## 5.3 Charakterisierung der zugehörigen Sprachklassen

In diesem Abschnitt werden die von ρ-SZA und ρ-SUZA mit verschiedenen Pipeline-Perioden ρ erkannten Sprachklassen untersucht. Besonderes Interesse gilt dabei den Fällen $\rho = O(\ln n)$ und $\rho = O(n^{\frac{1}{2}})$, die zugehörigen Sprachklassen SZA(ρ) und SUZA(ρ) werden mit den entsprechenden Klassen der zustandsänderungsbeschränkten ZA und UZA verglichen.

**Satz 5.1** **Für $g: \mathbb{N} \longrightarrow \mathbb{N}$ gilt:**

**a) SZA(g) $\subseteq$ ÄZA(g)**

**b) SUZA(g) $\subseteq$ ÄUZA(g)**

Beweis: In einem g-SZA ist jede Zelle während der Verarbeitung einer Eingabe höchstens g(n) Schritte aktiv. Daher kann jede Zelle auch nicht mehr als g(n) Zustandsänderungen durchführen.

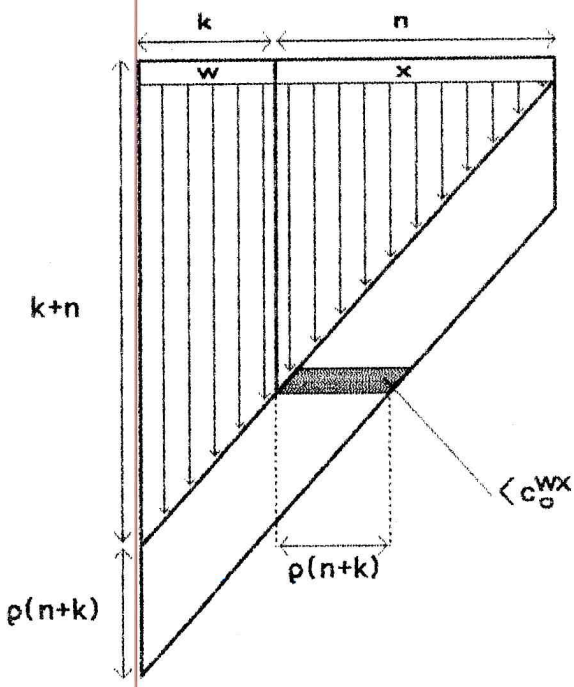
◇

In Satz 3.3 wurde für die ρ-änderungsbeschränkten Sprachen eine obere Schranke für die Anzahl der Äquivalenzklassen bezüglich der Relation $_k\sim$ bewiesen. Diese Schranke gilt aufgrund von Satz 5.1 natürlich auch für die entsprechenden Klassen SUZA(ρ). Es ist aber möglich, sowohl für SUZA als auch für SZA eine wesentlich bessere Schranke zu zeigen:

**Satz 5.2** **Es sei $L \subseteq V^*$ und $\rho: \mathbb{N} \longrightarrow \mathbb{N}$ mit $\rho(n) \leq n$. Wird L von einem ρ-SZA mit Zustandsmenge A erkannt, so gilt für alle $k, n \in \mathbb{N}$:**

$$Äq\left(\overline{V}^n, {}_k\sim \pmod L\right) \leq |A|^{\rho(n+k)+1}.$$

*Abb. 5.6 Die in Satz 5.2 betrachtete Situation*




**Das Ergebnis der Berechnung hängt nur vom Teilwort w und dem eingezeichneten Verhaltensfolgeausschnitt ab; dieser ist wiederum nur vom Teilwort x abhängig.**

Beweis: In einem SZA $\mathfrak{A}$ mit Pipeline-Periode $\rho$ hängt das Ergebnis der Berechnung bei Eingabe eines Wortes $wx$ mit $w \in \overline{V^k}$ und $x \in \overline{V^n}$ nur vom Teilwort $w$ und dem Verhaltensfolgeausschnitt $\langle c_o^{wx}(n+1, n+\rho(n+k)+1)\rangle$ ab (siehe Abb. 5.6). Dieser Verhaltensfolgeausschnitt ist wiederum eindeutig durch das Teilwort $x$ bestimmt.

Sind die zu zwei Wörtern $x$ und $y$ aus $\overline{V^n}$ gehörenden Verhaltensfolgeausschnitte $\langle c_o(n+1, n+\rho(n+k)+1)\rangle$ gleich, so folgt demnach für alle $w \in \overline{V^k}$

$$wx \in L_{\mathfrak{A}} \iff wy \in L_{\mathfrak{A}}.$$

Nach Definition 3.4 gilt dann $x \; {}_k\!\sim y \pmod{L_{\mathfrak{A}}}$.

Also ist die Anzahl der Äquivalenzklassen in $\overline{V^n}$ beschränkt durch die Anzahl der verschiedenen Verhaltensfolgeausschnitte $\langle c_o(n+1, n+\rho(n+k)+1)\rangle$ und damit durch $|A|^{\rho(n+k)+1}$.

◇

Mit einer konstanten Pipeline-Periode lassen sich in SZA, im Gegensatz zu den ZA mit konstanter Zustandsänderungsbeschränkung, nur reguläre Sprachen erkennen.

**Satz 5.3** $\mathbf{SZA(O(1)) = SUZA(O(1)) = \mathfrak{L}_3}$

Beweis: Zu zeigen ist:

a) $SZA(O(1)) \subseteq \mathfrak{L}_3$.

Wird L von einem s-SZA mit $s \in \mathbb{N}$ erkannt, so erhält man durch Einsetzen in Satz 5.2

$$Äq\left(\overline{V^n}, {}_k\!\sim \pmod{L}\right) \leq |A|^{s+1}.$$

Nach Definition der Relationen ${}_k\!\sim$ und $\sim_k$ in Abschnitt 3.4 ist dies äquivalent zu

$$Äq\left(\overline{V^n}, \sim_k \pmod{(L)^R}\right) \leq |A|^{s+1}.$$

Da dieser Ausdruck unabhängig von den Größen n und k ist, folgt nach Definition 3.3 b)

$$Äq\left(V^+, \sim \pmod{(L)^R}\right) \leq |A|^{s+1}.$$

Die Sprache $(L)^R$ ist also von endlichem Index und damit regulär. Wegen der Abgeschlossenheit von $\mathfrak{L}_3$ bezüglich Spiegelung ist auch L regulär.

b) $\mathfrak{L}_3 \subseteq SUZA(O(1))$

Die regulären Sprachen können von einem O(1)-SUZA mit dem für änderungsbeschränkte ZA in Abschnitt 2.2 beschriebenen Verfahren erkannt werden.

◇

Mit Hilfe von Satz 5.2 lassen sich weitere Resultate zum Verhältnis von SZA und zustandsänderungsbeschränkten ZA beweisen:

**Satz 5.4** Für $g \in \{n^{\frac{1}{2}}, \ln n, 1\}$ gelten

a) $SZA(O(g)) \subset \ddot{A}ZA(O(g))$

b) $SUZA(O(g)) \subset \ddot{A}UZA(O(g))$

Beweis: Der Beweis wird für $g = n^{\frac{1}{2}}$ anhand der Sprache $L_{RQ}$ aus Definition 3.1 geführt. Der Beweis für $g = \ln n$ verläuft analog mit Hilfe von $L_{RD}$, der Fall $g = 1$ folgt direkt aus Satz 5.3.

Nach Lemma 3.7 ist $L_{RQ}$ in $\ddot{A}UZA_R(O(n^{\frac{1}{2}}))$ und damit auch in $\ddot{A}UZA(O(n^{\frac{1}{2}}))$ und $\ddot{A}ZA(O(n^{\frac{1}{2}}))$. Es läßt sich aber mit Satz 5.2 zeigen, daß $L_{RQ}$ nicht in $SZA(O(n^{\frac{1}{2}}))$ und damit natürlich auch nicht in $SUZA(O(n^{\frac{1}{2}}))$ ist.

Zu jedem Wort v der Länge $k^2$ mit k Zeichen aus $\{a,b\}$ gibt es genau ein Wort $w = v^R YYQ_{\#}(\$^k)ZZ\$^k\#^t$ mit $vXXw \in L_{RQ}$. Sind $v_1XXw_1$ und $v_2XXw_2$ zwei verschiedene Wörter der Länge $n = 6k^2+2k+12$ in $L_{RQ}$, so würde aus

$$XXw_1 \ {}_{(k^2)}\!\sim XXw_2$$

folgen, daß auch $v_1XXw_2$ und $v_2XXw_1$ in $L_{RQ}$ sind. Also sind $XXw_1$ und $XXw_2$ in verschiedenen Äquivalenzklassen bezüglich ${}_{(k^2)}\!\sim (\mathrm{mod}\ L_{RQ})$.
Es gibt genau

$$\binom{k^2}{k} 2^k$$

Wörter v der Länge $k^2$ mit k Zeichen aus $\{a,b\}$, dies ist folglich auch die Anzahl der Wörter der Form $XXv^R YYQ_{\#}(\$^k)ZZ\$^k\#^t$ mit der Länge $m = 5k^2+2k+12$. Es folgt

$$\ddot{A}q\left(\overline{V^m}, {}_{(k^2)}\!\sim (\mathrm{mod}\ L_{RQ})\right) \geq \binom{k^2}{k} 2^k$$

$$\geq \left(k^2\right)^{\left(\frac{k}{2}\right)} \cdot 2^k \geq 2^{k \cdot \mathrm{ld}\, k} \quad \text{nach Lemma 3.5.}$$

Würde $L_{RQ}$ von einem $sn^{\frac{1}{2}}$-SZA mit $s \in \mathbb{R}$ erkannt werden, so würde nach Satz 5.2 gelten :

$$\ddot{A}q\left(\overline{V^m}, {}_{(k^2)}\!\sim (\mathrm{mod}\ L_{RQ})\right) \leq |A|^{s \cdot (m+k^2)^{\frac{1}{2}}+1}$$

$$\leq 2^{(7k^2)^{\frac{1}{2}} \cdot s \cdot \mathrm{ld}\,|A| + \mathrm{ld}\,|A|} \quad \text{wegen } 6k^2 > m \text{ für große } k$$

$$= 2^{c \cdot k + \mathrm{ld}\,|A|} \quad \text{mit } c = 7^{\frac{1}{2}} \cdot s \cdot \mathrm{ld}\,|A|$$

$$< 2^{k \cdot \mathrm{ld}\, k} \quad \text{für große } k.$$

Also kann $L_{RQ}$ von keinem $sn^{\frac{1}{2}}$-SZA erkannt werden.

◇

Vergleicht man die SUZA mit den Klassen der von änderungsbeschränkten UZA in Realzeit erkannten Sprachen, so erhält man das folgende Ergebnis :

**Satz 5.5** a) $SUZA(O(1)) \subset \ddot{A}UZA_R(O(1))$

b) $SUZA(O(\ln n)) \not\supseteq \ddot{A}UZA_R(O(\ln n))$

c) $SUZA(O(n^{\frac{1}{2}})) \not\supseteq \ddot{A}UZA_R(O(n^{\frac{1}{2}}))$

Beweis: a) folgt direkt aus Satz 5.3.

b) $L_{RD}$ ist in $\ddot{A}UZA_R(O(\ln n))$, aber nach dem Beweis von Satz 5.4 nicht in $SUZA(O(\ln n))$. Andererseits kann $L_p := \{a^{2^n} \mid n \in \mathbb{N}\}$ von keinem UZA in Realzeit erkannt werden und ist damit nicht in $\ddot{A}UZA_R(O(\ln n))$. Einen Algorithmus für $L_p$ in einem $O(\ln n)$-SUZA erhält man unter Benutzung eines Binärzählers, der in der in Beispiel 5.1 beschriebenen Weise von rechts nach links durch den Automaten läuft. Wenn der Zähler am linken Rand der Retina angekommen ist, wird geprüft, ob alle Stellen außer der ersten Stelle auf null stehen.

c) folgt analog zu b) anhand von $L_{RQ}$ und $L_p$.

◇

**Satz 5.6** **Die Klassen $SZA(O(g))$ und $SUZA(O(g))$ sind für $g \in \{n^{\frac{1}{2}}, \ln n\}$ nicht abgeschlossen bezüglich Spiegelbild.**

Beweis: Die Sprache $L_B$ aus Abschnitt 3.5 ist von einem $O(\ln n)$-SUZA erkennbar. Das Spiegelbild $(L_B)^R$ ist aber von keinem $O(n^{1-\varepsilon})$-SZA erkennbar. Dies folgt aus Teil b) des Beweises von Satz 3.4, da die in Satz 5.2 aufgestellte Schranke für die Anzahl der Äquivalenzklassen wesentlich kleiner ist als die dort verwendete Schranke aus Satz 3.3.

◇

**Satz 5.7** a) $SZA(O(n)) \supset SZA(O(n^{\frac{1}{2}})) \supset SZA(O(\ln n)) \supset SZA(O(1))$

b) $SUZA(O(n)) \supset SUZA(O(n^{\frac{1}{2}})) \supset SUZA(O(\ln n)) \supset SUZA(O(1))$

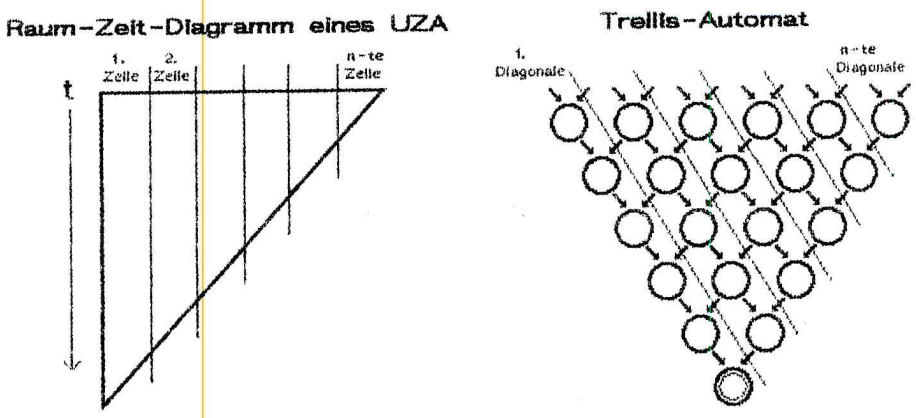
Beweis: Nach Satz 5.3 enthalten die Klassen $SZA(O(1))$ und $SUZA(O(1))$ nur die regulären Sprachen. Die nichtreguläre Sprache $L_p := \{a^{2^n} \mid n \in \mathbb{N}\}$ wird nach dem Beweis von Satz 6.5 von einem $O(\ln n)$-SUZA erkannt und ist daher in den Klassen $SUZA(O(\ln n))$ und $SZA(O(\ln n))$. Mit Hilfe von Satz 6.2 läßt sich zeigen, daß für $L_{QQ} := \{(Q_\#(u))^R X X Q_\#(u) \mid u \in \{a,b\}^+\}$ eine Pipeline-Periode von $O(n^{\frac{1}{2}})$ und für $L_R = \{uYYu^R \mid u \in \{a,b\}^+\}$ eine Pipeline-Periode von $O(n)$ benötigt wird. Algorithmen, die diese beiden Sprachen in der entsprechenden Pipeline-Periode erkennen, lassen sich einfach finden. ◇

## 5.4 Zeit-Raum-Transformationen

In den vorangegangenen Abschnitten wurde bewiesen, daß es mit SZA möglich ist, für viele Sprachen bereits mit O(n) Zellen eine Pipeline-Periode kleiner als O(n), also zum Beispiel von $O(n^{\frac{1}{2}})$ oder O(ln n), zu erreichen. In diesem Abschnitt wird nun demonstriert, wie man unter Ausnutzung dieser Ergebnisse mit möglichst wenigen Zellen eine Pipeline-Periode von eins erreicht.

Wie bereits in Abschnitt 5.1 erwähnt wurde, ergibt sich der hohe Durchsatz eines Trellis-Automaten gegenüber einem Realzeit-UZA bei gleicher Erkennungsmächtigkeit durch eine entsprechende Vervielfachung der Zellenzahl.
Jede Zelle des UZA wird im Trellis-Automaten durch eine ganze Diagonale von Zellen simuliert. Dabei wird die akzeptierende Zelle des UZA, die ja während der gesamten Berechnung aktiv ist, durch die 1. Diagonale mit insgesamt n Zellen simuliert. Die letzte Zelle eines Realzeit-UZA ist bei einer Berechnung nur im ersten Schritt aktiv und kann daher im Trellis-Automaten von der letzten (n-ten) Diagonale, die nur aus einer Zelle besteht, simuliert werden.

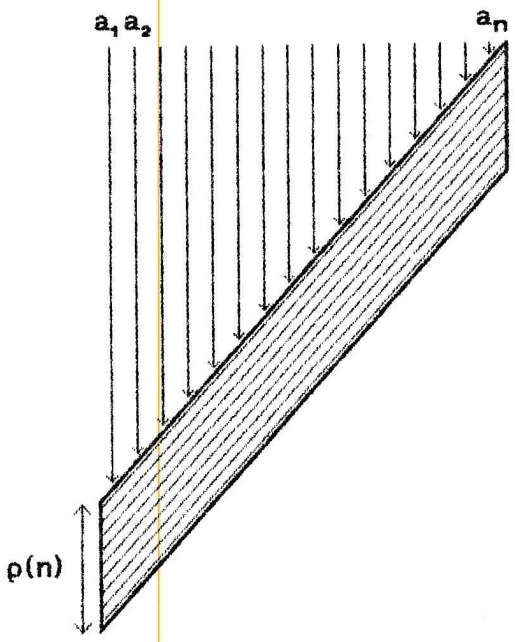
*Abb. 5.7 Zeit-Raum-Transformation am Beispiel eines Realzeit-UZA*




Diese Technik, durch eine entsprechende Vergrößerung der Zellenzahl den Durchsatz von zellularen und iterativen Automaten zu erhöhen, wird in [Gr84] als "Zeit-Raum-Transformation" (time-space- transformation) bezeichnet. Die Grundidee dieser Transformation besteht nämlich darin, für jeden Schritt einer im Automaten stattfindenden Berechnung eine eigene Schicht von Zellen vorzuhalten. Anschaulich betrachtet wird also das Raum-Zeit-Diagramm eines zellularen oder iterativen Automaten mit zusätzlichen Zellen "ausgefüllt".
Im Beispiel des Trellis wird jede Zelle des UZA durch eine Diagonale von Zellen des Trellis simuliert, man spricht deshalb in diesem Fall auch von "d-(diagonal)unrolling". Wird jede Zelle in der Zeit-Raum-Transformation durch eine Spalte von Zellen ersetzt, so wird von "c-(column)unrolling" gesprochen.

Diese Technik der Raum-Zeit-Transformation kann auch für SUZA verwendet werden, um eine Pipeline-Periode von O(1) zu erreichen. In einem ρ-SUZA ist zu jedem Zeitpunkt der Berechnung nur ein zusammenhängender Block von ρ(n) Zellen aktiv. Daher kann man durch Verwendung von n Schichten aus je ρ(n) Zellen, die im Verlauf einer Berechnung nacheinander aktiviert werden, auf eine Pipeline-Periode von eins kommen.

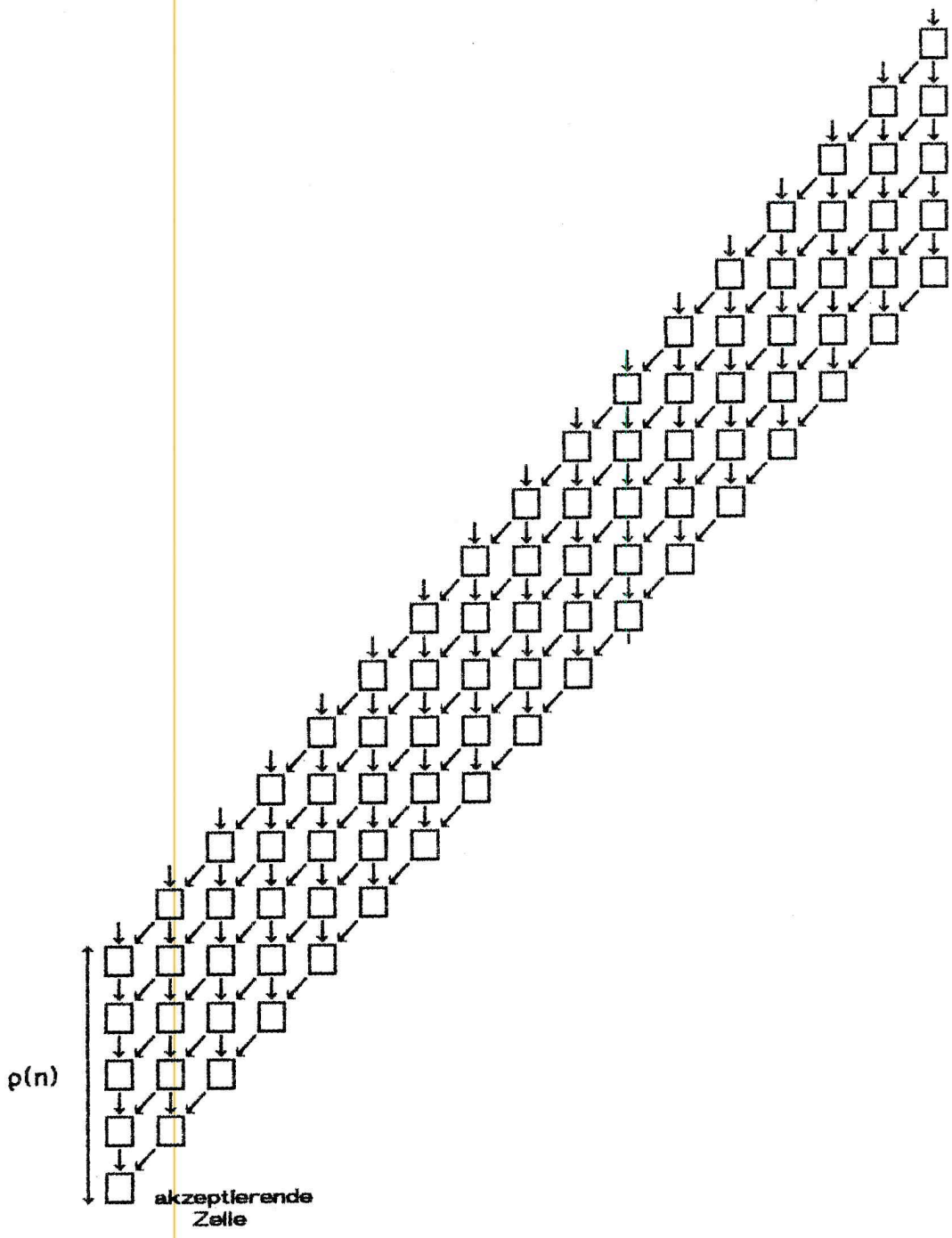
*Abb. 5.8 Raum-Zeit-Diagramm eines ρ-SUZA*




Ersetzt man nun durch "c-unrolling" jede Zelle des ρ-SUZA durch eine Spalte von ρ(n) Zellen, so erhält man den in Abbildung 5.9 dargestellten systolischen Automaten. Die Eingabe erfolgt in diesem Automaten genauso wie beim SZA durch serielles Einlesen der Zeichen in verschiedene Zellen. Im Gegensatz zum ρ-SUZA kann jedoch in jedem Schritt mit dem Einlesen eines neuen Eingabewortes begonnen werden, in jedem Takt werden also n Zeichen aus n verschiedenen Eingabewörtern eingelesen.

Ein durch eine Zeit-Raum-Transformation erhaltener systolischer Automat unterscheidet sich von dem ursprünglich vorliegenden zellularen oder iterativen Automaten nur in der Pipeline-Periode, nicht jedoch in der Menge der erkannten Sprachen. Erhält man also aus zwei verschiedenen Automaten durch Zeit-Raum-Transformationen gleiche oder äquivalente systolische Automaten, so erkennen auch die ursprünglichen Automaten die gleiche Sprachklasse. Diese Eigenschaft wird in Abschnitt 5.6 benutzt, um die Äquivalenz der SUZA mit den im nächsten Abschnitt eingeführten unidirektionalen iterativen Arrays auf anschauliche Weise zu zeigen.

Abb. 5.9 Systolischer Automat mit $\rho(n) \cdot n$ Zellen und Pipeline-Periode $O(1)$ für Sprachen aus $SUZA(O(\rho))$



## 5.5 Unidirektionale iterative Arrays

In diesem Abschnitt wird ein weiteres Polyautomaten-Modell, das unidirektionale iterative Array ("one-way iterative array", aus [CIV88], [IbJ87]), vorgestellt und einige bereits bekannte Ergebnisse aufgeführt.

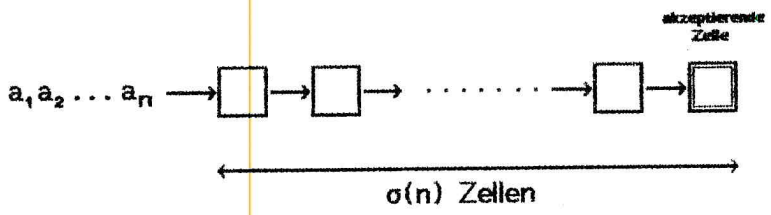
Ein unidirektionales iteratives Array (UIA) besteht aus einer einseitig unendlichen Kette von identischen endlichen Automaten, die ihren Zustand in jedem Berechnungsschritt in Abhängigkeit von ihrem eigenen Zustand und dem ihres linken Nachbarn verändern.

Zum Zeitpunkt 0 befinden sich alle Zellen des UIA in einem speziellen Ruhezustand, den sie nur verlassen können, wenn im vorhergegangenen Schritt ihr linker Nachbar seinen Zustand ändert. Die Eingabe wird nun seriell in die Zelle am linken Rand des UIA eingegeben, dabei wird im Gegensatz zu den in den zitierten Arbeiten betrachteten UIA das letzte Zeichen des Eingabewortes zuerst eingegeben, das ganze Eingabewort also vor der Eingabe umgedreht. Dadurch wird der Vergleich mit den SZA vereinfacht, bei denen ja auch die hinteren Zeichen zuerst eingelesen werden. Das Ende des Eingabevorganges wird dem UIA durch die Eingabe eines speziellen Zeichens $\varepsilon$ angezeigt.

Im Gegensatz zu den bisher betrachteten Automaten hängt es beim UIA von der Länge der Eingabe ab, welches die akzeptierende Zelle ist. Ein unidirektionales iteratives Array heißt $\sigma$-UIA mit $\sigma : \mathbb{N} \longrightarrow \mathbb{N}$, wenn bei einer Eingabe der Länge n die $\sigma(n)$-te Zelle von links über Annahme oder Nichtannahme des Eingabewortes entscheidet.

Die Funktion eines $\sigma$-UIA wird in der folgenden Abbildung noch einmal veranschaulicht.

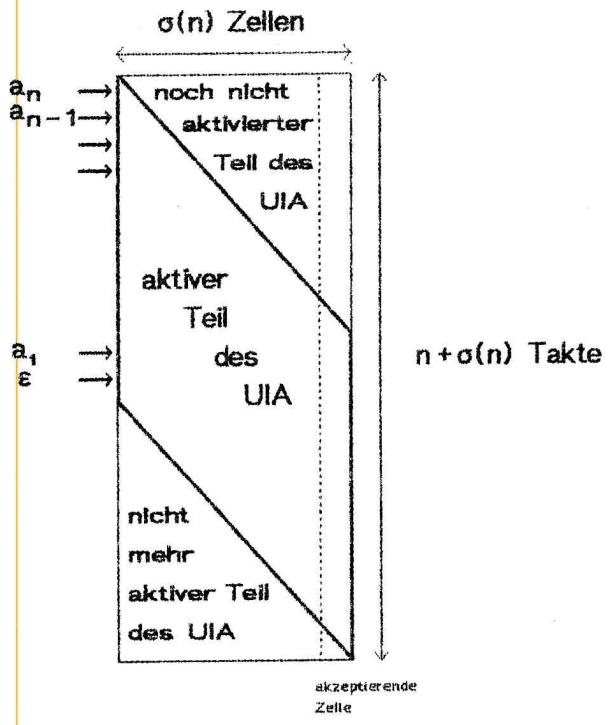
*Abb. 5.10 Unidirektionales iteratives Array mit $\sigma(n)$ Zellen und Eingabe $a_1 a_2 \ldots a_n$*




Die kleinste Zeit, in der sinnvolle Sprachen von einem $\sigma$-UIA erkannt werden können, beträgt $n+\sigma(n)$ Schritte. Diese Zeit wird deshalb auch Pseudo-Realzeit genannt. Die Menge aller von $\sigma$-UIA erkannten Sprachen wird mit $UIA(\sigma)$, die Menge aller von $\sigma$-UIA in Pseudo-Realzeit erkannten Sprachen mit $UIA_R(\sigma)$ bezeichnet.

Das folgende Raum-Zeit-Diagramm zeigt die zu verschiedenen Zeiten aktiven Zellen in einem auf Pseudo-Realzeit beschränkten $\sigma$-UIA.

*Abb. 5.11 Raum-Zeit-Diagramm eines σ-UIA bei Eingabe von $a_1 a_2 \ldots a_n$*




In den folgenden Sätzen werden einige bekannte Ergebnisse über UIA aufgeführt, entsprechende Beweise finden sich in den angegebenen Quellen.

Zu beachten ist, daß in den in diesem Abschnitt eingeführten UIA, im Gegensatz zu den UIA in [IbJ87] und [CIV88], das Eingabewort spiegelverkehrt eingelesen wird. Dies ändert aber nichts an der Form von Satz 5.8, da sich andererseits die akzeptierende Zelle des hier betrachteten ZA am anderen Ende der Retina als in [IbJ87] befindet.

**Satz 5.8** $UIA_R(O(n)) = UZA_{2n} = ZA_R$. [IbJ87]

Auch der nächste Satz kann aufgrund der Abgeschlossenheit von $\mathfrak{L}_3$ bezüglich Spiegelung ohne Änderung übernommen werden: Es können keine nichtregulären Sprachen von UIA mit weniger als O(ln n) Zellen erkannt werden.

**Satz 5.9** Es sei $\sigma : \mathbb{N} \longrightarrow \mathbb{N}$ mit $\sigma < O(\ln n)$. Dann gilt $UIA(O(\sigma)) = \mathfrak{L}_3$. [CIV88]

## 5.6 Vergleich von SUZA und UIA

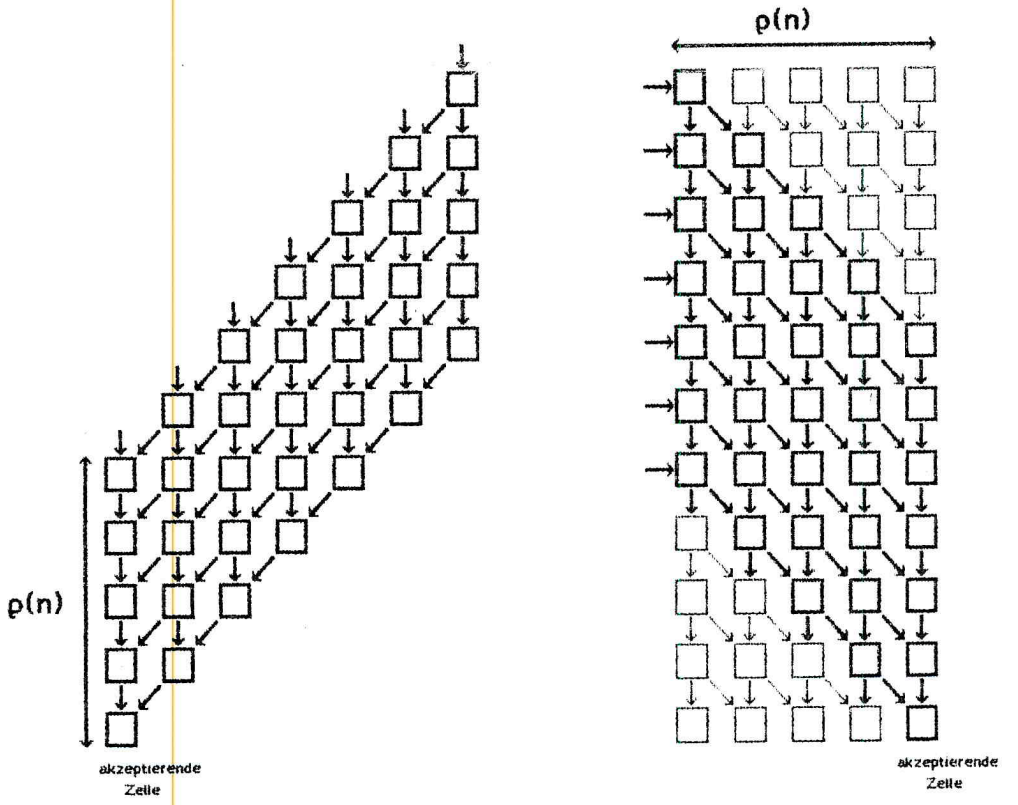
Mit Hilfe der Zeit-Raum-Transformation aus Abschnitt 5.4 wird in diesem Abschnitt bewiesen, daß die von ρ-UIA mit Pseudo-Realzeit-Beschränkung und die von ρ-SUZA erkannten Sprachklassen gleich sind. Dadurch können die im letzten Abschnitt zitierten Ergebnisse über UIA auf SUZA angewendet werden.

**Satz 5.10** Es sei $\sigma: \mathbb{N} \longrightarrow \mathbb{N}$ mit $\sigma(n) \leq n$. Dann gilt $SUZA(\sigma) = UIA_R(\sigma)$.

Beweis: Die Konstruktion des in Abb. 5.12 a) skizzierten systolischen Automaten aus einem SUZA durch eine Zeit-Raum-Transformation wurde bereits in Abschnitt 5.4 beschrieben.
Mit derselben Transformation erhält man den Automaten in Abbildung 5.12 b) aus dem in Abbildung 5.11 gezeigten Raum-Zeit-Diagramm eines ρ-UIA. Die dünn eingezeichneten Zellen stehen dabei für die nicht aktiven Bereiche des Raum-Zeit-Diagrammes des UIA, diese Zellen können daher auch weggelassen werden und werden beim Vergleich der beiden Automaten nicht beachtet. Man erkennt unschwer, daß die beiden in Abbildung 5.12 abgebildeten Automaten äquivalent sind, indem man die Zeilen des linken Automaten unter Beibehaltung der Verbindungen so gegeneinander verschiebt, daß sich der rechte Automat ergibt.

◇

*Abb. 5.12* *a) "c-unrolling" eines ρ-SUZA* *b) "c-unrolling" eines ρ-UIA*



Aus den Sätzen 5.8, 5.9 und 5.10 folgen nun unmittelbar die folgenden Ergebnisse:

**Satz 5.11** Es sei $\rho : \mathbb{N} \longrightarrow \mathbb{N}$ mit $\rho < O(\ln n)$. Dann gilt $SUZA(O(\rho)) = \mathfrak{L}_3$.

Ein SUZA erkennt also mit Pipeline-Perioden kleiner als $O(\ln n)$ nur noch die regulären Sprachen, eine Hierarchie der Komplexitäten $\ln_{(k)} n$, $k \in \mathbb{N}$ wie bei den änderungsbeschränkten ZA existiert damit für SUZA nicht.

**Satz 5.12** $SUZA(n) = UIA_R(n) = UZA_{2n} = ZA_R = \ddot{A}ZA_R(n)$.

Ein SUZA erkennt also mit Pipeline-Periode $O(n)$ alle Linearzeitsprachen der UZA. Es ist auch möglich, den Beweis dafür auf direkte Weise ohne Betrachtung der UIA zu führen und dann auf UIA zu übertragen. Die Betrachtung von SUZA führt so in einer Reihe von Fällen zu recht einfachen Beweisen für bekannte Ergebnisse über Pseudo-Realzeit-UIA. Zum Beispiel läßt sich die in Satz 5.2 bewiesene Beschränkung der Äquivalenzklassen bezüglich $_k\sim$ auf UIA übertragen und damit eine Hierarchie der Pseudo-Realzeit-UIA mit verschiedenen Zellenzahlen beweisen.

Es läßt sich zeigen, daß jeder SZA mit Pipeline-Periode n von einem UIA mit n Zellen in Linearzeit erkannt werden kann und umgekehrt. Daher ist $SZA(n)$ nach [IbJ87] genau die Menge der in Linearzeit von ZA erkannten Sprachen. Damit ist die Frage, ob $SUZA(\rho)$ echt in $SZA(\rho)$ enthalten ist, zurückgeführt auf die noch unbeantwortete Frage nach der Äquivalenz von Realzeit-ZA und Linearzeit-ZA.
Ist L eine in Linearzeit von einem ZA erkennbare Sprache, so erhält man $L_\#$ aus L durch Anfügen von $2^{|w|}$ #-Zeichen an jedes Wort $w \in L$. $L_\#$ kann dann bereits von einem $O(\ln n)$-SZA erkannt werden, der mit einem Zähler die Anzahl der # überprüft. Falls L nicht in $ZA_R$ ist, so würde damit $SUZA(\ln n)$ echt in $SZA(\ln n)$ enthalten sein.

In einem Pseudo-Realzeit-UIA ist mit $O(\rho)$ Zellen eine Pipeline-Periode von n möglich, da man sofort nach Beendigung einer Eingabe ein neues Eingabewort eingeben kann. In einem $\rho$-SUZA wird degegen mit n Zellen eine Pipeline-Periode von $O(\rho)$ erreicht. In einem durch Zeit-Raum-Transformation erhaltenen systolischen Automaten aus $O(n \cdot \rho)$ Zellen erhält man schließlich die Pipeline-Periode 1. Es ergibt sich also eine Abhängigkeit zwischen der Anzahl der Zellen und der erreichbaren Pipeline-Periode.
In einem gewissen Sinne ist es sogar möglich, mit einer einzigen "Zelle" eine Pipeline-Periode von $O(n \cdot \rho)$ zu erhalten. Zwar kann ein einzelner endlicher Automat nur reguläre Sprachen erkennen, verwendet man jedoch stattdessen eine geeignete sequentielle Maschine, so können alle Sprachen aus $SUZA(\rho)$ in $n \cdot \rho$ Schritten von einer solchen Maschine erkannt werden. Diese Möglichkeit, die von UIA und damit auch die von SUZA erkannten Sprachen durch sequentielle Maschinen zu charakterisieren, wird auch als "Sweep"-Komplexität bezeichnet und zum Beispiel in [CIV88] und in [IbJ87] betrachtet.