Karlsruhe Institute of Technology

Vergleich der Sprachklassen verschiedener Typen von Zellularautomaten

Bachelorarbeit
von

Henning Dieterichs

am Institut für Theoretische Informatik
der Fakultät für Informatik

Erstgutachter:	Prof. Dr.-Ing. R. Vollmar
Zweitgutachter:	Prof. Dr. Hartmut Prautzsch
Betreuender Mitarbeiter:	Dr. Thomas Worsch

Bearbeitungszeit: 26. März 2018 – 25. Juli 2018

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt, die wörtlich oder inhaltlich übernommenen Stellen als solche kenntlich gemacht und die Satzung des KIT zur Sicherung guter wissenschaftlicher Praxis in der jeweils gültigen Fassung beachtet habe.

Karlsruhe, den 25. Juli 2018

Inhaltsverzeichnis

1. Einleitung

Mithilfe von Zellularautomaten können räumlich und zeitlich diskrete, dynamische Systeme formal beschrieben werden. Das globale Verhalten eines solchen Zellularautomaten ergibt sich dabei aus lokalen Regeln. Dies ermöglicht und erzwingt hochgradig paralleles Vorgehen, da räumlich unabhängige Bereiche auch unabhängig voneinander betrachtet und ausgeführt werden können.

Zellularautomaten können auch zum Erkennen von formalen Sprachen verwendet werden und stellen dadurch ein alternatives Berechnungsmodell zur Turingmaschine dar. Offensichtlich berechnungsäquivalent zu Turingmaschinen, werfen Zellularautomaten durch ihr paralleles Vorgehen eine Reihe von interessanten Fragestellungen auf, die sich in der Art bei Turingmaschinen nicht ergeben.

Eine der interessantesten und immer noch offenen Fragestellungen in diesem Bereich ist, ob Linearzeit-Zellularautomaten Sprachen erkennen können, die von Echtzeit-Zellularautomaten nicht erkannt werden können. Ausgehend von dieser Fragestellung werden in dieser Arbeit verschiedene Erweiterungen von Echtzeit-Zellularautomaten untersucht, um besser zu verstehen, wie mächtig diese sind.

Zunächst werden in Kapitel 2 verschiedene Typen von Zellularautomaten definiert, verglichen und teilweise bekannte Resultate in Entsprechung zu den in dieser Arbeit eingeführten Definitionen neu bewiesen. Anschließend werden in Vorbereitung auf folgende Kapitel in Kapitel 3, Kapitel 4 und Kapitel 5 eine Reihe von interessanten Sätzen gezeigt. Insbesondere die Konstruktion aus Kapitel 5 zur Simulation von Echtzeit-Zellularautomaten innerhalb von Echtzeit mit Rücksetzfunktionalität erlaubt neuartige Ansätze, um Erweiterungen von Echtzeit-Zellularautomaten auf normale Echtzeit-Zellularautomaten zurückzuführen. Solche Erweiterungen werden in Kapitel 6 betrachtet. In Kapitel 7 werden abschließend eingeschränkte Echtzeit-Zellularautomaten untersucht, um Einsicht darüber zu erhalten, wie leicht Echtzeit-Zellularautomaten hinsichtlich ihrer Mächtigkeit eingeschränkt werden können.

2. Grundlagen

2.1 Notation

In diesem Abschnitt werden Besonderheiten der in dieser Arbeit verwendeten Notation geklärt.

Sofern nicht anders vermerkt, bezeichnen Σ und Γ beliebige endliche Mengen, auch Alphabete genannt.

In dieser Arbeit werden die Zeichen eines Wortes mit 1 beginnend durchnummeriert. Für ein Wort $w \in \Sigma^*$ gilt also die Identität $w = w_1 w_2 ... w_{|w|}$. Der Ausdruck $w[m...n]$ meint das Teilwort von w, das die Zeichen w_m, w_{m+1}, ... bis einschließlich w_n umfasst.

$\mathcal{P}(M)$ bezeichne die Potenzmenge einer Menge, also die Menge aller ihrer Teilmengen.

Weiter bezeichne $\mathbb{B} := \{\mathtt{0}, \mathtt{1}\}$ das Binär-Alphabet.

Wenn ein Tupel T durch $T := (A, B, ..., Z)$ definiert wird, wobei A, B, ..., Z symbolische Variablen sind, die sich auf einen entsprechenden Wert beziehen, bezeichnen A_T, B_T, ..., Z_T die Komponenten des Tupels T. Es gilt dann offensichtlich: $T = (A_T, B_T, ..., Z_T)$.

Für eine Fallunterscheidung

$$x = \begin{cases} 1 & \text{falls } A, \\ 2 & \text{sonst, falls } B, \\ 3 & \text{sonst, falls } C, \\ 4 & \text{sonst.} \end{cases}$$

meint ein „sonst“, dass alle darüberstehenden Fälle nicht zutreffen. So ist $x = 2$, falls B gilt, aber nicht A und $x = 3$, falls weder A noch B gilt, aber C.

2.2 Definition Zellularautomat

In diesem Abschnitt werden grundlegende Objekte und Definitionen im Zusammenhang mit den in dieser Arbeit betrachteten Zellularautomaten eingeführt.

Definition 2.2.1 (Zellularautomat)**.** Ein eindimensionaler Zellularautomat mit einer Nachbarschaft von Radius 1 wird in dieser Arbeit als Tupel (Q, δ) definiert, wobei Q eine endliche, nicht-leere Menge ist und δ eine Abbildung mit $\delta : Q^3 \to Q$. Q wird auch Zustandsmenge und δ lokale Überführungsfunktion genannt. Es bezeichne CA^* die Menge aller solcher Tupel. In dieser Arbeit werden nur solche Zellularautomaten betrachtet, weswegen die Dimension und Nachbarschaft im Folgenden nicht mehr erwähnt wird.

Es werden nun Begriffe eingeführt, mithilfe derer bestimmte Verhaltensweisen von Zuständen in Zellularautomaten gefordert werden können.

Definition 2.2.2. Sei $(Q, \delta) \in \mathrm{CA}^*$ ein Zellularautomat.

- $R \subseteq Q$ heißt passive Zustandsmenge, wenn $\forall a, b, c \in R : \delta(a, b, c) = b$.
- $q \in Q$ heißt passiver Zustand, wenn $\{q\}$ eine passive Zustandsmenge ist.
- $R \subseteq Q$ heißt δ-abgeschlossene Menge, wenn $\forall b \in R : \forall a, c \in Q : \delta(a, b, c) \in R$.
- $q \in Q$ heißt toter Zustand, wenn $\{q\}$ eine δ-abgeschlossene Menge ist.
- δ heißt links bzw. rechts-unabhängig, wenn $\forall a_1, a_2, b, c \in Q : \delta(a_1, b, c) = \delta(a_2, b, c)$ bzw. wenn $\forall a, b, c_1, c_2 \in Q : \delta(a, b, c_1) = \delta(a, b, c_2)$.
- $q \in Q$ heißt initial, wenn $\delta(a, b, c) = q \Rightarrow b = q$.

Die globale räumliche Situation eines Zellularautomaten zu einem festen Zeitpunkt wird mithilfe einer Konfiguration beschrieben.

Definition 2.2.3 (Konfiguration)**.** Eine Konfiguration über einer endlichen Zustandsmenge Q ist eine Abbildung $c \in Q^{\mathbb{Z}}$. Der Zustand an Position i einer Konfiguration c wird durch $c_i := c(i)$ beschrieben.

Die lokalen Regeln eines Zellularautomaten bestimmen, wie aus einer Konfiguration eine Folgekonfiguration entsteht.

Definition 2.2.4 (Folgekonfiguration)**.** Die Funktion Δ_C bildet in Abhängigkeit eines Zellularautomaten $C = (Q, \delta) \in \mathrm{CA}^*$ eine Konfiguration c auf die Folgekonfiguration $\Delta_C(c)$ ab:

$$\Delta_C : Q^{\mathbb{Z}} \to Q^{\mathbb{Z}},\ (\Delta_C(c))(i) := \delta(c_{i-1}, c_i, c_{i+1})$$

$\Delta_C^t(c)$ bildet eine Konfiguration c auf die t-te Folgekonfiguration für ein $t \in \mathbb{N}_0$ ab. Es gilt $\Delta_C^0 = \mathrm{id}$ und $\Delta_C^t = \Delta_C \circ \Delta_C^{t-1}$.

Ausgehend von einer Konfiguration und dem Begriff einer Folgekonfiguration kann ein Zellularautomat nun verwendet werden, um ein Berechnungsmodell zu bilden. Zunächst werden aber Objekte eingeführt, um von verschiedenen solcher Modelle abstrahieren zu können.

Definition 2.2.5 (Σ^* erkennender Zellularautomat)**.** Ein Σ^* erkennender Zellularautomat ist ein Tupel $C := (Q, \delta, \Sigma, \#, L, S)$ mit $(Q, \delta) \in \mathrm{CA}^*$, $\Sigma \subseteq Q$, $\# \in Q \setminus \Sigma$ und $L \subseteq \Sigma^*$ beliebig. S ist ein beliebiges Objekt derart, dass die Klasse aller solcher S eine Menge bildet.

wird bei der Einbettung von Wörtern in Konfigurationen als Rand verwendet. Im Gegensatz zu üblichen Definitionen von Zellularautomaten zur Erkennung von Sprachen, die etwa fordern, dass # passiv oder gar tot ist, gibt es in dieser Definition keine Einschränkungen. Es wird später gezeigt, dass # immer passiv und in den in dieser Arbeit relevanten Fällen auch tot gewählt werden kann.

Es bezeichne CA die Menge aller solcher Tupel. Wegen der Forderung an S ist diese Menge wohldefiniert, aber nicht notwendigerweise abzählbar. Durch Spezialisierung der Sprache L, die in dieser Definition keiner Einschränkung unterliegt, und dem ebenfalls frei wählbaren Objekt S werden im Folgenden verschiedene Typen von Zellularautomaten eingeführt. Die Menge CA erlaubt es, von diesen Spezialisierungen zu abstrahieren.

Für eine Teilmenge $M \subseteq \mathrm{CA}$ definiere die von M erzeugte Sprachklasse:

$$\mathcal{L}(M) := \{L_C \mid C \in M\}$$

Definition 2.2.6 (CA-Funktor)**.** Eine Funktion $\mathcal{F} : \mathcal{P}(\mathrm{CA}) \to \mathcal{P}(\mathrm{CA})$ heißt CA-Funktor. Seien $\mathcal{F}_1, ..., \mathcal{F}_n$ CA-Funktoren. Definiere:

$$\mathrm{CA}^{\mathcal{F}_1\text{-}...\text{-}\mathcal{F}_n} := (\mathcal{F}_n \circ ... \circ \mathcal{F}_1)(\mathrm{CA})$$

Im Laufe dieser Arbeit werden eine Reihe solcher Funktoren eingeführt. Funktoren ermöglichen es, bequem eine Klasse von Zellularautomaten zu spezialisieren oder in eine andere zu transformieren.

Bisher wurde noch nicht beschrieben, wie Zellularautomaten mit endlichen Wörtern interagieren. Dazu werden diese zu einer Konfiguration fortgesetzt.

Definition 2.2.7 (Einbettung von Wörtern in Konfigurationen)**.** Für einen erkennenden Zellularautomaten $(Q, \delta, \Sigma, \#, L, S) \in \mathrm{CA}$ lässt sich jedes Wort $w \in \Sigma^*$ wie folgt zu einer Konfiguration fortsetzen:

$$[w] : \mathbb{Z} \to Q,\ [w](p) := \begin{cases} w_p & p \geq 1 \wedge p \leq |w|, \\ \# & \text{sonst.} \end{cases}$$

Beispiel 2.2.8 (Konfiguration). Für $w = \texttt{abc}$ wird $[w]$ durch folgende Tabelle dargestellt:

p	...	-2	-1	0	1	2	3	4	5	6	...
$[w]_p$	...	#	#	#	a	b	c	#	#	#	...

Die Position einer Zelle in einer Konfiguration wird hier durch die Variable p beschrieben.

2.2.1 Definition F-haltender Zellularautomat

Ausgehend von den eingeführten Objekten kann nun ein erstes Berechnungsmodell zum konkreten Erkennen von Sprachen definiert werden.

Definition 2.2.9 (F-haltender Zellularautomat)**.** Es sei ein Zellularautomat $C := (Q, \delta, \Sigma, \#, L, (F, F^+, p)) \in \text{CA}$ mit $F \subseteq Q$, $F^+ \subseteq F$ und $p : \mathbb{N}_0 \to \mathbb{N}_0$, $p(0) = 0$ gegeben.

Anschaulich gesprochen repräsentiert F die Menge der finalen Zustände, F^+ die Menge der akzeptierenden finalen Zustände und $F^- := F \setminus F^+$ die Menge der nicht-akzeptierenden finalen Zustände. p ist eine Funktion, die abhängig von der Länge des Eingabeworts die Position der Zelle angibt, welche anzeigt, ob das Eingabewort akzeptiert wird.

Für ein Wort $w \in \Sigma^*$ gibt die Funktion time_C an, wie viele Schritte der Zellularautomat C braucht, um das Wort w zu akzeptieren:

$$\text{time}_C : \Sigma^* \to \mathbb{N}_0 \cup \{\infty\},\ w \mapsto \min(\{t \in \mathbb{N}_0 \mid \Delta_C^t([w])_{p(|w|)} \in F\} \cup \{\infty\})$$

Dies lässt sich auf die Wortlänge übertragen:

$$\text{worst-time}_C : \mathbb{N}_0 \to \mathbb{N}_0 \cup \{\infty\},\ n \mapsto \max\{\text{time}_C(w) \mid w \in \Sigma^n\}$$

C heißt *F-haltend*, wenn $L = L'$ mit

$$L' := \{w \in \Sigma^* \mid \text{time}_C(w) \neq \infty \wedge \Delta_C^{\text{time}_C(w)}([w])_{p(|w|)} \in F^+\}$$

Das F in F-haltend ist symbolisch gemeint und bezieht sich nicht auf den Wert der Menge F.

Mit folgender Definition des CA-Funktors FH bezeichnet CA^{FH} die Menge aller F-haltenden Zellularautomaten:

$$\text{FH}(M \subseteq \text{CA}) := \{C \in M \mid C \text{ ist } F\text{-haltend}\}$$

Ist $\mathcal{F}$ ein CA-Funktor, sodass die Menge $\{p_C \mid C \in \text{CA}^{\text{FH-}\mathcal{F}}\}$ abzählbar ist, dann ist auch die Menge $\text{CA}^{\text{FH-}\mathcal{F}}$ abzählbar. Die später eingeführten Funktoren L, R und M haben diese Eigenschaft.

2.2.2 Definition t-haltender Zellularautomat

Analog werden die t-haltenden Zellularautomaten definiert, bei denen keine finalen Zustände das Ende der Berechnung signalisieren, sondern beliebige Funktionen. Auf die Unterschiede zu F-haltenden Automaten wird in Abschnitt 2.3 eingegangen.

Definition 2.2.10 (t-haltender Zellularautomat)**.** Gegeben ein Zellularautomat $C := (Q, \delta, \Sigma, \#, L, (F^+, t, p)) \in \text{CA}$ mit $F^+ \subseteq Q$ und $t, p : \mathbb{N}_0 \to \mathbb{N}_0$, $t(0) = 0$, $p(0) = 0$. F^+ repräsentiert die Menge der akzeptierenden Zustände und t und p Funktionen, die abhängig von der Länge des Eingabeworts angeben, wie viele Schritte auszuführen sind bzw. welche Zelle entscheidet, ob das Wort zu akzeptieren ist.

C heißt *t-haltend*, wenn $L = L'$ mit

$$L' := \{w \in \Sigma^+ \mid \Delta_C^{t(|w|)}([w])_{p(|w|)} \in F^+\}$$

Das t in t-haltend ist symbolisch gemeint und bezieht sich nicht auf die konkrete Funktion t.

Mit folgender Definition des CA-Funktors TH bezeichnet $\mathrm{CA}^{\mathrm{TH}}$ die Menge aller t-haltender Automaten:

$$\mathrm{TH}(M \subseteq \mathrm{CA}) := \{C \in M \mid C \text{ ist } t\text{-haltend}\}$$

2.2.3 Verschiedene Funktoren

Es werden nun verschiedene Funktoren definiert, über die t- oder F-haltende Automaten weiter eingeschränkt werden können.

Definition 2.2.11 (L-, M- und R-CA-Funktoren)**.** Ein t- oder F-haltender Zellularautomat C heißt

- *links-erkennend*, falls $p_C(n) = 1$ (definiert CA-Funktor L),
- *mittig-erkennend*, falls $p_C(n) = \lceil n/2 \rceil$ (definiert CA-Funktor M) und
- *rechts-erkennend*, falls $p_C(n) = n$ (definiert CA-Funktor R).

Definition 2.2.12 (time-, RT-, LT- und Mid-CA-Funktoren)**.** Sei $F \subseteq {\mathbb{N}_0}^{\mathbb{N}_0}$ eine Menge von Funktionen. Definiere:

$$\mathrm{time}(F)(M \subseteq \mathrm{CA}) := \{C \in \mathrm{FH}(M) \mid \text{worst-time}_C \in F\} \cup \{C \in \mathrm{TH}(M) \mid t_C \in F\}$$

Mit folgenden Definitionen ist $\mathrm{CA}^{\text{L-RT}}$ bzw. $\mathrm{CA}^{\text{L-LT}}$ die Menge der links-erkennenden Echtzeit- bzw. Linearzeit-Zellularautomaten:

- $\mathrm{RT} := \mathrm{time}(\{n \mapsto n-1\}) \circ \mathrm{TH}$
- $\mathrm{LT} := \mathrm{time}(O(n)) \circ \mathrm{FH}$
- $\mathrm{Mid} := \mathrm{time}(\{n \mapsto \lfloor n/2 \rfloor\}) \circ \mathrm{TH} \circ M$

Echtzeit-Zellularautomaten werden über t-haltende Automaten definiert, da ein F-haltender Automat bei einem Wort w als Eingabe offensichtlich nicht in Zeit $|w| - 1$ halten kann - die erste Zelle sieht zu diesem Zeitpunkt noch nicht das Ende des Wortes. t-haltende Automaten hingegen können bestimmte Eigenschaften eines Wortes w in Zeit $|w| - 1$ erkennen und erkennen diese Eigenschaft damit gleichzeitig auch für alle Präfixe.

Linearzeit-Zellularautomaten werden über F-haltende Automaten definiert, da in der Menge $O(n)$ unberechenbare Funktionen enthalten sind. F-haltende Automaten müssen selbstständig ihr Berechnungsende angeben, sodass unberechenbare Zeiteinschränkungen nicht problematisch werden. Diese Thematik wird in Abschnitt 2.3 weiter untersucht.

Definition 2.2.13 (OCA_L- und OCA_R-CA-Funktoren)**.** Der OCA_L bzw. OCA_R Funktor wählt diejenigen links- bzw. rechts-erkennenden Automaten aus, deren Zustandsübergangsfunktion links- bzw. rechts-unabhängig ist.

2.3 Vergleich von t-haltenden und F-haltenden Automaten

Es folgt ein Lemma für F-haltende Automaten. Es zeigt, dass solche Automaten stets so umgebaut werden können, dass wenn sie einmal einen akzeptierenden oder ablehnenden Zustand einnehmen, diesen nicht mehr verlassen, sich also nicht mehr umentscheiden können. Ganz trivial ist das Lemma nicht, schließlich können Zustände aus F auch während der Berechnung in Zellen angenommen werden, die von der Funktion p nicht ausgewählt werden.

Lemma 2.3.1 (F^+ und F^- können δ-abgeschlossen gewählt werden)**.** *Sei* $C = (Q, \delta, \Sigma, \#, L, (F, F^+, p))$ *ein* F*-haltender Zellularautomat. Dann existiert ein* F*-haltender Zellularautomat* $C' = (Q', \delta', \Sigma, \#, L', (F', F'^+, p))$ *mit* $\text{time}_C = \text{time}_{C'}$ *und* $L = L'$*, wobei* F'^+ *und* F'^- δ*-abgeschlossene Mengen sind.*

Beweis. Setze $Q' := Q \times \{-1, 0, 1\}$ und identifiziere Σ mit $\Sigma \times \{0\}$ und $\#$ mit $(\#, 0)$. Wähle δ' wie folgt:

$$\delta'((a, f_a), (b, f_b), (c, f_c)) := \begin{cases} (\delta(a, b, c), f_b) & \text{falls } f_b \neq 0, \\ (\delta(a, b, c), -1) & \text{falls } f_b = 0 \text{ und } \delta(a, b, c) \in F^-, \\ (\delta(a, b, c), 1) & \text{falls } f_b = 0 \text{ und } \delta(a, b, c) \in F^+, \\ (\delta(a, b, c), 0) & \text{sonst.} \end{cases}$$

Setze $F' := Q \times \{-1, 1\}$ und $F'^+ := Q \times \{1\}$. Dann sind die Mengen F'^+ und F'^- offensichtlich δ'-abgeschlossen.

Sei $w \in \Sigma^*, p \in \mathbb{Z}, t \in N_0$ und $(q, f) := \Delta^t_{C'}([w])_p$. Dann gilt $q = \Delta^t_C([w])_p$. $(q, f) \notin F'$ impliziert $f = 0$ und damit, dass $\forall k \in \{0, ..., t\} : \Delta^k_C([w])_p \notin F$. Anderseits folgt aus $(q, f) \in F'$, dass es ein $k \in \{0, ..., t\}$ gibt, sodass $\Delta^k_C([w])_p \in F$. Da der Automat bei finalen Zuständen stoppt, gilt dann wegen der ersten Überlegung, dass $k = t$. Daraus folgt, dass $\text{time}_C = \text{time}_{C'}$ und $L = L'$. □

t-haltende Automaten müssen in Abhängigkeit zur Wortlänge stets eine Position und einen Zeitpunkt angeben, zu dem feststehen muss, ob das Wort akzeptiert oder abgelehnt werden soll. F-haltende Automaten haben die Möglichkeit, nicht zu terminieren. Da die Funktion zur Wahl des Zeitpunktes bei einem t-haltenden Automaten nicht berechenbar sein muss, sind t-haltende Automaten trotzdem nicht weniger mächtig als F-haltende, wie der folgende Satz zeigt.

Satz 2.3.2 (t-haltende Automaten sind mindestens so mächtig wie F-haltende)**.** *Sei* $C = (Q, \delta, \Sigma, \#, L, (F, F^+, p))$ *ein* F*-haltender Zellularautomat. Dann gibt es einen* t*-haltenden Zellularautomat* C' *mit* $L_{C'} = L_C$*,* $p_{C'} = p_C$ *und* $\forall n \in \mathbb{N}_0 : t_{C'}(n) \leq \text{worst-time}_C(n)$.

Beweis. Wegen Lemma 2.3.1 sind OBdA. F^+ und F^- δ-abgeschlossene Mengen.

Betrachte folgende Funktion t:

$$t : \mathbb{N}_0 \to \mathbb{N}_0,\ n \mapsto \max(\{0\} \cup \{\text{time}_C(w) \mid w \in \Sigma^n,\ \text{time}_C(w) \neq \infty\})$$

Wähle $C' := (Q, \delta, \Sigma, \#, L', (F^+, t)) \in \mathrm{CA}^{\mathrm{TH}}$. Es gilt offensichtlich $\forall n \in \mathbb{N}_0 : t(n) \leq \text{worst-time}_C(n)$.

Sei $w \in L_C$. Dann gilt $\text{time}_C(w) \neq \infty$ und $\Delta_C^{\text{time}_C(w)}([w])_{p(|w|)} \in F^+$. Wegen $t(|w|) \geq \text{time}_C(w)$ und weil F^+ δ-abgeschlossen ist, folgt $\Delta_{C'}^{t(|w|)}([w])_{p(|w|)} \in F^+$, also $w \in L_{C'}$.

Sei $w \in L_{C'}$. Dann gilt $f_1 := \Delta_{C'}^{t(|w|)}([w])_{p(|w|)} \in F^+$. Dann ist $\text{time}_C(w) \leq t(|w|)$, da in diesem Fall $t(|w|) = 0 \Rightarrow \text{time}_C(w) = 0$. Nach Definition von time folgt $f_2 := \Delta_C^{\text{time}_C(w)}([w])_{p(|w|)} \in F$. Wäre $f_2 \in F^-$, dann wäre auch $f_1 \in F^- = F \setminus F^+$, da auch F^- δ-abgeschlossen ist. Also folgt $f_2 \in F^+$. Also $w \in L_{C'}$.

Insgesamt ergibt sich $L_C = L_{C'}$. □

Anmerkung. Das t aus dem vorherigen Beweis ist im Allgemeinen tatsächlich auch nicht berechenbar - andernfalls wären semientscheidbare Sprachen stets entscheidbar, da sich Turingmaschinen auf Zellularautomaten reduzieren lassen und umgekehrt.

Unter bestimmten Annahmen an die Funktion t zur Zeitauswahl können allerdings t-haltende Automaten auch zu F-haltenden Automaten umgebaut werden. Die Funktion $t(n) := n - 1$ aus Definition 2.2.12 erfüllt diese Annahmen nicht, wohl aber die Funktion $t(n) := n$.

Satz 2.3.3 (F-haltende Automaten sind unter Annahmen mindestens so mächtig wie t-haltende)**.** *Sei C ein t-haltender Zellularautomat. Wenn es einen F-haltenden Zellularautomaten X gibt mit $p_X = p_C$, $|\Sigma_X| = 1$ und* $\text{worst-time}_X = t_C$*, dann gibt es auch einen F-haltenden Zellularautomat C' mit $L_{C'} = L_C$, $p_{C'} = p_P$ und* $\text{worst-time}_{C'} = t_C$.

Beweis. Führe X und C parallel aus, indem $\delta_{C'}$ durch δ_X und δ_C auf $Q_{C'} := Q_X \times Q_C$ definiert wird.

Benutze X, um zu erkennen, wann eine Zelle in einen Final-Zustand gehen soll, indem $F_{C'} := \{(q_1, q_2) \mid (q_1, q_2) \in Q_X \times Q_C,\ q_1 \in F_X\}$ gewählt wird.

Benutze C, um zu erkennen, ob dieser Final-Zustand akzeptierend oder ablehnend sein soll, indem $F_{C'}^+ := \{(q_1, q_2) \mid (q_1, q_2) \in Q_X \times Q_C,\ q_1 \in F_X, q_2 \in F_C^+\}$ gewählt wird. Der so definierte Automat C' erfüllt die Behauptung. □

Folgende Proposition zeigt, dass die Annahmen aus dem vorherigen Satz notwendig sind.

Proposition 2.3.4 (t-haltende Automaten sind mächtiger als F-haltende)**.** *Es gibt einen t-haltenden links-erkennenden Zellularautomaten mit $t(n) \in \{0, 1\}$, welcher eine nicht-semientscheidbare Sprache erkennt. Diese Sprache kann offensichtlich nicht von einem F-haltenden Automaten erkannt werden, da sie sonst semientscheidbar wäre.*

Beweis. Sei $L \subseteq \Sigma^*$ eine nicht-semientscheidbare Sprache und $\text{cod} : \mathbb{N}_0 \to \Sigma^*$ eine berechenbare surjektive Funktion. Dann ist $L' := \{w \in \Sigma^* \mid \text{cod}(|w|) \in L\}$ auch nicht-semientscheidbar. Setze $t(n) := 1_{cod(n) \in L}$ und $p(n) := 1$. Konstruiere t-haltenden Zellularautomat mit $\Sigma \cap F^+ = \emptyset$ und $\delta(\cdot, \cdot, \cdot) \in F^+$. Dieser Automat erkennt dann offensichtlich L'. □

2.4 Nützliche Eigenschaften

Die folgenden zwei Sätze sind äußerst wichtig, um den Rand von Zellularautomaten in den Griff zu kriegen. Üblicherweise wird schon in der Definition von Zellularautomaten ein toter Rand gefordert, was einige elegante Konstruktionen verhindert. Es wird zunächst gezeigt, dass der Rand ohne Beschränkung der Allgemeinheit passiv und initial gewählt werden kann.

Satz 2.4.1 (Wahl eines passiven und initialen Randes)**.**
Sei $C = (Q, \delta, \Sigma, \#, L, (F, F^+, p)) \in CA^{FH}$. Dann kann C so zu C' mit $L(C) = L(C')$ umgebaut werden, dass $\#_{C'}$ ein passiver und initialer Zustand ist. Gleichzeitig bleibt dabei die Links- bzw. Rechts-Unabhängigkeit von δ_C erhalten.

Beweis. Setze $Q_{C'} := Q^2 \cup \{\#_{C'}\}$. Identifiziere Σ mit $\Sigma \times \{\#\}$ und $\#_{C'}$ mit $\begin{pmatrix}\perp\\ \perp\end{pmatrix}$, $\perp \notin Q$.

Für $q \in Q \cup \{\perp\}$ und $p \in Q$ definiere folgende abkürzende Notation $q^p \in Q$:

$$q^p := \begin{cases} q & \text{falls } q \neq \perp, \\ p & \text{sonst.} \end{cases}$$

Es sei $\delta_{C'}$ definiert durch:

$$\delta'(\begin{pmatrix}a_1\\a_2\end{pmatrix}, \begin{pmatrix}b_1\\b_2\end{pmatrix}, \begin{pmatrix}c_1\\c_2\end{pmatrix}) := \begin{cases} \begin{pmatrix}\delta(a_1^x, b_1^x, c_1^x)\\ \delta(x,x,x)\end{pmatrix} & \text{falls } \{x\} = \{a_2, b_2, c_2\} \setminus \{\perp\}, \\ \#_{C'} & \text{sonst.} \end{cases}$$

Falls δ links-unabhängig ist, streiche a_2 aus der Menge in obiger Definition. Falls δ rechts-unabhängig ist, streiche c_2.

Offensichtlich ist nach dieser Konstruktion $\#'_C$ sowohl passiv als auch initial.

Setze $t_r := t_l := t$. Falls δ links-unabhängig ist, setze jedoch $t_l := 0$. Falls δ rechts-unabhängig ist, setze stattdessen $t_r := 0$.

Es reicht nun, für alle $t \in \mathbb{N}_0$ und $i \in \mathbb{Z}$ Folgendes zu zeigen:

$$\Delta^t_{C'}([w])_i = \begin{cases} \begin{pmatrix}\Delta^t_C([w])_i\\ \Delta^t_C([\varepsilon])\end{pmatrix} & \text{falls } -t_r < i \leq t_l + |w|, \\ \#_{C'} & \text{sonst.} \end{cases}$$

Definiere $c^t_i := \Delta^t_{C'}([w])_{i-1}$. Die Behauptung gilt offensichtlich für $t = 0$. Die Behauptung gelte nun für ein $t \in \mathbb{N}_0$. Angenommen, $-(t+1)_r < i \leq (t+1)_l + |w|$. Dann folgt nach Induktionsvermutung und Definition von δ':

$$\{(c^t_{i-1})_2, (c^t_i)_2, (c^t_{i+1})_2\} \setminus \{\perp\} = \{\Delta^t_C([\varepsilon])\} =: \{x\}$$

Angenommen, $i - 1 \leq -t_r$. Dann folgt mit der Induktionsvermutung $((c^t_{i-1})_1)^x = \Delta^t_C([\varepsilon]) = \Delta^t_C([w])_{i-1}$. Ansonsten $((c^t_{i-1})_1)^x = \Delta^t_C([w])_{i-1}$.

Angenommen, $i+1 > t_l + |w|$. Dann $((c_{i+1}^t)_1)^x = \Delta_C^t([\varepsilon]) = \Delta_C^t([w])_{i+1}$. Ansonsten $((c_{i+1}^t)_1)^x = \Delta_C^t([w])_{i+1}$.

Analog gilt für $i = -t_r$, $i = t+1+|w|$ und alle anderen i im angenommenen Intervall $(c_i^t)^x = \Delta_C^t([w])_i$.

Folglich gilt $c_i^{t+1} = \begin{pmatrix} \Delta_C^{t+1}([w])_i \\ \Delta_C^{t+1}([\varepsilon]) \end{pmatrix}$.

Angenommen, $i \leq -(t+1)_r$. Dann folgt mit der Induktionsannahme $c_i^t = c_{i-1}^t = \#_{C'}$. Außerdem gilt $c_{i+1}^t = \#_{C'}$ oder δ ist rechts-unabhängig. Angenommen, $i > (t+1)_l + |w|$. Auch dann gilt nach Induktionsannahme $c_i^t = c_{i+1}^t = \#_{C'}$. Analog gilt dann $c_{i-1}^t = \#_{C'}$ oder δ ist links-unabhängig.

In allen Fällen gilt dann $c_i^{t+1} = \#_{C'}$. □

Für Linearzeit-Zellularautomaten kann der Rand sogar ohne Beschränkung der Allgemeinheit tot gewählt werden. Da ein toter Rand einen Speicherverbrauch nach sich zieht, der nur linear von der Wortlänge abhängt, folgt aus den Platzhierarchie-Sätzen für Turingmaschinen, dass der Rand für allgemeine Zellularautomaten nicht tot gewählt werden kann: Schließlich können sich Zellularautomaten und Turingmaschinen mit polynomiellen Mehraufwand für Zeit und Platz gegenseitig simulieren.

Satz 2.4.2 (Wahl eines toten Randes bei Linearzeit-Zellularautomaten)**.** *Sei $C \in CA^{L\text{-}LT}$ ein Linearzeit-Zellularautomat. Dann kann C ohne Veränderung der Laufzeit so zu C' mit $L(C) = L(C')$ umgebaut werden, dass $\#_{C'}$ ein toter Zustand ist. Für eine passive Zustandsmenge $M \subseteq \Sigma_C$ mit $\#_C \in M$ ist die entsprechende Teilmenge von $\Sigma_{C'}$ auch passiv.*

Beweis. Da C ein Linearzeit-Zellularautomat ist, gibt es ein $k \in \mathbb{N}_0$, sodass $\forall n \in \mathbb{N}_0 : \text{worst-time}_C(n) \leq kn$. Setze $K := \{-k, ..., k\}$ und $Q' := Q_C^K \cup \{\#\}$. Die Elemente von $f \in Q_C^K$ können auf $\mathbb{Z}$ durch $f(n \in \mathbb{Z} \setminus K) := \#_C$ fortgesetzt werden.

Für $i \in K$, $q \in Q'$ und $c \in Q_C$ definiere folgende abkürzende Notation:

$$q_i^c := \left\{ \begin{array}{ll} c & \text{falls } q = \#, \\ q(i) & \text{sonst.} \end{array} \right\} \in Q_C$$

Definiere δ' für $a, c \in Q'$ und $b \in Q_C^K$ durch

$$\begin{aligned} \delta'(a, \#, c) &:= \# \\ (\delta'(a, b, c))(i \in K) &:= \begin{cases} \delta(a_i^{b(i-1)}, b(i), c_i^{b(i+1)}) & \text{falls } i \text{ gerade ist,} \\ \delta(c_i^{b(i-1)}, b(i), a_i^{b(i+1)}) & \text{sonst.} \end{cases} \end{aligned}$$

Bette $c \in \Sigma$ durch $\phi(c)$ wie folgt in Q' ein:

$$(\phi(c))(i \in K) := \begin{cases} c & \text{falls } i = 0, \\ \#_C & \text{sonst.} \end{cases}$$

Setze $F' := \{q \in Q_C^K \mid q(0) \in F\}$ und $F'^+ := \{q \in Q_C^K \mid q(0) \in F^+\}$.

Es bleibt nun die Korrektheit dieser Konstruktion zu zeigen. Definiere dazu die Funktion $f_n : \mathbb{Z} \to \{1, ..., n\}$ für $n \in \mathbb{N}$:

$$f_n(i) := \begin{cases} ((i-1) \bmod n) + 1 & \text{falls } (i \bmod 2n) \leq n, \\ n - ((i-1) \bmod n) & \text{sonst.} \end{cases}$$

Folgende Tabelle zeigt einige Werte von f_4:

i	-1	0	1	2	3	4	5	6	7	8	9	10	11	12
$f_4(i)$	2	1	1	2	3	4	4	3	2	1	1	2	3	4

f_n hat einen steigenden und einen fallenden Teil, die Funktion pendelt zwischen 1 und n und es gilt stets $|f_n(i) - f_n(i+1)| \leq 1$.

Für die Korrektheit reicht es nun zu zeigen, dass für $w \in \Sigma^*$, $t \in \{0, ..., k|w|\}$ und $p \in \{-k|w| + t, ..., k|w| - t\}$ gilt:

$$\Delta_C^t([w])_p = (\Delta_{C'}^t([w])_{f_{|w|}(p)})(\lfloor \frac{p-1}{|w|} \rfloor)$$

Diese Aussage kann leicht induktiv gezeigt werden. Dabei müssen die Fälle betrachtet werden, dass $f_{|w|}(p+1)$ bzw. $f_{|w|}(p+1)$ kleiner, größer oder gleich $f_{|w|}(p)$ sein kann. Abhängig davon befindet sich die Zelle mit Position p dann direkt neben dem Rand $\#_{C'}$ bzw. entscheidet sich die Parität von $\lfloor \frac{p-1}{|w|} \rfloor$.

□

Der folgende praktische Satz besagt, dass es auf endlich viele Ausnahmen in Sprachen nicht ankommt. Insbesondere kann damit der Fall ignoriert werden, dass Sprachen das leere Wort enhalten können.

Satz 2.4.3. *Sei C ein Zellularautomat, $L \subseteq \Sigma^*$ eine Sprache und $M \subseteq \Sigma^*$, $|M| \leq \infty$ eine endliche Menge von Wörtern, sodass $L_C \setminus M = L \setminus M$. Dann existiert ein Zellularautomat C', der L in der selben Zeit erkennt.*

Beweis. Offensichtlich sind dann $M_1 := L \cap M$ und $M_2 := (\Sigma^* \setminus L) \cap M$ auch endlich. M_1 und M_2 sind disjunkt und ihre Vereinigung ist genau M. M_1 enthält die Wörter, die L hat, aber L_C möglicherweise nicht, während M_2 die Wörter enthält, die L nicht hat, L_C aber möglicherweise schon. Da endliche Sprachen regulär sind und reguläre Sprachen von Echtzeit-Zellularautomaten erkannt werden können, existieren Zellularautomaten C_1 und C_2 mit $L_{C_1} = M_1$ und $L_{C_2} = M_2$. Akzeptiert weder C_1 noch C_2 ein Wort w, dann akzeptiert es C genau dann, wenn $w \in L$. Werden also die Automaten C, C_1 und C_2 parallel ausgeführt, kann leicht ein Echtzeit-Zellularautomat C' konstruiert werden mit $L_{C'} = L$. □

3. Links-unabhängige Zellularautomaten

Zellen in links-unabhängigen Zellularautomaten hängen in ihrer Berechnung nicht von ihrem linken Nachbarn ab. Informationen können sich also nur nach links verbreiten. Diese Einschränkung vereinfacht bestimmte Konstruktionen, was in Kapitel 4 ausgenutzt wird. In diesem Kapitel werden Zusammenhänge zwischen unbeschränkten und links-unabhängigen Zellularautomaten gezeigt. Ähnliche Zusammenhänge werden in [ChCu84] gezeigt.

Es ist allgemein bekannt, dass $\mathcal{L}(\mathrm{CA}^{\text{RT-L-OCA}_L})$ eine strikte Teilmenge von $\mathcal{L}(\mathrm{CA}^{\text{RT-L}})$ ist und dass $\mathcal{L}(\mathrm{CA}^{\text{LT-L-OCA}_L})$ mit $\mathcal{L}(\mathrm{CA}^{\text{RT-R}})$ zusammenfällt (siehe [Kutr09]).

Der erste Satz zeigt, dass unbeschränkte Zellularautomaten zu einem links-unabhängigen Automaten umgebaut werden können. Dabei verschiebt sich die Berechnung räumlich nach links und wird um den Faktor 2 langsamer.

Satz 3.0.1. *Sei $C = (Q, \delta) \in CA^*$ ein Zellularautomat (wie in Abbildung 3.1). Dann gibt es einen links-unabhängigen Zellularautomaten $C' = (Q' \supseteq Q, \delta')$ (wie in Abbildung 3.2), sodass für $\forall c \in Q^{\mathbb{Z}}, \forall t \in \mathbb{N}_0, \forall i \in \mathbb{Z}$ gilt:*

$$\Delta^t_{C'}(c)_i = \begin{cases} \Delta^{\frac{t}{2}}_C(c)_{i+\frac{t}{2}} & \text{falls } t \text{ gerade,} \\ (\Delta^{\frac{t-1}{2}}_C(c)_{i+\frac{t-1}{2}}, \Delta^{\frac{t-1}{2}}_C(c)_{i+\frac{t+1}{2}}) & \text{sonst.} \end{cases}$$

Beweis. Führe für diesen Beweis folgende Notation ein: ${c'}^t_i := \Delta^t_{C'}(c)_i$ und $c^t_i := \Delta^t_C(c)_i$.

Setze $Q' := Q \cup Q^2$. Definiere δ' für $b, c \in Q$ wie folgt:

$$\delta'(\cdot, b, c) := (b, c) \in Q'$$

und für $b, c \in Q \times Q$:

$$\delta'(\cdot, (b_1, b_2), (c_1, c_2)) := \delta(b_1, b_2, c_2)$$

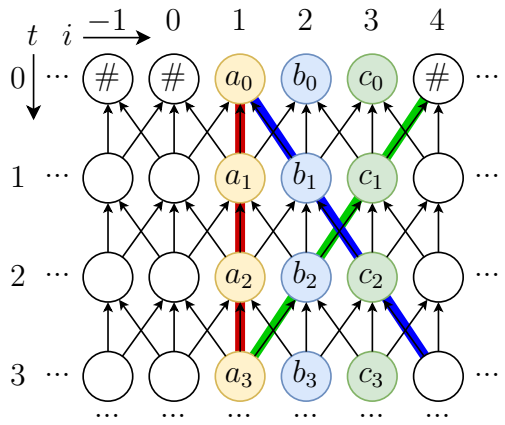

Abbildung 3.1: Teilweise Ausführung des Automaten C

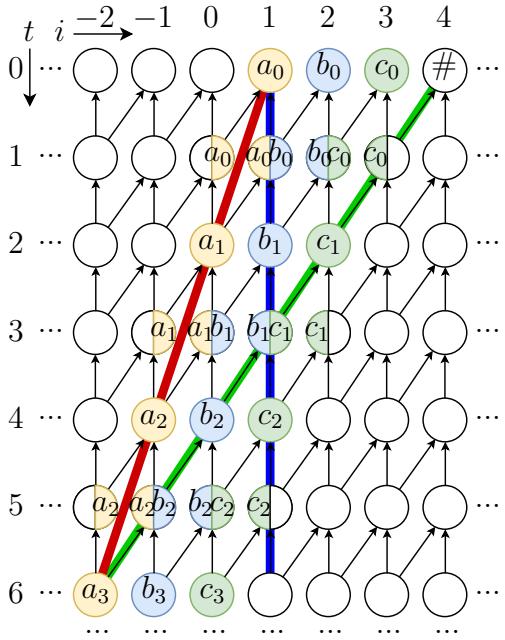

Abbildung 3.2: Teilweise Ausführung des links-unabhängigen Automaten C'

Definiere $\delta'(\cdot, b, c)$ beliebig für alle anderen Fälle. Offensichtlich ist δ' dann links-unabhängig.

Sei $c \in Q^{\mathbb{Z}}$. Beweis der Behauptung durch Induktion über t. Für $t = 0$ trivial.

Sei nun $t > 0, i \in \mathbb{Z}$.

1. Fall: t ist ungerade. Dann gilt nach Induktionsannahme

$$\forall j \in \mathbb{Z} : {c'}_j^{t-1} = c_{j+\frac{t-1}{2}}^{\frac{t-1}{2}} \in Q$$

Und damit

$$\begin{aligned} & {c'}_i^t \\ &= \delta'({c'}_{i-1}^{t-1}, {c'}_i^{t-1}, {c'}_{i+1}^{t-1}) \\ &= ({c'}_i^{t-1}, {c'}_{i+1}^{t-1})) \\ &= (c_{i+\frac{t-1}{2}}^{\frac{t-1}{2}}, c_{i+\frac{t+1}{2}}^{\frac{t-1}{2}}) \end{aligned}$$

2. Fall: t ist gerade. Damit $t \geq 2$. Dann gilt nach Induktionsannahme

$$\forall j \in \mathbb{Z} : {c'}_j^{t-1} = (c_{j+\frac{t-2}{2}}^{\frac{t-2}{2}}, c_{j+\frac{t}{2}}^{\frac{t-2}{2}}) \in Q \times Q$$

Und damit

$$\begin{aligned}
&{c'}_i^t \\
&= \delta'({c'}_{i-1}^{t-1}, {c'}_i^{t-1}, {c'}_{i+1}^{t-1}) \\
&= \delta'(q, (c_{i+\frac{t-2}{2}}^{\frac{t-2}{2}}, c_{i+\frac{t}{2}}^{\frac{t-2}{2}}), (c_{i+1+\frac{t-2}{2}}^{\frac{t-2}{2}}, c_{i+1+\frac{t}{2}}^{\frac{t-2}{2}})) \\
&= \delta(c_{i+\frac{t-2}{2}}^{\frac{t-2}{2}}, c_{i+\frac{t}{2}}^{\frac{t-2}{2}}, c_{i+1+\frac{t}{2}}^{\frac{t-2}{2}}) \\
&= c_{i+\frac{t}{2}}^{\frac{t}{2}}
\end{aligned}$$

□

Der nächste Satz zeigt die andere Richtung: Links-unabhängige Zellularautomaten können wieder zu einem Automaten ohne Einschränkung umgebaut werden. Das allein überrascht nicht, schließlich sind links-unabhängige Automaten eine Spezialisierung der Zellularautomaten ohne Einschränkung. Interessant ist jedoch, dass sich dabei die Berechnung nach rechts verschieben und um den Faktor 2 beschleunigen lässt.

Satz 3.0.2. *Sei $C = (Q, \delta) \in CA^*$ ein links-unabhängiger Zellularautomat (wie in Abbildung 3.3 gezeigt). Dann gibt es einen Zellularautomaten $C' = (Q' \supseteq Q, \delta')$ (wie in Abbildung 3.4 gezeigt), sodass für $\forall c \in Q^{\mathbb{Z}}$, $\forall t \in \mathbb{N}_0$, $\forall i \in \mathbb{Z}$ gilt:*

$$\Delta_{C'}^t(c)_i = \Delta_C^{2t}(c)_{i-t}$$

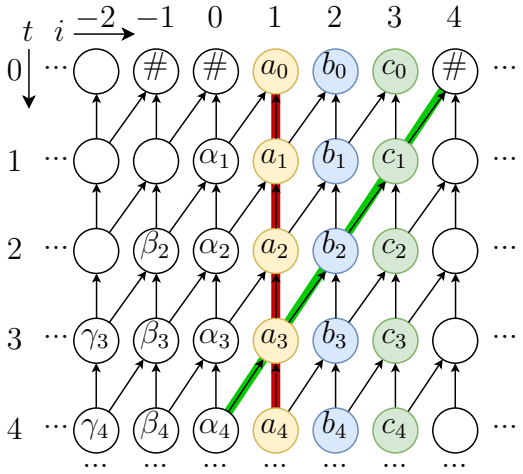

Abbildung 3.3: Teilweise Ausführung des links-unabhängigen Automaten C

Beweis. Führe für diesen Beweis folgende Notation ein: ${c'}_i^t := \Delta_{C'}^t(c)_i$ und $c_i^t := \Delta_C^t(c)_i$.

Da Q nicht leer ist, gibt es ein $q \in Q$. Setze $Q' := Q$ und für $a, b, c \in Q'$:

$$\delta'(a, b, c) := \delta(q, \delta(q, a, b), \delta(q, b, c))$$

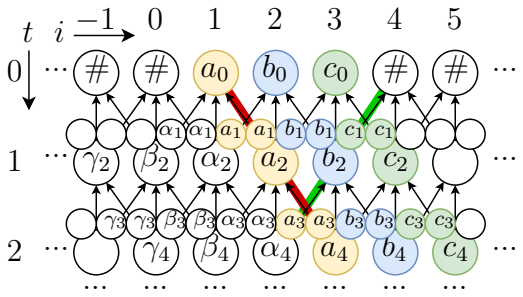

Abbildung 3.4: Teilweise Ausführung des Automaten C'

Sei $c \in Q^{\mathbb{Z}}$. Beweis der Behauptung durch Induktion über t. Für $t = 0$ trivial. Sei $t > 0$, $i \in \mathbb{Z}$.

Nach Induktionsannahme gilt $\forall j \in \mathbb{Z} : {c'}^{t-1}_{j+t-1} = c^{2t-2}_j$.

Es folgt:

$$
\begin{aligned}
&{c'}^{t}_{j+t} \\
&= \delta'({c'}^{t-1}_{j+t-1}, {c'}^{t-1}_{j+t}, {c'}^{t-1}_{j+t+1}) \\
&= \delta'(c^{2t-2}_j, c^{2t-2}_{j+1}, c^{2t-2}_{j+2}) \\
&= \delta(q, \delta(q, c^{2t-2}_j, c^{2t-2}_{j+1}), \delta(q, c^{2t-2}_{j+1}, c^{2t-2}_{j+2})) \\
&= \delta(q, c^{2t-1}_j, c^{2t-1}_{j+1}) \\
&= c^{2t}_j
\end{aligned}
$$

□

Mit beiden Sätzen zusammen kann ein Automat zunächst zu einem links-unabhängigen Automaten umgebaut, dann transformiert und wieder zurück umgebaut werden. Dies wird in Satz 4.2.2 aus dem nächsten Kapitel ausgenutzt.

4. Speedup-Konstruktionen

Bei der Fragestellung, wie sich Linearzeit-Zellularautomaten zu Echtzeit-Zellularautomaten verhalten, ist es essentiell, zu untersuchen, inwiefern sich Zellularautomaten beschleunigen lassen. Sogenannte Speedup-Sätze für Zellularautomaten sind schon einige bekannt (siehe [Kutr09, MaRe92]). Um dem Formalismus dieser Arbeit gerecht zu werden, werden einige dieser Sätze in diesem Kapitel neu bewiesen bzw. verallgemeinert. In Kapitel 6 und Kapitel 7 werden die hier vorgestellten Speedup-Konstruktionen wieder aufgegriffen.

4.1 Speedup um konstant viele Schritte

Ein zentraler Speedup-Satz zeigt, dass Zellularautomaten um einen Schritt beschleunigt werden können, wenn Annahmen über den Rand getroffen werden können. Satz 2.4.2 zeigt, dass wir diese Annahmen treffen dürfen. Eine Besonderheit der hier vorgestellten Konstruktion ist, dass sie in einem gewissen Rahmen passive Zustände erhält. Das wird in Kapitel 7 wieder aufgegriffen.

Definition 4.1.1 (1-Schritt-Speedup-Konstruktion)**.** Sei $(Q, \delta) \in \mathrm{CA}^*$ ein Zellularautomat. Definiere den Zellularautomaten $Sp(C) := (Q', \delta')$ mit $Q' := Q \times (Q^Q \cup \{\bot\})$. Identifiziere Q mit $Q \times \{\bot\}$. Für ein Element $q \in Q'$ bezeichnen $q_q \in Q$ und $q_f \in Q^Q \cup \{\bot\}$ die erste bzw. zweite Komponente, sodass gilt: $q = (q_q, q_f)$. Die Zustandsübergangsfunktion δ' sei wie folgt definiert:

$$\delta'((a_q, a_f), (b_q, b_f), (c_q, c_f)) := \begin{cases} (\delta(a_q, b_q, c_q), \bot) & \text{falls } c_f = \bot \\ \phi_{c_f(b_q)}(\delta(a_q, b_q, c_q)) & \text{sonst} \end{cases}$$

Definiere weiter $\phi_c : Q \to Q'$ für ein $c \in Q$:

$$\phi_c(b) := (b,\ Q \ni a \mapsto \delta(a, b, c) \in Q)$$

Satz 4.1.2 (Korrektheit der 1-Schritt-Speedup-Konstruktion)**.** *Sei $C = (Q, \delta) \in CA^*$ ein Zellularautomat, $\# \in Q$ ein ausgezeichneter Zustand und $c \in Q^{\mathbb{Z}}$ eine Konfiguration. Setze $C' := Sp(C)$. Sei $c' \in {Q_{C'}}^{\mathbb{Z}}$ eine Konfiguration mit $c'_j \in \{c_j, \phi_\#(c_j)\}$ für alle $j \in \mathbb{Z}$. Sei $t \in \mathbb{N}_0$ und $i \in \mathbb{Z}$.*

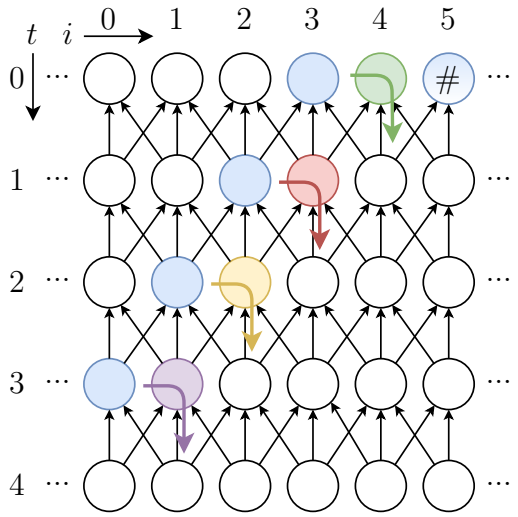

Abbildung 4.1: 1-Schritt Speedup
Für einen Zustand q beschreiben die eingezeichneten Pfeile die Abbildung q_f: Der jeweils linke Zustand wird auf den unteren abgebildet, sofern der Zustand rechts zum Zeitpunkt 0 bekannt ist.

(I) *Falls* $c'_{t+i} \in Q \times Q^Q$, *dann* $\Delta^t_{C'}(c')_i \in Q \times Q^Q$.

(II) *Es gilt:* $(\Delta^t_{C'}(c')_i)_q = \Delta^t_C(c)_i$.

(III) *Falls* $\Delta^t_{C'}(c')_i \in Q \times Q^Q$ *und* $c_{i+t+1} = \#$, *dann gilt, wie in Abbildung 4.1 veranschaulicht:*

$$(\Delta^t_{C'}(c')_i)_f((\Delta^t_{C'}(c')_{i-1})_q) = \Delta^{t+1}_C(c)_i$$

Beweis. Führe für diesen Beweis folgende Notation ein: $c'^t_i := \Delta^t_{C'}(c')_i$ und $c^t_i := \Delta^t_C(c)_i$.

(I) Die Aussage gilt offensichtlich für $t = 0$. Angenommen, die Aussage gilt nun für ein $t \in \mathbb{N}_0$ und alle $i \in \mathbb{Z}$. Sei $c'_{t+1+i} \in Q \times Q^Q$. Dann gilt nach Definition von $\delta_{C'}$ und wegen $c'^t_{i+1} \in Q \times Q^Q$:

$$c'^{t+1}_i = \delta_{C'}(c'^t_{i-1}, c'^t_i, c'^t_{i+1}) \in Q \times Q^Q$$

Damit gilt die Aussage auch für $t + 1$.

(II) Gilt offensichtlich.

(III) Beweis durch Induktion über t. Für $t = 0$ gilt mit $c'^0_i \in Q \times Q^Q$ und $c_{i+1} = \#$:

$$\begin{aligned} & (c'^0_i)_f((c'^0_{i-1})_q) \\ = \; & (c'_i)_f((c'_{i-1})_q) \\ = \; & (\phi_s(c_i))_f(c_{i-1}) \\ = \; & \delta(c_{i-1}, c_i, c_{i+1}) = c^1_i \end{aligned}$$

Es gelte nun die Behauptung für ein $t \in \mathbb{N}_0$ und alle $i \in \mathbb{Z}$. Sei $c'^{t+1}_i \in Q \times Q^Q$ und $c_{i+(t+1)+1} = \#$.

Nach Definition von $\delta_{C'}$ gilt $(c'^{t}_{i+1})_f \in Q^Q$. Wegen $c_{(i+1)+t+1} = \#$ gilt nach Induktionsvoraussetzung:

$$(c'^{t}_{i+1})_f((c'^{t}_{i})_q) = c^{t+1}_{i+1}$$

Es gilt:

$$\begin{aligned}
(c'^{t+1}_{i})_f &= \delta_{C'}(c'^{t}_{i-1}, c'^{t}_{i}, c'^{t}_{i+1})_f \\
= \quad & \phi_{(c'^{t}_{i+1})_f((c'^{t}_{i})_q)}(\delta((c'^{t}_{i-1})_q, (c'^{t}_{i})_q, (c'^{t}_{i+1})_q))_f \\
= \quad & \phi_{c^{t+1}_{i+1}}(c^{t+1}_{i})_f \\
= \quad & Q \ni a \mapsto \delta(a, c^{t+1}_{i}, c^{t+1}_{i+1}) \in Q
\end{aligned}$$

Und damit:

$$\begin{aligned}
& (c'^{t+1}_{i})_f((c'^{t+1}_{i-1})_q) \\
= \quad & \delta(c^{t+1}_{i-1}, c^{t+1}_{i}, c^{t+1}_{i+1}) \\
= \quad & c^{t+2}_{i}
\end{aligned}$$

□

Rekursiv lässt sich mithilfe des gezeigten Satzes ein Zellularautomat um konstant viele Schritte beschleunigen. Wird Satz 2.4.2 ausgenutzt, lässt sich die folgende Aussage zeigen.

Satz 4.1.3 (k-Schritt Speedup)**.** *Es gilt:*

$$\mathcal{L}(CA^{RT\text{-}L}) = \mathcal{L}(CA^{TH\text{-}L\text{-time}(\{n\mapsto\max\{0,n+k-1\}\mid k\in\mathbb{N}_0\})})$$

Beweis. Es reicht zu zeigen, dass für alle $k \in \mathbb{N}_0$ gilt:

$$\mathcal{L}(\mathrm{CA}^{\text{TH-L-time}(n\mapsto\max\{0,n+k-1\})}) \supseteq \mathcal{L}(\mathrm{CA}^{\text{TH-L-time}(n\mapsto n+k)})$$

Induktiv gilt dann die Behauptung.

Sei $L \in \mathcal{L}(\mathrm{CA}^{\text{TH-L-time}(n\mapsto n+k)})$. Dann existiert ein t-haltender Zellularautomat C, sodass $\Delta_C^{|w|+k}([w])_1 \in F_C^+ \Leftrightarrow w \in L$. Wegen Satz 2.4.2 kann angenommen werden, dass $\#_C$ ein toter Zustand ist.

Setze $C' := Sp(C)$. Identifiziere $c \in \Sigma_C$ mit $\phi_{\#_C}(c)$ in $\Sigma_{C'}$ und setze $\#_{C'} := \phi_{\#_C}(\#_C)$. Definiere $F^+_{C'}$ wie folgt:

$$F^+_{C'} := \{q \in Q_C \times Q_C^{Q_C} \mid q_f(\#_C) \in F_C^+\}$$

Es kann angenommen werden, dass $|w| + k - 1 \geq 0$, da mit Satz 2.4.3 das leere Wort ignoriert werden kann. Da $[w]_{|w|+k} \in Q_C \times Q_C^{Q_C}$, $[w]_{|w|+k+1} = \#_C$ und $\Delta_C^{|w|+k-1}([w])_0 = \#_C$, folgt mit Satz 4.1.2 $\Delta_{C'}^{|w|+k-1}([w])_1 \in F'^{+}_{C} \Leftrightarrow \Delta_C^{|w|+k}([w])_1 \in F_C^+ \Leftrightarrow w \in L$. Also $L \in \mathcal{L}(\mathrm{CA}^{\text{TH-L-time}(n\mapsto\max\{0,n+k-1\})})$. □

Mithilfe der Speedup-Sätze kann beispielsweise nun einfach gezeigt werden, dass Wortlängen einer $\mathrm{CA}^{\mathrm{RT\text{-}L}}$-Sprache OBdA. Vielfache einer beliebigen Zahl $k \in \mathbb{N}$ sind:

Satz 4.1.4. *Sei $L \subseteq \Sigma^*$ eine Sprache, $x \notin \Sigma$, $k \in \mathbb{N}$. Definiere zu L:*

$$L(k) := \{wx^{\min\{i \in \mathbb{N}_0 \mid (|w|+i) \in k\mathbb{N}_0\}} \mid w \in L\}$$

Es gilt: $L(k) \in \mathcal{L}(CA^{RT\text{-}L})$ genau dann, wenn $L \in \mathcal{L}(CA^{RT\text{-}L})$.

Beweis. Angenommen $L(k) \in \mathcal{L}(\mathrm{CA}^{\mathrm{RT\text{-}L}})$. Dann existiert ein Echtzeit-Automat C' mit $L_{C'} = L(k)$. Da bei links-erkennenden Echtzeit-Zellularautomaten das Akzeptanzverhalten unabhängig vom Verhalten des rechten Rands ist, kann der rechte Rand mit x identifiziert werden (OBdA. kann der rechte Rand vom linken unterschieden werden). Dieser Automat erkennt nun L in t Schritten mit $t \in \{n-1, ..., n-1+k-1\}$. Setze C'_i auf den i-Schritt-Speedup-Automaten von C' und konstruiere den Automaten C, der die endlich vielen Automaten C'_0, C'_1 bis C'_{k-1} parallel ausführt und genau dann akzeptiert, wenn einer von ihnen akzeptiert. Damit erkennt C die Sprache L in Echtzeit.

Umgekehrt sei nun $L \in \mathcal{L}(\mathrm{CA}^{\mathrm{RT\text{-}L}})$. Dann existiert ein Echtzeit-Automat C mit $L_C = L$. Konstruiere den Automat C' wie folgt: C' löscht in genau $k-1$ Schritten die Zellen am rechten Rand im Zustand x. Sind es mehr als $k-1$ solcher Zellen, lehne das Wort ab, bei weniger wird gewartet, bis die $k-1$ Schritte vorbei sind. Anschließend wird auf dieser neuen Konfiguration nun der Automat C ausgeführt. Parallel wird durch einen Automaten X getestet, ob die Länge des Wortes ein Vielfaches von m ist. Das Eingabewort soll nun genau dann von C' akzeptiert werden, falls C und X akzeptieren. Dieser Automat C' akzeptiert nun $L(k)$ in $n-1+k-1$ vielen Schritten. Da k fest ist, garantiert Satz 4.1.3 nun die Existenz eines $L(k)$-erkennenden Echtzeitautomaten. □

4.2 Speedup von Echtzeit-Zellularautomaten

Im vorherigen Abschnitt wurde gezeigt, dass Zellularautomaten, die nur eine konstante Anzahl Schritte von Echtzeit entfernt sind, auf Echtzeit beschleunigt werden können. In diesem Kapitel wird gezeigt, dass sich Echtzeit-Zellularautomaten in gewisser Hinsicht auch beschleunigen lassen. Offensichtlich ist die Zeiteinschränkung von Echtzeit-Zellularautomaten bereits minimal: Wenn ein Wort w in weniger als $|w| - 1$ Schritten erkannt wird, kann das letzte Zeichen des Eingabewortes nicht zur Akzeptanz beitragen. Allerdings lässt sich die Anzahl benötigter Schritte tatsächlich weiter verringern, wenn dafür die Position der Zelle, die Akzeptanz oder Ablehnung signalisiert, in Richtung Mitte wandert.

Zunächst wird aber das folgende Lemma benötigt.

Lemma 4.2.1. *Sei $C \in CA^*$ ein links-unabhängiger Zellularautomat (wie in Abbildung 4.2) und $2 \leq k \in \mathbb{N}$. Sei ferner $\# \in Q_C$ ein passiver und initialer Zustand. Dann gibt es einen links-unabhängigen Zellularautomaten $C' \in CA^*$, sodass für $i \in \mathbb{Z}$ mit $i \leq 0$ und $t \in \mathbb{N}_0$ und alle Konfigurationen c mit $c_p = \#$ für $p \leq 0$ oder $p \geq p_0$ gilt:*

$$\Delta^t_{C'}(c)_i = w \in Q^k \text{ mit } w_j := \Delta^{t+i-ki+k-j}_C(c)_{ki-k+j}$$

Für $k = 2$ erhält man, wie in Abbildung 4.3 gezeigt:

$$\Delta^t_{C'}(c)_i = (\Delta^{t-i+1}_C(c)_{2i-1}, \Delta^{t-i}_C(c)_{2i})$$

Für $k = 3$ ergibt sich:

$$\Delta^t_{C'}(c)_i = (\Delta^{t-2i+2}_C(c)_{3i-2}, \Delta^{t-2i+1}_C(c)_{3i-1}, \Delta^{t-2i}_C(c)_{3i})$$

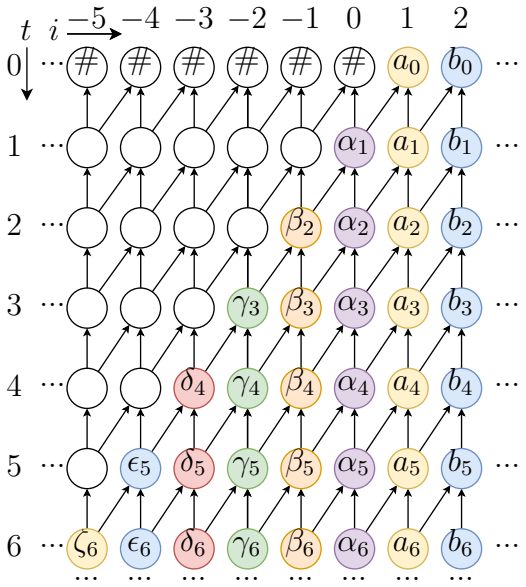

Abbildung 4.2: Teilweise Ausführung des Automaten C

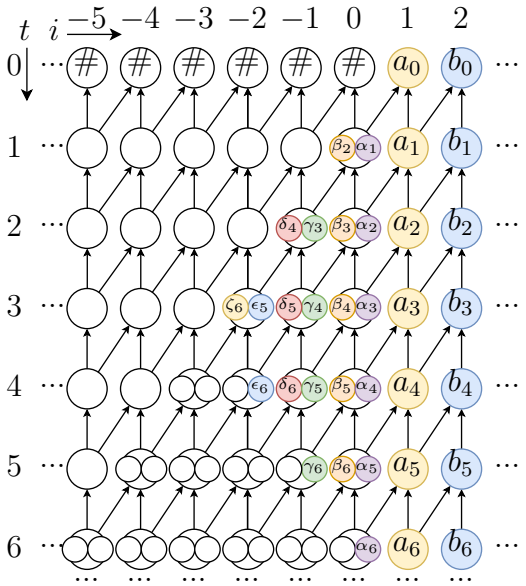

Abbildung 4.3: Teilweise Ausführung des Automaten C' für $k = 2$

Beweis. Definiere $\delta^w \in Q^{|w-1|}$ für $w \in Q^*$ mit $|w| \geq 2$ wie folgt:

$$\delta^w_i := \begin{cases} \delta(\#, w_{|w|-1}, w_{|w|}) & \text{falls } i = |w| - 1 \\ \delta(\#, w_i, \delta^w_{i+1}) & \text{sonst} \end{cases}$$

Beispielsweise gilt $\delta^{q_1q_2q_3} = \delta(\#, q_1, \delta(\#, q_2, q_3))\delta(\#, q_2, q_3)$.

Setze $Q' := Q_C \cup Q_C^k$. Identifiziere $\#$ mit $\#^k$. Definiere δ' wie folgt:

$$\begin{aligned}
\delta'(\cdot, q_b \in Q_C \setminus \{\#\}, q_c \in Q_C) &:= \delta(\#, q_b, q_c) \\
\delta'(\cdot, w_b \in Q_C^k, \quad q_c \in Q_C) &:= \delta^{w_b q_c} \\
\delta'(\cdot, x_b \in Q', \quad w_c \in Q_C^k) &:= \delta'(\#, x_b, (w_c)_1)
\end{aligned}$$

Da $\#$ ein initialer Zustand in C ist und auch in δ' durch die Identifikation mit $\#^k$ passiv bleibt, verhält sich δ' für Zellen mit positiver Position genauso wie δ.

Definiere nun die Hilfsfunktionen ψ und ϕ:

$$\begin{aligned}
\psi(i,j) &:= ki + j - k \\
\phi(t,i,j) &:= t + i - \psi(i,j) \\
c_i^t &:= \Delta_i^t(c)
\end{aligned}$$

Für ϕ und ψ gelten für $j \in \{1, ..., k\}$ folgende triviale Fakten, die im Folgenden verwendet werden:

$$\begin{aligned}
\psi(i,j) &\leq 0 \\
\phi(0,i,j) &\leq -\psi(i,j) \\
\psi(1,1) &= 1 \\
\psi(i+1,1) &= \psi(i,k) + 1 \\
\psi(i,j) &= \psi(i,j-1) + 1 \\
\phi(t,1,1) &= t \\
\phi(t,i+1,1) &= \phi(t,i,k) \\
\phi(t+1,i,j) &= \phi(t,i,j-1)
\end{aligned}$$

Da im Beweis nur diese erwähnten Fakten über ψ und ϕ verwendet werden, ist es nicht verwunderlich, dass diese Fakten ψ und ϕ für $i \leq 0$ und $t \geq 0$ eindeutig bestimmen.

Für die Wahl $C' := (Q', \delta')$ ist nun zu für alle $i \leq 0$ und $t \in \mathbb{N}_0$ zeigen:

$$(\Delta_{C'}^t(c)_i)_j = c_{\psi(i,j)}^{\phi(t,i,j)}$$

Beweis der Aussage über Induktion nach $t \in \mathbb{N}_0$.

Sei $t = 0$. Da $\#$ passiv ist, gilt $c_{i'}^{t'} = \#$ für $0 \leq t' \leq -i'$ und $i \leq 0$. Wegen $\phi(0,i,j) \leq -\psi(i,j)$ und $\psi(i,j) \leq 0$ folgt $(\Delta_{C'}^0(c)_i)_j = \# = c_{\psi(i,j)}^{\phi(0,i,j)}$ für alle $j \in \{1, ..., k\}$.

Die Aussage gelte nun für ein $t \in \mathbb{N}_0$.

Sei $x_a := \Delta_{C'}^t(c)_i$ und $x_b := \Delta_{C'}^t(c)_{i+1}$. Wegen $i \leq 0$ folgt $x_a \in Q_C^k$.

Nach Induktionsvermutung gilt $(x_a)_j = c_{\psi(i,j)}^{\phi(t,i,j)}$.

Wähle $q := \begin{cases} x_b & \text{falls } x_b \in Q_C \\ (x_b)_1 & \text{falls } x_b \in Q_C^k \end{cases}$.

Für diese Wahl von q gilt nun $q = c_{\psi(i+1,1)}^{\phi(t,i+1,1)}$:

Angenommen, $x_b \in Q_C$. Nach Definition von δ' folgt $i = 0$ und $q = x_b = c_1^t = c_{\psi(1,1)}^{\phi(t,1,1)} = c_{\psi(i+1,1)}^{\phi(t,i+1,1)}$.

Andernfalls gilt $x_b \in Q_C^k$ und damit $i < 0$. Nach Induktionsvermutung gilt dann $q = (x_b)_1 = c_{\psi(i+1,1)}^{\phi(t,i+1,1)}$.

Zeige $(\Delta_{C'}^{t+1}(c)_i)_j = c_{\psi(i,j)}^{\phi(t+1,i,j)}$ durch endliche, absteigende Induktion über j.

Für $j = k$ gilt:

$$\begin{aligned}
&(\Delta_{C'}^{t+1}(c)_i)_k \\
&= \delta'(\#, x_a, x_b)_k \\
&= \delta_k^{x_a q} \\
&= \delta(\#, (x_a)_k, q) \\
&= \delta(\#, c_{\psi(i,k)}^{\phi(t,i,k)}, c_{\psi(i+1,1)}^{\phi(t,i+1,1)}) \\
&= \delta(\#, c_{\psi(i,k)}^{\phi(t,i,k)}, c_{\psi(i,k)+1}^{\phi(t,i,k)}) \\
&= c_{\psi(i,k)}^{\phi(t+1,i,k)}
\end{aligned}$$

Es gelte die Induktionsvermutung nun für $1 < j \leq k$. Dann gilt:

$$\delta_j^{x_a q} = (\Delta_{C'}^{t+1}(c)_i)_j = c_{\psi(i,j)}^{\phi(t+1,i,j)}$$

Zeige nun die Vermutung für $j - 1$:

$$\begin{aligned}
&(\Delta_{C'}^{t+1}(c)_i)_{j-1} \\
&= \delta'(\#, x_a, x_b)_{j-1} \\
&= \delta_{j-1}^{x_a q} \\
&= \delta(\#, (x_a)_{j-1}, \delta_j^{(x_a q)}) \\
&= \delta(\#, c_{\psi(i,j-1)}^{\phi(t,i,j-1)}, c_{\psi(i,j)}^{\phi(t+1,i,j)}) \\
&= \delta(\#, c_{\psi(i,j-1)}^{\phi(t,i,j-1)}, c_{\psi(i,j-1)+1}^{\phi(t,i,j-1)}) \\
&= c_{\psi(i,j-1)}^{\phi(t+1,i,j-1)}
\end{aligned}$$

Damit ist auch der Induktionsschritt gezeigt und der Beweis vollendet. □

Damit lässt sich dann der folgende Satz zeigen.

Satz 4.2.2. *Sei $C \in CA^*$ ein Zellularautomat. Dann gibt es einen Zellularautomaten $C''' \in CA^*$ und Funktionen g_1 und g_2, sodass für alle $p \in \mathbb{N}$ gilt:*

$$g_1(\Delta_{C'''}^{2p-1}(c)_p) = \Delta_C^{3p-2}(c)_1$$
$$g_2(\Delta_{C'''}^{2p}(c)_{p+1}) = (\Delta_C^{3p-1}(c)_1, \Delta_C^{3p}(c)_1)$$

Ferner existiert dann ein Automat C_1 und eine Funktion f, sodass für $i \geq 1$ gilt:

$$f(\Delta_{C_1}^{2i+1}(c)_i) = (\Delta_C^{3i-3}(c)_1, \Delta_C^{3i-2}(c)_1, \Delta_C^{3i-1}(c)_1)$$

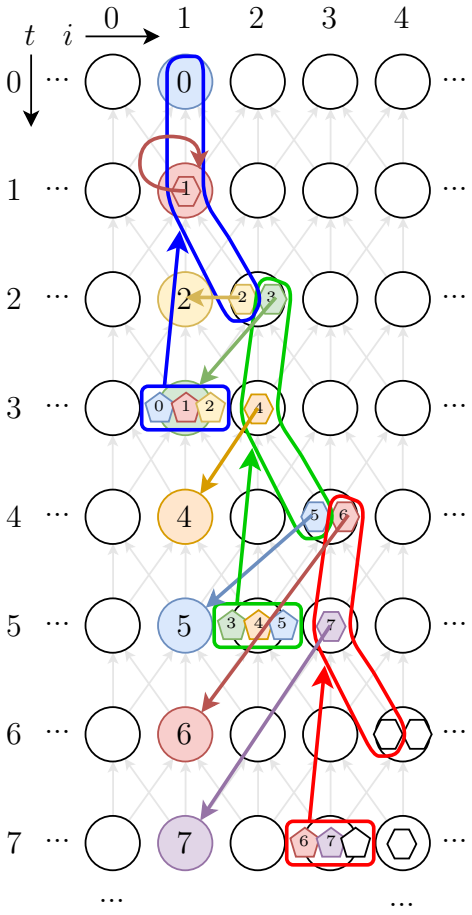

Abbildung 4.4: Visualisierung der Ausführung der Automaten C, C''' und C_1
Es werden die Zustands-Abhängigkeiten der Automaten zueinander gezeigt. Die Kreise stellen die Zustände der Ausführung des Automaten C dar, die Sechsecke zeigen das Bild von g_1 bzw. g_2 der Zustände von C''' und die Fünfecke das Bild von f der Zustände von C_1. Die eingezeichneten Pfeile zeigen, wodurch sich die Zustände ergeben.

Beweis. Wähle C' wie in Satz 3.0.1, sodass für $\forall c \in Q^{\mathbb{Z}}, \forall t \in \mathbb{N}_0, \forall i \in \mathbb{Z}$ gilt:

$$\Delta_{C'}^t(c)_i = \begin{cases} \Delta_C^{\frac{t}{2}}(c)_{i+\frac{t}{2}} & \text{falls t gerade,} \\ (\Delta_C^{\frac{t-1}{2}}(c)_{i+\frac{t-1}{2}}, \Delta_C^{\frac{t-1}{2}}(c)_{i+\frac{t+1}{2}}) & \text{sonst.} \end{cases}$$

Wähle C'' wie in Lemma 4.2.1, sodass für $i \in \mathbb{Z}$ mit $i \leq 0$ und $t \in \mathbb{N}_0$ und alle Konfigurationen c mit $c_p = \#$ für $p \leq 0$ oder $p \geq p_0$ gilt:

$$\Delta^t_{C''}(c)_i = (\Delta^{t-2i+2}_{C'}(c)_{3i-2}, \Delta^{t-2i+1}_{C'}(c)_{3i-1}, \Delta^{t-2i}_{C'}(c)_{3i})$$

Wähle C''' wie in Satz 3.0.2, sodass für $\forall c \in Q^{\mathbb{Z}}$, $\forall t \in \mathbb{N}_0$, $\forall i \in \mathbb{Z}$ gilt:

$$\Delta^t_{C'''}(c)_i = \Delta^{2t}_{C''}(c)_{i-t}$$

Wähle $g_1(q \in Q''') := q_3$ und $g_2(q \in Q''') := ((q_2)_1, q_1)$.

Es gilt dann:

$$\begin{aligned} & g_1(\Delta^{2p-1}_{C'''}(c)_p) \\ =&(\Delta^{4p-2}_{C''}(c)_{1-p})_3 \\ =&\Delta^{6p-4}_{C'}(c)_{3-3p} \\ =&\Delta^{3p-2}_{C}(c)_1 \end{aligned}$$

Außerdem gilt:

$$\begin{aligned} & g_2(\Delta^{2p}_{C'''}(c)_{p+1}) \\ =&g_2(\Delta^{4p}_{C''}(c)_{1-p}) \\ =&((\Delta^{6p-1}_{C'}(c)_{2-3p})_1, \Delta^{6p}_{C'}(c)_{1-3p}) \\ =&(\Delta^{3p-1}_{C}(c)_1, \Delta^{3p}_{C}(c)_1) \end{aligned}$$

Anhand von Abbildung 4.4 ist nun ersichtlich, wie C_1 und f konstruiert werden müssen. □

4.3 Speedup um einen Faktor durch Komprimierung

Wenn das Eingabewort komprimiert werden darf, lässt sich ein noch viel stärkerer Speedup-Satz zeigen.

Definition 4.3.1 (Speedup-Konstruktion $S_k(C)$, $S_k(c)$, γ_q)**.** Es sei ein Zellularautomat $C := (Q, \delta) \in \mathrm{CA}^*$ und ein $k \in \mathbb{N}$ gegeben. Definiere $S_k(C) := (Q', \delta')$ mit $Q' := Q^k$.

Definiere für $q \in Q$ und $l, r \in Q^*$ die Konfiguration $c(q, l, r) \in Q^{\mathbb{Z}}$:

$$c(q,l,r)_i := \begin{cases} (lr)_{i+|l|} & \text{falls } 1 - |l| \leq i \leq |r| \\ q & \text{sonst} \end{cases}$$

Wähle $q \in Q$ beliebig. Definiere $\delta' : {Q'}^3 \to Q'$ wie folgt:

$$\delta'(a,b,c) := \Delta^k_C(c(q, a, bc))[1..k]$$

Definiere die Konfigurationskomprimierung $S_k(c) \in Q'^{\mathbb{Z}}$ für eine Konfiguration $c \in Q^{\mathbb{Z}}$:

$$S_k(c)_i := c_{k(i-1)+1} c_{k(i-1)+2} \ldots c_{k(i-1)+k} \in Q^k$$

Definiere außerdem noch $\gamma_q : Q' \to Q$ mit $\gamma_q(a) := \Delta^{k-1}_C(c(q, \epsilon, a))_1$.

Satz 4.3.2 (Korrektheit der Speedup-Konstruktion)**.** *Es sei ein Zellularautomat* $C := (Q, \delta) \in CA^*$*, ein* $k \in \mathbb{N}$ *und eine Konfiguration* $c : \mathbb{Z} \to Q$ *gegeben. Für beliebige* $t \in \mathbb{N}_0, i \in \mathbb{Z}$ *und* $j \in \{1, ..., k\}$ *gilt dann:*

$$(\Delta^t_{S_k(C)}(S_k(c))_i)_j = \Delta^{kt}_C(c)_{k(i-1)+j}$$

Ferner gilt für $q := (\Delta^t_{S_k(C)}(S_k(c))_{i-1})_k$*:*

$$\gamma_q(\Delta^t_{S_k(C)}(S_k(c))_i) = \Delta^{kt+k-1}_C(c)_{k(i-1)+1}$$

Beweis. Die Aussage kann durch einfaches Nachrechnen gezeigt werden, auf das an dieser Stelle verzichtet wird. □

Die Konstruktion aus Definition 4.3.1 kann beispielsweise verwendet werden, um folgenden Satz zu beweisen.

Theorem 4.3.3. $\mathcal{L}(CA^{LT\text{-}L}) = \mathcal{L}(CA^{TH\text{-}L\text{-time}(\{n \mapsto 2n\})})$

Beweis. Mithilfe der in Kapitel 5 und Definition 4.3.1 vorgestellten Konstruktionen kann das Theorem leicht bewiesen werden. Weitere Beweise finden sich in [MaRe92] und [IbJi88] □

5. Erweiterte Nakamura-Konstruktion zur asynchronen Simulation

Dieses Kapitel bildet die Grundlage für das beeindruckende Theorem 6.2.12: Die hier vorgestellte Konstruktion erlaubt es, einen Automaten C durch einen Automaten A zu simulieren, wobei ein gegebener Automat M die Simulation steuern kann. Dabei können dynamisch Startzustände in der Simulation von C neu- und zurückgesetzt werden. Die Konstruktion kommt ohne globale Synchronisation aus, insbesondere wenn die Startzustände von C zu unterschiedlichen Zeiten von M festgelegt bzw. neu gesetzt werden.

Die ursprüngliche Nakamura-Konstruktion, wie in [Wors10] beschrieben, zeigt, wie ein Zellularautomat so umgebaut werden kann, dass er auch bei asynchroner Ausführung noch korrekt arbeitet. Von der ursprünglichen Konstruktion ist in dieser erweiterten Konstruktion allerdings nicht mehr viel übrig geblieben, außerdem dient diese Konstruktion einem anderen Zweck. Die hier vorgestellte Konstruktion kann aber weiterhin genutzt werden, um Zellularautomaten so umzubauen, dass sie auch asynchron korrekt ausgeführt werden.

5.1 Definition

Definition 5.1.1 (Erweiterte Nakamura-Konstruktion $A(M, f, C, P)$)**.** Es seien zwei Zellularautomaten $M, C \in \mathrm{CA}^*$, eine Funktion $f : Q_M \to \mathrm{Op}$ mit

$$\mathrm{Op} := \{\mathrm{reset}, \mathrm{set}(q), \mathrm{step} \mid q \in Q_C\}$$

und eine Menge $P \subseteq Q_C$ von passiven, initialen Zuständen gegeben. Der Automat M gibt Steuerbefehle über die Funktion f und der Automat C ist der asynchron simulierte Automat. Mithilfe der Menge P lässt sich in bestimmten Fällen die benötigte Zeit zum Simulieren verbessern.

Setze $N := \{0, ..., 5\}$, $Q' := (Q_C^3 \times N \times \mathbb{N}_0) \cup \{\bot\}$ und $Q := Q' \times Q_M$. Die Komponenten eines Elements $q \in Q_C^3$ seien symbolisch mit q_a, q_{a-1} und q_0 bezeichnet:

$q = (q_a, q_{a-1}, q_0)$. q_a meint dabei den aktuellen Zustand des simulierten Automaten, q_{a-1} den Zustand des vorherigen Schrittes (sofern dieser existiert) und q_0 den Zustand der simulierten Zelle in der Anfangskonfiguration. Diese Notation lässt sich auf die erste Komponente von Q' und dann Q übertragen. Die zweite Komponente eines Elements $q \in Q' \setminus \{\bot\}$ wird mit q_t und die dritte mit $q_{t'}$ bezeichnet. q_t speichert „grob“, wieviele Schritte im simulierten Automaten für diese Zelle schon berechnet wurden. $q_{t'}$ speichert diese Zahl exakt und dient nur zur Beweisführung. Da die Komponente $q_{t'}$ nach Konstruktion nicht relevant für das sichtbare Verhalten des Automaten sein wird, existiert ein äquivalenter Automat, der ohne diese Komponente auskommt. Dessen Zustandsmenge ist dann endlich. Für $q \in Q$ bezeichne q_q die erste Komponente und q_m die zweite.

Definiere zunächst $\delta' : Q'^3 \to Q'$ für $a, b, c \in Q'$ für den Fall $\bot \in \{a, b, c\}$:

$$\delta'(a,b,c) := \begin{cases} \bot & \text{falls } b = \bot, \\ ((b_0, b_0, b_0), 0, 0) & \text{sonst.} \end{cases}$$

Dem Zustand $\bot \in Q'$ kommt damit die Bedeutung zu, nichts über den Zustand der Zelle im simulierten Automaten zu wissen. Die Nachbarn einer Zelle in diesem Zustand können dann offensichtlich keinen Schritt berechnen.

Definiere für $t \in N$ die Funktion $s(t)$ zur Bestimmung eines Nachfolgers in N (siehe Abbildung 5.1):

$$s(t) := \begin{cases} t+1 & \text{falls } t < 5, \\ 3 & \text{falls } t = 5. \end{cases}$$

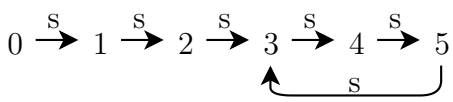

Abbildung 5.1: Wirkung von s

Definiere q^t zum Zugriff auf den möglicherweise in q gespeicherten Zustand der Zelle zum Zeitpunkt t' mit $s(t') = t$ für $q \in Q'$ und $t \in N$:

$$q^t := \begin{cases} q_0 & \text{falls } q_0 \in P \text{ und } q_t = 0, \\ q_{a-1} & \text{falls } q_0 \notin P \text{ und } q_t = t, \\ q_a & \text{falls } q_0 \notin P \text{ und } s(q_t) = t, \\ \bot & \text{sonst.} \end{cases}$$

Falls q_0 von Anfang an tot ist, beinhaltet q Zustandsinformationen der simulierten Zelle für alle Zeitpunkte, da tote Zustände in jedem Schritt gleich bleiben.

Definiere nun δ' für $a, b, c \in Q'$ falls $\bot \notin \{a, b, c\}$:

$$\delta'(a,b,c) := \begin{cases} \Big((b_0, b_0, b_0), 0, 0\Big) & \text{falls } b_0 \in P \text{ und } b_t = 0, \\ \Big((\delta(a^t, b^t, c^t), b^t, b_0), t, b_{t'} + 1\Big) & \text{sonst, falls } \bot \notin \{a^t, c^t\} \text{ für } t := s(b_t), \\ \Big((\delta(a^t, b^t, c^t), b^t, b_0), t, b_{t'}\Big) & \text{sonst, falls } \bot \notin \{a^t, c^t\} \text{ für } t := b_t, \\ \Big((\delta(a_0, b_0, c_0), b_0, b_0), 1, 1\Big) & \text{sonst.} \end{cases}$$

Der erste Fall hält tote Zustände in der Simulation fest, da sie nach Definition konstant sind. Dies ermöglicht nach Definition von q^t eine schnellere Simulation, da nicht auf Zellen in solch einem Zustand gewartet werden muss. Aus Komplexitätsgründen werden nur solche toten Zustände berücksichtigt, die von M mit set(q) schon direkt gesetzt wurden, was die Bedingung $b_t = 0$ erklärt.

Im zweiten Fall wird der nächste Schritt des simulierten Automaten für die zu aktualisierende Zelle berechnet. Dies ist offensichtlich nur möglich, wenn die Nachbarzellen Informationen zum aktuellen Schritt gespeichert haben.

Andernfalls, wenn die Nachbarzellen nur Informationen zum letzten Schritt gespeichert haben, greift Fall drei und die zu aktualisierende Zelle berechnet den aktuellen Schritt neu. Dies ist wichtig, falls sich Zellen zurücksetzen.

Im vierten Fall setzt die Zelle die Simulation lokal zurück und berechnet wieder den ersten Schritt der Simulation.

Abschließend ist δ für $a, b, c \in Q$ nun wie folgt definiert:

$$\delta(\left(a_q, a_m\right), \left(b_q, b_m\right), \left(c_q, c_m\right)) := \begin{cases} \text{Setze } c := \delta_M(a_m, b_m, c_m), & \\ \Big(((q,q,q),0,0), c\Big) & \text{falls } f(c) = \text{set}(q), \\ \Big(\bot, c\Big) & \text{falls } f(c) = \text{reset}, \\ \Big(\delta'(a_q, b_q, c_q), c\Big) & \text{falls } f(c) = \text{step}. \end{cases}$$

Damit kann der Automat M die Simulation kontrollieren und diese für einzelne Zellen auf einen neuen Anfangszustand zurücksetzen. Die Konstruktion stellt sicher, dass sich benachbarte Zellen einer solchen zurückgesetzten Zelle korrekt verhalten.

Definiere nun $A(M, f, C, P) := (Q, \delta)$.

Wenn nun eine Konfiguration $c_M : \mathbb{Z} \to Q_M$ von M gegeben ist, kann sie als Konfiguration $c : \mathbb{Z} \to Q$ von $A(M, f, C, P)$ aufgefasst werden:

$$c_i := \begin{cases} (((q,q,q),0,0), (c_M)_i) & \text{falls } f((c_M)_i) = \text{set}(q), \\ (\bot, (c_M)_i) & \text{sonst.} \end{cases}$$

Um über die Korrektheit der Konstruktion sprechen zu können, sind vorerst noch weitere Definitionen nötig.

Definition 5.1.2 (Sichtbare Startzustandszeit $st_o(t,i)$ und Startkonfiguration $c(t,i)$, sichtbare aktiven Zellen $A(t,i)$)**.** Es seien $M, C \in \text{CA}^*$, f und P wie in Definition 5.1.1 und eine Konfiguration $c_M : \mathbb{Z} \to Q_M$ gegeben. Seien $t \in \mathbb{N}_0$, $i \in \mathbb{Z}$ und $o \in \text{Op}$. Definiere die sichtbare Startzustandszeit $\text{st}_{M,C,f,P,o}(t,i) := \text{st}_o(t,i)$:

$$\text{st}_o(t,i) := \begin{cases} t' & \text{falls } f(\Delta_M^{t'}(c_M)_i) = o \\ & \text{für } t' := \max\{k \in \{0,...,t\} \mid f(\Delta_M^k(c_M)_i) \neq \text{step}\}, \\ \infty & \text{sonst.} \end{cases}$$

Offensichtlich gilt $o_1 = o_2$, falls $st_{o_1}(t,i) \neq \infty \wedge st_{o_2}(t,i) \neq \infty$ für $o_1, o_2 \in \text{Op}$. Setze $st_{\text{set}}(t,i) := \min\{st_{\text{set}(q)}(t,i) \mid q \in Q_C\}$.

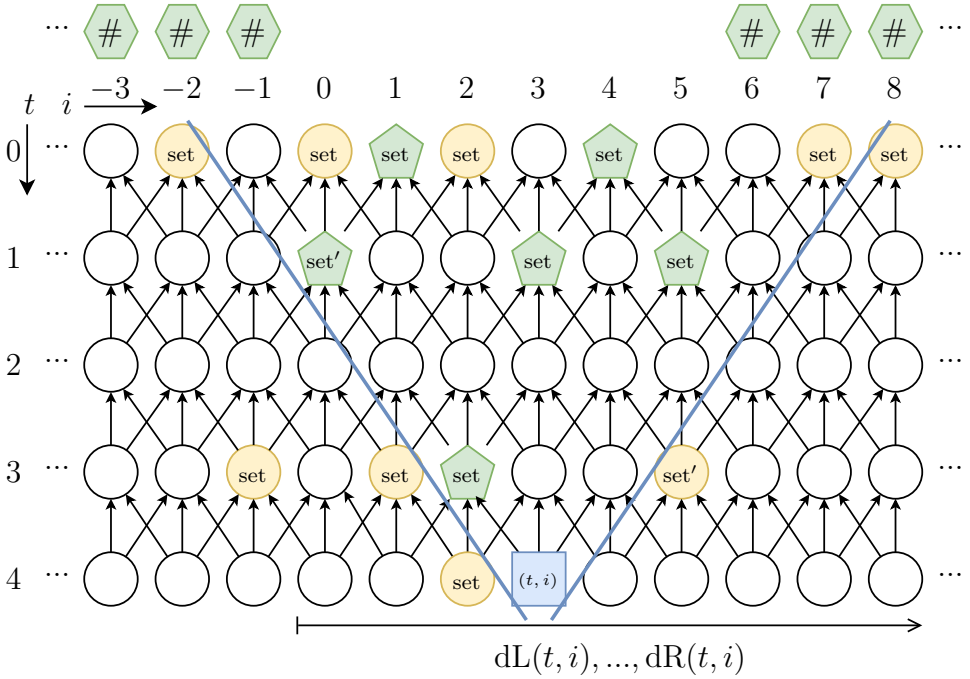

Abbildung 5.2: Sichtbare Startkonfiguration

Ein (gelb oder grün markiertes) „set“ in der Abbildung bedeutet, dass der Automat M über f an entsprechender Position und zum entsprechenden Zeitpunkt einen set(q)-Befehl an A übermittelt (das konkret gewählte q ist für dieses Beispiel egal, taucht aber als entsprechend gewähltes q in der sichtbaren Startzustandskonfiguration wieder auf). Ein „set′“ bedeutet, dass ein toter Zustand q aus P gesetzt wird. Runde Zustände (in weiß) stellen ein „step“ dar.
So ist die sichtbare Startzustandszeit $st_{\text{set}}(1, 6)$ in der Abbildung ∞, während $st_{\text{set}}(2, 5) = 1$ gilt.
Die fünf- und sechseckigen Zustände (in grün) reflekterien die sichtbare Startkonfiguration aus Sicht des viereckigen Zustandes (blau gefärbt, in der Abbildung $t = 4$ und $i = 3$). Die Sechsecke stellen dabei die von $c(t, i)_j$ eingefügten Rauten dar.
Der viereckige Zustand reflektiert einen Zustand des simulierten Automaten für einen Zeitpunkt, zu dem die so eingefügten Rauten allerdings noch keinen Einfluss nehmen konnten. Dieser Zeitpunkt kann also nur zwischen 0 und 2 liegen. Tatsächlich kann die Konstruktion nur einen Schritt simulieren, da der von (t, i) sichtbare Startzustand für die Zelle 2 erst zum Zeitpunkt 3 erfolgt.
Das eingezeichnete, rechts-unendliche Intervall kennzeichnet die von (t, i) aus sichtbaren aktiven Zellen. Da „set′“ bei $t = 3$ und $i = 5$ von (t, i) aus nicht sichtbar ist, spielt der gesetzte Zustand aus P an dieser Stelle keine Rolle.

Definiere die von (t, i) aus sichtbare Startkonfiguration $c_{M,f,\#}(t, i)_j := c(t, i)$ für $\# \in Q_C$, $t \in \mathbb{N}_0$ und $p \in \mathbb{Z}$ wie folgt:

$$c(t,i)_j := \begin{cases} q & \text{falls } \mathrm{st}_{\mathrm{set}(q)}(t - |i - j|, j) \neq \infty, \\ \# & \text{sonst.} \end{cases}$$

Definiere die von (t, i) aus sichtbare tote rechte Zelle $\mathrm{dR}(t, i)$ und linke Zelle $\mathrm{dL}(t, i)$:

$$\mathrm{dR}(t, i) := \min(\{j \in \mathbb{Z} \mid j \geq i \wedge c(t, i)_j \in P\} \cup \{\infty\})$$
$$\mathrm{dL}(t, i) := \max(\{j \in \mathbb{Z} \mid j \leq i \wedge c(t, i)_j \in P\} \cup \{-\infty\})$$

Definiere damit die von (t, i) aus sichtbaren aktiven Zellen $A(t, i)$:

$$A(t, i) := \{\mathrm{dL}(t, i), ..., \mathrm{dR}(t, i)\}$$

Abbildung 5.2 veranschaulicht die eingeführten Definitionen.

5.2 Korrektheit und Simulationsgeschwindigkeit

Aufgrund der Komplexität der erweiterten Nakamura-Konstruktion wird die Korrektheit und die Zeit, die für die Simulation benötigt wird, separat betrachtet. Erst Satz 5.2.2 zeigt, dass die Konstruktion überhaupt Fortschritte in der Simulation macht. Der folgende Satz zeigt, dass wenn die Konstruktion Schritte des zu simulierenden Zellularautomaten berechnet, diese korrekt sind.

Satz 5.2.1 (Korrektheit der erweiterten Nakamura-Konstruktion)**.** *Es seien $M, C \in CA^*$, f und P wie in Definition 5.1.1 und eine Konfiguration $c_M : \mathbb{Z} \to Q_M$ gegeben. Sei c die Auffassung von c_M als Konfiguration von $A := A(M, f, C, P)$.*

Es gelten folgende Aussagen für $t \in \mathbb{N}_0$, $i \in \mathbb{Z}$:

(I) A führt M korrekt aus:
$$(\Delta_A^t(c)_i)_m = \Delta_M^t(c)_i$$

(II) Sei $c \in \{-1, 1\}$ mit $\perp \notin \{a := (\Delta_A^t(c)_i)_q, b := (\Delta_A^t(c)_{i+c})_q\}$. Benachbarte Zellen mit ähnlicher „groben" Zeit haben ähnliche „exakte" Zeit:
$$s(a_t) = s(b_t) \Rightarrow a_{t'} = b_{t'} \text{ und } a_t = s(b_t) \Rightarrow a_{t'} = b_{t'} + 1$$
Für $t \in \{0, 1, 2\}$ gilt offensichtlich:
$$a_t = t \Rightarrow a_{t'} = t$$

(III) Sei $q := (\Delta_A^t(c)_i)_q$ mit $q \neq \perp$. Dann reflektiert q_a aus Sicht von (t, i) den korrekt simulierten Zustand:
$$q_a = \Delta_C^{q_{t'}}(c_{M,f,\#}(t, i))_i$$
Analog reflektiert q_{a-1} den vorherigen Zustand, falls $q_t > 0$:
$$q_{a-1} = \Delta_C^{q_{t'}-1}(c_{M,f,\#}(t, i))_i$$
Für $q_t = 0$ gilt $q_{a-1} = q_a$.

Beweis. (I) Offensichtlich.

(II) Kann induktiv leicht gezeigt werden. Beim Rücksetzen durch $\mathrm{set}(q)$ werden benachbarte Zellen ebenfalls auf ihren Startzustand zurückgesetzt, um diese Invariante zu erhalten.

(III) In dieser Arbeit wird aus Zeitgründen auf einen formalen Beweis dieser Aussage verzichtet. Die Komplexität der Definition der Konstruktion zieht sehr viele Fallunterscheidungen nach sich, die ohne geschickte Beweisführung nicht mehr nachvollziehbar sind. Der Leser überzeuge sich anhand von Beispielen von der Richtigkeit dieser Aussage.

□

Satz 5.2.2 (Zeitbedarf der erweiterten Nakamura-Konstruktion in einfachen Fällen)**.** *Es seien $M, C \in CA^*$, f und P wie in Definition 5.1.1 und eine Konfiguration $c_M : \mathbb{Z} \to Q_M$ gegeben. Sei außerdem c die Auffassung von c_M als Konfiguration von $A := A(M, f, C, P)$.*

Für $t \in \mathbb{N}_0$, $i \in \mathbb{Z}$ und $q := (\Delta_A^t(c)_i)_q$ setze:

$$\mathrm{tc}(t,i) := \max\{\hat{t} \in \mathbb{N}_0 \mid \forall k \in \{-\hat{t}, ..., \hat{t}\} \cap A(t,i) : \mathrm{st}_{\mathrm{set}}(t - |k|, i + k) \leq t - \hat{t}\}$$

(I) Für $q \neq \perp$ gilt $q_{t'} \leq \mathrm{tc}(t,i)$.

(II) Für $q \neq \perp$ gilt sogar $q_{t'} = \mathrm{tc}(t,i)$, falls für alle $k \in \{-\hat{t}, ..., \hat{t}\} \cap A(t,i)$ gilt:

$$t_1(t,k) > t_2(t,k) \ \wedge \ st_{\mathrm{reset}}(t_1(t,k) - 1, i + k) \leq t_2(t,k)$$

Wobei

$$t_1(t,k) := \min\{t_2(t,k-1), t_2(t,k+1)\}$$

und

$$t_2(t,k) := st_{\mathrm{set}}(t - |k|, i + k)$$

Diese zusätzlichen Bedingungen stellen sicher, dass das Neusetzen von Zuständen erst nach einem Zurücksetzen passiert und dass das Zurücksetzen ausreichend früh geschieht, sodass benachbarte Zellen in ihrer Simulation auf das Neusetzen warten und nicht ihre Simulation mit veralteten Startzuständen weiterführen. Warten sie nämlich nicht, müssen sie ebenfalls zurückgesetzt werden und die Simulation läuft langsamer ab.

(III) Sei $c' : \mathbb{Z} \to Q_C$ eine Konfiguration von C und n so, dass $c'_i \neq \# \Leftrightarrow i \in \{1, ..., n\}$. Angenommen, $P \subseteq \{\#\}$ und für $i \in \mathbb{Z}$ und $t \in \mathbb{N}_0$ gelte wie in Abbildung 5.3 dargestellt:

$$f(\Delta_M^t(c_M)_i) = \begin{cases} \mathrm{reset} & \text{falls } t = 0 \text{ und } i \in \{1, ..., n\}, \\ \mathrm{set}(c'_i) & \text{falls } t = 0 \text{ und } i \notin \{1, ..., n\}, \\ \mathrm{set}(c'_i) & \text{falls } i \in \{1, ..., n\} \text{ und } t = 2i + 1, \\ \mathrm{step} & \text{sonst.} \end{cases}$$

Dann gilt für $3 \leq t \leq 3n$:

$$(\Delta_A^t(c)_1)_a = \Delta_C^{\lfloor \frac{t}{3} \rfloor - 1}(c')$$

Beweis. (I), (II) Wie in Satz 5.2.1 wird auch für diese Aussagen auf einen formalen Beweis verzichtet. Der Leser überzeuge sich auch hier durch Beispiele, wie in Abbildung 5.3 gezeigt, von der Korrektheit der Aussage.

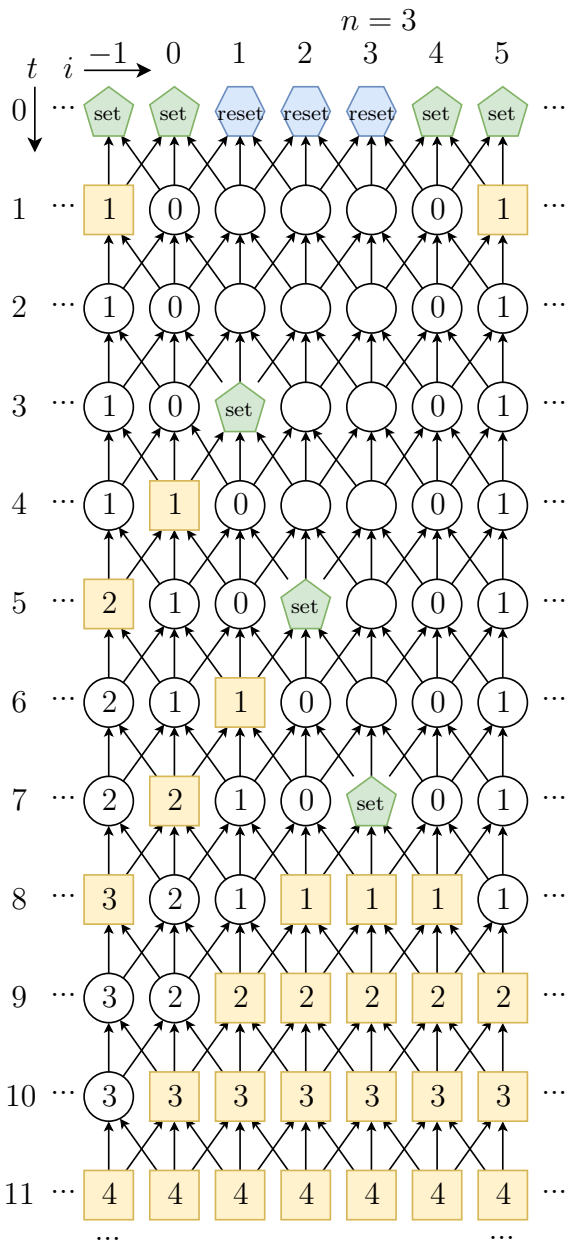

Abbildung 5.3: Simulierte Zeit im Vergleich zur Simulationszeit
Die Abbildung ist analog zu Abbildung 5.2 zu verstehen. Viereckige Zustände (in gelb) geben dabei an, dass für eine simulierte Zelle der nächste Zustand berechnet wurde. Die Zahl eines Zustands gibt an, wie viele Schritte für diese Zelle schon simuliert wurden. Nicht in jedem Simulations-Schritt kann auch ein simulierter Schritt berechnet werden, da die dafür notwendigen Startzustände erst zu einem späteren Simulations-Zeitpunkt gesetzt werden.

(III) Folgt aus (II).

Es gilt offensichtlich:

$$st_{\text{set}}(t,i) = \begin{cases} 0 & \text{falls } t \notin \{1,...,n\}, \\ \infty & \text{sonst, falls } t \leq 2i, \\ 2i+1 & \text{sonst.} \end{cases}$$

Sei $\hat{t} \in \{0, ..., n\}$ so, dass $t \in \{3\hat{t} + 3, ..., 3\hat{t} + 5\}$. Sei außerdem $k \in \{-\hat{t}, ..., \hat{t}\}$.

1. Fall: $-\hat{t} \leq k \leq -1$.
   Dann $\mathrm{st}_{\mathrm{set}}(t - |k|, 1 + k) = 0 \leq t - \hat{t}$.
2. Fall: $0 \leq k \leq \hat{t}$.
   Da $t \geq 3\hat{t} + 3$, folgt $t - |k| > 2k + 2$ und damit $\mathrm{st}_{\mathrm{set}}(t - |k|, 1 + k) = 2k + 3 \leq 2\hat{t} + 3 \leq t - \hat{t}$.
   Da $t \leq 3\hat{t}+5$, folgt $t-|\hat{t}+1| \leq 2(\hat{t}+1)+2$ und damit $\mathrm{st}_{\mathrm{set}}(t-|k|, 1+k) = \infty$.

Also $\mathrm{tc}(t, i) = \hat{t} = \lfloor \frac{t}{3} \rfloor - 1$ wodurch mit Satz 5.2.1 die Behauptung folgt - schließlich ist c' die einzig sichtbare Konfiguration.

□

6. Advice-Zellularautomaten

Dieses Kapitel beschäftigt sich damit, inwiefern die Mächtigkeit bestimmter Klassen von Zellularautomaten verändert wird, wenn der Automat zusätzlich zum Eingabewort ein vom Eingabewort abhängiges Unterstützungswort (im Folgenden Advice genannt) erhält. Insbesondere wird gezeigt, dass bestimmte solcher Klassen unter Zuhilfenahme ausgewählter Advices abgeschlossen sind. Solche Advices heißen dann verträglich bezüglich dieser Klasse. Diese Abschlusseigenschaft ist nützlich, um die Konstruktion einiger Echtzeit-Zellularautomaten zu vereinfachen und um besser verstehen zu können, wie mächtig Echtzeit-Zellularautomaten sind.

6.1 Definition

Es wird zunächst definiert, was ein Advice ist.

Definition 6.1.1. Ein Σ-Γ-*Advice* (Hinweis) $\mathcal{A}$ ist eine Abbildung $\mathcal{A} : \Sigma^* \to \mathcal{P}(\Gamma^*)$ mit $\forall w \in \Sigma^* : \mathcal{A}(w) \subseteq \Sigma^{|w|}$.

Wenn $\forall w \in \Sigma^* : |\mathcal{A}(w)| = 1$, kann $\mathcal{A}$ auch als Abbildung $\mathcal{A} : \Sigma^* \to \Gamma^*$ aufgefasst werden. $\mathcal{A}$ heißt dann *deterministisch*.

Ein Hinweis kann eine ganze Menge von Unterstützungswörtern bereitstellen. In diesem Fall wird ein Wort akzeptiert, wenn der Automat das Eingabewort mit Unterstützung irgendeines der bereitgestellten Wörter akzeptiert.

Um zu beschreiben, wann ein Automat mithilfe eines Advices ein Wort akzeptiert oder ablehnt, müssen zunächst die Funktionen split und join definiert werden.

Definition 6.1.2. Definiere injektive Funktion split : $(\Sigma \times \Gamma)^* \to (\Sigma^* \times \Gamma^*)$ um ein Wort über zwei Alphabete in zwei Wörter über je ein Alphabet zu zerlegen:

$$\forall v \in \Sigma^*, w \in \Gamma^{|v|} : \text{split}(\binom{v_1}{w_1} ... \binom{v_{|v|}}{w_{|v|}}) := \binom{v}{w}$$

Definiere join : split$((\Sigma \times \Gamma)^*) \to (\Sigma \times \Gamma)^*$, um zwei Wörter über je ein Alphabet zu einem Wort über zwei Alphabete zusammenzufügen:

$$\text{join}(w) := \text{split}^{-1}(w)$$

Beispiel 6.1.3.

$$\text{split}(\binom{1}{a}\binom{2}{b}\binom{3}{c}) = \binom{123}{abc}$$

$$\text{join}(\binom{123}{abc}) = \binom{1}{a}\binom{2}{b}\binom{3}{c}$$

Jetzt kann definiert werden, welche Auswirkung ein Advice auf einen Zellularautomaten hat.

Definition 6.1.4. Es sei ein Advice $\mathcal{A}$ und ein Zellularautomat $C \in \text{CA}$ gegeben. Mit Unterstützung des Advices $\mathcal{A}$ kann C nun folgende Sprache erkennen:

$$L_{\mathcal{A},C} := \{w \in \Sigma^* \mid \exists v \in \mathcal{A}(w) : \text{join}(\binom{w}{v}) \in L(C)\}$$

Für $M(\mathcal{A}) := \{\text{join}(\binom{w}{v}) \mid w \in \Sigma^* \wedge v \in \mathcal{A}(w)\}$ kann $L_{\mathcal{A},C}$ auch wie folgt aufgefasst werden:

$$L_{\mathcal{A},C} = \text{split}(L(C) \cap M(\mathcal{A}))_1$$

Folgender CA-Funktor stellt einer Klasse von Zellularautomaten einen gegebenen Advice zur Verfügung:

$$\text{Adv}(\mathcal{A})(M \subseteq \text{CA}) := \{(Q_C, \delta_C, \Sigma, \#_C, L_{\mathcal{A},C}, (S_C, C)) \mid C \in M,\ \Sigma_C = \Sigma \times \Gamma\}$$

Eine Fragestellung, die sich bei solchen Advices zwingend auftut, ist, ob ein gegebener Advice eine Klasse von Zellularautomaten echt mächtiger macht. Die folgende Definition führt dafür einen entsprechenden Begriff ein.

Definition 6.1.5 ($\mathcal{A}$ ist M-verträglich)**.** Sei $M \subseteq \text{CA}$ eine Menge von Zellularautomaten und $\mathcal{A}$ ein Advice. $\mathcal{A}$ heißt M-verträglich, falls gilt:

$$\mathcal{L}(M) = \mathcal{L}(M^{\mathcal{A}})$$

6.2 Ergebnisse

Offensichtlich verringert ein Advice die Mächtigkeit „wohlgeformter“ Klassen von Zellularautomaten nicht, wie im folgenden Satz formalisiert. Der Satz gilt nicht für beliebige Klassen, da sich zum Beispiel das Alphabet ändert und in der Konstruktion entsprechend gewählt werden muss. Mithilfe von Lemma 6.2.1 muss zum Zeigen der Verträglichkeit eines Advices immer nur die Inklusion „$\supseteq$“ beachtet werden.

Lemma 6.2.1. *Sei ein Advice $\mathcal{A}$ und eine Menge $F \subseteq {\mathbb{N}_0}^{\mathbb{N}_0}$ von Funktionen gegeben. Dann gilt: $\mathcal{L}(CA^{\text{time}(F)\text{-}L}) \subseteq \mathcal{L}(CA^{\text{time}(F)\text{-}L\text{-}Adv(\mathcal{A})})$.*

Beweis. Der Advice kann ignoriert werden, indem $\Sigma \times \Gamma$ auf Σ reduziert wird. □

Der nächste Satz zeigt ein Advice, das vermutlich nicht Echtzeit-Zellularautomaten-verträglich ist. Wäre es Echtzeit-Zellularautomaten-verträglich, wäre auch Echtzeit genauso mächtig wie Linearzeit.

Satz 6.2.2. *Sei* $(\Sigma \cup \{\square\})^2 \subseteq \Gamma, \square \notin \Sigma$. *Definiere den Komprimierungs-Advice* $\mathcal{A}_K$*:*

$$\mathcal{A}_K(w) := \binom{w_1}{w_2}\binom{w_3}{w_4}...\binom{w_{|w-1|}}{w_{|w|}}\binom{\square}{\square}^{\frac{|w|}{2}}$$

Falls $|w|$ *ungerade, setze* $w := w\square$. *Unter dem Advice* $\mathcal{A}_K$ *fällt Echtzeit mit Linearzeit zusammen:*

$$\mathcal{L}(CA^{RT\text{-}L\text{-}Adv(\mathcal{A})}) = \mathcal{L}(CA^{LT\text{-}L})$$

Beweis. Die Inklusion „$\subseteq$“ ist klar: Offensichtlich ist $\mathcal{A}_K$ $\mathcal{L}(\mathrm{CA}^{\mathrm{LT\text{-}L}})$-verträglich. Sei nun $L \in \mathcal{L}(\mathrm{CA}^{\mathrm{LT\text{-}L}})$. Nach Theorem 4.3.3 existiert ein Zellularautomat C, der L in $2n$ Schritten erkennt. Der Automat $C' := S_2(C)$ aus Definition 4.3.1 erkennt L dann in n Schritten auf der komprimierten Konfiguration $S_2([w])$. Da die komprimierte Konfiguration durch den Advice bereitgestellt wird, kann L mithilfe des Advices in Echtzeit erkannt werden. □

Nicht-deterministische Advices werden in diesem Kapitel nicht weiter untersucht. Trotzdem soll der folgende Zusammenhang nicht vorenthalten werden.

Satz 6.2.3. *Seien* $\mathcal{A}_1$ *und* $\mathcal{A}_2$ *zwei Advices. Definiere* $(\mathcal{A}_1 \cup \mathcal{A}_2)(w) := \mathcal{A}_1(w) \cup \mathcal{A}_2(w)$ *und* $(\mathcal{A}_1 \cap \mathcal{A}_2)(w) := \mathcal{A}_1(w) \cap \mathcal{A}_2(w)$.

Sind $\mathcal{A}_1$ *und* $\mathcal{A}_2$ $CA^{RT\text{-}L}$*-verträglich, so sind auch* $\mathcal{A}_1 \cup \mathcal{A}_2$ *und* $\mathcal{A}_1 \cap \mathcal{A}_2$ $CA^{RT\text{-}L}$*-verträglich.*

Beweis. Die Inklusion „$\subseteq$“ der Verträglichkeit ist mit Lemma 6.2.1 klar. Wähle $\oplus \in \{\cup, \cap\}$. Sei $L \in \mathcal{L}(\mathrm{CA}^{\mathrm{RT\text{-}L\text{-}}\mathcal{A}_1 \oplus \mathcal{A}_2})$. Es existieren nun die Echtzeit-Zellularautomaten C mit $L_{\mathcal{A}_1 \oplus \mathcal{A}_2, C} = L$, C_1 mit $L_{C_1} = L_{\mathcal{A}_1, C}$ und C_2 mit $L_{C_2} = L_{\mathcal{A}_2, C}$.

Offensichtlich gilt für beide Wahlen von $\oplus$ dann $L_{\mathcal{A}_1 \oplus \mathcal{A}_2, C} = L_{\mathcal{A}_1, C} \oplus L_{\mathcal{A}_2, C} = L_{C_1} \oplus L_{C_2}$. Da Echtzeit-Zellularautomaten unter Vereinigung und Schnitt abgeschlossen sind, folgt die Behauptung. □

Der Satz lässt sich auf eine endliche Menge von Vereinigungen bzw. Schnitten fortführen.

Die folgende Überlegung zeigt, dass die Fragestellung, ob Linearzeit mit Echtzeit zusammenfällt, auch für Advice-Zellularautomaten interessant ist.

Satz 6.2.4. *Sei* $\mathcal{A}$ *ein Advice mit* $\mathcal{L}(CA^{RT\text{-}L\text{-}Adv(\mathcal{A})}) \neq \mathcal{L}(CA^{LT\text{-}L\text{-}Adv(\mathcal{A})})$. *Dann* $\mathcal{L}(CA^{RT\text{-}L}) \neq \mathcal{L}(CA^{LT\text{-}L})$.

Beweis. Angenommen $\mathcal{L}_1 := \mathcal{L}(\mathrm{CA}^{\mathrm{RT\text{-}L\text{-}Adv}(\mathcal{A})}) \neq \mathcal{L}(\mathrm{CA}^{\mathrm{LT\text{-}L\text{-}Adv}(\mathcal{A})}) =: \mathcal{L}_2$. Wegen $\mathcal{L}_1 \subseteq \mathcal{L}_2$ gibt es laut Annahme ein $L \in \mathcal{L}_2 \setminus \mathcal{L}_1$. Dann gibt es einen Automaten $C \in \mathrm{CA}^{\mathrm{LT\text{-}L}}$, sodass gilt:

$$L = L_{\mathcal{A},C} = \{w \in \Sigma^* \mid \exists v \in \mathcal{A}(w) : \mathrm{join}(\binom{w}{v}) \in L(C)\}$$

Angenommen, es gäbe einen Automaten $C' \in \mathrm{CA}^{\mathrm{RT\text{-}L}}$ mit $L_{C'} = L_C$. Dann $L_{\mathcal{A},C} = \{w \in \Sigma^* \mid \exists v \in \mathcal{A}(w) : \mathrm{join}(\binom{w}{v}) \in L(C')\} = L_{\mathcal{A},C'}$ und damit $L \in \mathcal{L}_1$. Widerspruch zur Annahme! Also $L(C) \in \mathcal{L}(\mathrm{CA}^{\mathrm{LT\text{-}L}}) \setminus \mathcal{L}(\mathrm{CA}^{\mathrm{RT\text{-}L}})$. □

6.2.1 $\mathrm{CA}^{\text{RT-L}}$-verträgliche Advices

In diesem Kapitel werden eine Reihe von Advices vorgestellt, die sich als Echtzeit-verträglich herausgestellt haben. Die meisten hier vorgestellten Advices verwenden $\mathbb{B}$ als Advice-Alphabet und markieren damit ausgewählte Zeichen des Eingabeworts. Einfache Markierungen werden durch die im folgenden definierte Funktion erreicht. Diese Funktion wird auch in Kapitel 7 wieder aufgegriffen.

Definition 6.2.5. Seien $k, n \in \mathbb{N}_0$ zwei Zahlen. Die Funktion ones gibt ein Wort der Länge n zurück, sodass die k ersten Zeichen Einsen sind.

$$\text{ones}(k, n) := (1^k 0^n)[1..n]$$

Beispiel 6.2.6.

$$\text{ones}(2, 5) = 11000$$

$$\text{ones}(6, 5) = 11111$$

Eine Reihe einfacher Advices stützt sich auf Berechnungen in endlich vielen Schritten. Dazu muss zunächst definiert werden, was es heißt, wenn ein Zellularautomat eine Funktion berechnet.

Definition 6.2.7. Sei Γ ein endliches Alphabet. Ein F-haltender Zellularautomat $C \in \mathrm{CA}^{\text{FH}}$ berechnet eine Funktion $f : \Sigma^* \to \Gamma^*$, wenn $\Gamma \subseteq Q_C$, $\#_C \notin \Gamma$ und

$$\forall w \in \Sigma^* : \Delta_C^{\text{time}_C(w)}([w]) = [f(w)]$$

gilt.

Der folgende Satz zeigt gleich die Echtzeit-Verträglichkeit einer ganzen Klasse von „einfachen“ Advices.

Satz 6.2.8. *Wenn $\mathcal{A} : \Sigma^* \to \Gamma^*$ eine Funktion ist mit $\forall w \in \Sigma^* : |f(w)| = |w|$, die von einem F-haltenden Zellularautomaten B mit $\text{time}_C(\cdot) = k$ und $k \in \mathbb{N}_0$ berechnet wird, dann ist $\mathcal{A}$ $CA^{RT\text{-}L}$-verträglich.*

Beweis. Die Inklusion „$\subseteq$“ der $\mathrm{CA}^{\text{RT-L}}$-Verträglichkeit folgt durch Lemma 6.2.1.

Sei also $L \in \mathcal{L}(\mathrm{CA}^{\text{RT-L-Adv}(\mathcal{A})})$ und C der Echtzeit-Zellularautomat mit $L_{C,\mathcal{A}} = L$.

Es gibt einen Zellularautomaten I, der in k Schritten die Identität berechnet. Folglich gibt es einen Zellularautomaten A, der die Funktion $f : \Sigma^* \to (\Sigma \times \Gamma)^*$ mit $f(w) = \text{join}(\binom{w}{f(w)})$ in k Schritten berechnet, in dem I und B für k Schritte parallel ausgeführt werden.

Konstruiere den Automaten C' durch Hintereinanderausführung der Automaten A und C - dies ist möglich, da k fest ist und nicht von der Eingabe abhängt. Für ein Wort $w \in \Sigma^*$ braucht der Automat C' $k + |w|$ viele Schritte und es gilt $L_{C'} = L$.

Nach Satz 4.1.3 gibt es nun einen Automaten $C'' \in \mathrm{CA}^{\text{RT-L}}$ mit $L_{C''} = L_{C'}$. Also $L \in \mathcal{L}(\mathrm{CA}^{\text{RT-L}})$. □

Mit Satz 6.2.8 kann nun leicht die Echtzeit-Verträglichkeit einiger konkreter Advices gezeigt werden.

Korollar 6.2.9. *Sei $\Gamma = \mathbb{B}$, $k \in \mathbb{N}_0$. Folgende Advices sind $CA^{RT\text{-}L}$-verträglich:*

(I) $\mathcal{A}_1(w) := \text{ones}(k, |w|)$

(II) $\mathcal{A}_2(w) := \text{ones}(k, |w|)^{\text{Rev}}$

Beweis. Die Funktionen $\mathcal{A}_i$ können offensichtlich von einem F-haltenden Zellularautomat in k Schritten berechnet werden. Mit Satz 6.2.8 folgt die Behauptung. □

Es hat sich herausgestellt, dass die im folgenden definierte Eigenschaft auf viele interessante Advices zutrifft. Wird sie als Annahme gefordert, können ohne Beschränkung der Allgemeinheit weitere Eigenschaften angenommen werden - dies zeigt Lemma 6.2.11.

Definition 6.2.10 (Präfixstabil)**.** Ein Advice $\mathcal{A}$ heißt präfixstabil, wenn für alle $w, s \in \Sigma^*$ gilt:

$$\mathcal{A}(w) = \{v[1, ..., |w|] \mid v \in \mathcal{A}(ws)\}$$

Lemma 6.2.11. *Sei $\mathcal{A}$ ein präfixstabiler Advice, $k \in \mathbb{N}$ und $L \in \mathcal{L}(CA^{RT\text{-}L\text{-}Adv(\mathcal{A})})$. Dann gilt für $L(k)$ aus Satz 4.1.4:*

$$L(k) \in \mathcal{L}(CA^{RT\text{-}L\text{-}Adv(\mathcal{A})})$$

Sei $\mathcal{M}_k$ die Menge alle Sprachen, deren Wortlängen Vielfache von k sind. Es gilt dann folgende Implikation:

$$\begin{aligned} &\mathcal{L}(CA^{RT\text{-}L\text{-}Adv(\mathcal{A})}) \cap \mathcal{M}_k = \mathcal{L}(CA^{RT\text{-}L}) \cap \mathcal{M}_k \\ &\Rightarrow \mathcal{L}(CA^{RT\text{-}L\text{-}Adv(\mathcal{A})}) = \mathcal{L}(CA^{RT\text{-}L}) \end{aligned}$$

Beweis. Analog zum Beweis von Satz 4.1.4: Da $\mathcal{A}$ präfixstabil ist, können Zeichen am Ende des Wortes gelöscht werden, ohne dass sich der Advice auf dem übrig gebliebenen Teil des Wortes ändert.

Ist nun $L \in \mathcal{L}(\text{CA}^{\text{RT-L-Adv}(\mathcal{A})})$, gilt $L(k) \in \mathcal{L}(\text{CA}^{\text{RT-L-Adv}(\mathcal{A})}) \cap \mathcal{M}_k$ und damit auch $L(k) \in \mathcal{L}(\text{CA}^{\text{RT-L}}) \cap \mathcal{M}_k$. Mit Satz 4.1.4 folgt dann $L \in \mathcal{L}(\text{CA}^{\text{RT-L}})$. □

Das folgende Theorem 6.2.12 ist eine der zentralen Erkenntnisse dieser Arbeit. Es macht eine Aussage über die Echtzeit-Verträglichkeit von Advices, die sich auf bestimmte Art durch beliebige Zellularautomaten ergeben. Korollar 6.2.13 zeigt, wie mit diesem Theorem Echtzeit-Zellularautomaten in gewisser Hinsicht hintereinandergeschaltet werden können.

Zum Beweis werden Sätze aus allen vorangegangen Kapitel verwendet.

Theorem 6.2.12. *Sei $H = (Q, \delta) \in CA^*$ ein Zellularautomat mit $\Sigma \subseteq Q$. Der folgende Advice $\mathcal{A}_H$ ist $CA^{RT\text{-}L}$-verträglich:*

$$\mathcal{A}_H(w) := v \in Q^{|w|} \text{ mit } v_i := \Delta_A^{i-1}([w])_1$$

Beweis. Die Inklusion „$\subseteq$“ der $\mathrm{CA}^{\text{RT-L}}$-Verträglichkeit folgt durch Lemma 6.2.1. Sei nun $L \in \mathcal{L}(\mathrm{CA}^{\text{RT-L-Adv}(\mathcal{A}_H)})$. Da $\mathcal{A}_H$ präfixstabil ist, kann wegen Lemma 6.2.11 und Satz 2.4.3 OBdA. angenommen werden, dass die Wortlängen von L alle Vielfache von 3 sind und dass L nicht das leere Wort enthält.

Es existiert also ein Echtzeit-Zellularautomat C mit $L_{\mathcal{A},C} = L$. Nach Satz 2.4.2 kann der Rand von C tot gewählt werden. Sei $w \in \Sigma^*$ mit $|w| \in 3\mathbb{N}$, $c := [w]$, $v := \mathcal{A}_H(w)$ und $c' := [\mathrm{join}(\binom{w}{v})]$. Nach Satz 4.2.2 gibt es einen Automaten C_1 und eine Funktion f_1, sodass für $i \in \mathbb{N}$ gilt:

$$f_1(\Delta_{C_2}^{2i+1}(c)_i) = (\Delta_C^{3i-3}(c)_1, \Delta_C^{3i-2}(c)_1, \Delta_C^{3i-1}(c)_1)$$

Ferner lässt sich leicht ein Automat K konstruieren und eine Funktion f_2 angeben, sodass für $i \geq 1$ gilt:

$$f_2(\Delta_K^{2i+1}(c)_i) = (c_{3i-2}, c_{3i-1}, c_{3i})$$

Indem C_1 und K gleichzeitig ausgeführt werden, lässt sich ein Automat M konstruieren und eine Funktion f angeben, sodass für $n := \frac{|w|}{3}$ gilt:

$$f(\Delta_M^t(c)_i) = \begin{cases} \text{reset} & \text{falls } t = 0 \text{ und } i \in \{1, .., n\}, \\ \text{set}(\#\#\#) & \text{falls } t = 0 \text{ und } i \notin \{1, .., n\}, \\ \text{set}(s_i) & \text{falls } i \in \{1, ..., n\} \text{ und } t = 2 * i + 1, \\ \text{step} & \text{sonst.} \end{cases}$$

Wobei s_i für $i \in \mathbb{Z}$ wie folgt definiert ist:

$$s_i := \begin{cases} \binom{c_{3i-2}}{\Delta_C^{3i-3}(c)_1} \binom{c_{3i-1}}{\Delta_C^{3i-2}(c)_1} \binom{c_{3i}}{\Delta_C^{3i-1}(c)_1} \in (\Sigma \times Q_H)^3 & \text{falls } i \in \{1, ..., n\} \\ \#\#\# \in Q_C & \text{sonst} \end{cases}$$

Damit gilt aber gerade $s = S_3(c')$ wobei $S_3(c')$ die komprimierte c'-Konfiguration aus Definition 4.3.1 ist. Setze $S := S_3(C)$ auf die Speedup-Konstuktion, die drei Schritte in einem simuliert. Setze nun $A := A(M, f, S, \{\#\})$ aus Definition 5.1.1. Dann gilt nach Satz 5.2.2:

$$(\Delta_A^{|w|}(c)_1)_a = \Delta_S^{n-1}(s)$$

Da $\#$ in C und damit auch $\#\#\#$ in S tot ist, gilt damit nach Satz 4.3.2:

$$\gamma_\#((\Delta_A^{|w|}(c)_1)_a) = \gamma_\#(\Delta_S^{n-1}(s)) = \Delta_C^{|w|-1}(c')$$

Mit entsprechender Wahl der akzeptierenden Zustände und einem konstanten Speedup von einem Schritt durch Satz 4.1.3 folgt damit $L \in \mathcal{L}(\mathrm{CA}^{\text{RT-L}})$. □

Korollar 6.2.13. *Sei $L \in \mathcal{L}(CA^{RT\text{-}L})$. Der Advice $\mathcal{A}_L$ ist $CA^{RT\text{-}L}$-verträglich:*

$$\mathcal{A}_L(w) := v \in \mathbb{B}^{|w|} \text{ mit } v_i := \begin{cases} 1 & \text{falls } w_{[1...i]} \in L \\ 0 & \text{sonst} \end{cases}$$

Beweis. Nach Wahl von L gibt es einen Zellularautomaten $C \in \mathrm{CA}^{\mathrm{RT\text{-}L}}$ mit $L_C = L$. Damit gilt für ein $w \in \Sigma_C^*$ und alle $t \in \{0, ..., |w| - 1\}$:

$$\Delta_C^t([w])_1 \in \begin{cases} F_C^+ & \text{falls } w_{[1...t+1]} \in L \\ Q \setminus F_C^+ & \text{sonst} \end{cases}$$

Indem Zustände aus F_C^+ als 1 und Zustände aus $Q \setminus F_C^+$ als 0 aufgefasst werden, folgt mit Theorem 6.2.12 die Behauptung. □

Wenn L_1 also eine Echtzeit-Sprache ist und mit $\mathcal{A}_{L_1}$ gezeigt werden kann, dass L_2 eine Echtzeit-Sprache ist, dann ist auch der Advice $\mathcal{A}_{L_2}$ Echtzeit-verträglich und kann beispielsweise dazu benutzt werden, um zu zeigen, dass eine Sprache L_3 von einem Echtzeit-Zellularautomaten erkannt werden kann!

Die nächste Frage, die sich auftut, ist, ob es interessante Echtzeit-verträgliche Advices gibt, die nicht präfixstabil sind. Diese Advices können offensichtlich nicht durch Theorem 6.2.12 konstruiert werden, da $\mathcal{A}_H$ für jede Wahl von H präfixstabil ist. $\mathcal{A}_2$ aus Korollar 6.2.9 ist tatsächlich nicht präfixstabil und Echtzeit-verträglich, aber auch nicht besonders spannend. Der folgende Satz zeigt beschreibt ein „spannendes“, Echtzeit-verträgliches und nicht-präfixstabiles Advice. Tatsächlich ist der im folgenden beschriebene Advice der Grund, warum die erweiterte Nakamura-Konstruktion aus Kapitel 5 die Rücksetz-Funktionalität besitzt.

Satz 6.2.14. *Der folgende Advice $\mathcal{A}$ ist $CA^{RT\text{-}L}$-verträglich:*

$$\mathcal{A}(w) := \mathrm{ones}(2^{\lfloor \log_2 |w| \rfloor - 1}, |w|) \in \mathbb{B}^*$$

Beweis. Wegen Korollar 6.2.13 kann angenommen werden, dass die Zellen des Eingabewortes, deren Position eine Zweierpotenz ist, markiert sind (in Abbildung 6.1 durch rote Vierecke zu sehen). Die Idee ist nun, diese markierten Zellen als Signal für eine neue Mitte zu verwenden. Offensichtlich müssen frühere Berechnungen, die von anderen „Mitten“ ausgegangen sind, zurückgesetzt und neu gestartet werden. Um für die Simulation genügend Zeit zu haben, wird dazu der Ausgangsautomat nun unabhängig zwei Mal mithilfe der in Kapitel 5 vorgestellten Konstruktion simuliert. In Abbildung 6.1 werden die Setz-Befehle für die eine Simulation durch grüne Ovale und für die andere durch blaue dargestellt. In allen anderen Zuständen wird jeweils versucht, einen Simulationsschritt durchzuführen. Die markierten Zellen setzen dann jeweils abwechselnd den einen oder den anderen simulierten Automaten zurück und markieren jeweils eine Mitte mit größerer Position. Die erste Zelle verwendet für das Akzeptanzverhalten dann jeweils die eine oder die andere Simulation. Da sich die Abstände jeweils verdoppeln, reicht die Zeit aus.

□

Auf eine formale Beschreibung der Konstruktion bzw. einen formalen Beweis der Korrektheit wurde hier aus Verständlichkeits-, Umfangs- und Zeitgründen verzichtet. Da viele Überlegungen in Satz 6.2.14 gesteckt wurden und dies insbesondere die Rücksetz-Funktionalität der erweiterten Nakamura-Konstruktion begründet hat, wurde der Satz trotzdem aufgeführt.

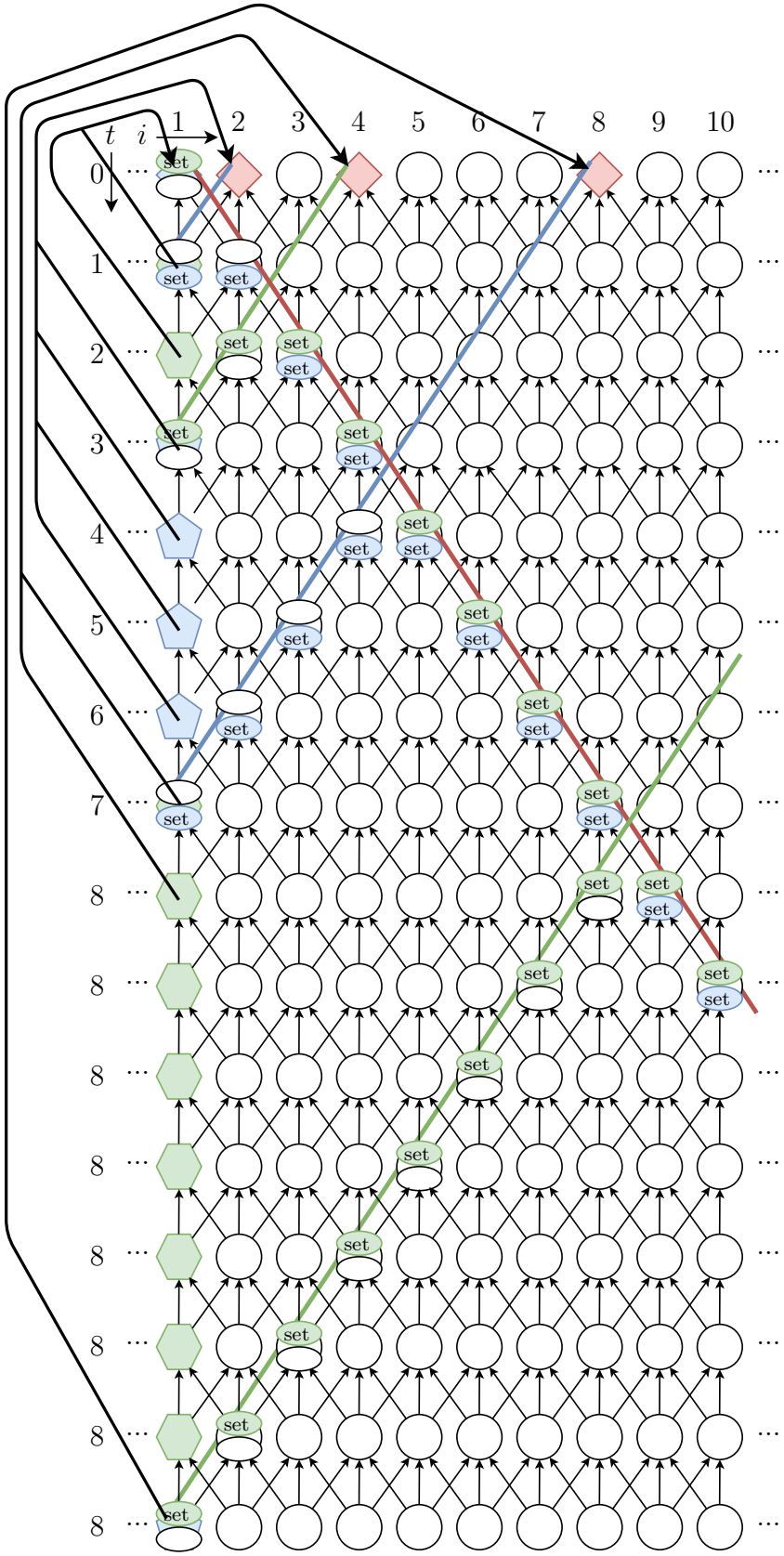

Abbildung 6.1: Visualisierung der Ausführung

6.3 Ausblick

Folgende Advices wurden außerdem noch auf Echtzeit-Verträglichkeit untersucht:

- $\mathcal{A}_1(w) := v \in \mathbb{B}^{|w|}$ mit $v_i = \texttt{1} \Leftrightarrow i = \lfloor \frac{|w|}{2} \rfloor$
- $\mathcal{A}_2(w) := v \in \mathbb{B}^{|w|}$ mit $v \in \texttt{0}^*\texttt{1}\texttt{0}^*$
- $\mathcal{A}_3(w) := \mathbb{B}^{|w|}$

Wie diese Advices Echtzeit-Zellularautomaten erweitern, ist nicht klar geworden. $\mathcal{A}_3$ hilft Echtzeit-Zellularautomaten sogar ggf. NP-schwere Probleme zu lösen. Wenn $\mathcal{A}_3$ Echtzeit-verträglich ist, so ist es auch $\mathcal{A}_2$. Wenn $\mathcal{A}_2$ Echtzeit-verträglich ist, so ist es auch $\mathcal{A}_1$.

Da bei $\mathcal{A}_1$ Echtzeit mit Linearzeit für unäre Sprachen zusammenfällt und die Markierung der Mitte ein äußerst schwieriges Unterfangen darstellt, vermutet der Autor, dass schon $\mathcal{A}_1$ nicht Echtzeit-verträglich ist. Interessant ist auch die Frage, ob $\mathcal{A}_2$ Echtzeit-verträglich ist, wenn angenommen wird, dass $\mathcal{A}_1$ Echtzeit-verträglich ist, bzw. allgemeiner, ob unter dieser Annahme Echtzeit mit Linearzeit zusammenfällt.

7. Eingeschränkte Zellularautomaten

In diesem Kapitel wird die Auswirkung der Forderung nach passiven Zellen im Eingabewort bei Echtzeit-Zellularautomaten untersucht.

7.1 Definition

Zunächst wird definiert, inwiefern Echtzeit-Zellularautomaten eingeschränkt werden. Dazu werden Advices aus dem vorherigen Kapitel verwendet und ein entsprechender Einschränkungs-Funktor definiert.

Definition 7.1.1. Zu einer gegebenen Funktion $d : \mathbb{N}_0 \to \mathbb{N}$ kann ein Σ-$\mathbb{B}$-Advice d^* wie folgt konstruiert werden:

$$d^*(w) := \text{ones}(d(|w|), |w|)^{Rev}$$

Dies definiert den Restr-CA-Funktor zum Einschränken von Zellularautomaten:

$$\begin{aligned}&\text{Restr}(d)(M \subseteq \text{CA}) := \\ &\qquad \{C \in \text{Adv}(d^*)(M) \mid (\Sigma_C \times \{0\}) \cup \{\#_C\} \text{ ist eine passive Zustandsmenge}\}\end{aligned}$$

Es mag auf den ersten Blick seltsam sein, dass Advices, die Zellularautomaten ggf. mächtiger machen, dazu verwendet werden, Zellularautomaten einzuschränken. Tatsächlich werden die Sprachklassen durch den Restr-Funktor nicht nur verkleinert, wie Satz 7.2.7 zeigt. Mit sinnvollen Wahlen der Einschränkungsfunktion d ergeben sich aber auch sinnvolle Einschränkungen.

7.2 Ergebnisse

Zunächst wird bemerkt, dass der k-Schritt Speedup für Zellularautomaten aus Satz 4.1.3 auch für eingeschränkte Zellularautomaten gilt.

Satz 7.2.1 (k-Schritt Speedup für eingeschränkte Zellularautomaten)**.** *Sei d eine Funktion $d : \mathbb{N}_0 \to \mathbb{N}$. Es gilt:*

$$\mathcal{L}(CA^{RT\text{-}L\text{-Restr}(d)}) = \mathcal{L}(CA^{TH\text{-}L\text{-time}(\{n\mapsto\max\{0,n+k-1\} \mid k\in\mathbb{N}_0\})\text{-Restr}(d)})$$

Beweis. Siehe Beweis von Satz 4.1.3: Satz 2.4.2 erhält passive #-enthaltende Mengen und in der Konstruktion des Automaten C' in Satz 4.1.3 kann $\Sigma_C \times \{0\}$ mit sich selbst und $q \in \Sigma_C \times \{1\}$ mit $\phi_\#(q)$ identifiziert werden, ohne dass der Beweis seine Gültigkeit verliert. Die Zelle am rechten Rand ist schließlich immer Element von $q \in \Sigma_C \times \{1\}$. Passive #-enthaltende Mengen bleiben also insgesamt erhalten, sodass der konstruierte Automat den Bedingungen eingeschränkter Zellularautomaten entspricht. □

7.2.1 Relation $\sim_{k,d,L}$

Es hat sich als beweistechnisch sinnvoll herausgestellt, eine Relation $\sim_{k,d,L}$ als Verallgemeinerung zur Nerode-Äquivalenzrelation zu betrachten.

Definition 7.2.2. Seien $k \in \mathbb{N}_0$, d eine Funktion $d : \mathbb{N}_0 \to \mathbb{N}$ und $L \subseteq \Sigma^*$. Angelehnt an die Nerode-Äquivalenzrelation wird auf Σ^* die Relation $\sim_{k,d,L}$ definiert. Seien dazu $v_1, v_2 \in \Sigma^*$.

$$v_1 \sim_{k,d,L} v_2 \;:\Leftrightarrow\; \forall w \in \Sigma^* : d(|wv_1|) = d(|wv_2|) = k \Rightarrow (wv_1 \in L \Leftrightarrow wv_2 \in L)$$

$\sim_{k,d,L}$ ist allerdings im Allgemeinen keine Äquivalenzrelation!

Eng mit der eingeführten Relation $\sim_{k,d,L}$ ist die Funktion $f_{C,k}$ verwandt, wie Satz 7.2.4 zeigt.

Definition 7.2.3. Sei $C \in \mathrm{CA}^{\text{RT-L-Restr}(d)}$ für ein $d : \mathbb{N}_0 \to \mathbb{N}$. Definiere die Funktion $f_{C,k} : \Sigma^* \to Q_C^*$ wie in Abbildung 7.1 gezeigt:

$$f_{C,k}(v) := w[1..\min(k+1, |v|)] \;\text{ mit }\; w := \Delta_C^{\max(0,|v|-k-1)}(\mathrm{join}(\begin{pmatrix} v \\ \mathrm{ones}(k,|v|)^{Rev} \end{pmatrix}))$$

Satz 7.2.4. *Sei $C \in CA^{RT\text{-}L\text{-Restr}(d)}$ für ein $d : \mathbb{N}_0 \to \mathbb{N}$. Es gilt für alle $k \in \mathbb{N}_0$ und $v_1, v_2 \in \Sigma^*$:*

$$f_{C,k}(v_1) = f_{C,k}(v_2) \Rightarrow v_1 \sim_{k,d,L} v_2$$

Beweis. Sei $k \in \mathbb{N}_0$. Seien $v_1, v_2 \in \Sigma^*$ so, dass $f_{C,k}(v_1) = f_{C,k}(v_2)$. Sei $w \in \Sigma^*$ mit $d(|wv_1|) = d(|wv_2|) = k$. Es bleibt zu zeigen, dass $wv_1 \in L$ genau dann gilt, wenn $wv_2 \in L$.

Angenommen, $|v_1| < k+1$. Dann

$$|f_{C,k}(v_2)| = |f_{C,k}(v_1)| = |v_1| < k+1$$

und damit $|f_{C,k}(v_2)| = \min(k+1, |v_2|) = |v_2|$, also $|v_1| = |v_2|$. Wegen $\max(0, |v_{i\in\{1,2\}}| - k - 1) = 0$ folgt $v_1 = v_2$, also gilt offensichtlich $v_1 \sim_{k,d,L} v_2$.

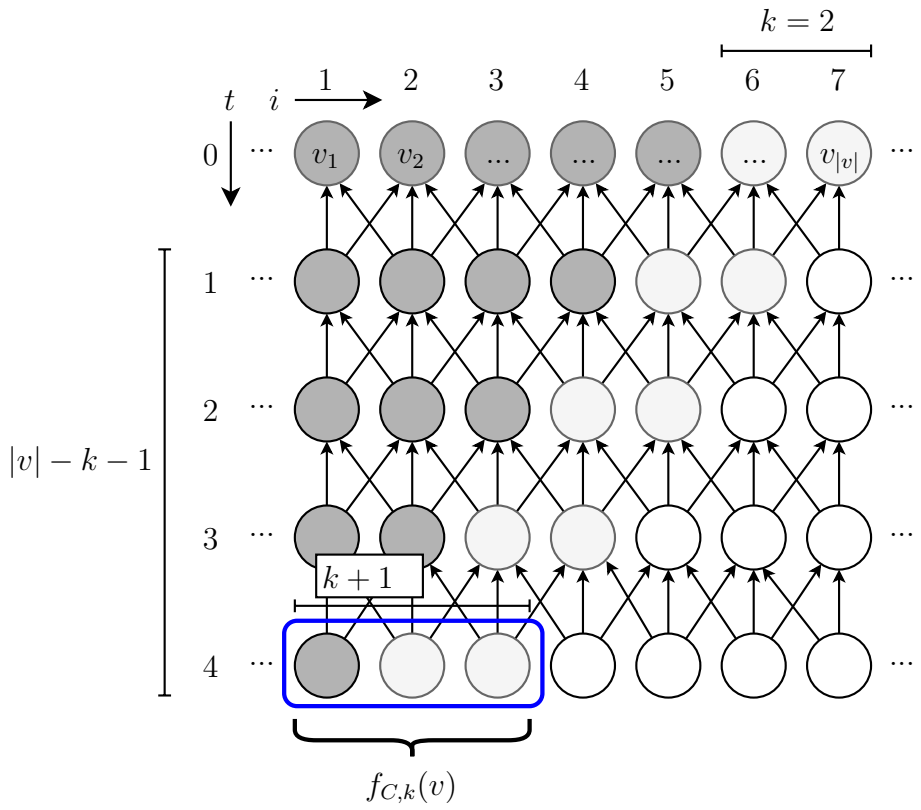

Abbildung 7.1: Berechnung von $f_{C,k}(v)$ für $k = 2$
Dunkelgraue Zustände sind passiv. Hellgraue Zustände sind für das Akzeptanzverhalten relevant.

Seien nun OBdA. $|v_1|, |v_2| > k$. Setze $u_{i\in\{1,2\}} := \text{join}(\begin{pmatrix} wv_i \\ d^*(wv_i) \end{pmatrix})$ und $c_i^t := \Delta_C^t(u_i)$.

Es gilt wegen der Forderung nach passiven Zuständen (rote Markierung in Abbildung 7.2, graue Zellen sind passiv):

$$(c_1^{|v_1|-k-1})_{[1..|w|]} = \text{join}(\begin{pmatrix} w \\ 0^{|w|} \end{pmatrix}) = (c_2^{|v_2|-k-1})_{[1..|w|]}$$

Und es gilt (blaue Markierung in Abbildung 7.2):

$$(c_1^{|v_1|-k-1})_{[|w|+1..|w|+k+1]} = |f_{C,k}(v_1)| = |f_{C,k}(v_2)| = (c_2^{|v_2|-k-1})_{[|w|+1..|w|+k+1]}$$

Also gilt zusammen (grüne Markierung in Abbildung 7.2):

$$(c_1^{|v_1|-k-1})_{[1..|w|+k+1]} = (c_2^{|v_2|-k-1})_{[1..|w|+k+1]}$$

Und damit ist auch der Zustand f aus Abbildung 7.2 in beiden Fällen gleich, es folgt also schließlich $wv_1 \in L \Leftrightarrow wv_2 \in L$. □

Damit ergibt sich dann folgende praktische Aussage.

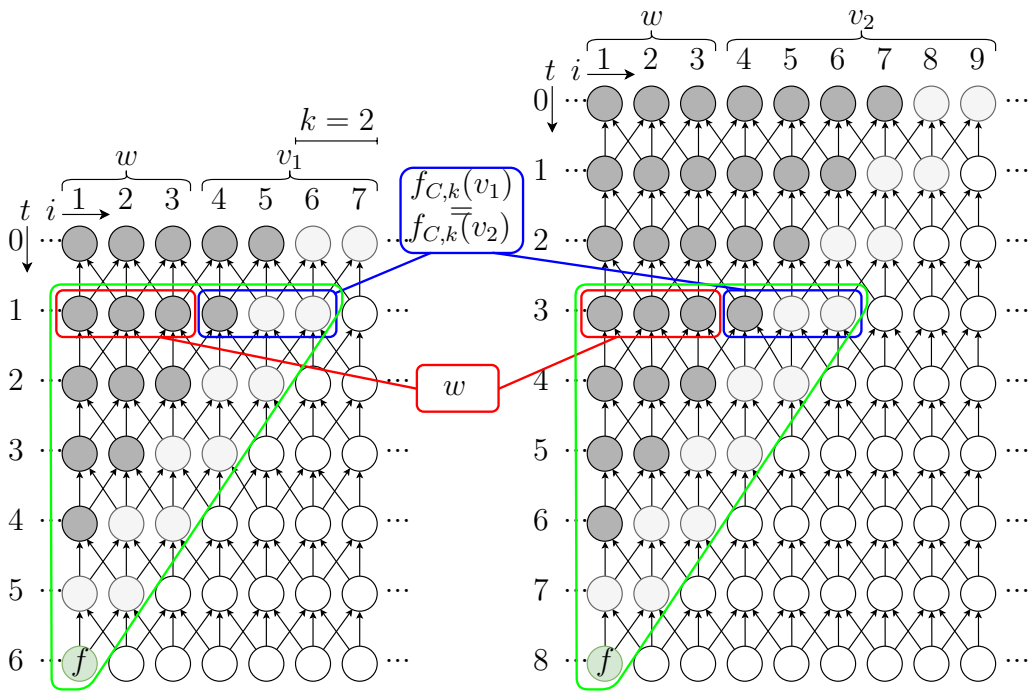

Abbildung 7.2: $f_{C,k}(v_1) = f_{C,k}(v_2) \Rightarrow v_1 \sim_{k,d,L} v_2$

Korollar 7.2.5. *Sei $C \in CA^{RT\text{-}L\text{-}\mathrm{Restr}(d)}$ für ein $d : \mathbb{N}_0 \to \mathbb{N}$, $k \in \mathbb{N}_0$ und $M \subseteq \Sigma^*$ eine Menge paarweiser nicht-ähnlicher Wörter bezüglich $\sim_{k,d,L}$. Es gilt:*

$$|M| \leq |f_{C,k}(\Sigma^*)| \leq |Q_C|^{k+2}$$

Beweis. Angenommen, $|M| > |f_{C,k}(\Sigma^*)|$. Wegen des Schubfachprinzips gibt es $m_1, m_2 \in M$ mit $f_{C,k}(m_1) = f_{C,k}(m_2)$ und wegen Satz 7.2.4 folgt mit $m_1 \sim_{k,d,L} m_2$ ein Widerspruch zur Wahl von M.

Die zweite Ungleichung folgt aus $|f_{C,k}(\cdot)| \leq k + 1$. □

7.2.2 Konstante Einschränkung

Nicht verwunderlich, kann ein Zellularautomat, der nur endlich viele Zellen zur Berechnung verwenden darf, auch nur reguläre Sprachen erkennen. Dies ist eine direkte Folgerung aus Korollar 7.2.5.

Satz 7.2.6. $\forall c \in \mathbb{N}, d(\cdot) := c : \mathcal{L}(CA^{RT\text{-}L\text{-}\mathrm{Restr}(d)}) = REG$

Beweis. Da $\sim_{c,d,L}$ für eine Sprache L gerade mit der Nerode-Äquivalenzrelation übereinstimmt, folgt aus Korollar 7.2.5, dass es nur endlich viele Nerode-Äquivalenzklassen gibt und damit die Regularität von L.

Umgekehrt ist die rechteste Zelle der Eingabe nie passiv. Zum Erkennen einer regulären Sprache L kann dann der endliche Automat, der L^{Rev} erkennt, benutzt werden, um einen entsprechenden Zellularautomaten zu konstruieren, der dann trotz Einschränkung L erkennt. □

Sind es zwar weiterhin nur endliche viele Zellen, die die Berechnung benutzen darf, z.B. eine oder zwei, hängt die genaue Anzahl aber wieder frei von der Länge des Eingabewortes ab, so lassen sich sogar unentscheidbare Sprachen erkennen.

Satz 7.2.7. *$\exists d : \mathbb{N}_0 \to \mathbb{N}$ mit $d(\cdot) \leq 2$, sodass $\mathcal{L}(CA^{RT\text{-}L\text{-}\mathrm{Restr}(d)})$ eine unentscheidbare Sprache enthält.*

Beweis. Es gibt einen Zellularautomaten, der das Eingabewort genau dann akzeptiert, wenn vom Advice genau zwei Zellen mit einer `1` markiert wurden. Die Funktion d kann so abhängig von der Länge des Wortes Informationen über unentscheidbare Sprachen an den Zellularautomaten weitergeben. □

7.2.3 Vergleich verschiedener Einschränkungen

Nicht verwunderlich, schränkt es einen Echtzeit-Zellularautomat kaum ein, wenn zur Berechnung nur endlich viele Zellen nicht verwendet werden dürfen.

Satz 7.2.8. *Sei $c \in \mathbb{N}_0$. Dann gilt für $d(n) := n - c$:*

$$\mathcal{L}(CA^{RT\text{-}L\text{-}\mathrm{Restr}(d)}) = \mathcal{L}(CA^{RT\text{-}L})$$

Beweis. Korollar 6.2.9 zeigt die Inklusion „$\subseteq$“. Für die andere Inklusion kann die asynchrone Simulations-Konstruktion aus Kapitel 5 verwendet werden, um dann mit einem entsprechenden Speedup-Faktor (siehe Definition 4.3.1) die eigentliche Berechnung ausführen, sobald nach c Schritten keine Zelle im Eingabewort mehr passiv ist. □

Das wohl interessanteste Ergebnis dieses Kapitels ist das folgende: Je weniger Zellen dem Automat für die Berechnung zur Verfügung stehen (asymptotisch in Abhängigkeit zur Wortlänge gesehen), desto weniger Sprachen kann er erkennen.

Satz 7.2.9. *Seien $d_1, d_2 : \mathbb{N}_0 \to \mathbb{N}$ zwei Funktionen, sodass $d_1(n) \leq \frac{n}{2}$ und $\liminf_{n\to\infty} \frac{d_2(n)}{d_1(n)} = 0$. Dann gibt es ein $L \in \mathcal{L}(CA^{RT\text{-}L\text{-}\mathrm{Restr}(d_1)}) \setminus \mathcal{L}(CA^{RT\text{-}L\text{-}\mathrm{Restr}(d_2)})$.*

Beweis. Wähle $L := \{w^{Rev}vw \mid v, w \in \mathbb{B}^*,\ |w| = d_1(|wvw|)\}$.

Es gilt $L \in \mathcal{L}(\mathrm{CA}^{\mathrm{RT\text{-}L\text{-}Restr}(d_1)})$, da die nicht-passiven Zellen genau das letzte Vorkommen von w markieren. Die aktiven Zeichen wandern dann nach links und versuchen, den Anfang von w^{Rev} zu finden. Bereits gefundenen Anfänge werden dann Schritt für Schritt verlängert. Anfänge, die nicht fortgesetzt werden können, werden verworfen.

Es gilt aber auch $L \notin \mathcal{L}(\mathrm{CA}^{\mathrm{RT\text{-}L\text{-}Restr}(d_2)})$: Sei $n \in \mathbb{N}_0$ und $k := d_1(n)$. Wegen $d_1(n) \leq \frac{n}{2}$ gibt es ein $v \in \mathbb{B}^*$, sodass $n = |v| + 2k$. Seien $w_1, w_2 \in \mathbb{B}^k$ zwei verschiedene Wörter der Länge k. Dann gilt $w_1^{Rev}vw_1 \in L$, aber $w_2^{Rev}vw_1 \notin L$, also $w_1 \not\sim_{d_2(n),d,L} w_2$. Damit ist $\mathbb{B}^k$ eine Menge paarweiser nicht-ähnlicher Wörter und $|\mathbb{B}^k| = 2^k$.

Angenommen, es gibt ein $C \in \mathrm{CA}^{\mathrm{RT\text{-}L\text{-}Restr}(d_2)}$ mit $L_C = L$.

Nach Korollar 7.2.5 gilt:

$$\forall n \in \mathbb{N}_0 : 2^{d_1(n)} = |\mathbb{B}^k| \leq |Q_C|^{d_2(n)+2} = 2^{\log_2(|Q_C|)(d_2(n)+2)}$$

Also gilt:

$$1 \leq \log_2(|Q_C|)(\frac{d_2(n)}{d_1(n)} + \frac{2}{d_1(n)})$$

Laut Annahme gibt es nun eine monoton steigende Folge a_n, sodass $\frac{d_2(a_n)}{d_1(a_n)}$ beliebig klein wird. Da dafür $d_1(a_n)$ beliebig groß wird, folgt ein Widerspruch zur Annahme. □

Anmerkung. Satz 7.2.9 besagt allerdings nicht, dass die Sprachklassen für langsamer wachsende Einschränkungen Teilmengen sind: Satz 7.2.7 zeigt ja gerade, dass selbst Automaten mit fast konstanter Einschränkung unentscheidbare Sprachen entscheiden können, die offensichtlich nicht von Automaten mit Einschränkung $d(n) = \frac{n}{2}$ erkannt werden können.

8. Zusammenfassung

Wie erwartet konnte die usprüngliche Fragestellung, ob Echtzeit mit Linearzeit zusammenfällt, nicht weiter geklärt werden - das Problem bleibt offen.

Allerdings hat Kapitel 6 gezeigt, dass Echtzeit-Zellularautomaten erstaunlich mächtig sind und mit Theorem 6.2.12 ein bemerkenswertes Werkzeug zur Verfügung gestellt, mit dem Echtzeit-Zellularautomaten in gewisser Hinsicht hintereinandergeschaltet werden können, ohne dabei in den Bereich der Linearzeit zu geraten. Es wurden Advices aufgezeigt, für die es lohnenswert sein könnte, sie weiter auf Echtzeit-Verträglichkeit zu untersuchen.

Die erweiterte Nakamura-Konstruktion aus Kapitel 5 hat sich durch seine Rücksetz-Funktionalität von Startzuständen ebenfalls als sehr praktisches Werkzeug zum asynchronen Simulieren von Automaten in Echtzeit herausgestellt. Detailliertere Korrektheitsbeweise der Konstruktion stehen allerdings noch aus.

Kapitel 7 hat gezeigt, dass sich Echtzeit-Zellularautomaten durch Vorgabe von passiven Zellen in der Eingabe bis hin zu regulären Sprachen einschränken lassen. Es bleibt offen, für welche Einschränkungen sich eine Hierarchie in den jeweiligen Sprachklassen ergibt: Satz 7.2.9 zeigt, dass Automaten, die stärkeren Einschränkungen unterliegen, bestimmte Sprachen nicht mehr erkennen können. Satz 7.2.7 zeigt allerdings, dass mitunter die stärksten Einschränkungen plötzlich unentscheidbare Sprachen erkennen lassen, die weniger starke Einschränkungen noch ausgeschlossen haben.

Literaturverzeichnis

[ChCu84] C. Choffrut und K. Culik. On real-time cellular automata and trellis automata. *Acta Informatica* 21(4), Nov 1984, S. 393–407.

[IbJi88] O. H. Ibarra und T. Jiang. Relating the power of cellular arrays to their closure properties. *Theoretical Computer Science* 57(2), 1988, S. 225 – 238.

[Kutr09] M. Kutrib. *Cellular Automata and Language Theory*, S. 800–823. Springer New York, New York, NY. 2009.

[MaRe92] J. Mazoyer und N. Reimen. A linear speed-up theorem for cellular automata. *Theoretical Computer Science* 101(1), 1992, S. 59 – 98.

[Wors10] T. Worsch. A note on (intrinsically?) universal asynchronous cellular automata. *Proceedings Automata*, 2010, S. 339–350.