 115–143.

Choffrut, C. und Culik II, K. *On real-time cellular automata and trellis automata.* Acta Informatica 21 (1984), 393–407.

Chomsky, N. *On certain formal properties of grammars.* Bericht QPR 65, Massachusetts Institute of Technology, 1962.

Church, A. *An unsolvable problem of elementary number theory.* American Journal of Mathematics 58 (1936), 345–363.

Codd, E. F. *Cellular Automata.* Academic Press, New York, 1968.

Cole, S. N. *Real-time computation by n-dimensional iterative arrays of finite-state machines.* IEEE Conference Record of 7^{th} Annual Symposium on Switching and Automata Theory, 1966, S. 53–77.

Dyer, C. R. *One-way bounded cellular automata.* Information and Control 44 (1980), 261–281.

Ginsburg, S. und Rice, H. G. *Two families of languages related to ALGOL.* Journal of the ACM 9 (1962), 350–371.

Ginsburg, S., Greibach, S. A., und Harrison, M. A. *One-way stack automata*. Information and Control (1967), 350–371.

Gvozdjak, P. *One letter context-free languages revisited*. Bulletin of the European Association for Theoretical Computer Science 49 (1993), 159–162.

Hillis, W. D. *The Connection Machine*. MIT Press, Cambridge, Massachusetts, 1985.

Ibarra, O. H. und Jiang, T. *Relating the power of cellular arrays to their closure properties*. Theoretical Computer Science 57 (1988), 225–238.

Kosaraju, S. R. *On some open problems in the theory of cellular automata*. IEEE Transactions on Computers C-23 (1974), 561–565.

Kutrib, M. *Kellererweiterte Polyautomaten*. Dissertation, Universität Gießen, Gießen, 1993a.

Kutrib, M. *Real-time language recognition by pushdown cellular automata*. Workshop on Parallel Processing, Lessach, 1993b.

Mazoyer, J. *A six-state minimal time solution to the firing squad synchronization problem*. Theoretical Computer Science 50 (1987), 183–238.

Moore, E. F. (Hrsg.). *Sequential Machines. Selected papers*. Addison-Wesley, Reading, Massachusetts, 1964.

Moore, F. R. und Langdon, G. C. *A generalized firing squad problem*. Information and Control 12 (1968), 17–33.

Müller, H. *Selbstreproduktion in zellularen Netzen*. In *Jahrbuch Überblicke Mathematik*. BI-Wissenschaftsverlag, Mannheim, 1978, S. 67–86.

Preston Jr., K. und Duff, M. J. B. *Modern Cellular Automata Theory and Applications*. Plenum Press, New York, 1984.

Reed, M. A. *Quantenpunkte*. Spektrum der Wissenschaft (1993), 52–57.

Rosenstiehl, P., Fiksel, J. R., und Holliger, A. *Intelligent graphs: Networks of finite automata capable of solving graph problems*. In Read, R. C. (Hrsg.), *Graph Theory and Computing*. Academic Press, New York, 1972, S. 219–265.

Salomaa, A. *Formal Languages*. Academic Press, New York, 1973.

Seidel, S. R. *Language recognition and the synchronization of cellular automata*. Bericht 79-02, Department of Computer Science, University of Iowa, Iowa City, 1979.

Shinahr, I. *Two- and three-dimensional firing-squad synchronization problems*. Information and Control 24 (1974), 163–180.

Smith III, A. R. *Cellular automata and formal languages*. IEEE Conference Record of 11^{th} Annual Symposium on Switching and Automata Theory, 1970, S. 216–224.

Smith III, A. R. *Cellular automata complexity trade-offs*. Information and Control 18 (1971a), 466–482.

Smith III, A. R. *Simple computation–universal cellular spaces*. Journal of the ACM 18 (1971b), 339–353.

Smith III, A. R. *Real-time language recognition by one-dimensional cellular automata*. Journal of Computer and System Sciences 6 (1972), 233–253.

Smith III, A. R. *Introduction to and survey of polyautomata theory*. In Lindenmayer, A. und Rozenberg, G. (Hrsg.), *Automata, Languages, Development*. North-Holland, Amsterdam, 1976, S. 405–422.

Toffoli, T. und Margolus, N. *Cellular Automata Machines*. MIT Press, Cambridge, Massachusetts, 1988.

Turing, A. *On computable numbers, with an application to the Entscheidungsproblem*. Proceedings of the London Mathematical

Society, Series 2 42 (1936), 231–265.

Umeo, H., Morita, K., und Sugata, K. *Deterministic one-way simulation of two-way real-time cellular automata and its related problems*. Information Processing Letters 14 (1982), 158–161.

Vollmar, R. *Algorithmen in Zellularautomaten*. Teubner, Stuttgart, 1979.

von Neumann, J. *Theory of Self-Reproducing Automata*. Herausgegeben und vervollständigt von A. W. Burks. University of Illinois Press, Urbana, 1966.

Waksman, A. *An optimum solution to the firing squad synchronization problem*. Information and Control 9 (1966), 66–78.