

# Zelluläre Automaten

Daniela Kern

22. April 2003

Ausgewählte Kapitel aus dem Bereich Softcomputing  
Sommersemester 2003

Experimental Computation  
(Entdeckung der Komplexität von simplen Programmen)

Erster Vortrag dieser Reihe: Grundlegendes zu zellulären Automaten  
(2. Kapitel und Teile des 3. Kapitels aus dem Buch *A New Kind of Science*  
von STEPHEN WOLFRAM)

## Zusammenfassung

STEPHEN WOLFRAM präsentiert mit seinem Buch *A New Kind of Science* das Resultat von 20 Jahren Forschungsarbeit. Die darin getroffenen Aussagen beruhen auf Serien von experimentellen Berechnungen, die unerwartete Ergebnisse hervorgebracht haben. WOLFRAM demonstriert mit seinem Buch, dass simple Programme ein erstaunlich komplexes Verhalten in ihrer Ausführung entwickeln können.

# Inhaltsverzeichnis

<b>1</b>	<b>Einführung</b>	<b>3</b>
<b>2</b>	<b>Komplexität und ihre Entstehung</b>	<b>3</b>
<b>3</b>	<b>The Principle of Computational Equivalence</b>	<b>4</b>
<b>4</b>	<b>Was ist ein zellulärer Automat?</b>	<b>6</b>
<b>5</b>	<b>Mathematica-Implementierung</b>	<b>7</b>
<b>6</b>	<b>Wesentliche Beispiele</b>	<b>9</b>
6.1	Regel 254 . . . . .	9
6.2	Regel 250 . . . . .	10
6.3	Regel 90 . . . . .	11
6.4	Regel 30 . . . . .	12
6.5	Regel 110 . . . . .	15
<b>7</b>	<b>Erkenntnisse</b>	<b>16</b>
<b>8</b>	<b>Weitere Automaten</b>	<b>17</b>
8.1	Weitere zelluläre Automaten . . . . .	17
8.2	Modifizierter Mathematica-Code . . . . .	18
8.3	Beispiele . . . . .	19
8.3.1	Regel 777 . . . . .	19
8.3.2	Regel 1110 . . . . .	20
8.3.3	Regel 600 . . . . .	20
8.3.4	Simple Muster . . . . .	21
8.3.5	Nested Patterns . . . . .	23
8.3.6	Komplexe Strukturen . . . . .	24
8.3.7	Hochkomplexe Strukturen . . . . .	26
8.3.8	Auslöschung . . . . .	27
8.4	Schlussfolgerungen . . . . .	28
<b>9</b>	<b>Anhang</b>	<b>29</b>
9.1	Die 256 elementaren zellulären Automaten . . . . .	29

# 1 Einführung

In dieser Ausarbeitung wird einführend das 2002 erschienene Buch *A New Kind of Science* von STEPHEN WOLFRAM behandelt. STEPHEN WOLFRAM ist der Entwickler des bekannten Programms *Mathematica*. Die Arbeit an *A New Kind of Science* hat grob 20 Jahre in Anspruch genommen und basiert in Teilen auf Erkenntnissen, die WOLFRAM durch die Entwicklung und das Arbeiten mit Mathematica gewonnen hat. WOLFRAM sieht sich mit diesem Buch als Gründer einer völlig neuen und revolutionären Wissenschaft. Im Gegensatz zu den klassischen Wissenschaften basiert sie nicht auf deduktivem und zweckorientiertem Vorgehen, sondern auf experimentellen Berechnungen. Dadurch seien plötzlich Ergebnisse erzielbar, die sich mit den klassischen Methoden nicht erreichen ließen, da niemand sie erreichen *wollte*. Der Fokus in den traditionellen Wissenschaften liegt auf Dingen, die intuitiv zumindest erahnbar sind.

Das übliche Vorgehen in den abstrakten Wissenschaften ist, reale Vorgänge mit mathematischen Begriffen zu modellieren. Dabei ist aber das durch die Mathematik gelieferte Handwerkszeug begrenzt, ebenso sind dann die Modellbildungen eher einseitig. Die gebräuchlichen Modelle kommen zum größten Teil aus der Algebra, der Analysis oder der Wahrscheinlichkeitslehre; alle diese Disziplinen haben nur einen beschränkten Fokus. Die mathematische Modellierung eignet sich gut für simple Systeme wie z.B. Atome, Planetensysteme etc., bei komplexeren greift sie in einem nicht ausreichenden Maße.

STEPHEN WOLFRAM wählt einen universelleren Ansatz, der mit einem minimalen Satz an Begriffen auskommt. Er beschränkt sich auf einfache Regeln und zeigt, dass sich auch damit komplexe Ergebnisse erzielen lassen. Wichtiges Beispiel für dieses sind die zellulären Automaten, die Thema dieser Ausarbeitung sind.

Mittels dieses universellen Ansatzes will WOLFRAM auch universelle Systeme modellieren können. Z.B. ist er sich gewiß, auf dem Weg zu sein, eine mächtige Theorie für die Physik zu entwickeln.

## 2 Komplexität und ihre Entstehung

Was WOLFRAM reizte, die Arbeit an *A New Kind of Science* aufzunehmen, war die Frage, wie *Komplexität* entsteht. Als Schuljunge hatte WOLFRAM auf einem Statistik-Buch die bildliche Darstellung von Teilchen-Diffusion gesehen. Die Anordnung der Teilchen auf den hinteren Bildern schien rein zufällig und ohne Ordnung. Hier lag ein *komplexes* System vor. Systeme ohne Ordnung oder solche Systeme, die mit extrem vielen Worten nur charakteri-

siert werden können, nennt Wolfram *komplex*. Eine korrekte, definierende Beschreibung eines komplexen Systems ist aufgrund der Ordnungslosigkeit des Systems in etwa mindestens genauso "groß" wie das System selber.

Die Teilchendiffusion auf WOLFRAMs Statistik-Buch ist ein Prozess, von dem einzelne Momentaufnahmen abgebildet wurden. Ausgehend von einem Startzustand entwickeln sich gemäß (teils unbekanntem) Gesetzen der Natur neue Zustände. WOLFRAM stellte sich die Frage, von welcher Beschaffenheit diese Regeln sein müssen. Der Gedanke liegt nahe, dass die Regeln aufgrund der Komplexität des Ergebnisses selbst eine hohe Komplexität aufweisen müssen. WOLFRAM hat diese Fragestellung auf Programme (meist in Mathematica) und deren Ausführungsverhalten übertragen. Er hat sich willkürlich einfache Berechnungsregeln gewählt und beobachtet, was die darauf basierenden Rechnungen ergaben. Dabei stellte er sich die Frage, wie sich die Komplexität eines Programms und die Komplexität der entsprechenden Berechnung verhalten. Die entscheidende Konsequenz, die er aus seinen Experimenten ziehen konnte, ist, dass auch simple Programme komplexes Verhalten erzeugen können. Programme und Berechnung weisen im allgemeinen also keine gleichwertige Komplexität auf. WOLFRAM zeigt dies an einigen einfachen, leicht selbst nachprüfbareren Beispielen. Die Beispiele werden im Folgenden noch eingehend thematisiert.

Durch die Aussage *Aus einfachen Regeln folgt komplexes Verhalten* kann man sich an die bereits ältere Erkenntnis aus der Informatik bzw. Berechenbarkeitstheorie erinnern, dass man mit einem festen, kleinen Satz von Rechenoperationen jegliche mögliche Berechnung realisieren kann. Anschaulich wird das an Computern, deren fester Satz an direkten Rechenoperationen recht klein und vor allem primitiv ist. Was WOLFRAM aber noch zusätzlich herausarbeitet, dass auch eine *schlichte Kombination* dieser Operationen schon zu komplexen Resultaten führen kann.

### 3 The Principle of Computational Equivalence

Wenn man davon ausgeht, dass man für (fast) beliebige Systeme (z.B. der Natur) dessen Regeln als korrespondierend zu einem Programm betrachten kann, dann kann man folglich auch das Verhalten des Systems als korrespondierend zu einer Berechnung (Computation) betrachten.

Durch diese Korrespondenz überträgt WOLFRAM seine durch Mathematica-Programmen gewonnenen Erkenntnisse auf die Phänomene der realen Welt.

WOLFRAM führt angeregt durch seine Entdeckungen einen neuen Begriff ein – *The Principle of Computational Equivalence*. Dieses Prinzip besagt gerade, dass man zu fast jedem beliebigen System (der realen Welt) eine Be-

rechnung (auf dem Computer) mit gleichwertiger Komplexität finden kann.

Wenn sich die verschiedenen Systeme der realen Welt auf Programme und deren Berechnungen übertragen lassen, wird die Vielzahl dieser Systeme plötzlich überschaubar und es lassen sich besser allgemeine Aussagen treffen als das mit den Mitteln der Mathematik bisher möglich gewesen ist. Das Entwickeln und Untersuchen von Programmen bedeutet dann gleichzeitig das Behandeln von ganzen Gruppen von Phänomenen der realen Welt.

In der Praxis hat das *Principle of Computational Equivalence* folgendes zur Konsequenz: selbst zu komplexen Phänomenen der realen Welt (z.B. menschliche Vorgänge wie Wahrnehmung und Analyse) existiert in den meisten Fällen ein einfaches Programm, das eine dem Phänomen äquivalente Berechnung liefert. Diese Aussage (so wahr) ist recht gravierend – z.B. könnte man in der Gen-Forschung eventuell auf einfachen Voraussetzungen basierende Verfahren entwickeln, mit denen sehr gute Produkte für medizinische Zwecke hergestellt werden können.

Es ergibt sich aber auch eine andere Betrachtungsweise von Prozessen in der Natur – wahrscheinlich sind die Regelwerke dort auch eher simpel.

Die Herangehensweise der klassischen Wissenschaften bewirkt, dass nur solche Probleme betrachtet werden, deren Verhalten in weiten Teilen vorhersagbar ist. Das Entwickeln einer Problemstellung in den klassischen Wissenschaften beginnt für gewöhnlich damit, dass ein Phänomen herangezogen wird und dieses soll dann deduktiv auf ein Regelwerk zurückgeführt werden. Und nur solche Probleme, für die dieses deduktive Vorgehen auch tatsächlich möglich ist, werden überhaupt als wissenschaftlich relevant weiter behandelt. Wie oben schon erwähnt, begrenzt dieses Vorgehen den Fokus der klassischen Wissenschaften beachtlich. Es werden hauptsächlich Vorgänge betrachtet, deren Entwicklungsschritte nicht im einzelnen nachvollzogen werden müssen, man kann wegen der Vorhersagbarkeit des Verhaltens einzelne Schritte auslassen. Systeme, bei denen keine Vorhersagen bezüglich ihrer Entwicklung getroffen werden können, bei denen also auch der Wissenschaftler jeden Schritt im einzelnen nachvollziehen muss, nennt WOLFRAM *computationally irreducible*. Wenn diese Irreduzibilität in der Berechnung vorliegt, ist das Ergebnis eines Programms bzw. Regelwerks also unvorhersehbar, menschliche Intuition hilft dann nicht weiter. Auch ein Mensch muss dann die gleichen stumpfen Rechenschritte wie eine Maschine machen, um an das Ergebnis der Berechnung zu gelangen. Somit sind die Begriffe *komplex* und *computationally irreducible* eng verknüpft: sie besagen in der Sache ungefähr das selbe, der Begriff *komplex* legt den Schwerpunkt aber auf die Betrachtung des gesamten Systems, z.B. auf den grafischen Output, den ein Programm liefert; *textitcomputationally irreducible* bezieht sich auf die Abfolge der Entwick-

lungsschritte, z.B. also auf die einzelnen Schritte bei der Ausführung eines Programms.

## 4 Was ist ein zellulärer Automat?

Anhand der zellulären Automaten werden die vorhergehenden Aussagen nun konkretisiert. Es wird ein Automatentyp eingeführt, der besonders gut veranschaulicht, dass es möglich ist, auch mit simplen Programmen komplexes Verhalten zu erhalten.

Ein (*elementarer*) *zellulärer Automat* besteht aus einem Feld von Zellen mit einem Zustand (im binären Fall 1 oder 0) und einer (elementaren) *Regel-Funktion* (vergleichbar mit der Steuerfunktion bei allgemeinen abstrakten Maschinen), die für die Berechnung des Zustands einer Zelle nur deren aktuellen Zustand und die aktuellen Zustände der direkten Nachbar-Zellen braucht.

Im folgenden werden nur die eindimensionalen binären zellulären Automaten betrachtet. Im eindimensionalen Fall berechnet sich der  $(n + 1)$ -te Zustand der  $i$ -ten Zelle aus dem  $n$ -ten Zustand der  $(i - 1)$ -ten,  $i$ -ten und  $(i + 1)$ -ten Zelle.

Die Regel ist für alle Zellen gleich, der  $(n + 1)$ -te Zustand der  $i$ -ten Zelle wird also nach dem gleichen Schema ermittelt wie der  $(n + 1)$ -te Zustand der  $j$ -ten Zelle für  $i \neq j$ .



Zellulärer Automat der Länge 11

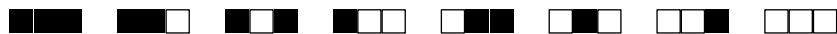
Die Zahl der möglichen Regeln ist wegen der Endlichkeit der möglichen Ein- und Ausgabewerte begrenzt. Im hier interessierenden eindimensionalen binären Fall berechnet sich die Anzahl der möglichen Regeln wie folgt:

Die Regel hat 3 Parameter (Zustand des linken Nachbarn der Zelle, Zustand der Zelle, Zustand des rechten Nachbarn der Zelle), die jeweils 0 oder 1 als mögliche Werte haben. Also gibt es  $2^3 = 8$  mögliche Eingabewerte. Des weiteren gibt es 2 mögliche Ausgabewerte (0 oder 1), daraus ergeben sich  $2^8 = 256$  mögliche Regeln.

**Bemerkung:** Die Regel-Funktionen bei eindimensionalen binären zellulären Automaten sind äquivalent zu einer Untergruppe der aus der Informatik bekannten *Booleschen Funktionen*  $f : \mathbb{B}^n \rightarrow \mathbb{B}$ , nämlich zu den Booleschen Funktionen mit  $n=3$ . Entsprechend gelten hier die bereits bekannten Eigenschaften der Booleschen Funktionen, insbesondere dass sich jede elementare

Regel durch AND und OR oder ein anderes komplettes System von elementaren binären Funktionen vollständig ausdrücken läßt. Des weiteren lassen sich die elementaren Regeln auch algebraisch schreiben, wenn man sie als Funktionen  $f : \mathbb{F}^3 \rightarrow \mathbb{F}$  auffasst;  $\mathbb{F}$  sei der Zahlenkörper, der nur die Elemente 0 und 1 enthält.

WOLFRAM hat eine recht einfache Darstellungsform für die Regelfunktionen gewählt. Er fixiert ein Reihenfolge für die 8 möglichen Eingabewerte und kann somit die Regeln als 8-stellige Listen schreiben, bei denen an der  $k$ -ten Stelle der Funktionswert zum  $k$ -ten möglichen (fixierten) Eingabewert steht. Die Fixierung der Eingabewerte lautet:



$\{1, 1, 1\}, \{1, 1, 0\}, \{1, 0, 1\}, \{1, 0, 0\}, \{0, 1, 1\}, \{0, 1, 0\}, \{0, 0, 1\}, \{0, 0, 0\}$

*Beispiel:*  $\{0, 0, 0, 1, 1, 1, 1, 0\}$  repräsentiert die Regel

$\{1, 1, 1\} \mapsto 0$	$\{1, 1, 0\} \mapsto 0$	$\{1, 0, 1\} \mapsto 0$	$\{1, 0, 0\} \mapsto 1$
$\{0, 1, 1\} \mapsto 1$	$\{0, 1, 0\} \mapsto 1$	$\{0, 0, 1\} \mapsto 1$	$\{0, 0, 0\} \mapsto 0$

Fasst man die zu den Regeln korrespondierenden Listen als Bitstrings auf und diese dann als binäre Zahlen, so erhält man eine Nummerierung der Regeln. Die Regel  $\{0, 0, 0, 0, 0, 0, 0, 0\}$  ist dann die *Elementary Rule 0*, die erste Regel;  $\{1, 1, 1, 1, 1, 1, 1, 1\}$  entsprechend die *Elementary Rule 255*, die 256. Regel. Allgemein:  $\{a, b, c, d, e, f, g, h\}$  ist Regel Nummer

$$2^7 \cdot a + 2^6 \cdot b + 2^5 \cdot c + 2^4 \cdot d + 2^3 \cdot e + 2^2 \cdot f + 2^1 \cdot g + 2^0 \cdot h.$$

## 5 Mathematica-Implementierung

WOLFRAM hat in den Anmerkungen zu seinem Buch *A New Kind of Science* den allgemeinen Quellcode zu den zellulären Automaten mitgeliefert. Das Programm ist bis auf eine kleine Abwandlung der Ausgabefunktion aus den Anmerkungen übernommen.

```

CenterList[n_Integer] :=
  ReplacePart[Table[0, {n}], 1, Ceiling[n/2]]
ElementaryRule[num_Integer] := IntegerDigits[num, 2, 8]
CAStep[rule_List, a_List] :=
  rule[[8 - (RotateLeft[a] + 2 (a + 2 RotateRight[a]))]]
CAEvolveList[rule_, init_List, t_Integer] :=
  NestList[CAStep[rule, #] &, init, t]
CAGraphics[history_List, size_] :=
  Graphics[Raster[1 - Reverse[history]],
    AspectRatio -> Automatic, ImageSize -> size]

```

*Erläuterung:* `CenterList` ist definiert als Funktion, die einen Integer-Wert  $n$  als Eingabe bekommt und eine Liste der Länge  $n$  erzeugt, bei der an Stelle  $\lceil n \rceil$  eine 1 steht. An allen anderen Positionen sind Nullen zu finden. Diese Liste ist die Repräsentation eines eindimensionalen (binären) zellulären Automaten der aus  $n$  Zellen besteht und sich im üblichen Startzustand befindet. Die mittlere Zelle ist schwarz gefärbt (ihr Zustand ist 1), die restlichen Zellen weiß (ihr Zustand ist 0).

`ElementaryRule` operiert auf einem Integer-Wert  $num$  und gibt die  $num$ -te Regel gemäß der Berechnungsregel im vorherigen Abschnitt zurück, also eine Liste, die  $num$  binär darstellt.

`CAStep` erhält zwei Listen  $rule$  und  $a$  als Eingabe. Vorweg sollte gesagt werden, dass Mathematica die Regel-Ausführung für alle Zellen gleichzeitig vornimmt und nicht die Liste sequentiell durchgeht. Dadurch müssen keine temporären Variablen benutzt werden, die die Liste des vorherigen Schritts speichern.  $rule$  wird im folgenden mit einer 8-stelligen Liste belegt sein und eine elementare Regel repräsentieren.  $a$  wird eine  $n$ -stellige Liste sein, die den Zustand eines zellulären Automaten der Größe  $n$  darstellt. `CAStep` berechnet aus diesen beiden Eingaben den Zustand des zellulären Automaten  $a$  nach einmaliger Anwendung der Regel  $rule$ .  $(\text{RotateLeft}[a] + 2 (a + \text{RotateRight}[a]))$  erzeugt eine 8-stellige Liste, an deren  $i$ -ter Position die Dezimaldarstellung der Binärzahl  $\{a(i-1), a(i), a(i+1)\}$  der originalen Liste  $a$  steht. (*Beachte:* Nummerierungen von Listen in Mathematica beginnen von links mit 1.) Da die Fixierung der Eingabewerte aber mit der größten Zahl  $\{1, 1, 1\}$  beginnt, muß jede Stelle von  $(\text{RotateLeft}[a] + 2 (a + \text{RotateRight}[a]))$  noch von 8 abgezogen werden. Diese  $n$ -stellige Liste dient nun als Index-Liste für  $rule$  – heraus kommt eine  $n$ -stellige Liste, an deren  $i$ -ter Position  $rule(8 - \text{Dezimal}(\{a(i-1), a(i), a(i+1)\}))$  steht, also insgesamt die Liste des neuen Zustands von  $a$ .

`CAEvolveList` mit den Parametern  $rule$ ,  $init$  und  $t$  führt die Funktion `CAStep`

nun iterativ  $t$ -mal mit der Ausgangsliste *init* durch und erzeugt eine Liste, die beginnend mit *init* alle Zwischenergebnisse und das Endergebnis liefert. Das Resultat der Ausführung von `CAEvolveList` ist also eine verschachtelte Liste, die als Matrix aufgefasst werden kann. Die  $k$ -te Zeile der Matrix besteht aus der Liste, die den  $k$ -ten Zustand des zellulären Automaten repräsentiert.

`CAGraphics` dient zur Visualisierung des zellulären Automaten. Für *history* muß eine 2-dimensionale Liste, also eine Matrix, eingesetzt werden. Die Matrix wird als Raster abgebildet, Nullen weiß und Einsen schwarz. Um das zu erreichen, wird jedes Element von *history* invertiert, indem es von 1 abgezogen wird. Denn Mathematica bildet Nullen schwarz und Einsen weiß (entsprechend den Lichtwerten) ab. Die `Reverse`-Operation wird durchgeführt, damit der Startzustand oben und der Endzustand unten zu finden ist. `Graphics` packt das Raster in ein Grafik-Objekt, zusätzlich können Parameter wie `AspectRatio` oder `ImageSize` abweichend von den Default-Einstellungen gesetzt werden. Das Grafik-Objekt kann dann durch `Show` auf dem Bildschirm ausgegeben werden.

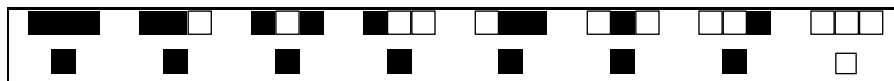
```
Show[CAGraphics[CAEvolveList[ElementaryRule[num],
                           CenterList[n], t], size]]
```

## 6 Wesentliche Beispiele

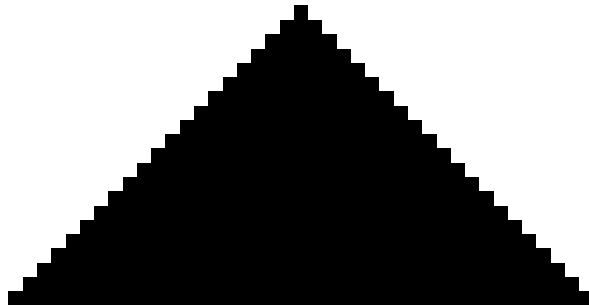
Bei den Beispielen sind beim Ausgangszustand des zellulären Automaten immer alle Zellen bis auf die mittlere weiß. Das ist ein ziemlich simpler Ausgangszustand und wirkt sich somit nur schwach als Einflussfaktor für die zu beobachtende Komplexität aus. Natürlich sind aber auch andere Ausgangskonfigurationen denkbar.

### 6.1 Regel 254

Als erstes einführendes Beispiel bietet es sich an, den zellulären Automaten mit der Regel-Funktion  $\{1, 1, 1, 1, 1, 1, 1, 0\}$  zu betrachten. Diese Regel besagt, dass eine Zelle dann mit 1 belegt werden soll, wenn die Zelle bereits selber oder einer der Nachbarzellen mit 1 belegt war – sozusagen eine dreistellige OR-Funktion.



Als Resultat ergibt sich eine simple symmetrische Struktur – ein Dreieck.



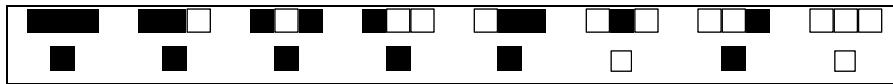
```
Show[CAGraphics[CAEvolveList[ElementaryRule[254],
CenterList[41], 20], Automatic]]
```

Das Verhalten dieses zellulären Automaten ließ sich auch ohne Implementierung auf dem Computer intuitiv vorhersagen.

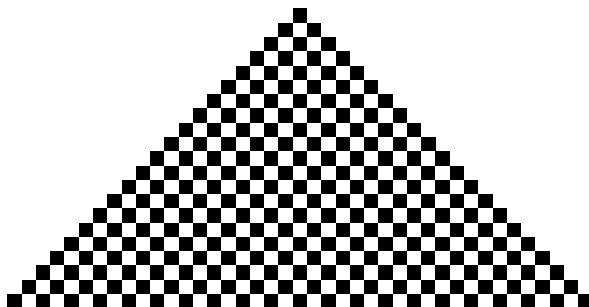
## 6.2 Regel 250

Ein weiteres, kaum komplizierteres Beispiel ist das durch die Regel  $\{1, 1, 1, 1, 1, 0, 1, 0\}$  gegebene.

Hier soll eine Zelle mit 1 belegt werden, genau dann wenn mindestens eine der beiden Nachbarzellen mit 1 belegt ist – das entspricht der Funktion *Zustand des linken Nachbarn OR Zustand des rechten Nachbarn*.



Resultat ist ein schachbrettartiges, spitzzulaufendes Dreieck.

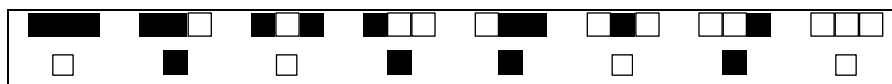


```
Show[CAGraphics[CAEvolveList[ElementaryRule[250],
CenterList[41], 20], Automatic]]
```

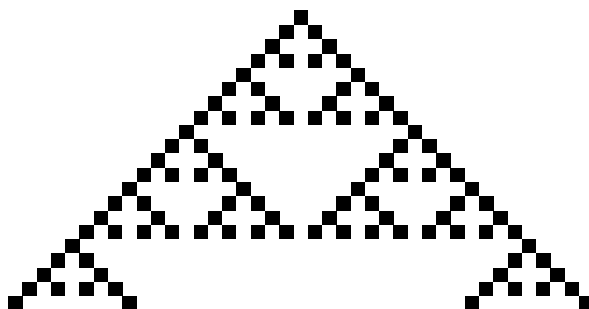
Auch hier ist das Ergebnis mit Denkarbeit und Intuition vorhersagbar und das Ergebnis sehr symmetrisch und regelmäßig.

### 6.3 Regel 90

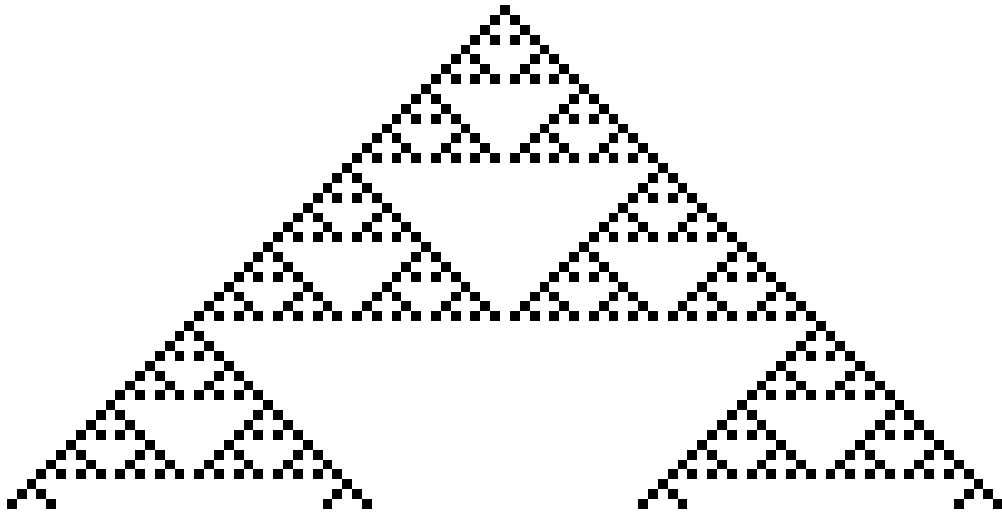
Ein zwar ebenfalls symmetrisches und regelmäßiges Dreieck, das aber schon nicht mehr (leicht) vorhersagbar ist, ist durch die Regel  $\{0, 1, 0, 1, 1, 0, 1, 0\}$  gegeben. Die Regel besagt, dass eine Zelle genau dann schwarz gefärbt werden soll, wenn eine der beiden Nachbarzellen, aber nicht beide, im vorherigen Schritt schwarz gefärbt war, in logischen Symbolen: *Zustand des linken Nachbarn XOR Zustand des rechten Nachbarn*.



Hier ergibt sich eine geschachtelte Struktur (*nested pattern*), bei der sich das Gesamtmuster in kleinen Teilabschnitten wiederholt.



```
Show[CAGraphics[CAEvolveList[ElementaryRule[90], CenterList[41],
20], Automatic]]
```

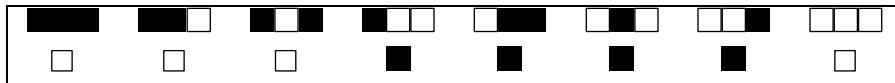


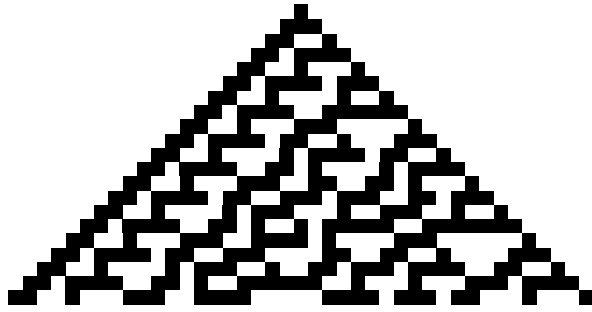
```
Show[CAGraphics[CAEvolveList[ElementaryRule[90],
CenterList[103], 50], 500]]
```

*Anmerkung:* Dieses Dreieck ist eine Darstellung des PASCALSchen Dreieck modulo 2. Schwarze Zellen korrespondieren mit den ungeraden Binomialkoeffizienten.

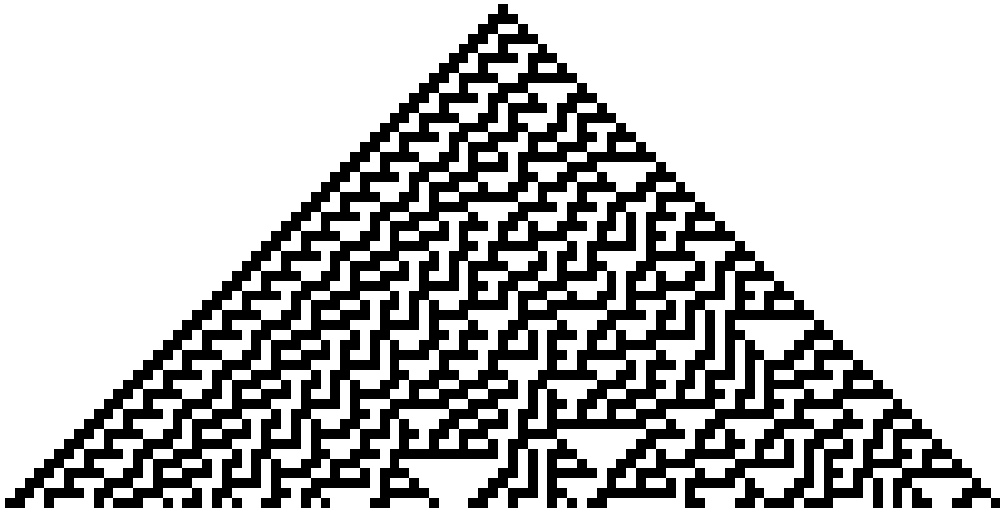
## 6.4 Regel 30

Regel 30 liefert nun das erste Beispiel für eine weder vorhersehbare noch symmetrische noch sonderlich regelmäßige Struktur. Das Muster erscheint in Teilen regelmäßig, in anderen aber sehr zufällig. Die zugrundeliegende Regel lautet:  $\{0, 0, 0, 1, 1, 1, 1, 0\}$ , in logischer Schreibweise *Zustand des linken Nachbarn XOR (Zustand der aktuellen Zelle OR Zustand des rechten Nachbarn)*. Für den Fall, dass sowohl die gerade zu betrachtende Zelle als auch ihr rechter Nachbar weiss gefärbt sind, so wird die aktuellen Zelle wie ihr linker Nachbar gefärbt; andernfalls bekommt die aktuelle Zelle genau die gegenteilige Farbe des linken Nachbarn.

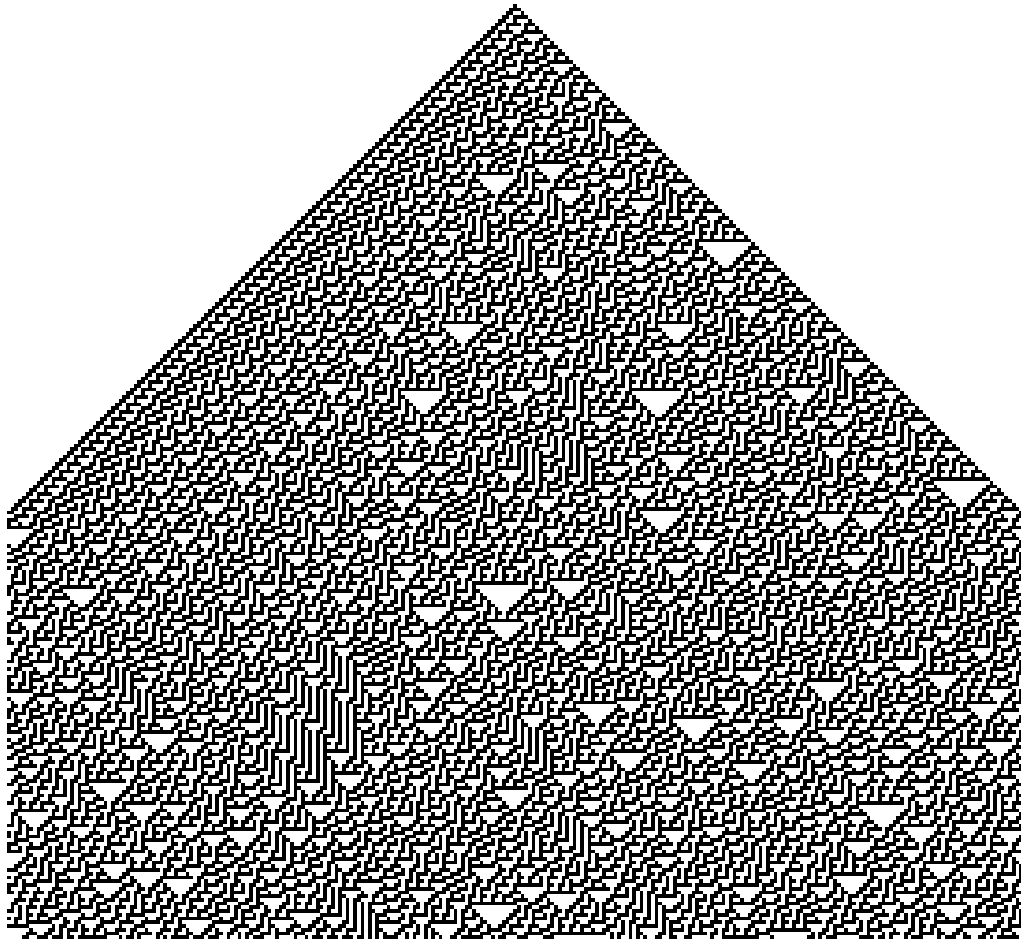




```
Show[CAGraphics[CAEvolveList[ElementaryRule[30], CenterList[41],  
20], Automatic]]
```



```
Show[CAGraphics[CAEvolveList[ElementaryRule[30],  
CenterList[103], 50], 500]]
```



```
Show[CAGraphics[CAEvolveList[ElementaryRule[30],  
CenterList[273], 250], 500]]
```

Spätestens hier hat man ein Beispiel für WOLFRAMS Aussage, dass man auch mit simplen Voraussetzungen komplexe Resultate erhalten kann.

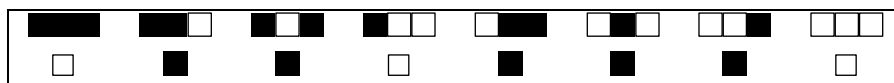
Eine Regelmäßigkeit ist aber auch hier noch geblieben: Die Verteilung der Nullen und Einsen ist ungefähr gleichmäßig, Nullen und Einsen treten ungefähr gleich oft auf. Doch findet man auch Beispiele, bei denen nicht mal mehr diese Regelmäßigkeit zu beobachten ist.

## 6.5 Regel 110

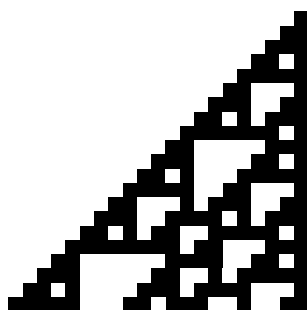
Es ist aber nicht nur möglich, Muster mit sehr verteilten Untermustern zu erhalten, sondern auch Strukturen, deren Verhalten völlig unvorhersehbar und untypisch ist. Das folgende von Regel 110 erzeugte ziemlich extreme Beispiel enthält "unnatürlich wirkende", schräg verlaufende, aus Dreiecken bestehende Linien, die sich nur bei Hunderten von Schritten überhaupt zu erkennen geben. Die Regel in Listenform lautet:  $\{0, 1, 1, 0, 1, 1, 1, 0\}$ . Wenn  $l$  den Zustand des linken Nachbarn bezeichnet,  $x$  den Zustand der aktuellen Zelle,  $r$  den Zustand des rechten Nachbarn, und ein Überstrich der Anwendung der Operation NOT entspricht, dann lautet die logische Schreibweise:

$(x \text{ AND } \bar{r}) \text{ OR } (\bar{l} \text{ AND } r) \text{ OR } (l \text{ AND } \bar{x} \text{ AND } r)$ .

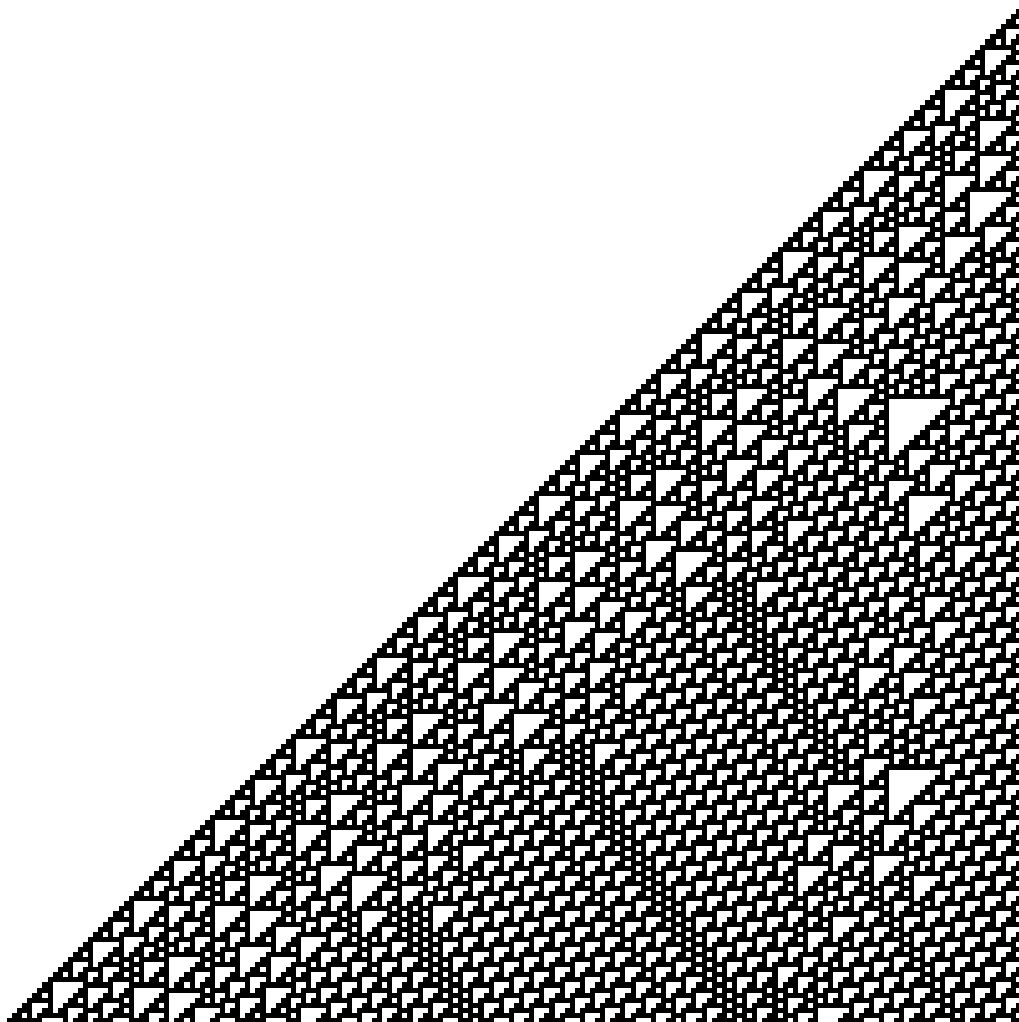
Anschaulich besagt die Regel folgendes: Wann immer die Zelle und ihre beiden Nachbarzellen im ursprünglichen Zustand die gleiche Färbung aufweisen, oder wenn die linke Nachbarzelle schwarz und die aktuelle Zelle und der rechte Nachbar beide weiß gefüllt gewesen sind, ist die neue Färbung der betrachteten Zelle weiß. In den anderen Fällen erhält sie eine schwarze Färbung.



Heraus kommt ein extrem unerwartbares Muster. Für wenige Ausführungen der Regel lassen sich die Unregelmäßigkeiten noch nicht erkennen, für mehrere Tausende von Ausführungen zeigt sich am linken Rand der Grafik noch immer eine ziemlich gleichmäßige Struktur von Dreiecken ungefähr gleicher Größe, in der Mitte der Grafik hat man ein ebenmäßiges Streifenmuster, am rechten Rand aber "verselbständigen" sich einige aus Dreiecken bestehende "Fäden", die keiner erkennbaren Regel folgen.



```
Show[CAGraphics[CAEvolveList[ElementaryRule[110],
CenterList[41], 20], Automatic]]
```



```
Show[CAGraphics[CAEvolveList[ElementaryRule[110],  
CenterList[401], 200], 1000]]
```

WOLFRAM hat durch Betrachten von mehreren Tausend Schritten erkannt, dass sich irgendwann doch eine einzige periodische Struktur herausbildet. Dennoch ist die durch Regel 110 erzeugte Struktur äußerst komplex.

## 7 Erkenntnisse

An den 5 Beispielen des vorhergehenden Abschnitts wird schrittweise deutlich, dass tatsächlich aus einem einfachen Startzustand auf den einfache Re-

geln angewendet werden, ein komplexe Struktur entstehen kann. Die Komplexität ist ausschließlich das Resultat der wiederholten Anwendung einer simplen Regel. Entscheidend ist, dass als Parameter für diese Regel nicht nur der alte Zustand der neu zu färbenden Zelle genutzt wird, sondern die beiden Nachbarn ebenfalls einen (gleichberechtigten) Einfluß nehmen. So wird die Abfolge der Zustandsveränderungen vor allem bei nicht-symmetrischen Regeln irgendwann unüberschaubar. Das experimentelle Vorgehen WOLFRAMS hat Formen hervorgebracht, die sonst nicht betrachtet worden wären; zum einen, weil normalerweise eher symmetrische oder anders regelmäßige Strukturen eher als interessant gewertet werden, zum anderen, weil es aber auch nicht möglich wäre, von den Strukturen auf die zugrunde liegenden Regeln zurückzuschließen. Man kann sich ihnen nicht mit den üblichen Analyse-Werkzeugen der traditionellen Wissenschaften annähern. Denn diese Analyse-Methoden beruhen prinzipiell alle auf mathematischem Vorgehen. Das heißt, sie stammen meistens aus der Analysis oder der Algebra, auf jeden Fall aber zerlegen sie Probleme (was ja auch die Bedeutung des Wortes *Analyse* ist), statt Lösungen (auch zu noch unbekanntem Fragen) durch Konstruktion zu erreichen.

## 8 Weitere Automaten

Die zellulären Automaten haben veranschaulicht, dass auch aus einfachen Regeln komplexe Strukturen erwachsen können. Nun stellt sich die Frage, wie typisch Automaten mit solchen Eigenschaften sind. Gelten die beobachteten Phänomene auch für andere Automatentypen wie z.B. die bekannte Turing-Maschine? Im dritten Kapitel von *A New Kind of Science* werden andere Automatentypen vorgeführt, unter anderem nicht-binäre zelluläre Automaten, die hier abschließend noch vorgeführt werden. Dabei soll sich herausstellen, dass diese verschiedenen Automatentypen durchaus ähnliche Ergebnisse produzieren.

### 8.1 Weitere zelluläre Automaten

Wie im Anhang ersichtlich sind die von den zellulären Automaten erzeugten grafischen Outputs teils recht unterschiedlich, aber es gibt einige Muster, die sich wiederholen. Oft sind es nur einzelne schwarze Zellen oder schmale schwarze Striche. Des weiteren sind oft einfache geschachtelte Strukturen zu erkennen. Etwa zwei Drittel der zellulären Automaten produzieren ein Muster von fester Größe, etwa ein Drittel aber produziert immer weiter wachsende Strukturen. Von diesen sind viel sehr regelmäßig, z.B. geschachtelt.

10 der Muster jedoch sind als komplex bzw. zufällig zu bezeichnen. Beispiel dafür ist der bereits bekannte zelluläre Automat mit der zugrunde liegenden elementaren Regel 110.

WOLFRAM fragt sich nun, ob durch Komplizierung der zellulären Automaten eine bessere Ausbeute an komplexen Mustern erreichbar ist. Er nimmt eine kleine Modifikation vor – er betrachtet tertiäre statt binärer zellulärer Automaten. Also solche zellulären Automaten, deren Zellen 3 mögliche Zustände besitzen. Die Zahl der möglichen Regeln beläuft sich dann auf  $3^{3^3} = 7,625,597,484,987$ . Diese Zahl ist zu groß, um empirische Studien zu betreiben, bei denen jeder dieser Automaten betrachtet wird. WOLFRAM begegnet diesem Problem dadurch, dass er den Begriff der *totalistischen* Regel prägt. Bei totalistischen Regeln ist nur die durchschnittliche Farbe des Eingabe-Tripels von Belang, also die Farbe, die sich ergibt, wenn man die drei Farben der drei Zellen miteinander mischt. Bei Farben schwarz, grau und weiß bzw. Werten 2, 1 und 0 gibt es 7 mögliche Mischfarbtöne, denn aus 3 Summanden mit möglichen Werten 2, 1, 0 können sich folgende mögliche Summen ergeben: 6, 5, 4, 3, 2, 1, 0 – also 7 mögliche Mischungen. Damit gibt es  $3^7 = 2187$  mögliche totalistische Regeln. Wie schon bei den binären, nicht totalistischen Regeln lassen sich diese Regeln einfach nummerieren. Man legt eine Fixierung der möglichen Eingabewerte fest, bei WOLFRAM vom dunkelsten bis zum hellsten Farbton, und kann die Regeln dann als 7-stellige Liste mit Werten 0, 1 oder 2 formulieren. Diese Liste aufgefasst als Zahl zur Basis 3 liefert eine eindeutige Nummer für die Regel.

## 8.2 Modifizierter Mathematica-Code

Es ergibt sich der folgende, modifizierte Mathematica-Code für totalistische ein-dimensionale zelluläre Automaten.

```
CenterList[n_Integer] :=
  ReplacePart[Table[0, {n}], 1, Ceiling[n/2]]
CASStep[TotalisticCARule[rule_List, 1], a_List] :=
  rule[[-1 - (RotateLeft[a] + a + RotateRight[a])]]
CAEvolveList[TotalisticCARule[rule_, 1], init_List, t_Integer] :=
  NestList[CASStep[TotalisticCARule[rule, 1], #] &, init, t]
CAGraphics[history_List, size_] :=
  Graphics[Raster[1 - Reverse[history / 2]],
    AspectRatio -> Automatic, ImageSize -> size]
```

*Erläuterung:* *rule* ist hier anders als bei den elementaren Regeln keine 8-, sondern eine 7-stellige Liste. Listen in Mathematica sind zyklisch. Für eine  $n$ -stellige Liste *liste* gibt der Aufruf `liste[[0]]` *List* zurück, also den Typ von *liste*. Für alle Indizes  $i, j$  gilt, dass für  $i \equiv j \pmod{n}$   $i$  und  $j$  Indizes für dieselbe Listenposition sind, also `liste[[i]]` dasselbe Element zurückgibt wie `liste[[j]]`.

`CAStep` funktioniert unter dem Gesichtspunkt, dass die Rechnung  $-1 - i$  bei Zugriff auf eine 7-stellige Liste einen gleichwertigen Index wie  $7 - i$  liefert, genau analog zu der Funktion `CAStep` bei den elementaren Regeln. Vor `RotateLeft[a]`, `a`, `RotateRight[a]` treten hier keine Faktoren auf, da nur die Mischung der Farben betrachtet wird.

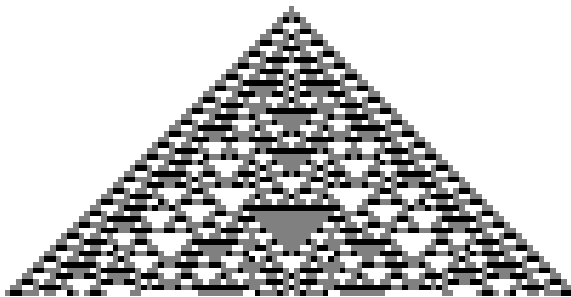
An `CAEvolveList` wurde nur eine syntaktische Veränderung vorgenommen. Die Methode `Raster` bildet 0 schwarz ab, die Werte zwischen 0 und 1 in Grauwerten und alle Wert gleich oder größer 1 weiß. *history* kann als Liste, die einen zellulären Automaten mit 3 möglichen Zuständen der Zellen repräsentiert, die Werte 0, 1 und 2 enthalten. Darum müssen die Werte der Eingabeliste *history* bei `CAGraphics` noch durch 2 geteilt werden, damit nicht alle Einträge ungleich 0 weiß abgebildet werden. Ansonsten entspricht `CAGraphics` der bereits von den elementaren Regeln bekannten Methode `CAGraphics`.

## 8.3 Beispiele

Betrachtet man eine große Auswahl von totalistischen dreifarbigem zellulären Automaten, so stellt sich heraus, dass sie sich im Verhalten nur relativ schwach von den elementaren zellulären Automaten unterscheiden. Und das, obwohl das zugrunde liegende Regelwerk oft schon entschieden komplexer ist. Die bereits bekannten symmetrischen geschachtelten Dreiecksmuster der elementaren zellulären Automaten treten in ähnlicher Form wieder auf. Neue, noch nicht dagewesene Muster sind dagegen seltener als zu erwarten gewesen wäre.

### 8.3.1 Regel 777

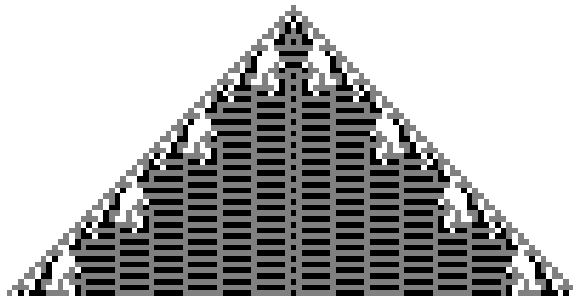
Regel 777 produziert ein symmetrisches, recht komplexes Muster, das eine ganz gute Verteilung der Farben weiß, schwarz und grau aufweist und somit ein gutes Beispiel für einen dreiwertigen zellulären Automaten gibt.



```
Show[CAGraphics[CAEvolveList[TotalisticCARule[IntegerDigits[777,3,9],1],CenterList[103,50], Automatic]]]
```

### 8.3.2 Regel 1110

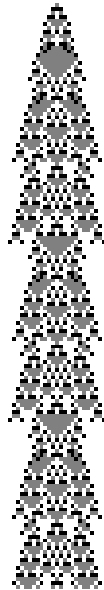
Totalistische zelluläre Automaten liefern oft Muster, die an den Rändern noch einigermaßen unregelmäßig, in der Mitte aber extrem geordnet sind. Regel 1110 gibt ein typisches Beispiel dafür.



```
Show[CAGraphics[CAEvolveList[TotalisticCARule[IntegerDigits[1110,3,9],1],CenterList[103,50], Automatic]]]
```

### 8.3.3 Regel 600

Regel 600 liefert ein Beispiel für eine wachstumsbeschränkte Struktur und somit für eine Struktur, die typisch für totalistische, aber nicht für elementare zelluläre Automaten ist. Das Muster ist recht simpel, nach ein paar Dutzend Schritten wiederholt es sich für immer. Von daher bietet es keine neue Quelle für Komplexität.

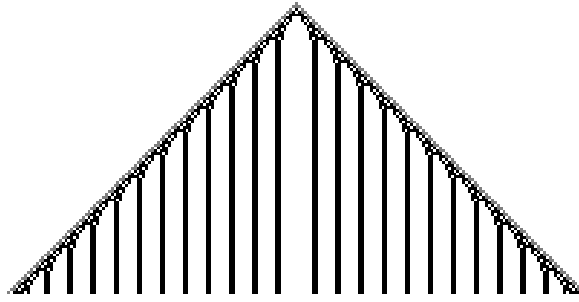


```
Show[CAGraphics[CAEvolveList[TotalisticCARule[IntegerDigits[600,3,9],1],CenterList[103,150], Automatic]]
```

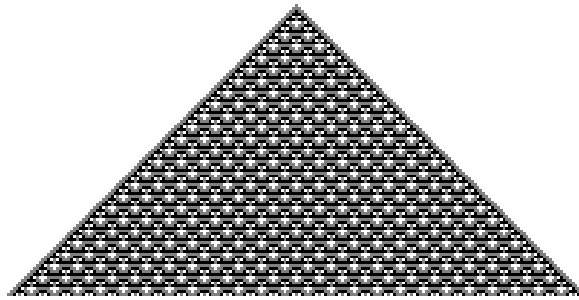
#### 8.3.4 Simple Muster

Totalistische zelluläre Automaten ergeben nicht selten einfache, durch Wiederholungen charakterisierte Muster. Diese unterscheiden sich in der Erscheinung zwar ein wenig von den von den elementaren Regeln bekannten Mustern, dennoch sind sie nicht weniger symmetrisch und keine Quelle für Komplexität.

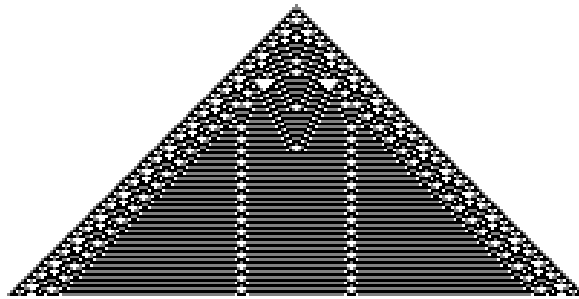
Regeln 219, 957, 966 und 1884 sind einige typische Beispiele für simple Strukturen, die von totalistischen Regeln erzeugt wurden.



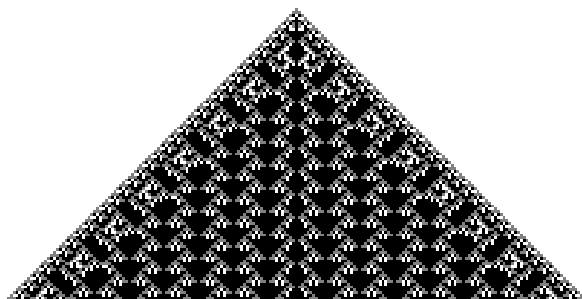
```
Show[CAGraphics[CAEvolveList[TotalisticCARule[IntegerDigits[219,3,9],1],CenterList[203],100], Automatic]]
```



```
Show[CAGraphics[CAEvolveList[TotalisticCARule[IntegerDigits[957,3,9],1],CenterList[203],100], Automatic]]
```



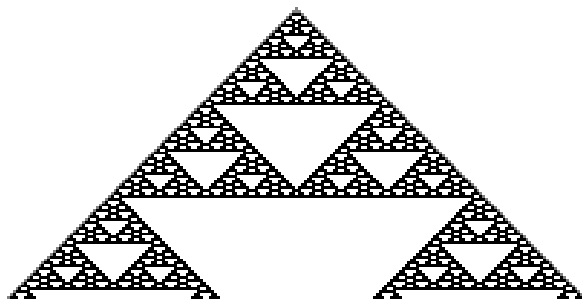
```
Show[CAGraphics[CAEvolveList[TotalisticCARule[IntegerDigits[966,3,9],1],CenterList[203],100], Automatic]]
```



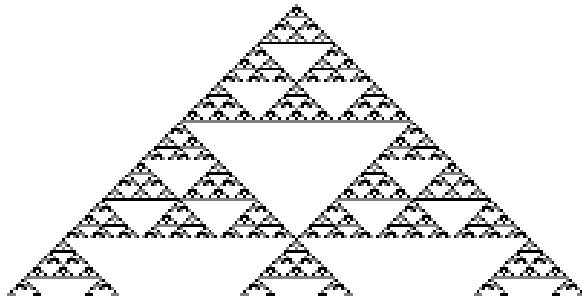
```
Show[CAGraphics[CAEvolveList[TotalisticCARule[IntegerDigits[  
1884,3,9],1],CenterList[203],100], Automatic]]
```

### 8.3.5 Nested Patterns

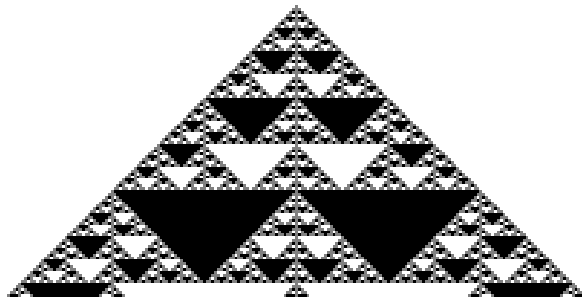
Auch bei den totalistischen zellulären Automaten treten geschachtelte Strukturen auf. Besonders die bereits bekannten Dreiecksmuster erscheinen oft, siehe dazu Regel 237, 420 und 1749. Neu ist, dass auch "runde" Formen auftreten, Beispiel hierfür liefert Regel 948.



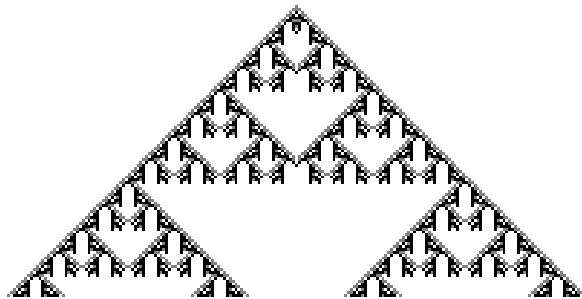
```
Show[CAGraphics[CAEvolveList[TotalisticCARule[IntegerDigits[  
237,3,9],1],CenterList[203],100], Automatic]]
```



```
Show[CAGraphics[CAEvolveList[TotalisticCARule[IntegerDigits[420,3,9],1],CenterList[203],100], Automatic]]
```



```
Show[CAGraphics[CAEvolveList[TotalisticCARule[IntegerDigits[1749,3,9],1],CenterList[203],100], Automatic]]
```

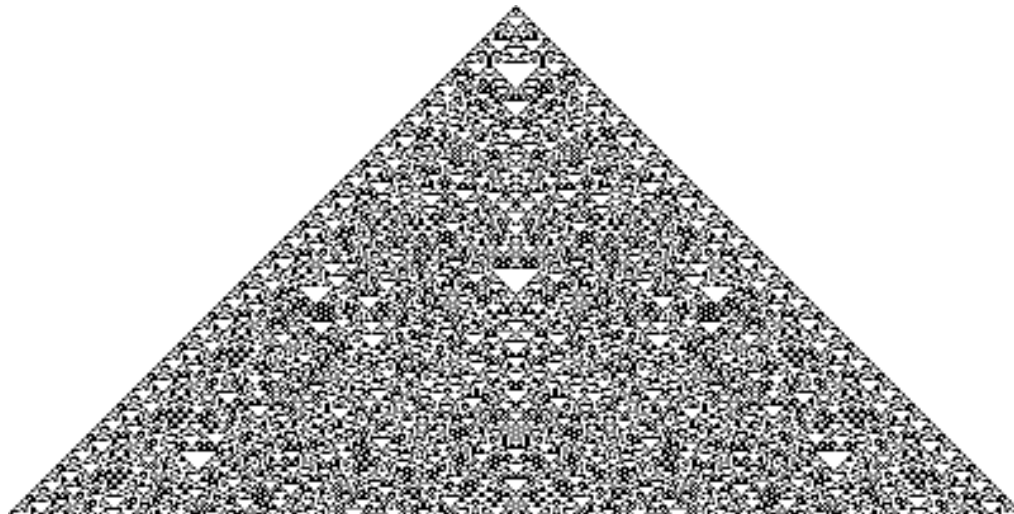


```
Show[CAGraphics[CAEvolveList[TotalisticCARule[IntegerDigits[948,3,9],1],CenterList[203],100], Automatic]]
```

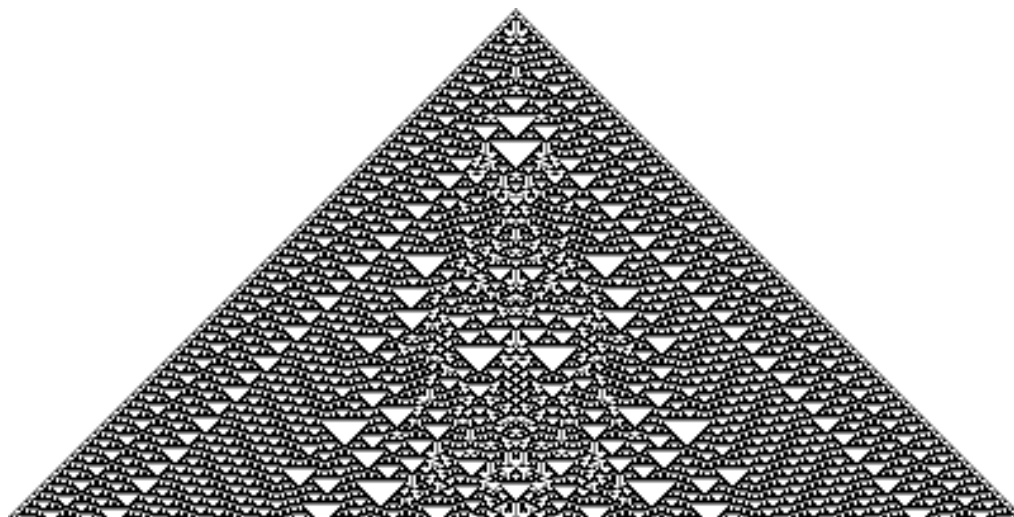
### 8.3.6 Komplexe Strukturen

Totalistische zelluläre Automaten bringen – genau wie die elementaren – mitunter Strukturen hervor, die zufällig erscheinen, also hohe Komplexität

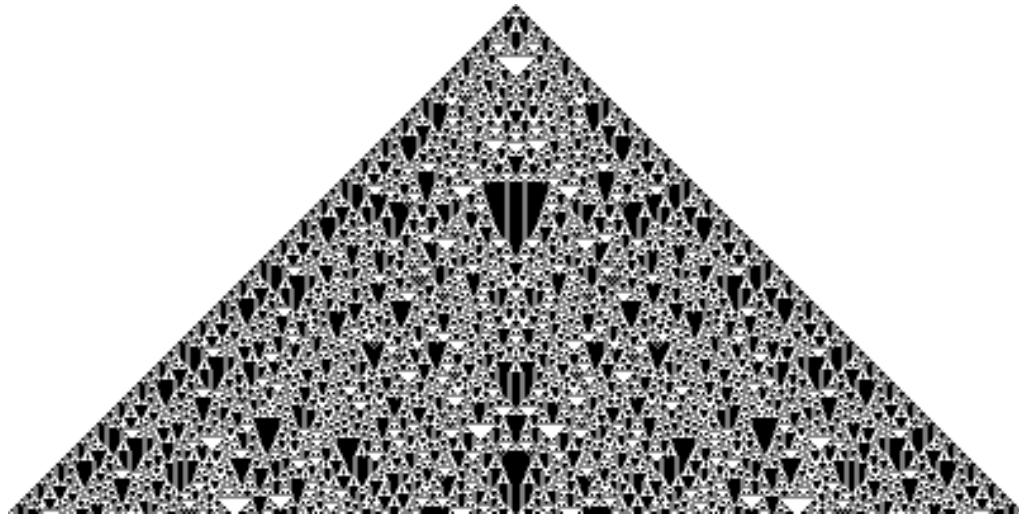
aufweisen. Beispiele hierfür geben Regeln 177, 912 und 2040. Die durch Regel 912 gegebene Figur hat als Besonderheit, dass die Komplexität an den Rändern des Dreiecks schwach ist und zur Mitte hin zunimmt. Dies erinnert an den elementaren zellulären Automaten, der durch die elementare Regel 30 definiert ist.



```
Show[CAGraphics[CAEvolveList[TotalisticCARule[IntegerDigits[  
177,3,9],1],CenterList[401],200],500]]
```



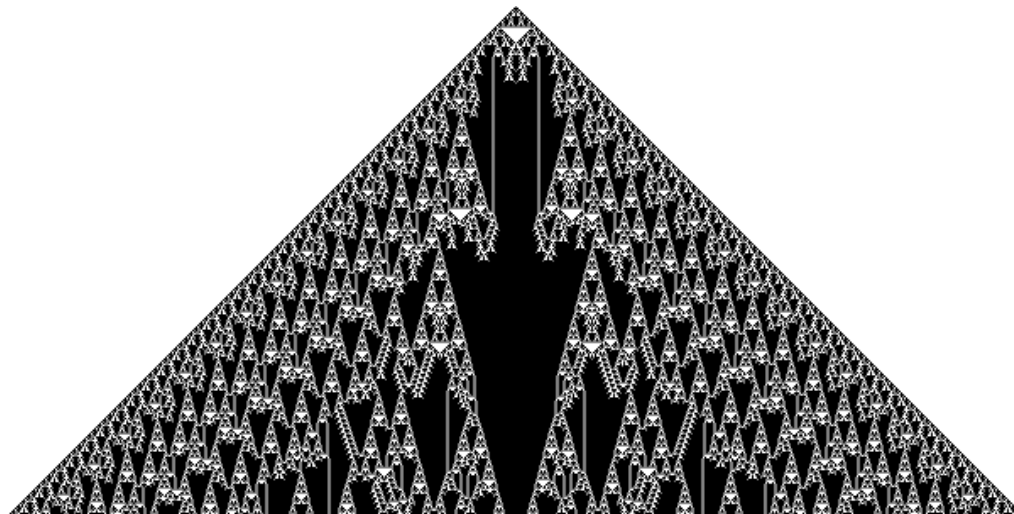
```
Show[CAGraphics[CAEvolveList[TotalisticCARule[IntegerDigits[  
912,3,9],1],CenterList[401],200],500]]
```



```
Show[CAGraphics[CAEvolveList[TotalisticCARule[IntegerDigits[  
2040,3,9],1],CenterList[401],200],500]]
```

### 8.3.7 Hochkomplexe Strukturen

Die elementare Regel 110 erzeugte ein Muster, das teils regelmäßig ist, teils von stark "ungeometrischen" Linienzügen durchzogen ist. Beim groben Übersehen der 2187 totalistischen zellulären Automaten (die ich wegen der zusätzlichen 61 Seiten mal nicht in den Anhang dieser Ausarbeitung aufgenommen habe) fallen kaum solch extreme Muster ins Auge, ein paar Beispiele gibt es aber doch, Regeln 1041, 1635 und 2049 liefern z.B. solche besonderen Resultate. Das folgende Bild zeigt 300 Schritte der Entwicklung des zellulären Automaten mit der zugrunde liegenden Regel 2049. Das Verhalten ist zwar symmetrisch, aber die Linienzüge, die die regelmäßige von Dreiecken gebildete Struktur durchbrechen erscheinen höchst zufällig.



```
Show[CAGraphics[CAEvolveList[TotalisticCARule[IntegerDigits[  
2049,3,9],1],CenterList[601],300],500]]
```

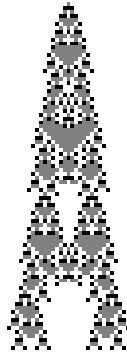
### 8.3.8 Auslöschung

Eine weitere Klasse von Strukturen zeichnet sich dadurch aus, dass sie komplex beginnt, dann aber in eine sehr einfache Form übergeht, z. B. nur noch aus weißen Kästchen besteht. Regeln 357 und 2058 bieten Beispiele hierzu.

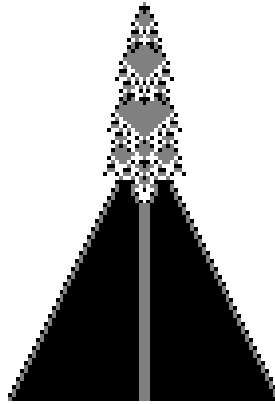
Regel 357 erzeugt ein Muster, das nach nur circa 100 Schritten zu wachsen aufhört und sogar völlig erlischt.

Regel 2058 produziert eine Struktur, die ebenfalls komplex beginnt und sich dann zu einem simplen Muster entwickelt. Hier ist das simple Muster ein schwarzes, konstant wachsendes Dreieck, das von einer regelmäßigen Struktur umrandet ist und von einem ebenfalls regelmäßigen grauen Balken durchzogen wird.

Hier sind jeweils 100 Schritte abgebildet.



*Regel 357*



*Regel 2058*

## 8.4 Schlussfolgerungen

Betrachtet man den erhaltenen grafischen Output, den die letzten Beispiele geliefert haben, stellt man fest, dass die sich ergebenden Strukturen kaum komplexer als bei den bereits bekannten elementaren zellulären Automaten ist. Man hat einerseits unmittelbar regelmäßige oder verschachtelte Strukturen und einige Ausnahmen mit komplexen Verhalten, die aber ähnlich zu den Beispielen der elementaren zellulären Automaten sind; andererseits bildet sich eine große Gruppe von wachstumsbeschränkten Formen heraus (z.B. das von Regel 600 erzeugte Muster) – aber diese weisen höchstens schwache Komplexität auf. Es gibt einige Beispiele, wo ein Grad von Komplexität erreicht wird, der mit den elementaren zellulären Automaten nicht zu erreichen war. Aber diese sind nicht zahlreich und so sieht die Gesamtheit der durch dreiwertige totalistische zelluläre Automaten erzeugten Muster beim groben Überblicken ähnlich aus wie die Gesamtheit der durch elementare Regeln erzeugten. Circa 85 Prozent der dreifarbig totalistischen zellulären Automaten erzeugen geordnete Muster. Dieser Anteil ist in etwa genau so hoch wie bei den elementaren zellulären Automaten. Man kann also die generelle Aussage formulieren, dass aus dem Hinzufügen von Komplexität in den zugrunde liegenden Regeln im allgemeinen *keine* erhöhte Komplexität im Verhalten folgt. Dies ist eine weitere Bestätigung für WOLFRAMs Aussage, dass komplexes Verhalten nicht zwingend auf komplexen zugrundeliegenden Regeln fußen muss, sondern dass vielmehr auch einfache Regeln bzw. Programme komplexes bzw. "zufälliges" Verhalten erzeugen können.

# 9 Anhang

## 9.1 Die 256 elementaren zellulären Automaten

