6 Einige grundlegende Techniken für Zellularautomaten

Ziel dieses Kapitels ist es, einige Techniken und Konzepte vorzustellen, die bei der Konstruktion von Algorithmen für ZA immer wieder auftauchen. Dazu gehören zum Beispiel Signale, das Markieren ausgezeichneter Zellen und Zeitpunkte (einfache Fälle; schwierigere wie z. B. Synchronisation werden im nächsten Kapitel behandelt). Dies ist Gegenstand des ersten Abschnittes. Im zweiten beschäftigen wir uns dann mit der Verwendung von „Zählern“ und einfacher Arithmetik mit Zahlen, deren Bits in aufeinander folgenden Zellen gespeichert sind.

6.1 Signale und Markierungen

Wir haben schon des öfteren Ausschnitte aus Raum-Zeit-Diagrammen dargestellt, wie zum Beispiel in 5.7. Wir stellen sie üblicherweise nur für eindimensionale Zellularautomaten dar und zwar so, dass horizontal die Raumkoordinate aufgetragen ist und die Zeit von oben nach unten wächst. Gelegentlich werden nicht die vollständigen Zustände angegeben, sondern nur „Teile“ davon, z. B. die Inhalte eines bestimmten Registers in jeder Zelle.

Als erstes gehen wir auf Signale ein, wie sie etwa schon in Beispiel 3.12 benutzt wurden.

6.1 Beispiel. Wenn sich in DRAHTWELT ein Elektron von links nach rechts durch einen einfachen, „waagerechten“ Draht bewegt, dann legt das Elektronenvorderteil in dem Sinne ein Signal $\mathbf{e}$ fest, dass für eine Reihe aufeinanderfolgender Zeitpunkte $t \in T$ die Koordinaten der Zelle festgelegt sind, in der sich das Signal „gerade befindet“, z. B. $\mathbf{e}(0) = (x, y)$, $\mathbf{e}(1) = (x+1, y)$, $\mathbf{e}(2) = (x+2, y)$, usw.

Wir sind hier natürlich nur an Signalen interessiert, die von Zellularautomaten „erzeugt“ werden können. Um dies präzisieren zu können, legen wir zunächst ein mal fest:

6.2 Definition Eine Abbildung $\mathbf{s} : T \to R$ mit $T = \mathbb{N}_0$ oder $T = \{0, \ldots, n\}$ heiße ein *Signal*. ◇

$\mathbf{s}(t)$ gibt also die Zelle an, in der sich das Signal zum Zeitpunkt t befindet. Damit das Signal *konstruierbar* ist, ist es jedenfalls *notwendig*, dass es sich in jedem Schritt höchstens von einer Zelle zu einer (bzgl. N) benachbarten Zelle „bewegt“. Mit anderen Worten muss für alle t gelten: $\mathbf{s}(t) - \mathbf{s}(t+1) \in N$. (Man überlege sich genau, dass hier kein Vorzeichenfehler vorliegt!) Zweitens sollte man das Signal „leicht erkennen“ können. Und drittens sollte die Konstruktion durch die Arbeit eines ZA bewerkstelligt werden, und nicht schon in der Startkonfiguration kodiert sein.

6.3 Übung. Man zeige, dass die dritte Forderung wesentlich ist: Ansonsten kann *jedes* Signal, das die beiden ersten Forderungen erfüllt, „konstruiert“ werden.

Die folgende Definition berücksichtigt diese Forderungen auf besonders einfache Weise.

6.4 Definition Ein Signal $\mathbf{s}$ heißt genau dann *(ZA-)konstruierbar*, wenn es einen Zellularautomaten $C = (Q, \dots)$, einen Zustand $s' \in Q \setminus \{\sqcup\}$ und eine Menge $L \subset Q^N$ lokaler Konfigurationen gibt, so dass C ausgehend von der Konfiguration

$$c^0 : i \mapsto \begin{cases} s' & \text{falls } i = \mathbf{s}(0) \\ \sqcup & \text{sonst} \end{cases}$$

nacheinander Konfigurationen $c^1, c^2, \dots$ durchläuft, für die alle gilt: $c^t_{i+N} \in L \iff \mathbf{s}(t) = i$. ◇

6.5 Algorithmus. (Signale mit konstanten Geschwindigkeiten) Für jede *rationale* Zahl $r \in \mathbb{Q}$ mit $0 < r < 1$ ist das Signal $\mathbf{r}_r : \mathbb{N}_0 \to \mathbb{Z} : t \mapsto \lfloor rt \rfloor$ ZA-konstruierbar. Die folgenden Tabellen skizzieren für den Fall $r = \frac{2}{5}$ ein geeignetes δ und einen Ausschnitt aus einem entsprechenden Raum-Zeit-Diagramm:

$l(-1)$	$l(0)$	$l(1)$	$\delta(l)$
⊔	1>	⊔	2>
⊔	2>	⊔	3>
⊔	3>	⊔	⊔
3>	⊔	⊔	4>
⊔	4>	⊔	5>
⊔	5>	⊔	⊔
5>	⊔	⊔	1>

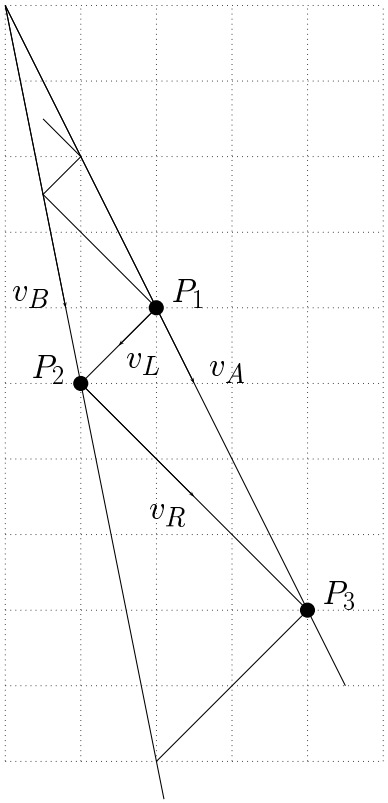
1			1>					
2			2>					
3			3>					
4				4>				
5				5>				
6					1>			
7					2>			
8					3>			
9						4>		
10						5>		

Für solche Signale hat es sich eingebürgert, von ihrer *Geschwindigkeit* zu sprechen. Damit ist die „im Durchschnitt pro Schritt zurückgelegte Entfernung", also r, gemeint. In eindimensionalen Zellularautomaten werden wir auch noch nach rechts laufenden Signalen positive Geschwindigkeiten und nach links laufenden die entsprechenden negativen Geschwindigkeiten zuordnen.

6.6 Es erscheint auf den ersten Blick nahezu unmöglich, Signale zu konstruieren, deren Geschwindigkeit zwar konstant, aber (jedenfalls in einem gewissen Sinne) eine irrationale Zahl, z. B. $\log_{10} 2$. Dass so etwas auch möglich ist, wurde von Korec (1993) gezeigt.

Bevor wir uns als nächstes ansehen, wie man kompliziertere Signale durch Verwendung mehrerer linearer Signale erzeugen kann, überlege man einmal selbst, wie man z. B. ein Zickzack-Signal oder eine Parabel konstruieren könnte.

6.7 Algorithmus. (Zickzack-Signal) Im folgenden nur sehr grob skizzierten Raum-Zeit-Diagramm eines eindimensionalen Zellularautomaten ist $v_L < 0 < v_B < v_A < v_R$.



6.8 Übung. Man gebe eine Zustandsmenge und eine lokale Überführungsfunktion an, die das Verhalten realisieren, das im vorangegangenen Algorithmus beschrieben wurde. Die folgende skizzenhafte Ausschnitt aus einem Raum-Zeit-Diagramm ist (vermutlich) hilfreich:

1		5>		a				
2			1>	b				
3			2	>	a			
4			3		<b			
5			4	<		a		
6			5>			b		
7				1>			a	
8				2	>		b	

6.9 Algorithmus. (Parabel-Signal) Im folgenden Raum-Zeit-Diagramm läuft im wesentlichen ebenfalls ein Signal zwischen zwei Grenzen hin und her. Dieses Mal bewegt sich die rechte Grenzmarkierung aber nicht „von selbst“ (und mit konstanter Geschwindigkeit) sondern sie wird „von dem Zickzack-Signal weitergeschoben“. Man überlege sich, dass der „Weg“ der rechten Grenzmarkierung im Raum-Zeit-Diagramm dadurch näherungsweise die Form einer Parabel hat!

1		(>)					
2		(| <)				
3		(< |)				
4		(> |)				
5		(| >)				
6		(		<)			
7		(	<	)			
8		(<		)			
9		(>		)			
10		(	>	)			
11		(		>)			
12		(			<)		
13		(		<	)		
14		(	<		)		
15		(<			)		
16		(>			)		
17		(	>		)		
18		(		>	)		
19		(			>)		
20		(				<)	
21		(			<	)	
22		(		<		)	
23		(	<			)	
24		(<				)	
25		(>				)	

6.10 Übung. Auf welchem Prinzip beruht Algorithmus 6.9? Wie kann man die Konstruktion verallgemeinern, um kubische, biquadratische, etc. Kurven zu realisieren?

Durch ein konstruierbares Signal wird zu einer Reihe aufeinanderfolgender Zeitpunkte jeweils eine Zelle markiert. Das verallgemeinern wir nun wie folgt:

6.11 Definition Eine beliebige Teilmenge $\mathbf{m} \subset R \times \mathbb{N}_0$ heißt eine *Menge von Markierungen*. Ist $\mathbf{m} \subset \{i\} \times \mathbb{N}_0$ für ein $i \in R$, so sprechen wir gelegentlich von *zeitlichen Markierungen*. Gibt es für jedes $i \in R$ höchstens ein $t \in \mathbb{N}_0$ mit $(i,t) \in \mathbf{m}$, so sprechen wir gelegentlich von *räumlichen Markierungen*. ◇

6.12 Definition Ein Menge $\mathbf{m}$ von Markierungen heißt *ausgehend von einer Konfiguration* c *(ZA-)konstruierbar*, wenn es einen Zellularautomaten $C = (Q, \dots)$ und eine Menge $L \subset Q^N$ lokaler Konfigurationen gibt, so dass C ausgehend von $c^0 = c$ nacheinander Konfigurationen $c^1, c^2, \dots$ durchläuft, für die gilt: Für alle i und alle t ist $c^t_{i+N} \in L \iff (i,t) \in \mathbf{m}$. ◇

Man mache sich klar, dass man Signale als spezielle Markierungen auffassen kann. Wie man sieht wurden die beiden Konstruierbarkeitsbegriffe so gewählt, dass sie miteinander verträglich sind.

Um Markierungen in einem Zellularautomaten vorzunehmen, können die Treffpunkte verschiedener Signale herangezogen werden.

6.13 Algorithmus. (MITTE-RAUM-MARK) Um die Mitte eines an den Enden gekennzeichneten Abschnittes zu markieren, kann man von einem Ende zwei Signale mit den Geschwindigkeiten

$\frac{1}{3}$ und 1 in die gleiche Richtung starten. Das schnellere wird am anderen Ende reflektiert. Die Signale treffen sich „genau“ in der Mitte.

6.14 Übung. (DOPPEL-RAUM-MARK) Man finde einen Algorithmus, der jede Konfiguration der Form

	(	.	.	.	.	.	)								

überführt in die korrespondierende Konfiguration

	(	.	.	.	.	.	)	(	.	.	.	.	.	)	

6.15 Übung. (WURZEL-RAUM-MARK) Man finde einen Algorithmus, der jede Konfiguration der Form

	(	.	.	.	.	.	.	.	)

überführt in die korrespondierende Konfiguration

	(	.	x	.	.	.	.	.	)

wobei die Länge des Abschnittes `( ··· x` gerade die Wurzel der Länge des Abschnittes `( ··· )` sein soll.

6.16 Algorithmus. (EXP-RAUM-MARK) Algorithmen der Art ZICKZACK-SIGNAL (siehe 6.7) können benutzt werden, um Zellen mit exponentiell wachsenden Abständen zu kennzeichnen. Im Raum-Zeit-Diagramm aus 6.7 gilt für die Punkte $P_i = (x_i, t_i)$:

$$\begin{aligned}
t_2 v_B &= x_1 + (t_2 - t_1) v_L \\
t_2 (v_B - v_L) &= x_1 - \frac{x_1}{v_A} v_L \\
t_2 &= x_1 \frac{v_A - v_L}{v_A (v_B - v_L)} \\
x_2 &= x_1 \frac{v_B (v_A - v_L)}{v_A (v_B - v_L)}
\end{aligned}$$

und weiter

$$\begin{aligned}
t_3 v_A &= x_2 + (t_3 - t_2) v_R \\
t_3 (v_A - v_R) &= x_1 \frac{v_B (v_A - v_L)}{v_A (v_B - v_L)} - x_1 \frac{(v_A - v_L) v_R}{v_A (v_B - v_L)} \\
t_3 &= x_1 \frac{(v_B - v_R)(v_A - v_L)}{v_A (v_B - v_L)(v_A - v_R)} \\
x_3 &= x_1 \frac{(v_B - v_R)(v_A - v_L)}{(v_B - v_L)(v_A - v_R)}
\end{aligned}$$

Also ist $\frac{x_3}{x_1}$ unabhängig von x_1 eine *Konstante* k. Folglich ist auch der Quotient aufeinanderfolgender Abstände

$$\frac{x_5 - x_3}{x_3 - x_1} = \frac{k^2 x_1 - k x_1}{k x_1 - x_1} = k$$

konstant. Die Abstände wachsen also *exponentiell*.

6.17 Beispiel. Wählt man etwa $v_L = -1$, $v_R = 1$, $v_B = \frac{1}{5}$ und $v_A = \frac{1}{2}$, so erhält man als Konstante gerade $k = \frac{-\frac{4}{5} \cdot \frac{3}{2}}{\frac{6}{5} \cdot (-\frac{1}{2})} = 2$.

Man überlege sich, dass jede rationale Konstante $k > 1$ „realisierbar“ ist. Was ist mit Rundungsproblemen?

6.18 Algorithmus. (QUADRAT-ZEIT-MARK) Algorithmus 6.9 kann benutzt werden, um die Zelle am linken Rand zu den Zeitpunkten $1, 4, 9, 16, \ldots$ zu markieren. Man beweise das!

6.19 Algorithmus. (PRIM-ZEIT-MARK) Man kann in einer Zelle auch die Zeitpunkte t markieren, die Primzahlen sind. Ein entsprechender Algorithmus wurde von Fischer (1965) vorgestellt. Er ist um einiges komplizierter als die bisher vorgestellten.

6.20 Algorithmus. (konzentrische Kreise in Realzeit) Der in den Augen des Verfassers dieses Skriptes faszinierendste Algorithmus in der ZA-Literatur überhaupt stammt aus der Dissertation von Laure Tougne. Sie hat einen 2-dimensionalen Zellularautomaten gefunden, der

- gestartet auf der Konfiguration, bei der alle Zellen aus einer einzigen im Ruhezustand sind,
- nach k Schritten genau die Zellen markiert, die der Diskretisierung des Kreises mit Radius k um die Ausgangszelle entsprechen.

Dieser ausgesprochen nichttriviale Algorithmus wurde zusammen mit Delorme, Mazoyer und Tougne (1999) veröffentlicht.

6.21 Definition Für ein Signal $\mathbf{s}$ heiße die i.a. partielle Abbildung $a_{\mathbf{s}} : R \to T : i \mapsto \min\{t \mid \mathbf{s}(t) = i\}$ die *Ankunftszeit(en)* von $\mathbf{s}$. ◇

6.22 Satz. (Terrier 1991) *Es sei $\mathbf{s} : \mathbb{N}_0 \to R$ ein Signal mit folgenden Eigenschaften:*

- *Es ist durch einen eindimensionalen ZA mit $H_1^{(1)}$-Nachbarschaft konstruierbar.*
- *$\mathbf{s}(0) = 0$ und $\forall t \in \mathbb{N}_0 : \mathbf{s}(t+1) \geqslant \mathbf{s}(t)$*
- *Die Ankunftszeiten sind streng monoton wachsend: $\forall i \in \mathbb{N}_0 : a_{\mathbf{s}}(i+1) > a_{\mathbf{s}}(i)$.*

Dann gilt:

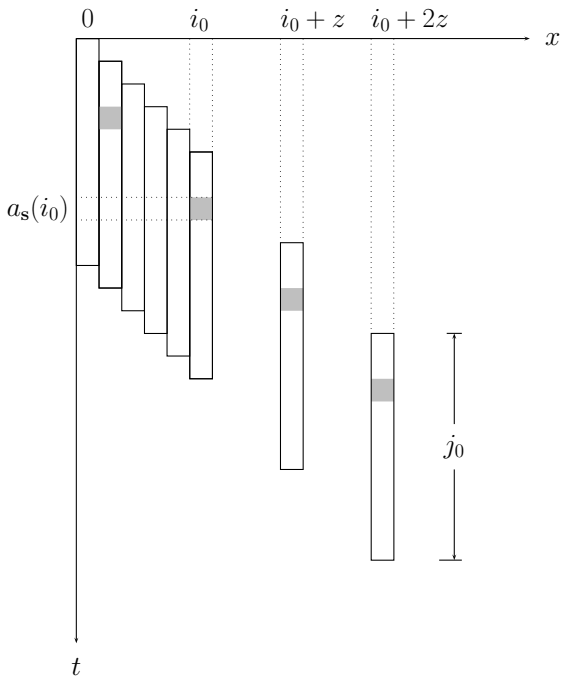
a) Entweder wird $a_{\mathbf{s}}(i) - i$ schließlich konstant,

b) oder es gibt eine Konstante $b \in \mathbb{N}_+$, so dass $\forall i \in \mathbb{N}_0 : a_{\mathbf{s}}(i) - i \geqslant \lfloor \log_b i \rfloor$.

Dieser Satz ist zum Beispiel deswegen bemerkenswert, weil er etwa das Signal $\mathbf{s}(t) = t - \lceil \log \lceil \log t \rceil \rceil$ als nicht konstruierbar kennzeichnet. Andererseits weiß man, dass das Signal $\mathbf{s}(t) = \lceil \log \lceil \log t \rceil \rceil$ konstruierbar ist. (Man überlege sich, wie!)

6.23 Beweis (Terrier 1991) Es sei $C = (Q, \ldots)$ ein Zellularautomat, der $\mathbf{s}$ konstruiert. Wir zeigen: falls nicht *b)* gilt, dann *a)*. Wenn *b)* nicht gilt, dann gibt es insbesondere für $b = |Q|$ ein i_0 derart, dass $a_{\mathbf{s}}(i_0) - i_0 < \lfloor \log_{|Q|} i_0 \rfloor$. Zur Abkürzung setzen wir $j_0 = \lfloor \log_{|Q|} i_0 \rfloor$.

Man betrachte im Raum-Zeit-Diagramm für die Anfangskonfiguration c^0 gemäß Definition 6.4 die Wörter $w_i = \Delta^i(c^0)(i)\Delta^{i+1}(c^0)(i) \cdots \Delta^{i+j_0-1}(c^0)$ der Länge j_0, die in der folgenden Abbildung durch die senkrechten Rechtecke skizziert sind; w_i ist also die Folge der Zustände, die Zelle i ab Zeitpunkt i durchläuft. Da $H_1^{(1)}$-Nachbarschaft vorausgesetzt wird, ist jede Zelle i vor dem Zeitpunkt i im Ruhezustand. Also kann man, wenn man nur w_i kennt, daraus bereits das w_{i+1} eindeutig bestimmen, d. h., es liegt ein funktionaler Zusammenhang vor.



Es gibt aber „nur“ $|Q|^{j_0} = |Q|^{\lfloor \log_{|Q|} i_0 \rfloor} \leqslant i_0$ paarweise verschiedene solche Wörter. Folglich muss w_{i_0} bereits unter den $\{w_0, w_1, \ldots, w_{i_0-1}\}$ mindestens einmal vorkommen. Also ist die Folge $(w_i)_{i \in \mathbb{N}_0}$ schließlich zyklisch; es sei $w_{i_0-z} = w_{i_0}$ und folglich für alle $k \in \mathbb{N}_0$ auch $w_{i_0+kz} = w_{i_0}$.

Nun kommt **s** nach Voraussetzung in Zelle i_0 zum Zeitpunkt $a_{\mathbf{s}}(i_0) < i_0 + j_0$ an. Der entsprechende Punkt im Raum-Zeit-Diagramm liegt also innerhalb von w_{i_0} und ist in der Abbildung grau dargestellt. Dabei handelt es sich um einen Zustand aus der Teilmenge Q' im Sinne von Definition 6.4. Es sei $a_0 = a_{\mathbf{s}}(i_0) - i_0$.

Da die Folge der w_i zyklisch ist, ist also auch in allen w_{i_0+kz} der entsprechende Zustand aus Q'. Das Signal **s** kommt also in den Zellen $i_0 + kz$ zum Zeitpunkt $i_0 + kz + a_0$ an. Da sich aber einerseits das Signal in jedem Schritt um höchstens eine Zelle weiterbewegen kann, andererseits die Ankunftszeiten streng monoton wachsen, muss **s** in *allen* Zellen $i \geqslant i_0$ *genau* zum Zeitpunkt $i + a_0$ ankommen. Also ist für $i \geqslant i_0$ mit anderen Worten $a_{\mathbf{s}}(i) - i = a_0$ konstant. ∎

Weitere Beispiele für Signale und Markierungen sowie Ergebnisse über Zusammenhänge zwischen verschiedenen Formen von Konstruierbarkeit findet man in dem Forschungsbericht von Mazoyer und Terrier (1999).

6.2 Zähler und einfache Arithmetik

Ein weiteres nützliches Hilfsmittel bei der Konstruktion von Zellularautomaten sind Zähler. Wir führen hier explizit nur die einfachste Version vor. An ihr sieht man aber auch schon, dass der Logarithmus als untere Schranke im Satz von Terrier optimal ist.

6.24 Algorithmus. (WANDER-ZAEHLER) Wir beschränken uns darauf, einen Ausschnitt aus einem Raum-Zeit-Diagramm anzugeben. Der Algorithmus realisiert einen Zähler, der durch einen eindimen-

sionalen Zellularautomaten „wandert“ und mit jeder Zelle um 1 erhöht wird. Wie man daran sieht, umfasst die Zustandsmenge unter anderem $\{(1), (1,1), 0), 1, 0\} \times \{(1, 1, 0\}$. Die erste (im RZD jeweils oben dargestellte) Komponente repräsentiert ein Bit das Zählers (ggf. mit Randkennzeichnung), die zweite (im RZD jeweils unten) einen Übertrag. Die Zähler-Bits, die nacheinander (mit niedrigstwertigem Bit voran) in Zelle i „vorbeikommen“ bilden die Dualzahldarstellung von i.

1	2	3	4	5	6	7	8	9	10	11	12	13
(1)												
0												
	0)											
	(1											
	(1	1)										
	0	0										
		(1	0)									
		0	1									
			0	1)								
			(1	0								
			(1	0	0)							
			0	0	1							
				(1	1	1)						
				0	0	0						
					(1	1	0)					
					0	0	1					
						(1	0	1)				
						0	1	0				
							0	0	0)			
							(1	0	1			
							(1	0	1	1)		
							0	0	0	0		
								(1	0	1	0)	
								0	0	0	1	
									(1	0	0	1)
									0	0	1	0

Wir verzichten wieder darauf, die lokale Überführungsfunktion explizit anzugeben.

Wie man an dem durch ein (gekennzeichneten höchstwertigen Bit sieht, realisiert der Algorithmus unter anderem auch ein Signal $\mathbf{s}$ mit $a_{\mathbf{s}}(i) = i + \log i + O(1)$.

6.25 Übung. Man modifiziere den Algorithmus WANDER-ZAEHLER so, dass nicht mehr alle Zellen gezählt werden, sondern nur noch die, die eine bestimmte Bedingung erfüllen (zum Beispiel am Anfang mit a initialisiert wurden).

WANDER-ZAEHLER kann auch als Algorithmus für die Umwandlung von Unär- in Binärdarstellung angesehen werden. Überlegen Sie sich einen Zellularautomaten, der die umgekehrte Umwandlung vornimmt.

6.26 Algorithmus. (ADDITION) Der folgende Algorithmus addiert zwei Dualzahlen. Er benötigt dafür in etwa so viele Schritte wie die größere Zahl lang ist. Wie man an Beispielen mit Summanden 2^{k-1} und 1 sieht, ist dieser Zeitbedarf bei der gewählten Zahldarstellung (bis auf einen kleinen konstanten Summanden) optimal.

Jede Zelle besteht aus vier Registern. Zwei werden für die Speicherung der Argumente benutzt, das dritte für die der Summe, und das vierte für den Übertrag. Es läuft ein Signal von rechts nach links vom niedrigst- zum höchstwertigen Bit, das jeweils den Übertrag von einer Stelle zur nächsten transportiert.

		(1	0	1	1)		
		(1	0	1	0)		
		(1	0	1	1)		
		(1	0	1	0)		
					1)		
					<0		
		(1	0	1	1)		
		(1	0	1	0)		
				0	1)		
				<1			
		(1	0	1	1)		
		(1	0	1	0)		
			1	0	1)		
			<0				
		(1	0	1	1)		
		(1	0	1	0)		
		0	1	0	1)		
		<1					
		(1	0	1	1)		
		(1	0	1	0)		
	(1	0	1	0	1)		

In der Darstellung sind die Summanden gleich lang. Es ist aber kein Problem, die Überführungsfunktion so gestalten, dass auch der Fall verschieden langer Summanden korrekt behandelt wird (sofern man sich zum Beispiel auf nichtnegative Zahlen beschränkt).

Die letzte arithmetische Operation, mit der wir uns beschäftigen wollen, ist die Multiplikation. Da wir sie nach der „Schulmethode“ realisieren wollen, zunächst noch ein Verschiebealgorithmus.

6.27 Algorithmus. (VERSCHIEBUNG) Beim folgenden Algorithmus bestehe jede Zelle aus drei Registern. Die in den ersten Registern gespeicherte Zahl wird (unter Zuhilfenahme der zweiten) nach links verschoben. Die Verschiebungen werden jeweils am rechten Ende von einem Signal initiiert, das in den dritten Registern mit Geschwindigkeit 1/2 nach links wandert, bis es in einer gekennzeichneten Zelle absorbiert wird.

					(1	0	1	1)	
				|				<a	
					(1	0	1		
								<1)	
				|				<b	
					(1	0	1)		
							<1		
				|			<a		
					(1	1			
						<0	<1)		
				|			<b		
					0	1)			
					<(1	<1			
				|		<a			
				(1	1				
					<0	<1)			
				|		<b			
				0	1)				
				<(1	<1				
				|	<a				
			(1	1					
				<0	<1)				
				|	<b				
			0	1)					
			<(1	<1					
				|					
		(1	1	1)					
			<0						
				|					
		0	1	1)					
		<(1							
				|					
	(1	0	1	1)					
				|					

6.28 Algorithmus. (MULTIPLIKATION) Wir benutzen im Prinzip die Methode, nach der man üblicherweise von Hand zwei Zahlen multipliziert.

```
          1  0  1  1  ·  1  0  1
          1  0  1  1
       0  0  0  0
 1  0  1  1
 1  1  0  1  1  1
```

Verschobene Kopien des ersten Faktors werden mit den Bits des zweiten Faktors multipliziert

und die entstehenden Zahlen addiert. Das Raum-Zeit-Diagramm für eine Beispielberechnung ist auf die nächsten beiden Abbildungen verteilt.

				(1	0	1	1)		
					(1	0	1)		
				|			<a		
				(1	0	1			
							<1)		
					(1	0	1)		
							1)		
							<0*1		
				|			<b		
				(1	0	1)			
						<1			
					(1	0	1)		
						1	1)		
						<0*1			
				|		<a			
				(1	1				
					<0	<1)			
					(1	0	1)		
					0	1	1)		
					<0*1	<0*0			
				|		<b			
				0	1)				
				<(1	<1				
					(1	0	1)		
				(1	0	1	1)		
				<0*1	<0*0				
				|	<a				

Die in der Darstellung obersten Register beinhalten den ersten Faktor. Darunter befindet sich die Schicht der Hilfsregister für die Bits, die gerade verschoben werden. Es wird der Algorithmus aus 6.27 benutzt. Die dritten Register speichern den zweiten Faktor und die vierten die Zwischensumme bzw. am Ende das Ergebnis der Multiplikation. Das Aufaddieren des (verschobenen) ersten Faktors erfolgt gemäß Algorithmus 6.26. Im fünften Register werden die Übertragsbits nach links geschoben (<0 und <1) sowie die Information darüber, ob der verschobene Faktor addiert werden muss (*1) oder nicht (*0). Im letzten Register läuft ein Signal mit Geschwindigkeit 1/2 nach links und startet die Verschiebungen des ersten Faktors usw. bis es den gesamten zweiten Faktor passiert hat.

			(1	1					
				<0	<1)				
					(1	0	1)		
				(1	1	1	1)		
				<0*0	<0*1				
				|	<b				
			0	1)					
			<(1	<1					
					(1	0	1)		
				0	1	1	1)		
			<0*0	<1*1					
		(1	1	1)					
			<0						
					(1	0	1)		
			1	0	1	1	1)		
			<0*1						
		0	1	1)					
		<(1							
					(1	0	1)		
		(1	1	0	1	1	1)		
		<0*1							
	(1	0	1	1)					
					(1	0	1)		
		(1	1	0	1	1	1)		

Man überzeugt sich recht schnell davon, dass der Zeitbedarf des Algorithmus höchstens $3n$ beträgt, wenn n die Länge des längeren Faktors ist.

Im Gegensatz zur Addition wirkt sich hier die Tatsache, dass im Vergleich etwa zu einer Turingmaschine „viele“ endliche Automaten an der Verarbeitung beteiligt sind, deutlich auf den Zeitbedarf aus.

6.29 Übung. Eine leicht einzusehende theoretische untere Schranke für den Zeitbedarf der Multiplikation ist $2n$.

- Wie kommt man zu dieser Schranke?
- Man versuche, einen Verfahren zu entwerfen, der dieser Schranke näher kommt als Algorithmus 6.28.

Die eben vorgestellten auf Bitebene arbeiten Algorithmen sind natürlich recht naiv. Weniger trivial ist es zum Beispiel schon, einen Divisionsalgorithmus entwickeln, der in einer Zeit proportional zu (oder gar identisch mit) der Länge des Dividenden arbeitet. Auch das geht.

Wer sich genauer für solche, mitunter sehr trickreiche, Bitalgorithmen interessiert, der findet in der Literatur zum Teil auch unter dem Stichwort *systolische Algorithmen* noch viele interessante Ideen.

Zusammenfassung

- Für jede rationale Zahl r, die kleiner oder gleich dem „Radius“ der Nachbarschaft ist, kann man Signale konstruieren, deren Durchschnittsgeschwindigkeit gleich r ist.
- Lässt man solche Signale geschickt zusammenwirken, dann kann man auch kompliziertere Signale realisieren.
- Häufig verwendet man Signale, um bestimmte Zellen oder/und Zeitpunkte zu markieren.
- Nichtnegative ganze Zahlen kann man speichern, indem man die Bits ihrer Dualzahldarstellung in nebeneinanderliegenden Zellen ablegt.
- Solche Zahlen kann man wandern lassen, und dabei zum Beispiel bei Vorliegen lokal entscheidbarer Kriterien de- oder inkrementieren.
- Mit solchen Zahldarstellungen kann man auch (schnell) einfache Arithmetik betreiben.

Literatur

Delorme, Marianne, Jacques Mazoyer und Laure Tougne (1999). «Discrete parabolas and circles on 2D cellular automata». In: *Theoretical Computer Science* 218.2, S. 347–417 (siehe S. 48).

Fischer, Patrick C. (1965). «Generation of Primes by a One-Dimensional Real-Time Iterative Array». In: *Journal of the ACM* 12, S. 388–394 (siehe S. 48).

Korec, Ivan (1993). «Irrational Speeds of Configurations Growth in Generalized Pascal Triangles». In: *Theoretical Computer Science* 112.2, S. 399–412 (siehe S. 44).

Mazoyer, Jacques und Véronique Terrier (1999). «Signals in one-dimensional cellular automata». In: *Theoretical Computer Science* 217.1, S. 53–80 (siehe S. 49).

Terrier, Véronique (1991). «Signals in Linear Cellular Automata». In: *Proceedings of a Workshop on Cellular Automata*. Centre for Scientific Computing. Espoo, Finland (siehe S. 48).

