4. ZA mit konstanter Zustandsänderungsbeschränkung

Schon mit einer konstanten Anzahl von Zustandsänderungen lassen sich relativ komplexe Sprachen erkennen, so zum Beispiel $\{a^n b^n c^n \mid n \in \mathbb{N}\}$ oder die in Abschnitt 3.5 vorgestellte Sprache L_B. Daher werden die Zellularautomaten der Zustandsänderungskomplexität O(1) in diesem Kapitel genauer untersucht und die entsprechenden Sprachklassen feiner strukturiert. Die vorgestellten Ergebnisse sind teilweise schon in [Vo81] enthalten, wo die Klasse ÄZA(O(1)) betrachtet wird.

4.1 Die Sprachen mit Zustandsänderungskomplexität O(1)

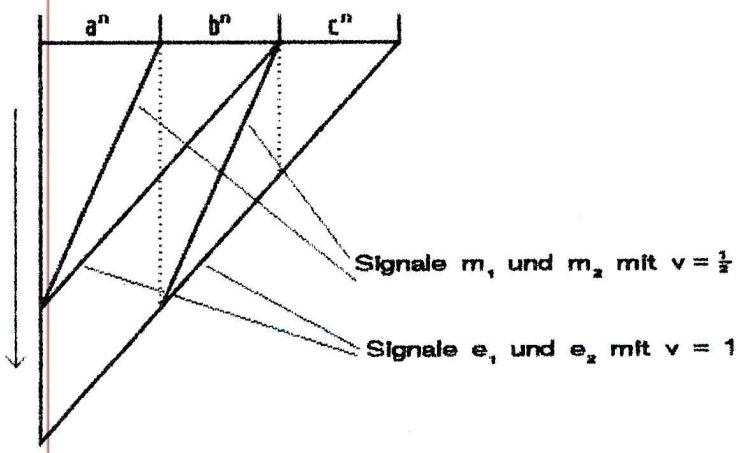
In den vorangehenden Abschnitten wurden bereits einige Sprachen mit Änderungskomplexität O(1) vorgestellt, so zum Beispiel die Sprachen L_B, L_{BB}, $L_{B\&}$ und ihre Spiegelbilder aus 3.5. Ein Ergebnis dieses Abschnittes war, daß die Zustandsänderungskomplexität einer Sprache vom betrachteten Zellularautomatenmodell abhängt, also von der Wahl der Nachbarschaft (H_1 oder $\overline{H}_1$) und einer eventuellen Beschränkung auf Realzeit. Gemeinsam ist jedoch allen Modellen, daß eine echte Obermenge der regulären Sprachen erkannt wird.
Eine einfache Charakterisierung dieser Sprachklassen oder eine Äquivalenz mit anderen, bereits untersuchten Sprachklassen ist nicht bekannt. Es ist auch nicht sicher, ob die Klasse ÄZA(O(1)) in der Menge der Realzeitsprachen der bidirektionalen ZA enthalten ist: In [BuC84] wird vermutet, daß $L_t := \{a^k b^n \mid k,n \in \mathbb{N} \text{ und } n \text{ teilt } k\}$ von keinem Realzeit-ZA mit H_1-Raster erkannt wird. L_t läßt sich aber von einem O(1)-änderungsbeschränkten ZA erkennen; ein entsprechendes Verfahren erhält man mit Satz 3.5 aus einem einfacher zu findenden O(1)-änderungsbeschränkten Verfahren für $(L_t)^R$.

Satz 4.1 **Für $Z \in \{ÄZA, ÄZA_R, ÄUZA, ÄUZA_R\}$ gilt:**
[Vo81] **a) $\mathfrak{L}_3 \subseteq Z(1)$**
b) $\mathfrak{L}_2 \not\supseteq Z(O(1))$.

Beweis: a) Am Anfang des Abschnitts 3.2 wurde ein Verfahren zur Simulierung eines endlichen Automaten durch einen UZA mit einer Zustandsänderung pro Zelle angegeben.

b) Nach Lemma 3.5 ist die kontextfreie Sprache $L_R = \{uYYu^R \mid u \in \{a,b\}^+\}$ nicht in $Z(O(1))$. Andererseits ist die Sprache $\{a^n b^n c^n \mid n \in \mathbb{N}\}$ aus $\mathfrak{L}_1 \setminus \mathfrak{L}_2$ mit 4 Zustandsänderungen pro Zelle in einem Realzeit-UZA erkennbar.

Abb. 4.1 Erkennung von $\{a^n b^n c^n \mid n \in \mathbb{N}\}$



Funktionsweise: Durch die Signale m_1 und e_1 wird die Anzahl der a's und b's, durch m_2 und e_2 die Anzahl der b's und c's auf Gleichheit überprüft. Durch e_2 wird die Eingabe zusätzlich auf die Form $a^k b^m c^n$ geprüft.

◇

Satz 4.2 $\ddot{A}UZA_R(1) = \ddot{A}UZA(1) = \ddot{A}ZA_R(1) = \ddot{A}ZA(1) = \mathfrak{L}_3$

Beweis: In Satz 4.1 a) wurde $\mathfrak{L}_3 \subseteq \ddot{A}UZA_R(1)$ gezeigt. Nimmt man an, daß die Aussage $\ddot{A}ZA(1) \subseteq \mathfrak{L}_3$ bereits bewiesen ist, so folgt mit Satz 2.1 die Behauptung.

Zu zeigen ist also lediglich noch $\ddot{A}ZA(1) \subseteq \mathfrak{L}_3$.

Ein ZA, in dem nur eine Zustandsänderung pro Zelle stattfindet, ist nach Lemma 3.1 auf Realzeit beschränkt. Bei einer Eingabe der Länge n nimmt der akzeptierende Automat des ZA zu einem Zeitpunkt $t_o \leq n$ einen Zustand aus $\{a, \omega\}$ an.

Eine Zustandsänderung kann aber in einer Zelle zu einem Zeitpunkt t nur auftreten, wenn entweder $t = 0$ gilt oder eine Zelle in der Nachbarschaft zum Zeitpunkt $t-1$ ihren Zustand gewechselt hat. Wenn in einem 1-max-zustandsänderungsbeschränkten ZA mit H_1-Nachbarschaft die akzeptierende Zelle in einen Zustand aus $\{\alpha, \omega\}$ wechselt, so muß zum Zeitpunkt t_o-1 die Zelle rechts von der akzeptierenden Zelle ihren Zustand wechseln. Auch für $t = t_o-2, t_o-3, \ldots, 2, 1$ kann die Zustandsänderung zum vorangehenden Zeitpunkt nur in der jeweils nächsten weiter rechts liegenden Zelle stattgefunden haben, da ja alle weiter links liegenden Zellen ihren Zustand zu einem späteren Zeitpunkt wechseln.
Das Ergebnis der Berechnung hängt also nur von einem einzigen, zum Zeitpunkt 0 gestarteten, mit Geschwindigkeit 1 nach links laufenden und zum Zeitpunkt t_o in der akzeptierenden Zelle ankommenden Signal ab.

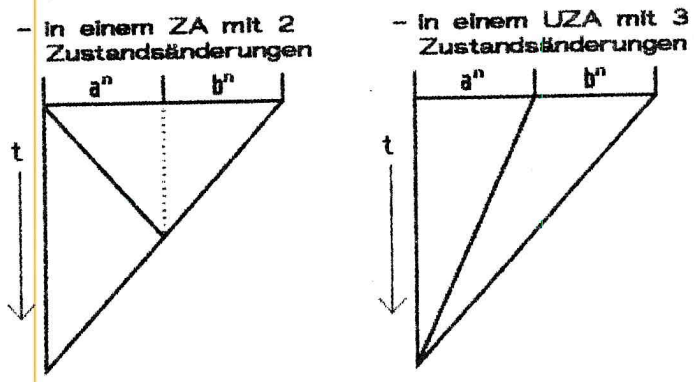
Gilt für alle Eingaben $t_o = n$, so wurde das Signal von der Zelle am rechten Rand der Retina gestartet. Der gesamte Zellularautomat läßt sich dann von einem endlichen Automaten simulieren, der die in den Zellen des ZA enthaltene Eingabe von rechts nach links liest und seinen Zustand entsprechend ändert.

Falls t_o kleiner als die Wortlänge ist, wird das Signal nicht vom rechten Rand, sondern von einer Zelle im Innern des ZA gestartet. Das Ergebnis der Berechnung ist dann nur von dem Präfix der Eingabe abhängig, der von dem Signal durchlaufen wird. Die Menge P dieser Präfixe kann genauso wie im Fall $t_o = n$ von einem endlichen Automaten erkannt werden. Die vom ZA erkannte Sprache hat dann die Form PX^* und ist damit ebenfalls regulär.

◇

Vermutlich können in einem UZA auch mit 2 Zustandsänderungen pro Zelle nur reguläre Sprachen erkannt werden, ein Beweis steht aber noch aus. Betrachtet man ZA mit zwei bzw. UZA mit drei Zustandsänderungen pro Zelle, so werden bereits einige nichtreguläre Sprachen erkannt, so zum Beispiel $\{a^n b^n \mid n \in \mathbb{N}\}$:

Abb. 4.2 Erkennung von $\{a^n b^n \mid n \in \mathbb{N}\}$



Wie im nächsten Abschnitt bewiesen wird, kann man durch Erhöhung der Anzahl der zulässigen Zustandsänderungen jeweils weitere Sprachen erkennen. Es existieren also unendliche Hierarchien von Sprachklassen innerhalb der Klassen $Z(O(1))$, $Z \in \{$ÄZA, $ÄZA_R$, ÄUZA, $ÄUZA_R\}$.
Um die entsprechenden Beweise zu erleichtern, wird vorher noch das folgende Lemma bewiesen. Das Kriterium für änderungsbeschränkte UZA aus Satz 3.3 läßt sich für die Sprachfamilien ÄUZA(r) und $ÄUZA_R(r)$ mit $r \in \mathbb{N}$ wie folgt vereinfachen:

Lemma 4.1 **Es sei $L \subseteq V^*$ und $r \in \mathbb{N}$. Dann gilt:**

a) Wird L von einem r-zustandsänderungsbeschränkten unidirektionalen ZA $\mathfrak{A} = (A, 1, \overline{H}_1, F)$ erkannt, so gilt für alle $n, k \in \mathbb{N}$

$$Äq\left(\overline{V}^n, {}_k\sim (\mathrm{mod}\, L)\right) < |A|^{r+1} (r(n+k))^r$$

b) Wird L von einem r-zustandsänderungsbeschränkten unidirektionalen ZA $\mathfrak{A} = (A, 1, \overline{H}_1, F)$ in Realzeit erkannt, so gilt für alle $n, k \in \mathbb{N}$

$$Äq\left(\overline{V}^n, {}_k\sim (\mathrm{mod}\, L)\right) = |A|^{r+1} n^r = Äq\left(\overline{V}^n, \sim (\mathrm{mod}\, L)\right).$$

Beweis: a) Nach Lemma 3.1 ist ein r-zustandsänderungsbeschränkter ZA $r \cdot n$-zeitbeschränkt. Setzt man diese Größen in Satz 3.3 ein und schätzt anschließend den entstehenden Term $\tau(n+k)-k$ durch $r \cdot (n+k)$ ab, so erhält man a).

b) Der Beweis erfolgt wieder durch Einsetzen in Satz 3.3. Bei einem auf Realzeit beschränkten ZA vereinfacht sich dann der Term $\tau(n+k)-k$ zu n. Da der Gesamtterm unabhängig von k ist, erhält man damit auch eine Abschätzung der Äquivalenzklassen bezüglich der Relation $\sim (\mathrm{mod}\, L)$.

◇

Lemma 4.1 a) wird im nächsten Abschnitt zum Beweis einer Hierarchie innerhalb der Klassen der O(1)-änderungsbeschränkten Sprachen benutzt. Bemerkenswert an Teil b) ist, daß die Abschätzung der Äquivalenzklassen bezüglich ${}_k\sim$ in O(1)-zustandsänderungsbeschränkten Realzeit-UZA unabhängig von k wird.

4.2 Die Hierarchien der ZA mit endlicher Zustandsänderungszahl

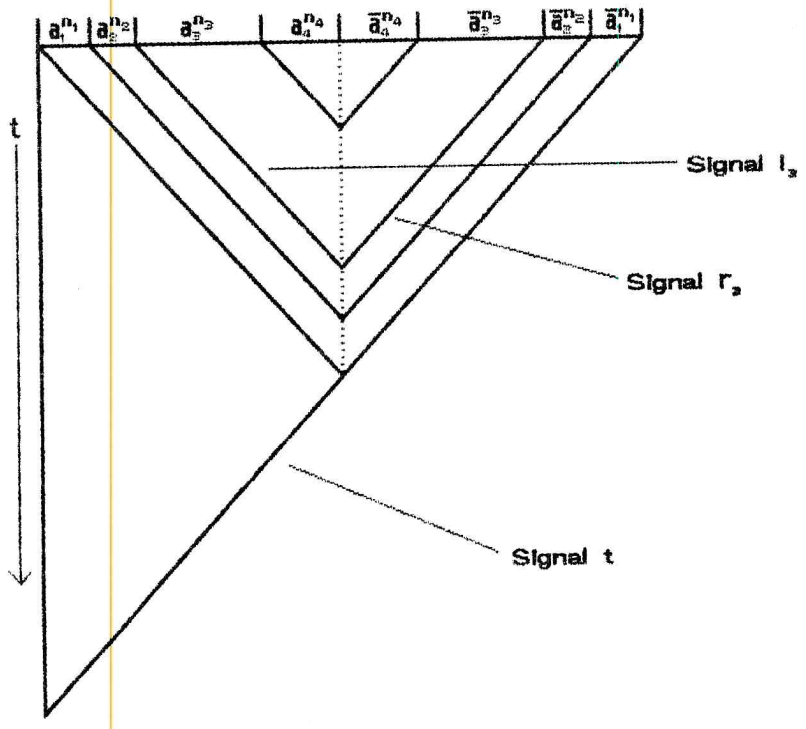
Anhand der in diesem Abschnitt definierten Sprachen $L_{(\lambda)}$ kann die Existenz einer unendlichen Hierarchie der Zellularautomaten mit konstanter Zustandsänderungsbeschränkung bewiesen werden. Dazu wird gezeigt, daß es zu jedem $\alpha \in \mathbb{N}$ ein $\lambda \in \mathbb{N}$ gibt, so daß die Klasse $Z(\alpha)$ für $Z \in \{ÄZA, ÄZA_R, ÄUZA, ÄUZA_R\}$ echt in der Klasse $Z(\lambda)$ enthalten ist.

Definition 4.1 Für $\lambda \in \mathbb{N}$ sei $L_{(\lambda)} := \{ a_1^{n_1} a_2^{n_2} \dots a_\lambda^{n_\lambda} \bar{a}_\lambda^{n_\lambda} \dots \bar{a}_2^{n_2} \bar{a}_1^{n_1} \mid n_\nu \in \mathbb{N} \}$.

Lemma 4.2 a) $L_{(\lambda-1)} \in ÄZA_R(\lambda)$
b) $L_{(\lambda-2)} \in ÄUZA_R(\lambda)$

Beweis: a) Ein Algorithmus für $L_{(\lambda)}$ wird anhand der folgenden Abbildung beschrieben.

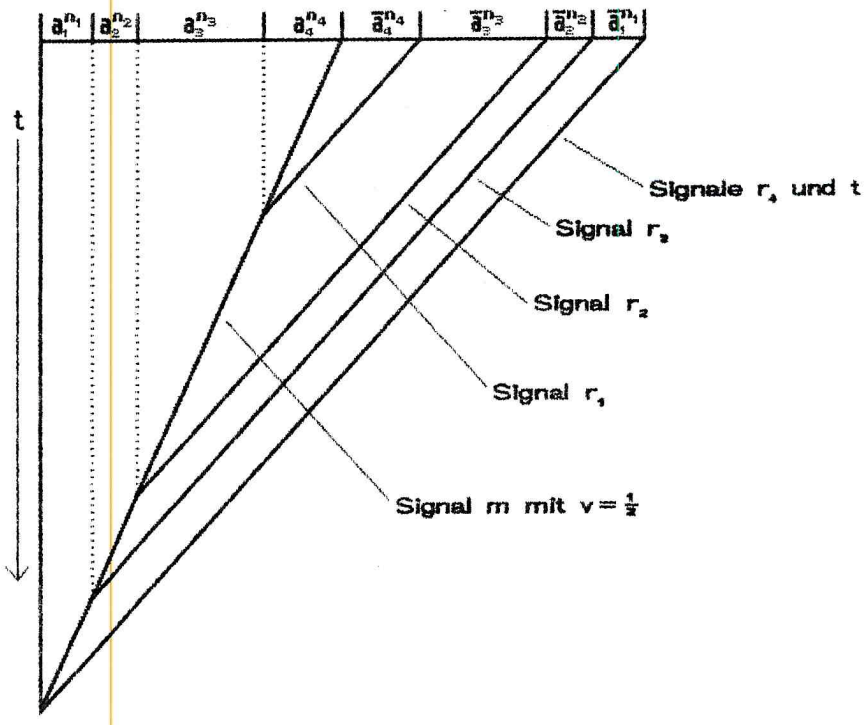
Abb. 4.3 Erkennung von $L_{(\lambda)}$ am Beispiel einer Eingabe aus $L_{(4)}$



Funktionsweise: Durch das vom rechten Rand der Retina nach links laufende Signal t wird die Eingabe auf die Form $\{a_1\}^+\{a_2\}^+ \dots \{a_\lambda\}^+\{\bar{a}_\lambda\}^+ \dots \{\bar{a}_2\}^+\{\bar{a}_1\}^+$ überprüft. Außerdem wird vom linken Rand jedes Teilwortes $a_k^{n_k}$, $k \le \lambda$ ein l_k-Signal mit Einheitsgeschwindigkeit nach rechts und vom rechten Rand jedes $\bar{a}_k^{n_k}$ ein r_k-Signal nach links geschickt.
Treffen sich für alle $k \le \lambda$ die Signale l_k und r_k genau zwischen dem letzten a_λ und dem ersten $\bar{a}_\lambda$, so ist das Eingabewort in $L_{(\lambda)}$.
Zur Erkennung der Sprache $L_{(\lambda)}$ werden dabei nicht mehr als $\lambda+1$ Zustandsänderungen benötigt.

b) Zur Erkennung von $L_{(\lambda)}$ in einem UZA wird folgendes Verfahren verwendet:

Abb. 4,4 Erkennung von $L_{(4)}$ in einem UZA



Funktionsweise: Das Signal t überprüft wieder die Eingabe auf Wohlgeformtheit. Von der mutmaßlichen Mitte der Eingabe wird ein Signal t mit der Geschwindigkeit $\frac{1}{2}$ gestartet. Gleichzeitig wird vom rechten Rand jedes Teilwortes $\bar{a}_k^{n_k}$, $k \leq \lambda$ ein Signal r_k mit Einheitsgeschwindigkeit nach links geschickt. Ist die Eingabe aus $L_{(\lambda)}$, so wird jedes r_k das langsamere m-Signal genau am linken Rand des Teilwortes $a_k^{n_k}$ einholen.

◇

Lemma 4.3 Für $\lambda \in \mathbb{N}$ gilt $L_{(2\lambda+2)} \notin \ddot{A}ZA(\lambda)$.

Beweis: Es wird das Beweisschema aus Lemma 3.6 verwendet:

Ist $\mathfrak{A}$ ein ZA mit $L_{\mathfrak{A}} = L_{(2\lambda+2)}$ und sind $w = uv$, $\bar{w} = \bar{u}\bar{v}$ zwei verschiedene Wörter aus $L_{(2\lambda+2)}$ mit $|u| = |v| = |\bar{u}| = |\bar{v}| = k$, so würde nach Lemma 3.2 aus $\langle c_o^w(k,k+1)\rangle = \langle c_o^{\bar{w}}(k,k+1)\rangle$ folgen, daß auch $u\bar{v}$ und $\bar{u}v$ von $\mathfrak{A}$ akzeptiert werden. Da diese Wörter jedoch nicht in der Sprache $L_{(2\lambda+2)}$ enthalten sind, muß für jedes Wort einer festen Länge $n = 2k$ aus der Sprache ein eigener Verhaltensfolgeausschnitt $\langle c_o(k,k+1)\rangle$ existieren.

Die Anzahl der Wörter der Länge $n = 2k$ in $L_{(2\lambda+2)}$ beträgt

$$\binom{k-1}{2\lambda+1} = O(n^{2\lambda+1}).$$

In einem λ-max-änderungsbeschränkten und daher nach Lemma 3.1 $\lambda \cdot n$-zeitbeschränkten ZA gibt es aber nach Lemma 3.4 mit $\rho = \lambda$ und $\tau = \lambda \cdot n$ weniger als

$$(\lambda \cdot n)^{2\lambda} |A|^{2\lambda+2} = O(n^{2\lambda})$$

Verhaltensfolgeausschnitte $\langle c_o(k,k+1) \rangle$.

Also kann $L_{(2\lambda+2)}$ von keinem λ-max-änderungsbeschränkten ZA erkannt werden.

◇

Aus Lemma 4.3 folgt unmittelbar

Satz 4.3 **a) Es gilt $Z(\lambda) \subset Z(2\lambda+4)$ für $\lambda \in \mathbb{N}$ und $Z \in \{ÄZA, ÄZA_R, ÄUZA, ÄUZA_R\}$.**
b) Für $\lambda \in \mathbb{N}$ gilt $ÄZA(\lambda) \subset ÄZA(2\lambda+3)$.

Beweis: a) Nach Lemma 4.2 b) ist $L_{(2\lambda+2)}$ in $Z(2\lambda+4)$, aber nicht in $Z(\lambda)$ enthalten.

b) Nach Lemma 4.2 a) ist $L_{(2\lambda+2)}$ in $ÄZA(2\lambda+3)$, aber nicht in $ÄZA(\lambda)$ enthalten.

◇

Für die Klassen der unidirektionalen ZA läßt sich die Hierarchie aus Satz 4.3 a) noch verfeinern. Mit Hilfe von Lemma 4.1 erhält man das folgende Ergebnis.

Lemma 4.4 **Für $\lambda \in \mathbb{N}$ gilt $L_{(\lambda+1)} \notin ÄUZA(\lambda)$.**

Beweis: Nach Lemma 4.1 a) gilt für eine von einem λ-zustandsänderungsbeschränkten UZA erkannte Sprache L

$$Äq\left(\overline{X^m}, {}_k\sim (\bmod L)\right) < |A|^{\lambda+1} (\lambda(m+k))^{\lambda} = O(n^{\lambda}) \quad \text{mit } n = m+k.$$

Jedes Wort w der Form $\{\bar{a}_{\lambda+1}^{n_{\lambda+1}}\}^* \ldots \{\bar{a}_2^{n_2}\}^* \{\bar{a}_1^{n_1}\}^*$ mit $|w| \le k$ ist in einer anderen Äquivalenzklasse bezüglich ${}_k\sim (\bmod L_{(\lambda+1)})$. Damit ist die Anzahl der Äquivalenzklassen größer oder gleich der Anzahl der Wörter dieser Form und es folgt mit $n = 2k$:

$$Äq\left(\overline{X^k}, {}_k\sim (\bmod L_{(\lambda+1)})\right) \ge \binom{k-1}{\lambda+1} = O(k^{\lambda+1}) = O(n^{\lambda+1})$$

Also kann $L_{(\lambda+1)}$ von keinem λ-max-änderungsbeschränkten UZA erkannt werden.

◇

Zusammen mit Lemma 4.2 b) folgt:

Satz 4.4 **Für $\lambda \in \mathbb{N}$ gilt $ÄUZA(\lambda) \subset ÄUZA(\lambda+3)$ und $ÄUZA_R(\lambda) \subset ÄUZA_R(\lambda+3)$.**

Auch für die Klassen $\ddot{A}ZA_R(\lambda)$, $\lambda \in \mathbb{N}$, läßt sich ein ähnliches Ergebnis erhalten. Dazu werden die Sprachen $L_{(\lambda)}$ durch Einfügen von &-Zeichen gemäß Abschnitt 3.5 modifiziert. Für die dadurch erhaltenen Sprachen $L_{(\lambda)\&}$ wird die Anzahl der Äquivalenzklassen bezüglich der Relation $\overset{}{}_k\approx$ aus Definition 3.5 abgeschätzt. Satz 3.7 liefert dann die gegenüber Satz 4.3 verfeinerte Hierarchie.

Definition 4.2 **Die Sprache $L_{(\lambda)\&}$ wird für $\lambda \in \mathbb{N}$ definiert durch**

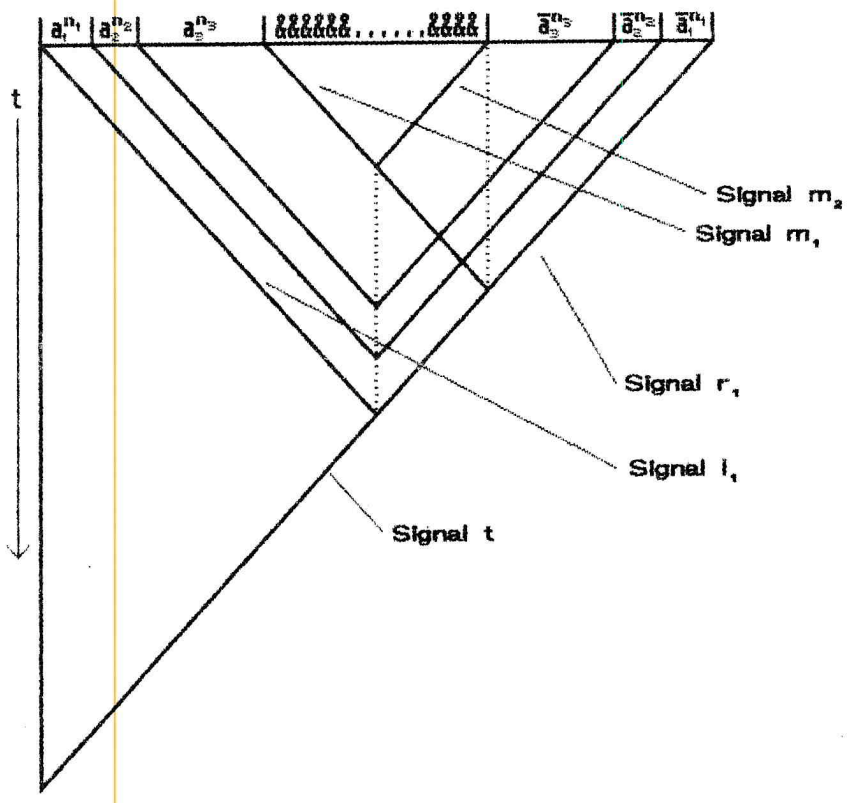
$$L_{(\lambda)\&} := \{ a_1^{n_1} a_2^{n_2} \dots a_\lambda^{n_\lambda} \&^{(n_1+n_2+\dots+n_\lambda)} \bar{a}_\lambda^{n_\lambda} \dots \bar{a}_2^{n_2} \bar{a}_1^{n_1} \mid n_\nu \in \mathbb{N} \}.$$

Lemma 4.5 **a)** $L_{(\lambda)\&} \in \ddot{A}ZA_R(\lambda+2)$.

b) $L_{(\lambda+2)\&} \notin \ddot{A}ZA_R(\lambda)$.

Beweis: a) Das folgende Verfahren erkennt $L_{(\lambda)\&}$ mit $\lambda+2$ Zustandsänderungen pro Zelle.

Abb. 4.5 Erkennung von $L_{(\lambda)\&}$ am Beispiel einer Eingabe aus $L_{(3)\&}$



Funktionsweise: Das vom rechten Rand der Retina kommende Signal t überprüft die Eingabe auf die Form $\{a_1\}^+\{a_2\}^+ \dots \{a_\lambda\}^+\{\&\}^+\{\bar{a}_\lambda\}^+ \dots \{\bar{a}_2\}^+\{\bar{a}_1\}^+$. Durch m_1 und m_2 wird die Mitte des Eingabewortes ermittelt, durch m_1 und t wird außerdem geprüft, ob in der Mitte die richtige Anzahl von &-Zeichen steht. Analog zum Verfahren für $L_{(\lambda)}$ in Lemma 4.2 a) werden Signale l_k und r_k, $k \le \lambda$, gestartet, die sich bei einem Wort aus der Sprache $L_{(\lambda)\&}$ jeweils genau in der Mitte treffen.

b) Nach Satz 3.7 gilt für eine von einem λ-zustandsänderungsbeschränkten ZA in Realzeit erkannte Sprache L

$$\ddot{A}q\left(X^m, {}_k\approx(\mathrm{mod}\, L)\right) < |A|^{\lambda+1}\, m^{\lambda} = O(n^{\lambda}) \text{ mit } n = 2m+k,$$

Jedes Wort der Form $\{\bar{a}_{\lambda+1}^{n_{\lambda+1}}\}^* \ldots \{\bar{a}_2^{n_2}\}^* \{\bar{a}_1^{n_1}\}^*$ mit Länge k ist in einer anderen Äquivalenzklasse bezüglich ${}_k\approx(\mathrm{mod}\, L_{(\lambda+2)\&})$. Damit ist die Anzahl der Äquivalenzklassen größer oder gleich der Anzahl der Wörter dieser Form und es folgt mit $n = 3k$

$$\ddot{A}q\left(X^k, {}_k\sim(\mathrm{mod}\, L_{(\lambda+2)\&})\right) \geq \binom{k-1}{\lambda+1} = O(k^{\lambda+1}) = O(n^{\lambda+1}).$$

Also kann $L_{(\lambda+2)\&}$ von keinem λ-max-änderungsbeschränkten ZA in Realzeit erkannt werden.

◇

Zu beachten ist, daß bei der Abschätzung der Anzahl der Äquivalenzklassen von ${}_k\approx$ im Gegensatz zum Beweis von Lemma 4.4 nur die Wörter mit genau der Länge k betrachtet werden. Dadurch ist das folgende Ergebnis über die Klassen $\ddot{A}ZA_R(\lambda)$ etwas schlechter als das für UZA in Satz 4.4.

Satz 4.5 Für $\lambda \in \mathbb{N}$ gilt $\ddot{A}ZA_R(\lambda) \subset \ddot{A}ZA_R(\lambda+4)$.

Vermutlich gilt sogar $Z(\lambda) \subset Z(\lambda+1)$ für $Z \in \{\ddot{A}ZA, \ddot{A}ZA_R, \ddot{A}UZA, \ddot{A}UZA_R\}$ und $\lambda > 1$, ein Nachweis dieser Vermutung ist aber noch nicht gelungen.

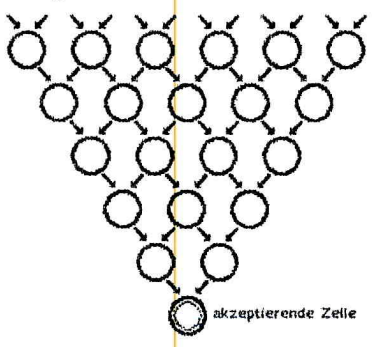
5. Pipeline-Verarbeitung in Zellularautomaten

In [Vo87] wurde ein Zellularautomatenmodell eingeführt, das sich besonders zur Pipeline-Verarbeitung eignet: Der Zellularautomat im "skewed input mode" (SZA). Während in den bisher betrachteten Zellularautomaten mit paralleler Eingabe erst nach O(n) Schritten ein neues Eingabewort eingelesen werden kann, können beim SZA durch serielles, versetztes Einlesen des Eingabewortes in die einzelnen Zellen Pipeline-Perioden kleiner als O(n) erreicht werden.
Im folgenden werden nach einer kurzen Einleitung und einer informellen Beschreibung der Zellularautomaten im "skewed input mode" die zugehörigen Sprachklassen charakterisiert und mit den von änderungsbeschränkten Zellularautomaten und anderen Polyautomaten erkannten Sprachen verglichen. Unter anderem wird bewiesen, daß die SZA die gleichen Sprachklassen erkennen wie ein anderer, bereits mehrfach untersuchter Typ von Polyautomaten, die unidirektionalen iterativen Arrays (aus [CIV88],[IbJ87]), die hier mit UIA abgekürzt werden. Dadurch können einige bekannte Resultate über UIAs auf die Zellularautomaten im "skewed input mode" übertragen werden.

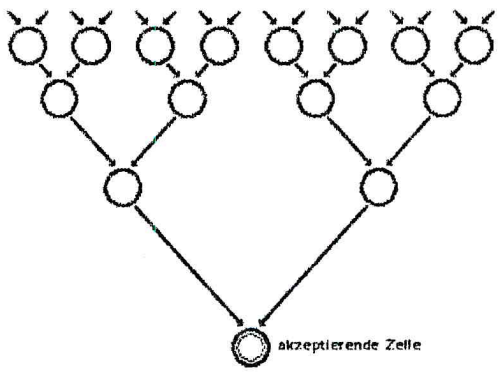
5.1 Systolische Automaten

Ein eindimensionaler Zellularautomat mit H_1- oder $\overline{H}_1$-Raster erkennt bereits in Realzeit viele recht komplizierte Sprachen, für die sequentielle Automaten, zum Beispiel Turing-Maschinen, wesentlich mehr Zeit benötigen. Eine neue parallele Eingabe in den ZA kann aber erst stattfinden, wenn die vorhergehende Berechnung abgeschlossen ist, also (ausgenommen triviale Fälle) frühestens nach O(n) Schritten, abhängig vom verwendeten Raster.
Ein solcher ZA ist daher ungeeignet für Pipeline-Verarbeitung, wenn Pipeline-Perioden kleiner als O(n) angestrebt werden. Für diesen Zweck werden andere, geeignetere Typen von Polyautomaten verwendet, die sich von den ZA in der Struktur der Verbindungen oder in der Art der Eingabe unterscheiden. Besonders interessant wäre dabei das Erreichen einer konstanten Pipeline-Periode, so daß unabhängig von der Länge der Eingabe nach je einer konstanten Anzahl von Schritten eine neues Wort eingegeben und verarbeitet werden kann.
Zwei bereits eingehend untersuchte Modelle für Pipeline-Verarbeitung sind der systolische Trellis [CGS84] und der systolische Baum [CSW84], die beide eine Pipeline-Periode von eins erreichen.

Abb. 5.1 a) Systolischer Trellis



b) Systolischer Baum



Die Funktionsweise dieser beiden Automatenmodelle wird nun mit Hilfe von Abbildung 5.1 kurz erläutert.
Beide Automaten bestehen aus mehreren übereinander liegenden Schichten von Zellen, die während einer Berechnung von oben nach unten durchlaufen werden. Dabei ist jede Zelle einer Schicht mit je zwei Zellen der darüberliegenden Schicht in der in Abbildung 5.1 a) bzw. b) angegebenen Weise verbunden.
Ein Eingabewort w der Länge n wird zum Zeitpunkt 0 parallel in die unterste Schicht mit ausreichender Breite eingelesen, beim systolischen Trellis also in die n-te Schicht von unten. Im Fall des systolischen Baumes wird die Eingabe durch Auffüllen mit einem speziellen Zeichen # auf eine Länge 2^k, $k \in \mathbb{N}$ gebracht und in die k+1-te Schicht von unten eingelesen.
Während der Berechnung ist der Folgezustand einer Zelle nur von den Zuständen der beiden mit ihm verbundenen Zellen aus der darüberliegenden Schicht abhängig, so daß das Ergebnis der Berechnung nach n-1 (systolischer Trellis) bzw. nach k Schritten (systolischer Baum) in der untersten Zelle vorliegt. In beiden Automaten findet also der Datenfluß stets nur in einer Richtung statt; nach jedem Berechnungsschritt kann sofort eine neue Eingabe eingegeben werden.

Für die Trellis-Automaten wurde in [ChC85] bewiesen, daß sie genau die Klasse UZA_R der von unidirektionalen Zellularautomaten in Realzeit akzeptierten Sprachen erkennen. Dazu werden jedoch $\frac{1}{2}n^2 = O(n^2)$ Zellen benötigt, der hohe Durchsatz im Vergleich zu einem unidirektionalen ZA wird also einfach durch eine entsprechende Vervielfachung der Anzahl der benötigten Zellen erreicht.
Die Untersuchung der Zustandsänderungskomplexität legt jedoch nahe, daß es bei vielen Sprachen möglich sein sollte, bereits mit n Zellen eine Pipeline-Periode von weniger als O(n) Takten zu erreichen. So ändert beispielsweise bei der Erkennung einer Sprache in einem O(ln n)-max-zustandsänderungsbeschränkten Zellularautomaten jede Zelle im Verlauf einer Berechnung höchstens an O(ln n) Zeitpunkten ihren Zustand. Während des überwiegenden Teils der Berechnung bleiben die Zellen also untätig. Gelänge es nun, einen Automatentyp zu finden, der diese Sprachen "effizienter" erkennt, so sollte man damit auch ohne eine entsprechende Erhöhung der Anzahl der Zellen eine kleinere Pipeline-Periode als O(n) erreichen können.
Ein Beispiel für einen Automaten, der schon mit O(n) Zellen eine Pipeline-Periode von eins erreicht, ist der systolische Baum. Die von ihm erkannten Sprachen sind jedoch stark durch die dem Automaten zugrundeliegende Binärbaumstruktur beschränkt. Neben den regulären Sprachen wird beispielsweise die Sprache $\{a^{2^n} \mid n \in \mathbb{N}\}$ aus $ZA_R \setminus UZA_R$ erkannt, andere, recht einfache Sprachen wie $\{a^n b^n \mid n \in \mathbb{N}\}$ jedoch nicht [CSW84].

Daher wird im folgenden ein weiterer Automatentyp vorgestellt, der auf dem eindimensionalen Zellularautomaten mit H_1- bzw. $\overline{H}_1$-Raster beruht. Dieser Automat unterscheidet sich von den bisher betrachteten Zellularautomaten und den eben vorgestellten systolischen Automaten durch ein sequentielles Einlesen der Eingabe.
Zunächst werden nur Automaten betrachtet, die aus einer einzigen Schicht von Zellen bestehen. Damit können für viele Sprachen bereits mit O(n) Zellen Pipeline-Perioden beispielsweise von O(ln n) oder $O(n^{\frac{1}{2}})$ erreicht werden. Durch die Verwendung mehrerer Schichten läßt sich dann mit weniger als $O(n^2)$ Zellen eine Pipeline-Periode von eins erreichen.

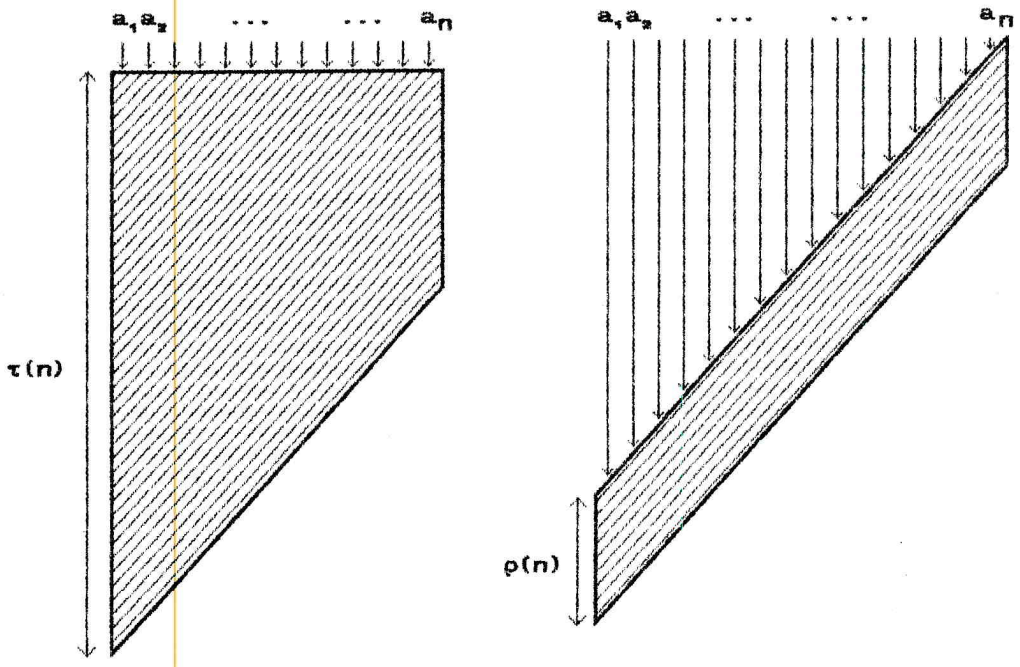
5.2 Zellularautomaten im "skewed input mode"

Die in diesem Kapitel betrachteten Zellularautomaten im "skewed input mode" (SZA) sind eindimensionale Zellularautomaten mit H_1- oder $\overline{H}_1$-Raster, bei denen die Eingabe nicht parallel zu Beginn, sondern nacheinander während der Berechnung in die verschiedenen Zellen eingelesen wird:

Bei einem Eingabewort $w = a_1 a_2 \ldots a_n$ wird das Zeichen a_n zum Zeitpunkt 0 in die letzte Zelle der Retina eingegeben. Im nächsten Schritt wird dann a_{n-1} in die $(n-1)$-te Zelle eingelesen, das Zeichen a_ν erscheint entsprechend zum Zeitpunkt $n-\nu$ in der ν-ten Zelle. Nach n Schritten wird die serielle Eingabe durch das Einlesen des ersten Zeichens a_1 in die Zelle am linken Rand der Retina beendet.

Ein Eingabewort wird vom SZA akzeptiert, wenn die Zelle am linken Rand der Retina den Zustand α annimmt. Wird dagegen der Zustand ω angenommen, so wird die Eingabe nicht akzeptiert. Ein SZA, in dem die akzeptierende Zelle für alle Eingaben $w \in V^+$ innnerhalb von $\rho(|w|)$ Schritten nach dem Ende des Einlesens einen Zustand aus $\{\alpha, \omega\}$ annimmt, wird als SZA mit Pipeline-Periode $\rho(n)$ oder einfach als ρ-SZA bezeichnet. Die Menge der von ρ-SZA erkannten Sprachen wird $SZA(\rho)$ geschrieben.

Abb. 5.2 Raum-Zeit-Diagramme der aktiven Zellen in einem τ-zeitbeschränkten ZA und in einem SZA mit Pipeline-Periode $\rho(n)$

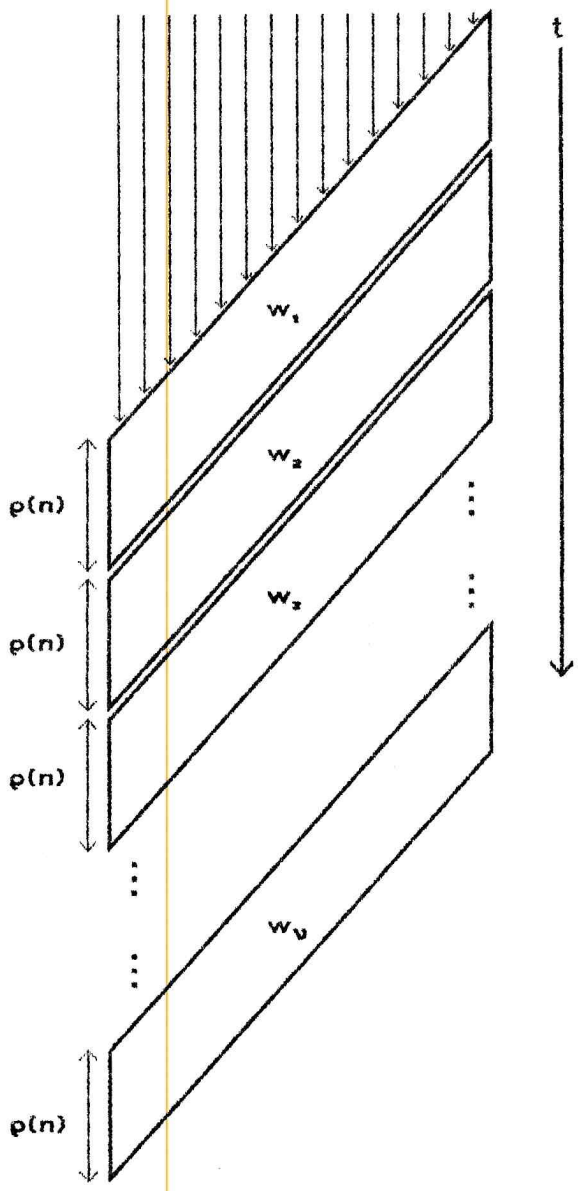


In einem ρ-SZA ist nur der in Abbildung 5.2 schraffierte Teil des Raum-Zeit-Diagrammes für das Ergebnis der Berechnung von Bedeutung. Es ist also keine Einschränkung, wenn man davon ausgeht, daß jede Zelle in einem ρ-SZA während der Verarbeitung eines Eingabewortes der Länge n nur maximal $\rho(n)$ Takte aktiv ist. Daher kann bereits $\rho(n)$ Schritte nach dem Einlesen des letzten Zeichens in die Zelle am rechten Rand

der Retina mit der Eingabe eines neuen Wortes begonnen werden. In dieser Weise ist in einem ρ-SZA Pipeline-Verarbeitung mit einer Periode von ρ(n) möglich.

Das folgende Raum-Zeit-Diagramm zeigt die Staffelung der verschiedenen Berechnungsvorgänge bei Eingabe von Wörtern $w_1, w_2, \ldots, w_\nu$ der Länge n. Während im rechten Bereich der Retina bereits ein neues Wort w_μ eingelesen wird, können in den anderen Bereichen noch die vorher eingegebenen Wörter $w_{\mu-1}, w_{\mu-2}, \ldots$ verarbeitet werden. Je kleiner die Pipeline-Periode ist, desto mehr Eingabewörter können gleichzeitig gestaffelt verarbeitet werden.

Abb. 5.3 Raum-Zeit-Diagramm der Pipeline-Verarbeitung in einem ρ-SZA



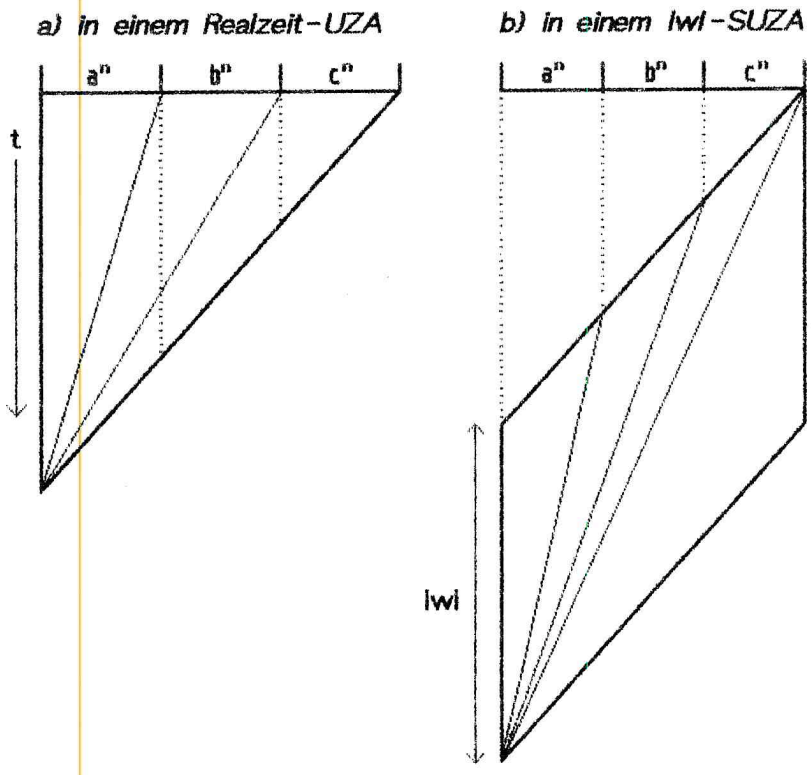
SZA mit $\bar{H}_1$-Raster, in denen also nur Datenfluß von rechts nach links möglich ist, werden im folgenden als SUZA bezeichnet. Entsprechend steht ρ-SUZA für einen SUZA mit Pipeline-Periode ρ(n) und SUZA(ρ) für die Menge der von ρ-SUZA erkannten Sprachen.

In den folgenden Abschnitten werden die von SZA und SUZA mit verschiedenen Pipeline-Perioden erkannten Sprachklassen untersucht. Dabei werden in erster Linie die Sprachen der unidirektionalen SZA betrachtet. Es ist noch nicht bekannt, ob mit bidirektionalen SZA überhaupt mehr Sprachen als mit SUZA erkannt werden können. Es läßt sich allerdings zeigen, daß dieses Problem für Pipeline-Perioden größer oder gleich $O(\ln n)$ eng zusammenhängt mit der Frage, ob die von Zellularautomaten in Realzeit und in Linearzeit erkannten Sprachen gleich sind (siehe Abschnitt 5.6).

Ein weiteres Beispiel für Zellularautomaten, mit denen Pipeline-Perioden von weniger als $O(n)$ erreicht werden, sind die Zellularautomaten mit parallelem Annahmeverhalten (aus [SoW83], [IPK85]). In diesen Automaten wird eine Eingabe akzeptiert, wenn alle Zellen der Retina einen Zustand α annehmen. Dadurch werden Annahmezeiten (und damit natürlich auch Pipeline-Perioden) von weniger als Realzeit möglich. Es läßt sich recht einfach zeigen, daß die Menge der von einem solchen Automaten in der Zeit $\tau(n) < n$ erkannten Sprachen echt in der Klasse $SZA(\tau)$ enthalten ist.

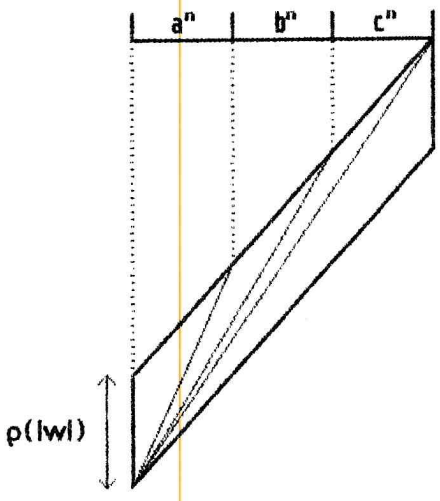
Beispiel 5.1 [Vo87] Die Funktionsweise der Zellularautomaten im "skewed input mode" wird am Beispiel der Sprache $\{a^n b^n c^n \mid n \in \mathbb{N}\}$ demonstriert. In einem UZA wird diese Sprache mit Hilfe von Signalen der Geschwindigkeit $\frac{1}{3}$, $\frac{2}{3}$ und 1 erkannt, die zum Zeitpunkt 0 nach links geschickt werden (Abbildung 5.4 a)). In einem SUZA werden diese Signale unmittelbar nach der Eingabe der entsprechenden Zeichen gestartet, sie haben beim einfachen Fall einer Pipeline-Periode $|w|$ die Geschwindigkeiten $\frac{1}{4}$, $\frac{2}{5}$ und $\frac{1}{2}$ (Abbildung 5.4.b)).

Abb. 5.4 Erkennung von $\{a^n b^n c^n \mid n \in \mathbb{N}\}$



Die Pipeline-Periode des SUZA kann jedoch mit einfachen Mitteln um beliebige Faktoren verkleinert werden, indem die Geschwindigkeiten der Signale erhöht werden (Abbildung 5.5). Dabei wird allerdings die Zustandsmenge A der Zellen in der Regel entsprechend vergrößert.

Abb. 5.5 Erkennung von $\{a^n b^n c^n \mid n \in \mathbb{N}\}$ in einem UZA mit Pipeline-Periode $\rho = O(|w|)$, $\rho(|w|) < |w|$



Will man jedoch die Pipeline-Periode wesentlich veringern, also beispielsweise von $O(n)$ auf $O(n^{\frac{1}{2}})$, so muß ein anderer, "besserer" Algorithmus verwendet werden. Für die in diesem Beispiel betrachtete Sprache läßt sich mit dem folgenden Erkennungsverfahren eine Pipeline-Periode von $O(\ln n)$ erreichen.

Vom rechten Rand der Retina werden drei Zähler nach links geschickt, die die a's , b's und c's im Eingabewort zählen. Die niederwertigste Stelle jedes Zählers läuft mit Einheitsgeschwindigkeit nach links und zieht die restlichen Stellen hinter sich her. Ein Überlauf in einer Stelle wird also einfach auf die nachfolgende, nächsthöhere Stelle aufaddiert. Die Zähler können dabei maximal eine Länge von $O(\ln|w|)$ erreichen. Die niederwertigste Stelle jedes Zählers erreicht die akzeptierende Zelle unmittelbar nach dem Einlesen des ersten Zeichens in diese Zelle, zum Vergleich der Zählerstände werden dann noch einmal $O(\ln|w|)$ Takte benötigt. Ein weiteres, vom rechten Rand mit Geschwindigkeit 1 nach links laufendes Signal prüft die Eingabe auf die Form $a^{n_1}b^{n_2}c^{n_3}$. Wenn dieses Signal ein positives Ergebnis liefert und die drei Zählerstände übereinstimmen, so wird die Eingabe akzeptiert.

Auch für dieses Verfahren kann die Pipeline-Periode noch um einen beliebigen Faktor verringert werden, indem die Basis des Zählers erhöht wird, dies führt natürlich zu einer Vergrößerung der Zustandsmenge A.