

Zellularautomaten mit 1-bit Zustand

(web edition)

Nicola Apicella

na<PLUS>seminar<AT>nicapicella<DOT>com

<http://www.nicapicella.com/>

Institut für Algorithmen und Kognitive Systeme
Universität Karlsruhe (TH)
Sommersemester 2004

Zusammenfassung

In dieser Seminararbeit setze ich mich mit der Frage auseinander, ob die Begrenzung eines Zellularautomaten auf 1-bit Zustand, also einer Zustandsmenge bestehend aus nur zwei Elementen, dessen Universalität beeinflusst.

Als Grundlage zu dieser Ausarbeitung gilt eine Veröffentlichung aus dem Jahre 1971 von A. R. Smith: „*Cellular Automata Complexity Trade-Offs*“ [Smi71]¹.

1 Einführung

Ein Zellularautomat ist eine besondere Art von Automat. Meist meint man mit dem Begriff „Zellularautomat“ ganz viele Einzelautomaten: diese Automaten sind nämlich relativ einfache Gebilde — ein Zellularautomat Z ist formal ein Tupel bestehend aus Gitter R , Nachbarschaft N , Zustandsmenge Q und einer Überföhrungsfunktion — deren Mächtigkeit in der Zusammenarbeit mit Seinesgleichen liegt. Die in einem regelmäßigem Gitter R (man stelle sich zum Beispiel ein zweidimensionales Gitter vor, an welchem an alle Kreuzungen zweier Linien ein Automat sei) angeordneten Automaten(system) Z , welches mit einer festen Anzahl Nachbarn in festem Muster N kommunizieren kann (zum Beispiel mit jeweils den direkten Nachbarn).

Die Regeln eines solches Systems sind für alle Automaten die selben; alle Automaten Z haben die gleiche Nachbarschaft N , Zustandsmenge Q und die gleiche Überföhrungsfunktion. Man kann durch die Zusammenarbeit vieler kleiner, einfacher Komponenten ein sehr komplexes Ganzes bilden, welches die Fähigkeiten des Einzelnen weit übersteigen.

Auch in der Natur sind solche Netzwerke unschwer zu finden: Ameisen, zum Beispiel, sind auch ziemlich hilflos, wenn auf sich alleine gestellt; allerdings kann ein Ameisenstaat, basierend auf der Zusammenarbeit von vielen kleinen Ameisen, solch komplizierte Bauten wie einen Ameisenhügel schaffen.

2 Komplexitätskompromisse bei Zellularautomaten

In der Veröffentlichung von Smith [Smi71] geht es primär um zwei verschiedene Einschränkungen. Die Erste sei nur der Vollständigkeit halber erwähnt, die Zweite ist das Thema dieser Ausarbeitung und wird ausführlicher besprochen werden.

¹Es wird darauf hingewiesen, dass in dieser Veröffentlichung einige Fehler zu vermerken sind. Drei sind sicher — zwei eher harmlose Fehler werden hier erwähnt: im Teil IV, im letzten Absatz, bei der Einführung von M sollte es $0 \leq j \leq m - 1$ (oder auch $0 \leq j < m$) statt $0 \leq j < m - 1$ heißen; weiterhin kommt in der Definition von σ_1 eine weitere Klammer („(“) gleich nach dem zweiten c' , vor $(2nm_0 + 2)$. Der schwerwiegendere Fehler wird erwähnt werden, wenn die dazugehörige Formel besprochen wird.

Um kurz vorweg zusammenzufassen, werden im ersten Teil „kleine“ Zellen (also Zellen eines Automaten mit einer relativ kleinen Zustandsmenge) mit einer großen Nachbarschaft durch „große“ Zellen mit kleiner Nachbarschaft ersetzt, um hierbei die Nachbarschaft zu minimieren. Im zweiten Teil soll das Gegenteil geschehen: hier ersetzen „kleine“ Zellen mit großer Nachbarschaft die „großen“ Zellen mit kleiner (hier: Von-Neumann) Nachbarschaft.

2.1 Reduktion der Nachbarschaft und Beschleunigung

Smith zeigt, dass man einen Zellularautomaten mit beliebiger Nachbarschaft N durch einen Zellularautomaten simulieren kann, welcher die Von-Neumann Nachbarschaft H_1 hat (also wo $H_1 = \{a \mid |a| \leq 1\}$ mit $|a| = \sum_{i=0}^{d-1} |a_i|$; siehe auch Abbildung 1) auf Kosten der Zustandsmenge Q , die bei diesem Prozess vergrößert werden muss. Dieses Ergebnis sei wichtig, weil viele Arbeiten in der Theorie der Zellularautomaten von der Von-Neumann Nachbarschaft ausgehen.

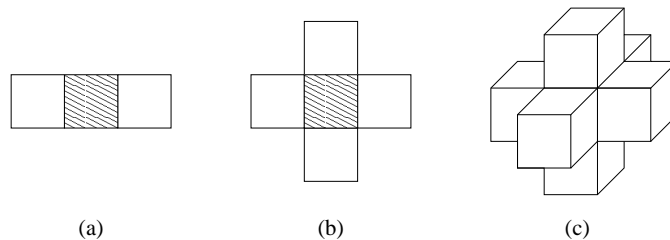


Abbildung 1: Die Von-Neumann Nachbarschaft bei $R = \mathbb{Z}^d$; (a) mit $d = 1$, (b) mit $d = 2$ und (c) mit $d = 3$ (bei letzterem ist die Ausgangszelle nicht sichtbar, da von den anderen Zellen komplett umschlossen).

Es wird außerdem noch bewiesen, dass eine noch kleinere Nachbarschaft als die Von-Neumann Nachbarschaft ausreichend sei, um die Eigenschaft der Universalität in einem Zellularautomaten zu bewahren. Weiterhin zeigt Smith noch, dass die Zeiteinbußen eines simulierenden Zellularautomaten vernachlässigbar seien, da diese Automaten (auf Kosten einer noch größeren Zustandsmenge) in Echtzeit und sogar mit einer Beschleunigung die Ausgangsautomaten simulieren können.

2.2 Reduktion der Zustandsmenge

Im zweiten Teil wird gezeigt, dass eine zweielementige Zustandsmenge Q eines Zellularautomaten genügt, um die Universalität zu bewahren. Der Bequemlichkeit und Anschaulichkeit halber werden im Beweis Zellularautomaten mit Von-Neumann Nachbarschaft betrachtet; sämtliche andere Automaten lassen sich — wie im ersten Teil bewiesen — auf diesen Spezialfall zurückführen.

Der genaue Satz sowie dessen Beweis werden im folgenden genauer analysiert und erklärt. Der Trick bei dem Beweis liegt darin, dass während die Zustandsmenge Q verkleinert wird, die Nachbarschaft N vergrößert wird um somit den „Informationsverlust“ des Zustandes ausgleichen zu können. Während Smith im ersten Teil der Veröffentlichung versucht, eine kleine Nachbarschaft N zu bewahren und dafür eine große Zustandsmenge Q hinnimmt, so muss man hier um eine kleine Zustandsmenge Q zu erhalten die Nachbarschaft N vergrößern.

3 1-bit Zustand als Einschränkung — Satz und Beweis

Es stellt sich die Frage, wie sich eine Einschränkung der Zustandsmenge auf nur zwei Elemente auf einen Zellularautomaten auswirkt. Kann man mit einem so beschrankten Zellularautomaten tatsachlich Zellularautomaten mit beliebigem Gitter $R = \mathbb{Z}^d$, Zustandsmenge Q und Nachbarschaft N simulieren? Die Antwort auf diese Frage ist ja.

Wie weiter oben schon erwahnt, setzen wir als Nachbarschaft des zu simulierenden Automaten die Von-Neumann Nachbarschaft voraus: jede andere kann auf diesen Spezialfall zuruckgefuhrt werden.

Der genaue Satz lautet folgendermaBen²:

Fur einen beliebigen Zellularautomaten Z mit $R = \mathbb{Z}^d$ und Q beliebig, N Von-Neumann Nachbarschaft existiert ein Zellularautomat Z' mit $R' = R, N' = T, |Q'| = 2$ und $|T| = 4nd - 2d + m + 1$, wobei n die kleinste natuerliche Zahl ist, so dass $|Q| \leq 2^n - n$ gilt, und m die kleinste natuerliche Zahl fuer die $n < 2^m$ gilt, welches Z in Echtzeit simuliert.

3.1 Allgemeine Vorgehensweise

Wir werden versuchen, zunachst einmal eine Zelle aus Z komplett in Z' zu simulieren. Mit folgender Vorgehensweise wird Z' aufgebaut: Jede einzelne Zelle von Z wird in Z' durch insgesamt $2n$ Zellen simuliert werden (siehe Abbildung 2 fuer ein Beispiel dieser „Aufspreizung“ im Zweidimensionalen). Hierbei ist n die Zahl, die im Satz der Bedingung genuegen muss, die kleinste Zahl $\in \mathbb{N}$ zu sein, fuer die $n < 2^m$ gilt. Wie und warum es gerade $2n$ Zellen sind, wird in den naechsten beiden Abschnitten erklart.

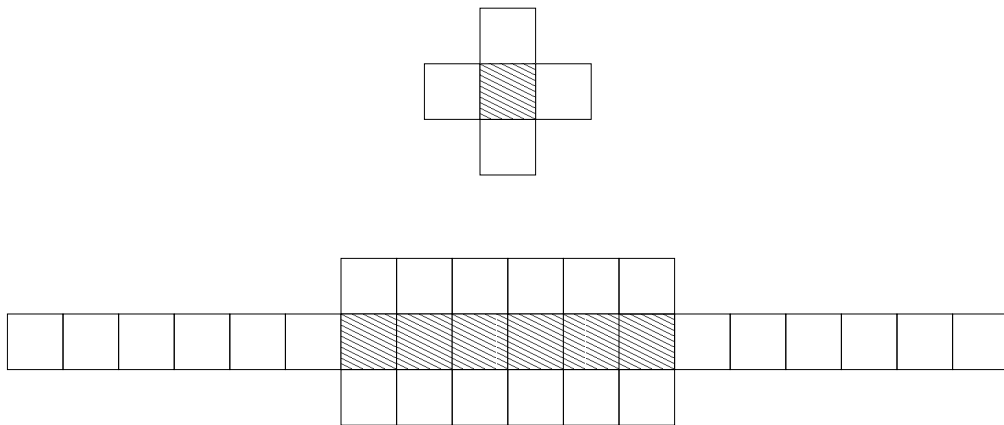


Abbildung 2: Jede Zelle eines (in diesem Beispiel zweidimensionalen) Zellularautomaten wird durch $2n$ Zellen simuliert.

Durch diese Art der Simulation ergibt sich die Notwendigkeit, eine Funktion γ zu definieren, welche jedem Index i einer Zelle c_i aus Z den Wert $2in$ zuweist, da ja die Zellen c_i in Z' wie gesagt durch $2n$ Zellen simuliert werden. Kurz gefasst wird also eine Zelle c_i aus Z durch

² ACHTUNG: In der Veroffentlichung von Smith liegt gerade in diesem Satz ein groeerer Fehler vor: Die Gleichung der Laenge der Nachbarschaft sollte $|T'| = 4nd - 2d + m + 1$ heißen, also am Ende ein „+1“ statt ein „-1“. Dies kann mit einfachen mathematischen Kenntnissen sogar ausgerechnet werden, da der vorangehende Schritt zur Berechnung von $|T'|$ im funften Abschnittes des Beweises zu finden ist. Dort steht, dass $|T'| = n|H_1| + (2d - 1)(n - 1) + m$ ist, was ausmultipliziert **nicht** das dort angegebene, sondern das hier erwahnte Ergebnis (zur Erlauterung: H_1 steht in der Veroffentlichung fuer die Von-Neumann Nachbarschaft, wobei bekanntlich $|H_1| = 2d + 1$) liefert.

die Zellen $\{c'_{\gamma(i)}, c'_{\gamma(i)+1}, \dots, c'_{\gamma(i)+(2n-1)}\}$ in Z' simuliert. Der Übersichtlichkeit wegen wollen wir nun $\gamma(i) = j$ setzen und davon ausgehen, dass eine Zelle c_i durch c'_j und dessen $2n - 1$ Nachfolger³ simuliert wird.

3.2 Voraussetzung: binäre, nullfreie Schieberegister

Laut Golomb, „Shift Register Sequences“ [Gol67] gilt:

$\forall m, n \in \mathbb{N}$ mit $n < 2^m \exists$ ein Schieberegister $s \in \{0, 1\}^n$ mit $s \neq \mathbf{0}$ und Grad m .

Es existiert also ein *binäres, nullfreies* Schieberegister der Länge n und mit Grad m . Die Existenz eines solchen Schieberegister soll hier nur erwähnt werden — dieses wird im weiteren Verlauf wichtig sein. Auf die einzelnen Eigenschaften dieser Folge sowie deren Bedeutungen wird im weiteren Verlauf noch eingegangen. Es soll noch darauf hingewiesen werden, dass die Folge s aus welcher der Schieberegister besteht, seine (lokale) Eindeutigkeit beibehält, wenn man s an sich selbst anhängt; dadurch bilden sich genau n zyklische Permutationen von s , dessen ersten m Stellen eindeutig eine Position codieren.

3.3 Zustandscode

Es werden n der in Abschnitt 3.1 erläuterten $2n$ Zellen dazu genutzt, um die Zustandsmenge Q codiert in Z' zu speichern.

Um die Zustandsmenge Q von Z binär codieren zu können, werden offensichtlich n Bits benötigt, so dass $|Q| \leq 2^n$ gilt. Um den Automaten möglichst minimal zu halten, wählen wir n so, dass es die kleinste Zahl $\in \mathbb{N}$ ist, welche die obige Ungleichung erfüllt.

Damit es unmöglich wird, die Codierung des Zustandes mit der Codierung der Position zu verwechseln (welche auch n Stellen lang ist und im kommenden Abschnitt eingegangen wird), stellen wir außerdem noch die Forderung, dass $n + n \leq 2^n$ gelten soll und wir somit die Ausgangsbedingung $n \leq 2^n - n$ erreichen.

3.4 Positionscodierung

Gerade haben wir den Positionscodierung schon erwähnt, welcher auch aus n Bits besteht. Der Code soll dazu dienen, die Position der jeweiligen Zelle aus Z' zu identifizieren, um Verwechslungen zu vermeiden.

Dieser soll also irgendwie für jede Stelle eine Eindeutigkeit darstellen; er muss auch noch binär (wegen $|Q'| = 2$) und keine Nullfolge sein, damit man ihn nicht mit der Anfangskonfiguration verwechselt.

Der aufmerksame Leser wird nun die Ähnlichkeit zwischen dem Positionscodierung sowie dessen benötigten Eigenschaften mit dem in Abschnitt 3.2 angesprochenen *binären, nullfreien* Schieberegister der Länge n festgestellt haben. Genau hierfür benötigen wir einen solchen „Schieberegister“, oder besser gesagt dessen Bitfolge s . Die Existenz einer solchen Folge haben wir übrigens dadurch garantiert, als wir uns ein m mit $n < 2^m$ definiert haben. Auch hier soll m als kleinste Zahl $\in \mathbb{N}$ gewählt werden, um den Automaten möglichst minimal zu halten.

3.5 Zusammenfassung und weiteres Vorgehen

Bis jetzt haben wir zwei verschiedene Bitfolgen der Länge n definiert. Offenbar bilden diese beide zusammen die schon erwähnten $2n$ Zellen die in Z' benötigt werden, um eine Zelle c_i aus Z zu simulieren. Aber wie genau?

Wir bilden nun unsere $2n$ -lange Bitfolge für Z' indem wir jeweils ein Bit des Zustandscodes neben einem des Positionscodes setzen. Es entsteht somit eine Folge, deren jedes zweite Bit

³Die Richtung, in welche wir die Nachfolger (und später die Nachbarn) aufsuchen, ist beliebig; allerdings sollte man bei allen Betrachtungen der Nachbarzellen immer in die selbe Richtung schauen.

betrachtend den Positionscode bzw. den Zustandscode bilden. Eine veranschaulichung dieser Anordnung ist in Abbildung 3 zu sehen.

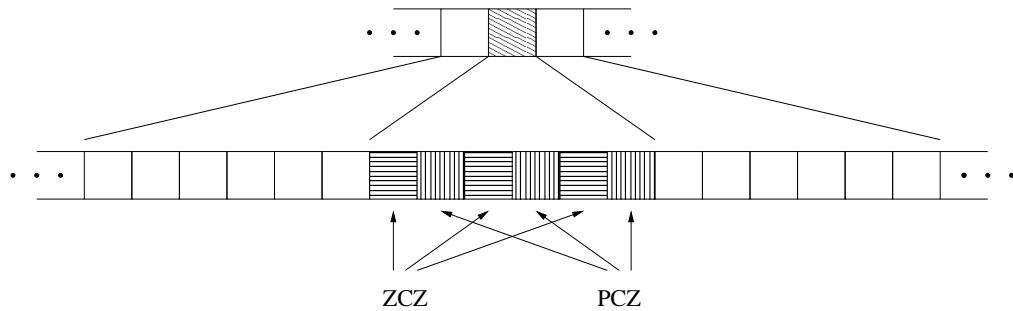


Abbildung 3: Veranschaulichung der Aufteilung in ZCZ und PCZ einer in Z' simulierten Zelle.

Jede Zelle c'_j kann nun, durch Betrachten von sich selbst und $n-1$ zweite Nachbarn in beiden Richtungen (also $\{c'_j, c'_{j+2}, \dots, c'_{j+2(n-1)}\}$ und auch $\{c'_j, c'_{j-2}, \dots, c'_{j-2(n-1)}\}$, wie in Abbildung 4 veranschaulicht) feststellen, ob sie eine Zelle des Zustandscodes ist oder eine des Positionscode. Sind die Zelle mit ihren zweiten Nachbarn beider Richtungen eine zyklische Permutation des Positionscode, so ist die Zelle (sowie auch ihre zweite Nachbarn) eine Positionscodezelle (PCZ). Sind die Zellen beider Richtungen keine solche Permutation, aber auch nicht der Ruhezustand, so ist die Zelle mit ihren Nachbarn eine Zustandscodezelle (ZCZ).

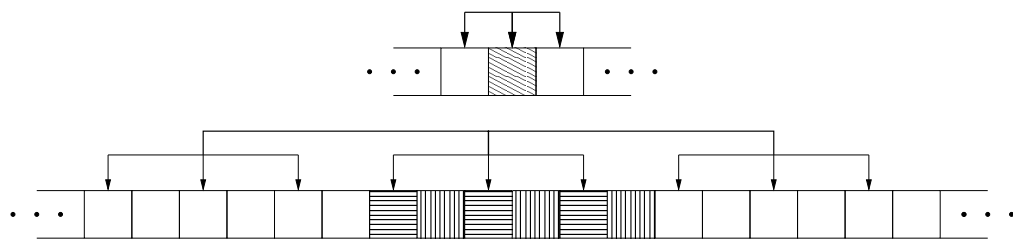


Abbildung 4: Auszulesende Zustände beim betrachten einer Zelle in Z und auszulesende ZCZ bei Betrachtung der selben in Z' simulierten Zelle.

Es ist vielleicht nicht ohne weiteres ersichtlich, warum man in beide Richtungen zur Zellenbestimmung schauen muss. Dies ist so, denn trotz der Mühe bei der Wahl des Positionscode und des Zustandscodes kann es passieren, dass die Endbits einer Zustandscodierung mit den Anfangsbit der darauf folgenden Codierung zufälligerweise eine Permutation des Positionscode darstellen. Wäre es allerdings eine Positionscodierung, so müssen es auch die Zellen des Nachbarn in der anderen Richtung sein.

Bevor wir weitermachen noch ein Hinweis: wie weiter oben schon gezeigt, ist die Position einer Zelle von nur den ersten m Positionscodezellen eindeutig festgelegt. Es ist also nicht nötig, um die Position zu bestimmen, alle n Positionscodezellen auszulesen.

3.6 Überföhrungsfunktion

Eine PCZ ändert ihren Zustand nicht. Sobald die Überföhrungsfunktion eine PCZ als solche erkannt hat, so wird diese unverändert in die nächste Konfiguration übernommen.

Eine ZCZ muss allerdings mehrere Schritte vornehmen: Erst einmal muss sie überprüfen, an welcher Stelle sie sich befindet. Durch Auslesen ihres Nachbarn c'_{i+1} und dessen m zweite Nachbarn (wir haben oben gezeigt, dass m Zellen genügen; wir lesen also $\{c'_{j+1}, c'_{j+3}, \dots, c'_{j+(2m-1)}\}$ aus), kann die Überföhrungsfunktion feststellen, welche Position des Zustandscode die zu analysierende Zelle ausmacht. Je nach Ergebnis werden dann auch alle anderen Stellen des Zustandscodes ausgelesen, um den binär codierten Zustand der Zelle c_i aus Z festzustellen. Nun können, immer noch dank der ermittelten Position von c'_j , auch die Zustände der Nachbarzellen von c_i , also c_{i-1} (welche von $c'_{\gamma(i-1)}$ und $n - 1$ Nachbarzellen simuliert wird) und c_{i+1} ($c'_{\gamma(i+1)}$ und Nachbarn), festgestellt werden — es handelt sich hierbei um jeweils n Zellen aus Z' , welche ausgelesen werden müssen. Man hat nun alle nötigen Informationen, um den Folgezustand der Zelle in Z zu berechnen, um dann die vorher ermittelte Stelle der Codierung des Folgezustandes in Z' zu nehmen und diesen Bit als Folgezustand für die aktuell betrachtete Zelle in die nächste Konfiguration von Z' zu übernehmen.

3.7 Nachbarschaft

Bei der Nachbarschaft T' von Z' ist wohl das interessanteste die Aussage die über die Anzahl der Zellen, die betrachtet werden. Die sogenannte Länge der Nachbarschaft kann nämlich durch die Formel

$$|T'| = 4nd - 2d + m + 1$$

berechnet werden⁴.

Da diese Formel nicht ohne weiteres verständlich ist, werde ich noch einige Worte über deren Herleitung verlieren. Die Nachbarschaft stellt eine Art Muster dar, welches bestimmt, auf welche Nachbarzellen eine Zelle bei der Berechnung ihres eigenen Folgezustand mit einbeziehen soll. T' muss zur korrekten Berechnung folgendes leisten:

- Als erstes muss sie, um feststellen zu können, ob es sich bei der aktuellen Zelle um eine PCZ oder eine ZCZ handelt (siehe hierzu Abschnitt 3.5), sich selbst und n zweite Nachbarn ($\{c'_j, c'_{j+2}, \dots, c'_{j+2(n-1)}\}$) auslesen. Ist die Zelle eine PCZ, so ist diese Information für die Überföhrungsfunktion schon ausreichend. **Es müssen n Zellen berücksichtigt werden.**
- Handelt es sich bei der Zelle um eine ZCZ, so muss man noch Informationen über die Aktuelle Position über $\{c'_{j+1}, c'_{j+3}, \dots, c'_{j+(2m-1)}\}$ in Erfahrung bringen. **Weitere m Zellen.**
- Natürlich brauchen wir auch Informationen über die Zustände der Nachbarzellen⁵: sie sind in jeweils n Zellen für jede Nachbarzelle der Von-Neumann Nachbarschaft H_1 in Z' gespeichert. Bei der Von-Neumann Nachbarschaft werden also $n|H_1|$ Zellen benötigt — da $|H_1| = 2d + 1$ ist, muss die Nachbarschaft weitere $2nd + n$ Zellen mit einbeziehen. Da wir allerdings den Zustand der Ausgangszelle für die Nachbarschaft vorhin schon dazugezählt haben, bleiben also noch $2nd$ Zellen. **Noch $2nd$ Zellen.**
- Im Gegensatz zum Positionscode, bei welchem die m Zellen $\{c'_{j+1}, c'_{j+3}, \dots, c'_{j+(2m-1)}\}$ **immer** die richtige Position angeben da der Code zyklisch immer wieder den gleichen (immer wieder korrekten) Wert liefert, gibt es beim Zustandscode noch eine kleine Tücke. Wenn wir nämlich das Nachbarschaftsmuster so wählen, um garantieren zu können, dass wir sämtliche benötigte Zustandscodierungen von den Nachbarzellen auslesen können, dann reicht es nicht aus, für jede Dimension nur n Zellen des vorherigen Nachbarn und n Zellen des Nachfolgenden Nachbarn zu erfassen.

Würden wir nämlich solch ein Muster auswählen, so könnten wir dadurch nur garantieren, dass es für eine bestimmte Position der Codierung gilt. Das Nachbarschaftsmuster jedoch ist im voraus festgelegt und muss für jede Zelle identisch sein. Um zu gewährleisten,

⁴Siehe zu dieser Formel auch den Hinweis in Fußnote 2.

⁵Zur Erinnerung: wir haben uns der Anschaulichkeit- und Bequemlichkeit halber auf den Spezialfall einer Von-Neumann Nachbarschaft beschränkt.

dass auch jede Zelle mit der genau gleichen Nachbarschaft (bzw. Nachbarschaftsmuster) auskommt, müssen wir für alle Richtungen aller Dimensionen mit Ausnahme von der eigenen Zelle und der in der Richtung liegenden, in welche wir unsere simulierende Zellen ausgebreitet haben (also eine Richtung einer Dimension), noch weitere $n - 1$ Zellen anschauen. Daraus ergeben sich noch $(n - 1)(2d - 1) = 2nd - n - 2d + 1$ Zellen, die berücksichtigt werden müssen. **Weitere $2nd - n - 2d + 1$ Zellen zu betrachten.**

Wie wir an dieser Auflistung gut sehen können, brauchen wir die oben genannte Anzahl von $n + m + 2nd + 2nd - n + 1 = 4nd - 2d + m + 1$ Zellen.

4 Zusammenfassung

Wir haben gesehen, dass es durchaus möglich ist, beliebige Zellularautomaten durch „äquivalente“ Zellularautomaten zu ersetzen, deren Zustandsmengen auf zwei Elemente beschränkt sind. Natürlich haben wir etwas in Kauf nehmen müssen, nämlich eine wesentlich größere Nachbarschaft, um unser Ziel zu erreichen.

Wir haben einerseits die Zustände binär kodiert und andererseits eine Möglichkeit erläutert, diese Codes wieder korrekt auszulesen. Insgesamt mussten wir jede Zelle c_i in Z durch eine eindimensionale, zusammenhängende Gruppe an Zellen aus Z' simulieren, denn nur somit konnten wir die zwei im vorherigen Paragraph angegebenen Merkmale erfolgreich in Z' unterbringen. Zuletzt haben wir Aussagen über die Nachbarschaft gemacht und einen Ansatz für die Überföhrungsfunktion gegeben, um eine korrekte Simulation gewährleisten zu können.

Literatur

- [Smi71] Alvy Ray Smith III, „Cellular Automata Complexity Trade-Offs“, Information and Control, Vol. 18, No. 5, pp. 466–482, Jun 1971.
<ftp://ftp.alvyray.com/Acrobat/TradeOff.pdf> (letzter Zugriff: 07/Dez/04)
- [Gol67] Solomon W. Golomb, „Shift Register Sequences“, Holden-Day Inc., San Francisco, 1967; 2nd, Revised Edition, Aegan Park Press, May, 1982.