

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	Blockpermutationen	7
2.2	Verschmelzen der Zustände und Permutationen	12

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:

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

Bei einer Blockpermutation mit Ursprung o (engl. origin) wird die Einteilung in Blöcke um o verschoben, bevor die Blockfunktion angewendet wird.

Wenn e eine Funktion aber nicht notwendigerweise eine Permutation ist, sprechen wir von einem Blockzellularautomaten statt einer Blockpermutation.

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

Das Ziel der Arbeit von Durand-Lose ist es, beliebige reversible Zellularautomaten mit reversiblen Blockzellularautomaten zu simulieren. Um dieses Ziel zu erreichen, beschreibt er zuerst eine Reihe von Blockpermutationen, die reversible Zellularautomaten simulieren. Anschließend beweist er, dass diese Blockpermutationen zu einem Blockzellularautomaten zusammengefasst werden können.

2.1 Blockpermutationen

Betrachten wir zunächst die Hilfsmengen

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

Diese Mengen sind Hyperwürfel, die mit wachsendem i gleichmäßig in allen Dimensionen verschoben werden. Ein Beispiel für diese Mengen ist in Abbildung 2.1 gegeben.

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

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

Beispiele für diese Mengen werden in Abbildungen 2.2 und 2.3 gezeigt.

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,

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

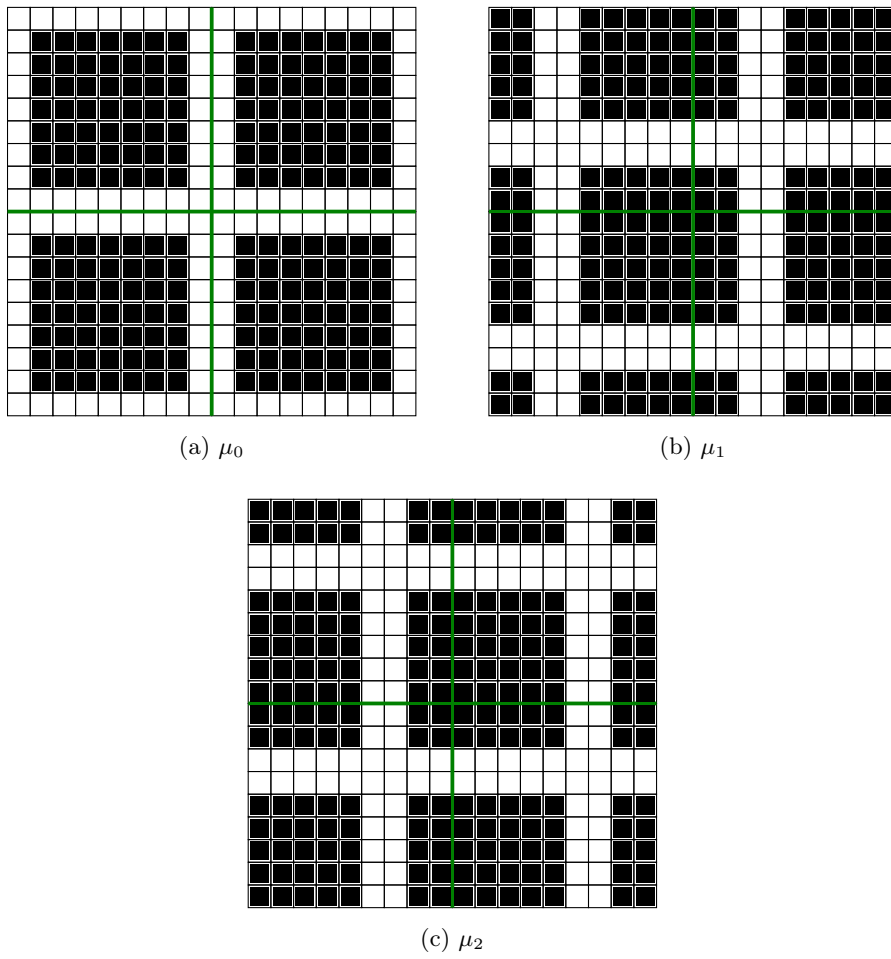
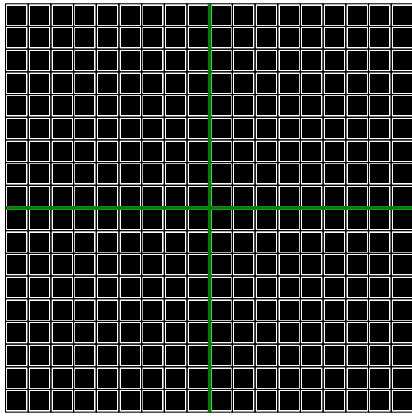
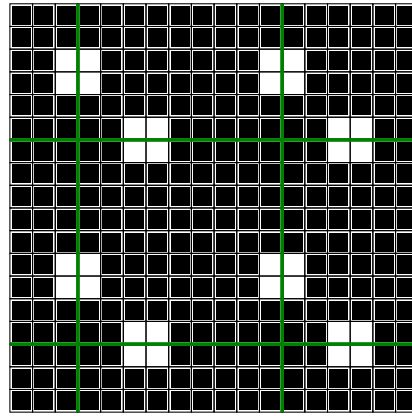


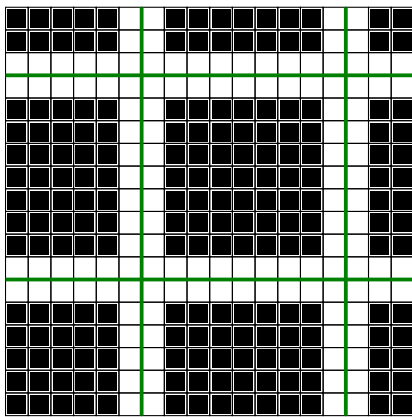
Abbildung 2.1: Die Hilfsmengen für $d = 2$ und $r = 1$. Es sind jeweils vier Blöcke gezeichnet, die Blockgrenzen sind grün markiert.



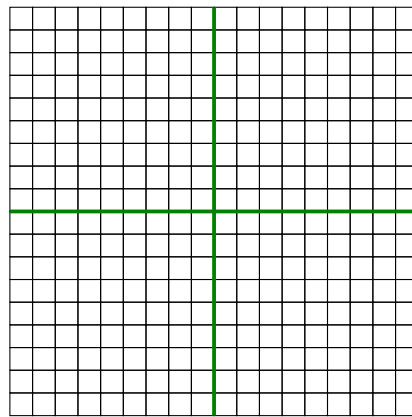
(a) E_0^P



(b) E_1^P

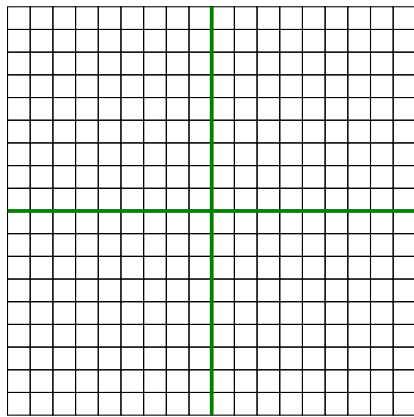


(c) E_2^P

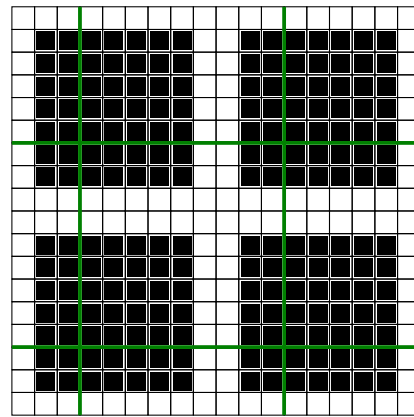


(d) E_3^P

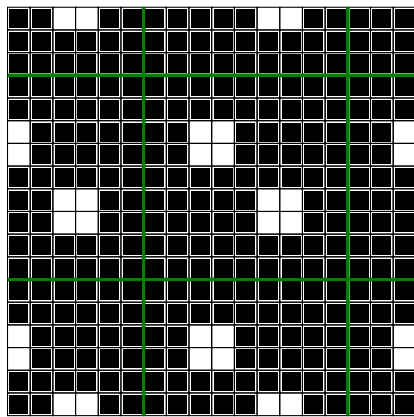
Abbildung 2.2: E_λ^P für je vier Blöcke. Die Blockgrenzen sind grün eingezeichnet (jeweils für den Schritt λ).



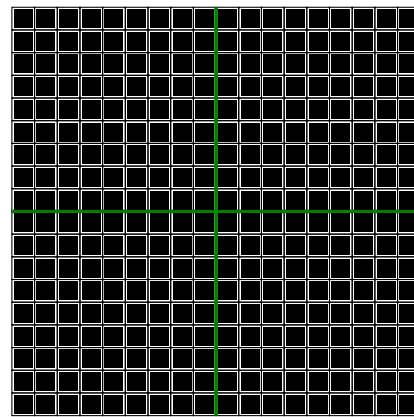
(a) E_0^N



(b) E_1^N



(c) E_2^N



(d) E_3^N

Abbildung 2.3: E_λ^N für je vier Blöcke. Die Blockgrenzen sind grün eingezeichnet (jeweils für den Schritt λ).

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

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

Lemma 3?

Die Blockpermutationen benutzen als Zustandsmenge $(S \cup \{\perp\}) \times (S \cup \{\perp\})$. Ein Teil wird zum Speichern des vorherigen Zustands benutzt, der andere zum Speichern des nächsten Zustands. Das Symbol \perp wird verwendet wenn ein Zustand fehlt.

Wir definieren nun folgende Konfigurationen:

$$\forall c \in C \quad \forall \lambda \in [0, d+1] \quad \mathcal{E}_\lambda(c) = (c|_{E_\lambda^P}, \mathcal{G}(c)|_{E_\lambda^N})$$

Dabei ist $\mathcal{E}_\lambda(c)$ die Konfiguration der Blockpermutationen nach λ Schritten der Berechnung von $\mathcal{G}(c)$.

Sei B_λ die Blockpermutation, die $\mathcal{E}_\lambda(c)$ auf $\mathcal{E}_{\lambda+1}(c)$ abbildet. B_λ hat Breite $3(d+1)r$ und Ursprung $3\lambda r$. Da wir bei jedem Schritt alte Zustände vergessen ($\mathcal{E}_{\lambda+1} \subsetneq \mathcal{E}_\lambda$) ist nicht offensichtlich, dass in jedem Schritt alle Informationen vorhanden sind, die benötigt werden.

Lemma 1. Für alle λ aus $[0, d]$ enthält $\mathcal{E}_\lambda(c)$ genug Informationen, um $\mathcal{E}_{\lambda+1}(c)$ zu berechnen.

Beweis. Die Menge der Zellen, für die ein neuer Zustand berechnet werden muss, ist

$$\Delta_\lambda = E_{\lambda+1}^N \setminus E_\lambda^N = \mu_\lambda \setminus \bigcup_{0 \leq i < \lambda} \mu_i$$

Für alle Zellen $x \in \Delta_\lambda$ gilt

$$x \in \mu_\lambda = (3\lambda r, 3\lambda r, \dots, 3\lambda r) + [r, 3(d+1)r - r - 1]^d$$

Dementsprechend sind x und alle seine Nachbarn in

$$\begin{aligned} & (3\lambda r, 3\lambda r, \dots, 3\lambda r) + [r - r, 3(d+1)r - r - 1 + r]^d \\ &= (3\lambda r, 3\lambda r, \dots, 3\lambda r) + [0, 3(d+1)r - 1]^d \end{aligned}$$

Das ist genau die Partitionierung der Blockpermutationen B_λ , da diese den Ursprung $(3\lambda r, 3\lambda r, \dots, 3\lambda r)$ und Breite $3(d+1)r - 1$ hat. Wir haben nun also gezeigt, dass alle Zellen, für die ein neuer Zustand berechnet werden soll, sowie alle ihre Nachbarn innerhalb eines Blockes liegen. Es bleibt zu zeigen, dass alle vorherigen Zustände, die dafür benötigt werden, noch vorhanden sind.

Sei $i \in [0, \lambda - 1]$. Aus der Definition von Δ_λ folgt, dass $x \notin \mu_i \forall x \in \Delta_\lambda$. Das heißt, dass es einen Index j_i gibt, so dass

$$x_{j_i} \notin 3ir + [r, 3(d+1)r - r - 1]$$

wobei x_{j_i} der j_i -te Eintrag des d -Tupels x ist. Daraus folgt, dass $x_{j_i} \in 3ir + [-r, r - 1]$. Da die Mengen $3ir + [-r, r - 1]$ für $i \in [0, \lambda - 1]$ disjunkt sind, heißt das, dass es λ paarweise verschiedene j_i geben muss.

Warum?

Sei y eine beliebige Zelle, die zur Berechnung den nächsten Zustands von x benötigt wird. y gehört zu $x + [-r, r]^d$. Für alle $i \in [0, \lambda - 1]$ ist y_{j_i} in $3ir + [-2r, 2r - 1]$. Nehmen wir an, dass ein y existiert, das nicht zu E_λ^P gehört. Dann gilt für alle $v \in [\lambda, d + 1]$, dass ein k_v existiert, sodass y_{k_v} nicht zu $vr + [r, 3(d + 1)r - r - 1]$ gehört, oder äquivalent $y_{k_v} \in 3vr + [-r, r - 1]$. Da die Mengen $3vr + [-r, r - 1]$ disjunkt sind, gibt es $d + 1 - \lambda$ paarweise verschiedene k_v .

Zusammen existieren $\lambda + d + 1 - \lambda = d + 1$ Werte für j_i und k_v , es gibt aber nur d mögliche Werte. Daraus folgt, dass i_0 und v_0 existieren, sodass $j_{i_0} = k_{v_0}$. Das bedeutet, dass die Schnittmenge von $3i_0r + [-2r, 2r - 1]$ und $3v_0r + [-r, r - 1]$ nicht leer ist. Das ist nur möglich, wenn $i_0 = v_0$, was aber nicht möglich ist, da $i_0 \in [0, \lambda - 1]$ und $v_0 \in [\lambda, d + 1]$, also $i_0 < v_0$.

Unsere Annahmen, dass $y \notin E_\lambda^P$ war also falsch und $y \in E_\lambda^P$. Damit sind alle vorherigen Zustände, die benötigt werden, vorhanden, und der nächste Zustand der Zelle x kann berechnet werden. \square

Durch die Symmetrie von E^N und E^P können die Zustände, die gelöscht werden, aus den verbleibenden vorherigen Zuständen sowie den neuen Zuständen berechnet werden. Das bedeutet, dass die partielle Funktion e_λ zu einer Permutation vervollständigt werden kann.

2.2 Verschmelzen der Zustände und Permutationen

Bisher haben wir $(S \cup \{\perp\})^2$ Zustände und $d + 1$ Permutation. In diesem Abschnitt zeigen wir, dass sich die Zustände auf $S \cup S^2$ reduzieren und zeigen, dass sich die Permutationen zu einer einzelnen Permutation zusammenfassen lassen.

Wenn wir nur noch eine Permutation haben, dann müssen wir "wissen", welchen Schritt diese gerade ausführen soll. Dies können wir herausfinden, indem wir uns anschauen, in welchen Zellen zwei Zustände vorhanden sind und in welchen nicht.

Lemma 2. *Die aktuelle Blockpermutation B_λ kann anhand der Positionen der Doppelzustände identifiziert werden.*

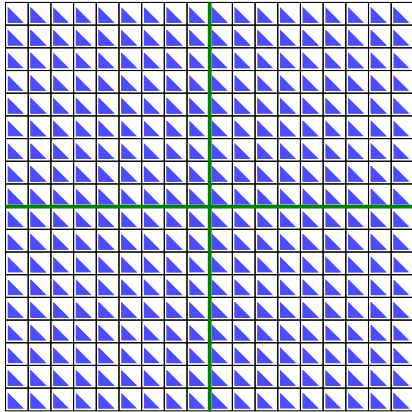
Beweis. Für $i \in [0, d]$ definieren wir:

$$\begin{aligned}\epsilon_0 &= (3\lambda r, 3\lambda r, \dots, 3\lambda r) \\ \epsilon_1 &= (3\lambda r, 3\lambda r, \dots, 3\lambda r) + (-3r, -3r, -3r, \dots, -3r) \\ \epsilon_2 &= (3\lambda r, 3\lambda r, \dots, 3\lambda r) + (-3r, -6r, -6r, \dots, -6r) \\ \epsilon_j &= (3\lambda r, 3\lambda r, \dots, 3\lambda r) + (-3r, -6r, -9r, \dots, -3(j-1)r, -3jr, \dots, -3jr) \\ \epsilon_d &= (3\lambda r, 3\lambda r, \dots, 3\lambda r) + (-3r, -6r, -9r, \dots, -3dr)\end{aligned}$$

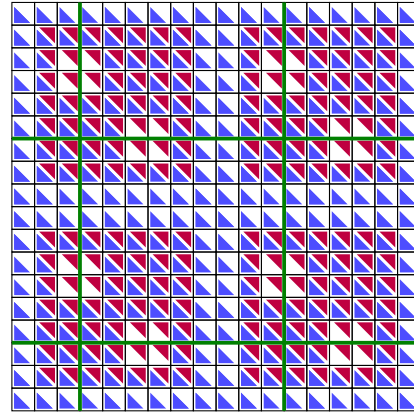
Für $\lambda \leq d$ gilt

$$\epsilon_j \in \mu_\lambda \subset E_\lambda^{P1}$$

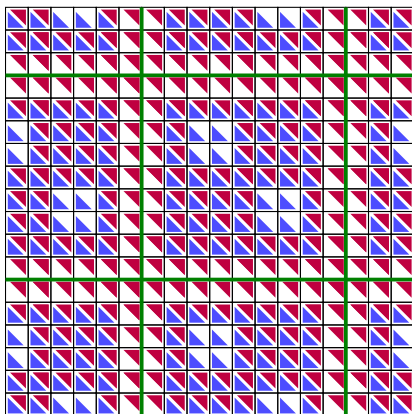
¹In [Dur01] steht hier E_λ^N . Dies scheint ein Tippfehler zu sein.



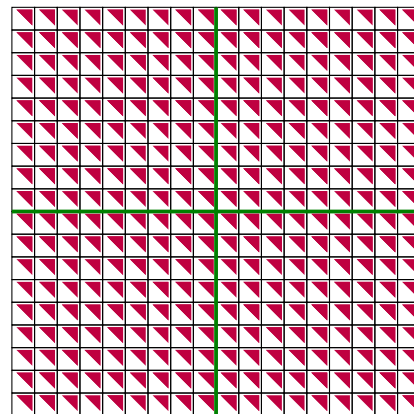
(a) Vor dem ersten Teilschritt.



(b) Nach dem ersten Teilschritt.



(c) Nach dem zweiten Teilschritt.



(d) Nach dem dritten Teilschritt.

Abbildung 2.4: Ein Simulationsschritt für einen Automaten mit $d = 2$ und $r = 1$. Zellen, die ihren vorherigen Zustand noch kennen sind blau markiert, Zellen, die schon ihren neuen Zustand kennen rot.

Der einzige Fall, in dem $\lambda \leq d$ nicht gegeben ist, ist, wenn ein Simulationsschritt abgeschlossen ist. Dann fängt aber ein neuer Simulationsschritt an und alle Zellen haben nur einen Zustand, der (für unsere Zwecke in diesem Beweis) sowohl als alter als auch als neuer Zustand betrachtet werden kann. ϵ_j ist damit immer in E_λ^P .

Das Tupel ϵ_j gehört genau dann zu E_λ^N , wenn $\epsilon_j \in \bigcup_{0 \leq i < \lambda} \mu_i$, also

$$(-3r, -6r, \dots, -3jr, -3jr) \in \bigcup_{-\lambda \leq i < 0} ((3ir, \dots, 3ir) + [[r, 3(d+1)r - r - 1]]^d)$$

Da $-3r, -6r, \dots$ und $-3jr$ im Intervall sein müssen, ist das höchste j , für das dies gegeben ist, $\lambda - 1$. Also ist λ das höchste j , für das ϵ_j zwei Zustände besitzt, plus eins. Falls kein solches j existiert ist $\lambda = 0$.

Innerhalb eines Blockes kann ϵ_j zu $(-3r, -6r, \dots, -3jr)$ vereinfacht werden, und ist damit unabhängig von λ . Es reicht, die Positionen der Zellen eines Blockes mit zwei Zuständen zu kennen, um die richtige Permutation e_λ anzuwenden. \square

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>.