

Seminar

Zellularautomaten und diskrete komplexe Systeme
im Sommersemester 2016

Ausarbeitung

von **Philipp Adolf**, Matr.nr. 1669101

Thema

Jérôme Durand-Lose (2001)

*Representing Reversible Cellular Automata with Reversible Block
Cellular Automata*

Discrete Models: Combinatorics, Computation, and Geometry, DM-
CCG 2001, Band **AA**, S. 145–154

Erklärung

gemäß §6 (11) der Prüfungsordnung Informatik (Master)

Ich versichere wahrheitsgemäß, die Seminausarbeitung zum Seminar „Zellularautomaten und diskrete komplexe Systeme“ im Sommersemester 2016 selbstständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Abänderungen entnommen wurde.

(Philipp Adolf, Matr.nr. 1669101)

Inhaltsverzeichnis

1	Einleitung	4
1.1	Definitionen	4
1.1.1	Zellularautomaten	5
1.1.2	Blockpermutationen	5
1.1.3	Reversibilität	5
1.1.4	Simulation	6
2	Simulation von reversiblen Zellularautomaten durch Blockpermutationen	7
2.1	Partitionierung	7

Kapitel 1

Einleitung

Zellularautomaten sind ein beliebtes Modell, um parallele Berechnungen zu beschreiben und zu analysieren. Dabei wird der Zustand eines Zellularautomaten über die Zustände seiner Zellen festgelegt – jede Zelle hat die gleiche Menge an möglichen Zuständen. Für die eigentlichen Berechnungen werden lokale Überföhrungsfunktionen angegeben.

Eine besondere Klasse von Zellularautomaten sind jene, bei denen für jede Konfiguration ein eindeutiger Vorgänger existiert. Diese sogenannten reversiblen Zellularautomaten eignen sich besonders zur Simulation von isolierten Systemen, also solchen, die weder Energie noch Masse mit ihrer Umgebung austauschen. Das Problem hierbei ist, dass bei Zellularautomaten mit zwei oder mehr Dimensionen die Frage der Umkehrbarkeit unentscheidbar ist.

Bei Blockzellularautomaten ist dies hingegen einfach entscheidbar. Bei dieser Art von Zellularautomaten gibt es keine lokale Überföhrungsfunktion, die den nächsten Zustand einer Zelle angibt. Stattdessen wird eine Funktion angegeben, die für einen ganzen Block von Zellen die nächsten Zustände berechnet. Dabei bekommt diese Funktion nur einen Block als Eingabe, sie kann nicht auf umliegende Blöcke zugreifen. Damit trotzdem zwischen den Blöcken Informationen ausgetauscht werden können, werden die Grenzen der Blöcke bei aufeinanderfolgenden Schritten verschoben.

In [Dur01] stellt Durand-Lose eine Konstruktion vor, die zeigt, wie sich ein beliebiger reversibler Zellularautomat durch einen reversiblen Blockzellularautomaten simulieren lässt.

1.1 Definitionen

Im folgenden sei

$$[a, b] = \{x \mid x \in \mathbb{N} \wedge a \leq x \leq b\}$$

Also ist $[a, b]$ das geschlossene Intervall von a bis b (jeweils einschließlich). Dies erweitern wir wie folgt auf d -dimensionale Tupel:

$$[a, b]^d = \{(x_1, x_2, \dots, x_d) \mid x_i \in [a, b] \forall i \in [1, d]\}$$

1.1.1 Zellularautomaten

Ein Zellularautomat \mathcal{A} wird durch ein Tupel (d, S, r, f) definiert, wobei d die Dimensionalität des Automaten angibt. S ist die Menge der Zustände, die eine einzelne Zelle annehmen kann, und r ist der Radius der Nachbarschaft. Die lokale Überföhrungsfunktion ist durch $f : S^{(2r+1)^d} \rightarrow S$ gegeben.

Für einen Schritt wird für jede Zelle die lokale Überföhrungsfunktion ausgewertet um den Nachfolgezustand zu bestimmen. Dies passiert für alle Zellen gleichzeitig.

Die Funktion $c \in S^{\mathbb{Z}^d}$ wird (globale) Konfiguration genannt. Die globale Überföhrungsfunktion bildet eine globale Konfiguration auf eine neue globale Konfiguration ab, indem für jede Zelle die lokale Überföhrungsfunktion ausgewertet wird.

Wir schreiben kurz $c|_E$ für die Einschränkung von c auf $E \subset \mathbb{Z}^d$.

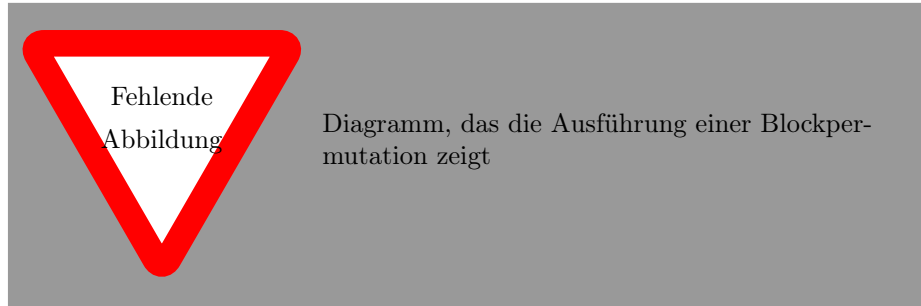
1.1.2 Blockpermutationen

Eine Blockpermutation wird durch (d, S, w, o, e) definiert. Dabei ist $w \in \mathbb{N}^+$ die Breite (engl. width). Das Volumen der Blockpermutation ist $V = [0, w-1]^d \subsetneq \mathbb{Z}^d$. Die Blockfunktion $e : S^V \rightarrow S^V$ ist eine Permutation.

Um die Blockpermutation anzuwenden, werden die Zellen in Blöcke eingeteilt. Diese Blöcke sind Hyperwürfel mit Kantenlänge w . Auf jeden Block wird dann die Blockfunktion f angewendet.

Etwas formaler: Um den Nachfolger einer globalen Konfiguration c zu berechnen, suchen wir für jede Zelle ihren nächsten Zustand. Sei $i \in \mathbb{Z}^d$ der Index einer beliebigen Zelle und $a = i \div w$ und $b = i \bmod w$ mit $a \in \mathbb{Z}^d$ und $b \in [0, w-1]^d$. Dann gibt a an, in welchem Block sich die Zelle befindet und b welche Position die Zelle in ihrem Block hat. Wir wenden nun die Blockfunktion an und nehmen aus dem Ergebnis den Zustand für diese Zelle:

$$T(c)_i = e(c|_{a*w+V})_b$$



1.1.3 Reversibilität

Zellularautomaten, Blockzellularautomaten und Blockpermutationen definieren Funktionen \mathcal{G} von $S^{\mathbb{Z}^d}$ nach $S^{\mathbb{Z}^d}$. Der Automat \mathcal{A} ist genau dann reversibel, wenn die zugehörige Funktion $\mathcal{G}_{\mathcal{A}}$ bijektiv ist und ein Automat \mathcal{B} mit $\mathcal{G}_{\mathcal{B}} = \mathcal{G}_{\mathcal{A}}^{-1}$ existiert.

Für eindimensionale Zellularautomaten existiert ein Algorithmus, der prüft, ob ein Automat umkehrbar ist oder nicht. Für höherdimensionale Zellularautomaten wurde gezeigt, dass dieses Problem unentscheidbar ist.

Blockpermutation sind hingegen durch ihre Konstruktion trivial reversibel: Man nimmt einfach die inverse Permutation bei gleicher Aufteilung. Blockzellularautomaten sind genau dann reversibel, wenn ihre Blockfunktion eine Permutation ist.

1.1.4 Simulation

Durand-Lose benutzt folgende Definition einer Simulation:

Gegeben zwei Funktionen, $f : F \rightarrow F$ und $g : G \rightarrow G$, sagen wir, dass f von g in linearer Zeit τ simuliert wird, wenn es zwei Kodierungsfunktionen $\alpha : F \rightarrow G$ und $\beta : G \rightarrow F$, sodass

$$\forall x \in F, \forall n \in \mathbb{N} \quad f^n(x) = \beta \circ g^{\tau n} \circ \alpha(x)$$

Anders ausgedrückt: Wir übersetzen eine Konfiguration eines Automaten in eine Konfiguration eines anderen Automaten, lassen diesen τ Schritte machen und übersetzen dann wieder zurück. Das Ergebnis ist dann äquivalent dazu, einen Schritt mit dem ursprünglichen Automaten zu machen.

Kapitel 2

Simulation von reversiblen Zellularautomaten durch Blockpermutationen

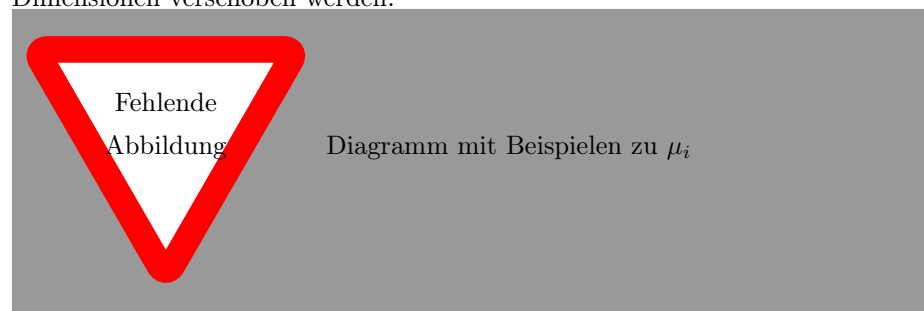
Die Konstruktion läuft in mehreren Schritten ab: Zuerst werden $d + 1$ Blockpermutationen angegeben, die, wenn sie nacheinander ausgeführt werden, den reversiblen Zellularautomaten simulieren. Anschließend wird gezeigt, dass sich die Blockpermutationen zu einer einzelnen Blockpermutation vereinigen lassen, so dass sich ein reversibler Blockzellularautomat ergibt.

2.1 Partitionierung

Betrachten wir zunächst die Hilfsmengen

$$\mu_i = (3i, 3i, \dots, 3i) + [r, 3(d+1)r - r - 1]^d \quad 0 \leq i \leq d$$

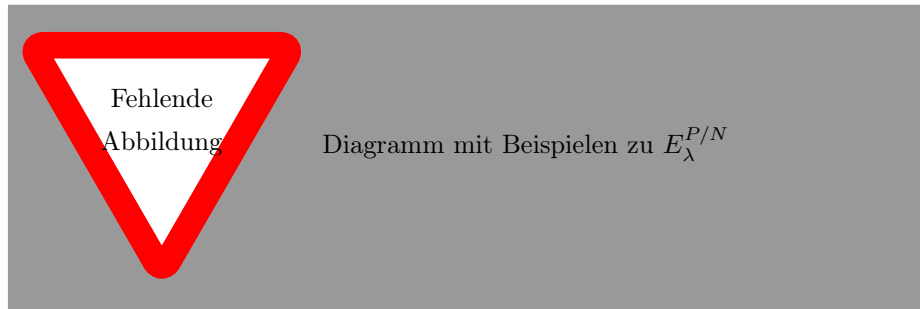
Diese Mengen sind Hyperwürfel, die mit wachsendem i gleichmäßig in allen Dimensionen verschoben werden.



Mit diesen Mengen können wir nun für $0 \leq \lambda \leq d$ folgende Mengen bilden:

$$E_\lambda^P = \bigcup_{\lambda \leq i \leq d} \mu_i$$

$$E_{\lambda}^N = \bigcup_{0 \leq i < \lambda} \mu_i$$



Dabei ist E_{λ}^P die Menge der Zellen, die in Schritt λ der Simulation noch den Ausgangszustand kennen (von engl. previous) und E_{λ}^N die Menge der Zellen, die bereits den nächsten Zustand kennen (von engl. next). Dadurch wird schon klar, dass für $\lambda = 0$, also zu Beginn der Simulation,

$$E_{\lambda}^P = V \quad \wedge \quad E_{\lambda}^N = \emptyset$$

gelten muss. Genauso gilt für $\lambda = d$

$$E_{\lambda}^P = \emptyset \quad \wedge \quad E_{\lambda}^N = V$$

Literatur

- [Dur01] Jérôme Durand-Lose. “Representing Reversible Cellular Automata with Reversible Block Cellular Automata”. In: *Discrete Models: Combinatorics, Computation, and Geometry, DM-CCG 2001*. Hrsg. von Robert Cori u. a. Bd. AA. DMTCS Proceedings. Discrete Mathematics und Theoretical Computer Science, 2001, S. 145–154. URL: <http://www.dmtcs.org/proceedings/html/dmAA0110.abs.html>.