Elementary Cellular Automata

by

Abel Mengistu

Advisor: Dr. Andrew Rich

Math Senior Research 2011

Manchester College

May, 2011

Table of Contents

Introduction

Cellular automata are abstract mathematical systems that live in a discrete space-time. In the computational universe of cellular automata, physical quantities take on a finite set of discrete values which evolve with time. CA consist of a regular grid of cells, each of which is in a finite number of states. Each cell evolves in time depending on the state of its neighbors and a given set of rules. Cellular automata are often studied as models for biological systems; however, they have been used to model freeway traffic, earthquakes, forest fire, fluids etc. The most fundamental property of cellular automata is the type of grid on which it is computed. It can vary from a simple one-dimensional line to a k-neighbor multi-dimensional space. As time evolves, generations of new cells are created according to a mathematical function that assigns to each old cell its new state based on the states of its neighboring cells. The goal of this project was to study some properties of cellular automata.

History

In the 1940s John von Neumann was working on constructing a self-replicating system. His initial design was based on the notion of one robot building another one. This design is known as the kinematic model. Developing a kinematic model based self-replicating system turned out to be a very difficult task and von Neumann, with advice from his colleague Stanislaw Ulam, changed his approach to using a mathematical abstraction. This led to development of von Neumann's two-dimensional cellular automata. The result was a universal copier and constructor that was able to replicate itself within the given computational universe (Pesavento).

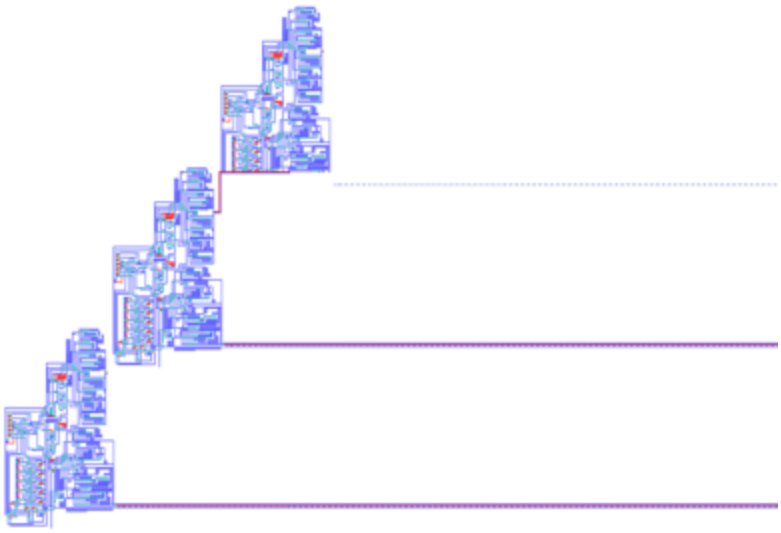
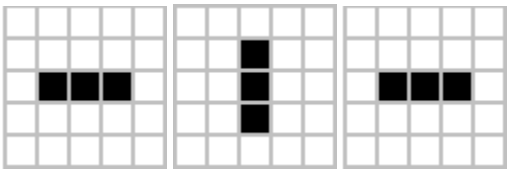


Figure 1. Implementation of von Neumann's universal copier and constructor

In the 1970s cellular automata was popularized by Conway's Game of Life, a two-dimensional Moore-neighborhood CA, invented by John Conway with the following rules: if a cell has two black neighbors, it stays the same. In the case it has 3 black neighbors, it becomes black. For all other cases it becomes white. The following illustration shows a pattern within the CA that oscillates as time goes.



$t = 0$ $t = 1$ $t = 2$

There are other structures such as blocks (stable), spaceships (move through space with constant velocity), and guns (creates spaceships). After much effort it has been shown that the Game of Life can emulate a universal Turing machine (Chapman).

Elementary Cellular Automata

One-dimensional nearest-neighbor two-state cellular automata are commonly known as Elementary Cellular Automata (ECA). They are the simplest one-dimensional CA. Each cell has two values (0 or 1), and rules that only depend on its current state and the states of its nearest neighbors. Practically, a CA is implemented as having a finite number of cells in finite space. A consequence of this is the two boundary cells only have one neighbor. There are two remedies to this problem – either treat the empty space next to the cells as 0, or let space wrap around as a cylinder allowing the edge cells to be neighbors of each other.

The evolution of an ECA can be described completely by a table specifying the state of a given cell will have in the next generation given the state of the cell on its left, its own state and the state of the cell on its right. The rule is a mathematical function that assigns to a three bit number, a one bit number which corresponds to the new state of the middle cell. Since there are 8 possible binary states for 3 cells, there are a total of $2^8 = 256$ ECA.

x	f(x)
000	0
001	1
010	0
011	0
100	1
101	0
110	0
111	0

Table 1

A generalization to n-number of states with a neighborhood size (width) of k can be made. n^k possible states for k cells, which means there are a total of n^{n^k} CA with n-states and neighborhood size k.

Wolfram code

Stephen Wolfram, a major researcher in the field of Cellular Automata and complex systems, developed an intuitive naming convention for the 256 different ECA. We enumerate all 256 ECA rules with the corresponding output column, $\alpha = [\alpha_0\alpha_1 \dots \alpha_7]$. To each rule we assign an integer,

$$N = \sum_{i=0}^{7} \alpha_i 2^i$$

Using this naming scheme, the rule that always assigns 0, will be named rule 0, and the rule that always assigns 1 will be named rule 255. For example, for the rule stated in Table 1, the corresponding wolfram code is 18. It is important to note this scheme can be generalized to an n-state nearest neighbor. For an n-state CA, by the same reasoning as before there are a total of n^{n^3} possibilities. Therefore our output column is $\alpha = \left[\alpha_0\alpha_1 \dots \alpha_{n^{n^3}-1}\right]$. To each rule we assign an integer,

$$N = \sum_{i=0}^{n^{n^3}-1} \alpha_i 2^i$$

All 256 ECA rules with their corresponding Wolfram code applied to a starting configuration of a single 1 cell in the middle can be found in the appendix. Note that in the appendix, a white pixel represents state 0.

Essentially the Wolfram code employs a decimal representation of the output column to identify a rule.

Totalistic Cellular Automata

The number of possible CA with a given number of state and neighborhood size grows extremely fast, already reaching an order of 10^{12} for 3-state nearest neighbor CA, and 10^{38} for a 4-state nearest neighbor CA. Wolfram describes another class of CA called totalistic CA that reduce the number of different rules significantly.

A totalistic CA is very similar to a standard CA except for the fact that the domain of the mapping function is an integer. Specifically, the domain integer represents the sum of all the states in a given input configuration. For example, in a totalistic two-state nearest neighbor CA, there are 8 possible combinations of input cells (Table 1). When we add up the state of each input cell, we get a range between 0 and 3. Generally for an n-state k-neighbor CA, we get a range between 0 and $(n-1)k$. For each input, we can assign n different states for output, giving us a total of $n^{(n-1)k+1}$ rules for an n-state k-neighbor CA.

An example of a 3-state nearest neighbor totalistic CA is provided below along with its space-time diagram.

x	f(x)
0	1
1	1
2	0
3	2
4	1
5	0
6	1

Table 2

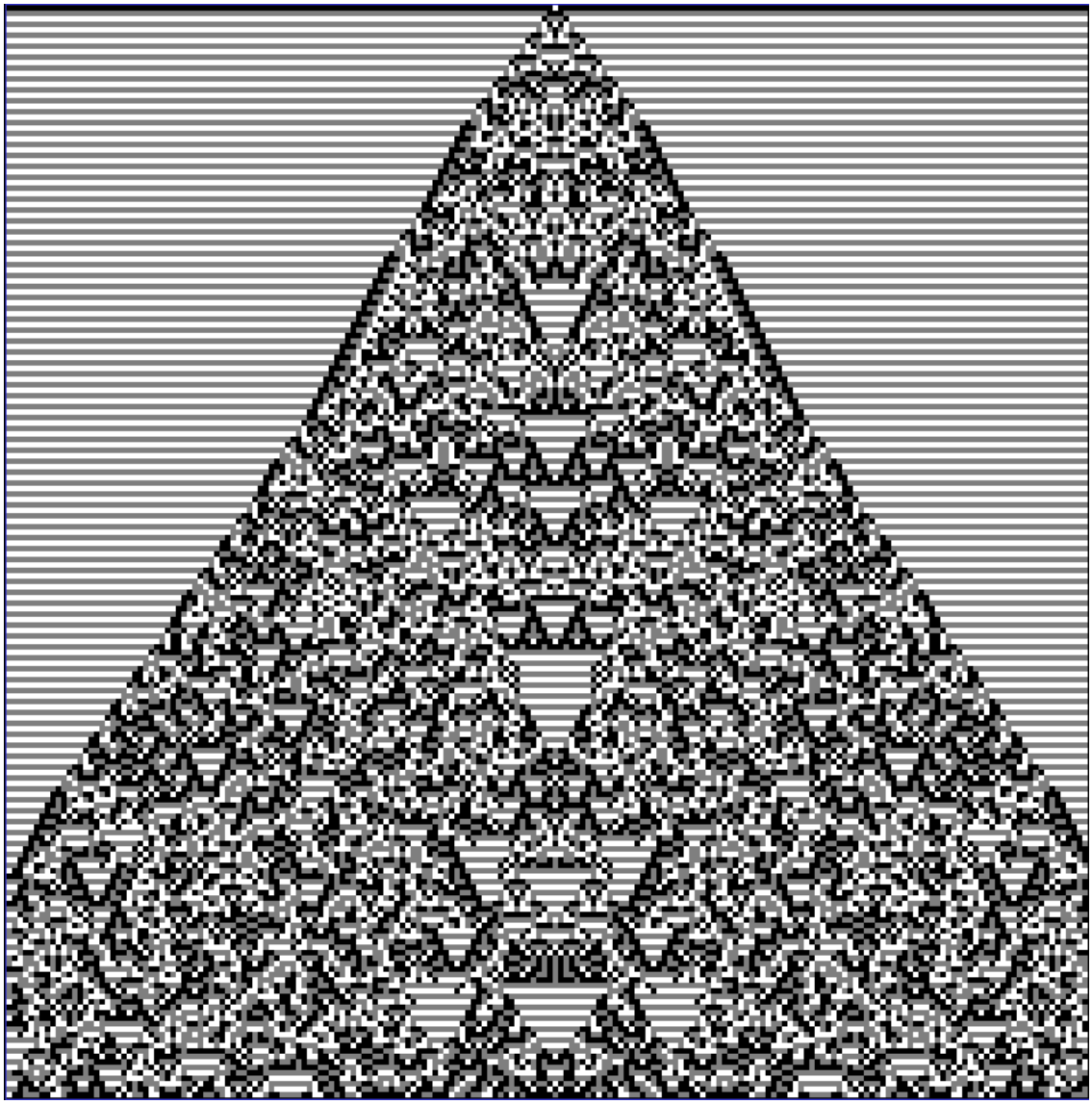


Figure 2: Space-time of totalistic CA with rule stated in Table 2

Note that in figure 2, each cell represents a cell, and the brightness of the cell represents the state. Therefore, white pixels represent state 2, gray cells state 1, and black cells state 0 in this case. We can also employ a similar strategy as the Wolfram code for naming totalistic rules by using the decimal representation of the output column. In the case of the rule in Table 2, it would be identified as rule 868 3-state totalistic CA.

Computational Universality

As Minsky said, a universal system is "a completely general instruction-obeying mechanism." Stephen Wolfram while studying cellular automata in the 1980s suggested that rule 110 is universal – that is, Turing complete. In computability theory, a system of data manipulation rules is said to be Turing complete if and only if it can be used to simulate any single-taped Turing machine. A Turing machine, named after the English mathematician and computer scientist Alan Turing, is a theoretical device that manipulates symbols on a strip of tape according to a given set of rules. Later in 1990, Matthew Cook, a research assistant to Wolfram proved the conjecture and showed that rule 110 is indeed Turing complete (Cook).

Wolfram Classification Scheme

A popular method of classifying cellular automata and several other simple computational models was defined by Wolfram in his book a *New Kind of Science* (Wolfram). In order of increasing complexity the classes are:

- Class 1: Nearly all initial patterns quickly evolve into a stable, homogenous state. Any randomness in the initial pattern disappears.
- Class 2: Nearly all initial patterns quickly evolve into stable or oscillating structures. Some of the randomness in the initial pattern remains. Local changes to the initial pattern tend to remain local.
- Class 3: Nearly all initial patterns evolve in a pseudo-random and chaotic manner. Stable structures and oscillating structures quickly disappear after formation due to neighboring noise. Local changes to the initial pattern tend to spread indefinitely.
- Class 4: Nearly all initial patterns evolve into structures that interact in complex and interesting ways. Might eventually transition to exhibiting class 2 type behavior; but the number of steps required to reach this state is generally very large. Local changes to the initial pattern may spread indefinitely.

This scheme has been criticized for being ambiguous and not well defined. The classification is made by visually looking at the space-time of a CA and picking the best fitting category. More importantly, it fails to capture the idea of universal computation.

Analysis

Analysis of the CA requires fast computation capabilities which doing by hand does not provide. Also, a free open-source program that enables the generation of CA that supported many features could not be found. Therefore, as part of this project, a software program called Cellular Automata User Interface (CAUI) was developed. The

program generated grayscale images of desired width and height which represent the space-time of one-dimensional CA. As input the program takes the number of states, a neighborhood definition, and the rule (either in Wolfram code or a “bit” string). It has features such as random initial configuration, wrap-around, etc. The time complexity of the algorithm is $n^2$where n is the number of generations computed.

While studying the properties of ECA we came across many questions, as a result a major part of this project included proposing possible answers for the questions we encountered.

Recurrence of the Sierpinski triangle

The Sierpinski triangle is a fractal, a geometric shape that can be split into parts, each of which is a reduced copy of the whole, described in 1915 by the Polish mathematician Waclaw Sierpinski. A Sierpinski triangle is constructed from a triangle by shrinking it to half the size and making three copies which are positioned such that each shrunken triangle touches the other two at a corner. This process is repeated.

The above diagram shows the construction of 4 steps of the Sierpinski triangle. Looking through the appendix the occurrence of this pattern is eye catching. Out of the 256 ECA such behavior is exhibited in 19 of them namely: rules 18, 22, 26, 60, 82, 90, 126, 129, 137, 146, 154, 161, 165, 167, 181 ,182, 193, 195, 210 and 218 when

starting with a single one in the middle column. Some of these have to be reoriented in order to match the above description.

We look for a set of sufficient conditions that guarantee the formation of the Sierpinski triangle starting with a single 1 state. We can assert that $f(0,0,0) = 0$, this is clearly necessary. Furthermore, the conditions $f(0,0,1) = 1$ and $f(1,0,0) = 1$ also must be satisfied in order for the triangles edges to extend. However, these are necessary but not sufficient conditions for the formation of a Sierpinski triangle. Along with the two additional conditions, $f(1,0,1) = 0$ and $f(0,1,0) = 0$ nonetheless, we have sufficient conditions for the formation of a Sierpinski triangle. All the other cases, $f(1,1,1), f(0,1,1)$ and $f(1,1,0)$ need not be assigned to a particular bit. This sufficient condition accounts for 8 of the 19 occurrences, specifically the rules who written in binary are of the form $** 01 * 010$ where $*$ indicates either 0 or 1. Given a rule N, we can check if it satisfies the condition by checking if $N\ mod\ 64 \equiv 18\ \ or\ N\ mod\ 64 \equiv 26$. A similar approach will allow us to have sufficient conditions for the other unaccounted for rules who also exhibit Sierpinski triangle behavior, although somewhat differently.

Isomorphism

Another question we addressed is the possibility of emulating a CA with more number of states (higher-order) with a lower-order CA and vice versa. Note that in this section we are strictly considering a single 1 middle column initial configuration. Also by emulate we mean to find a CA that qualitatively behaves like another one. Emulating a lower-order CA with a higher-order CA is a trivial task and it only requires every

possible configuration in the higher-order CA that is also in the lower-order CA to be mapped to the same state as in the lower-order CA.

On the other hand, emulating a higher-order CA with a lower-order one is not an easy task and requires a cleverly constructed rule for a CA that in width, or neighborhood size, is significantly larger than that being emulated. Nevertheless, using the fact that rule 110 is computationally universal, we can state that any cellular automata can be emulated by it, in turn showing that it is possible for a lower-order CA to emulate a higher-order one.

Symmetry along central column

When looking at an ECA starting with a single 1 in the middle column, we can consider the symmetry along the middle column. Symmetry, throughout this paper would be used in that sense.

64 out of the 256 ECA are symmetric, which led us to look for both necessary and sufficient conditions for this property. In order to have symmetry along a middle column, we want the rule to preserve the symmetry of every 3-bit input. Out of the eight possible inputs (look at Table 1), '000','111','101' and '010' are symmetric along the middle column therefore we need not worry about what they are assigned to. However, we have to make sure '100' and '001' are assigned to the same bit in order to preserve the symmetry. Also '110' and '011' for the same reason have to be assigned to the same bit. These two conditions give us both necessary and sufficient conditions for symmetry. To summarize, a rule of the form $*A*BA*B*$ is symmetric. $*, A, B$ are each either 0 or 1.

Random initial configuration

The behavior in the evolution of CA when starting with randomly set states can be classified according to the Wolfram classification scheme. We will first discuss some ECA that keep all the information that is stored in the initial configuration.

There is the trivial rule that keeps the state of a cell the same from one generation to the next. That is the ECA with rule $f(1,1,1) = 1, f(1,1,0) = 1, f(1,0,1) = 0$ and so on.

There are two other rules that also completely preserve the initial information other than shifting their position in space. These are the rules that move the state of each cell in either direction. For example consider the rule given by the following table:

x	f(x)
000	0
001	0
010	0
011	0
100	1
101	1
110	1
111	1

Table 3

We see that this rule in fact takes on the bit of the leftmost input bit, effectively moving the entire row one step to the right in the next generation. Such a rule is not interesting and trivial. However, while we were studying the effect of a symmetric CA on a random initial configuration, we found that a symmetric CA in its evolution does not have structures that move in one direction for a long period of time. And a significant proportion of CA that did not satisfy the symmetry condition stated earlier

in this paper, had structures that eventually form and propagate in one direction for long periods of time before interference or in some cases forever.

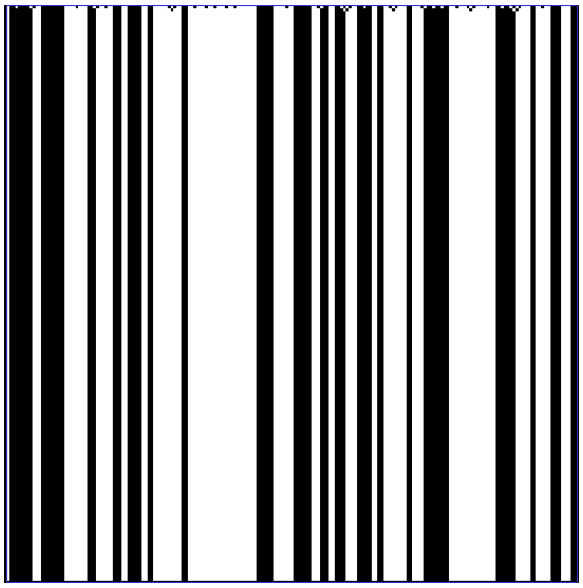


Figure 3: rule 232

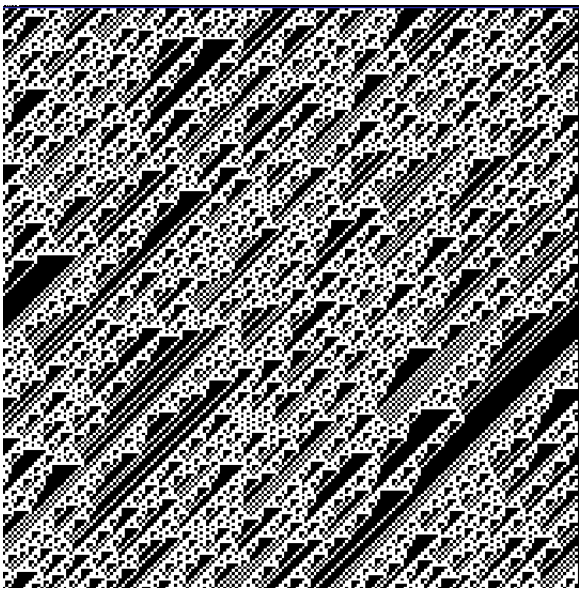


Figure 4: rule 106

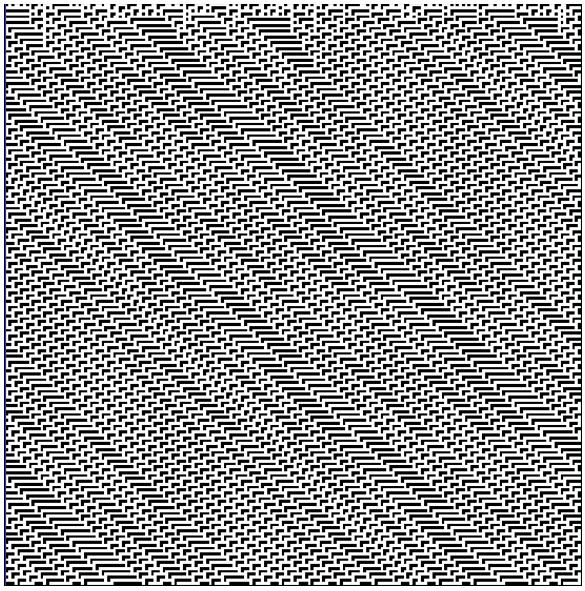


Figure 5: rule 110

Figure 3 is the space-time diagram of a symmetric CA with random initial configuration; we can see there is no apparent propagation of structures in a particular direction. Figures 4 and 5 however show the space-time diagram of asymmetric CA with random initial configurations; in both cases we see structures that propagate in one direction.

Conclusion

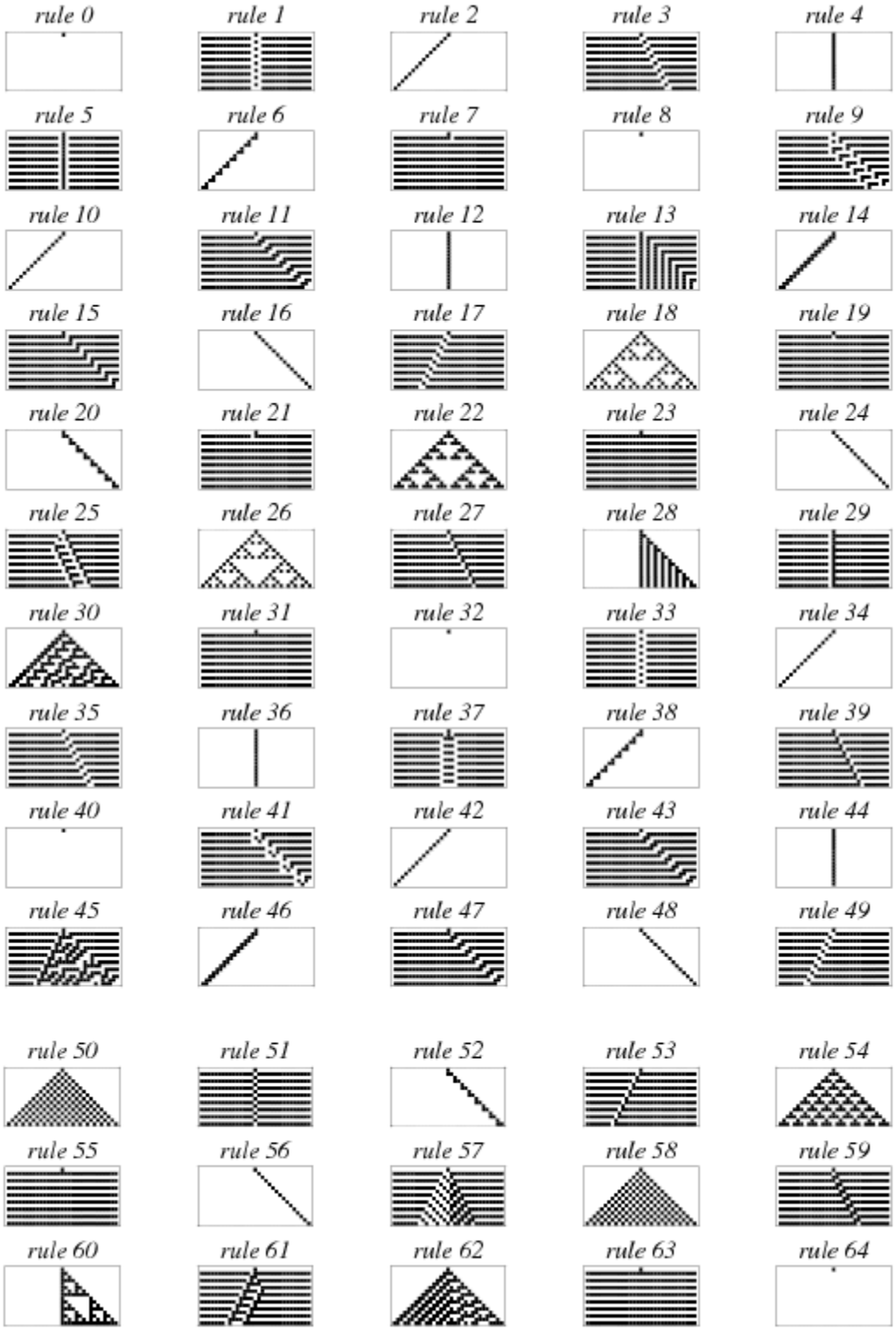
The goal of this project was to study some properties of CA, and we have done that. There is beauty in the complexity that arises from such simple systems. The more we studied CA the more questions we had. This is a subject that has not been studied as much and there is a lot of room development.

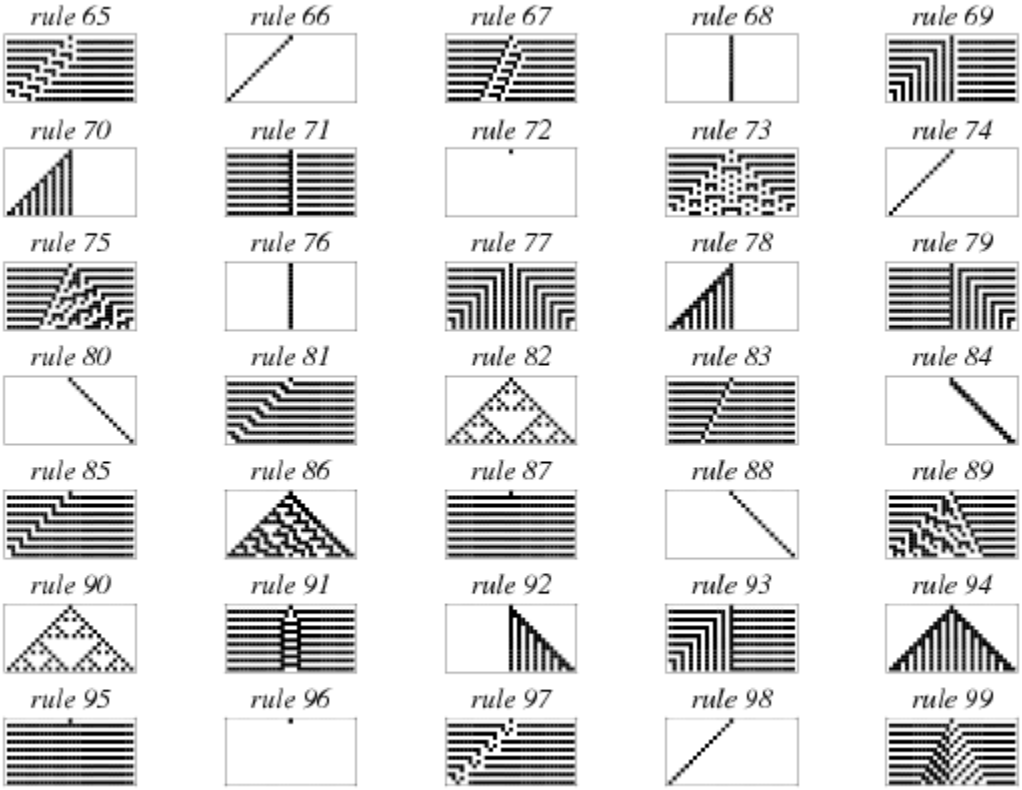
Future work

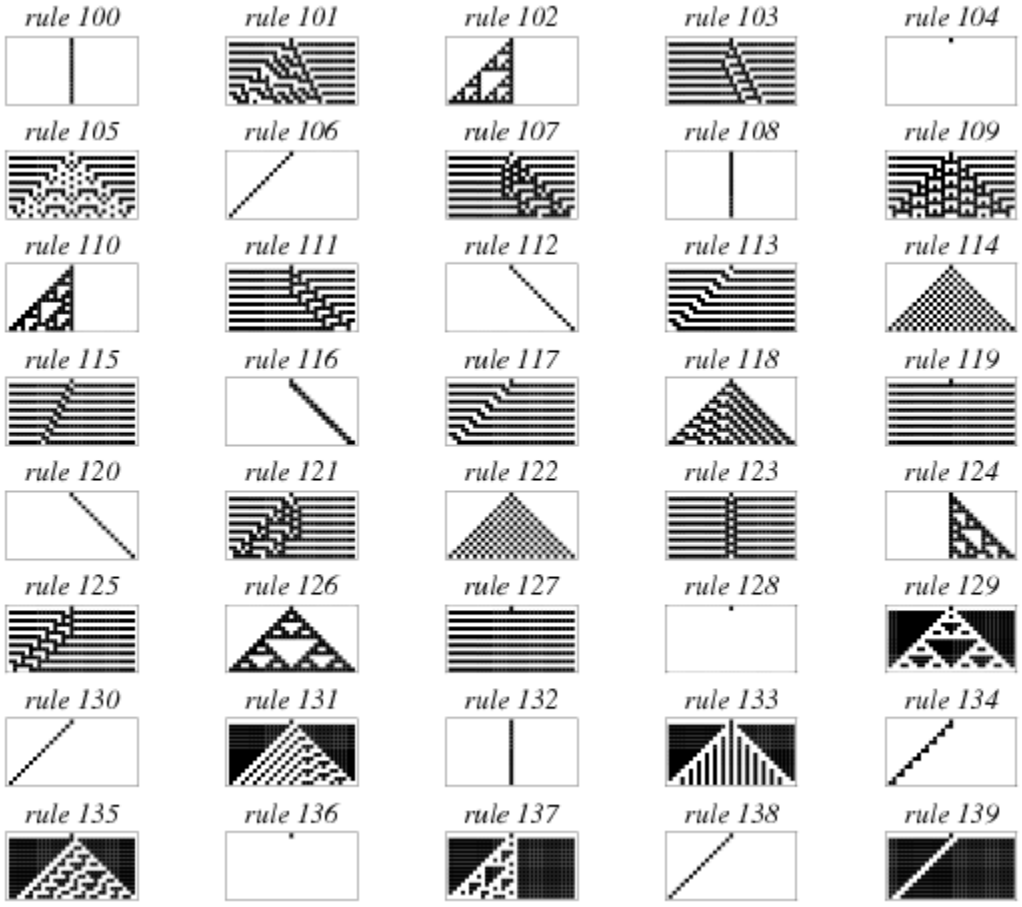
Some possible ways of continuing this work include:

- studying similar properties for totalistic CA
- studying CA with more states or that live in more than one dimension
- look for other classification schemes that improve on the current standard

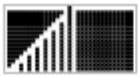
Appendix


rule 0
rule 1
rule 2
rule 3
rule 4
rule 5
rule 6
rule 7
rule 8
rule 9
rule 10
rule 11
rule 12
rule 13
rule 14
rule 15
rule 16
rule 17
rule 18
rule 19
rule 20
rule 21
rule 22
rule 23
rule 24
rule 25
rule 26
rule 27
rule 28
rule 29
rule 30
rule 31
rule 32
rule 33
rule 34
rule 35
rule 36
rule 37
rule 38
rule 39
rule 40
rule 41
rule 42
rule 43
rule 44
rule 45
rule 46
rule 47
rule 48
rule 49
rule 50
rule 51
rule 52
rule 53
rule 54
rule 55
rule 56
rule 57
rule 58
rule 59
rule 60
rule 61
rule 62
rule 63
rule 64


rule 65
rule 66
rule 67
rule 68
rule 69
rule 70
rule 71
rule 72
rule 73
rule 74
rule 75
rule 76
rule 77
rule 78
rule 79
rule 80
rule 81
rule 82
rule 83
rule 84
rule 85
rule 86
rule 87
rule 88
rule 89
rule 90
rule 91
rule 92
rule 93
rule 94
rule 95
rule 96
rule 97
rule 98
rule 99


rule 100
rule 101
rule 102
rule 103
rule 104
rule 105
rule 106
rule 107
rule 108
rule 109
rule 110
rule 111
rule 112
rule 113
rule 114
rule 115
rule 116
rule 117
rule 118
rule 119
rule 120
rule 121
rule 122
rule 123
rule 124
rule 125
rule 126
rule 127
rule 128
rule 129
rule 130
rule 131
rule 132
rule 133
rule 134
rule 135
rule 136
rule 137
rule 138
rule 139

rule 140 

rule 141 

rule 142 

rule 143 

rule 144 

rule 145 

rule 146 

rule 147 

rule 148 

rule 149 

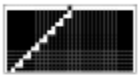
rule 150 

rule 151 

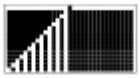
rule 152 

rule 153 

rule 154 

rule 155 

rule 156 

rule 157 

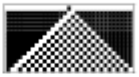
rule 158 

rule 159 

rule 160 

rule 161 

rule 162 

rule 163 

rule 164 

rule 165 

rule 166 

rule 167 

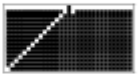
rule 168 

rule 169 

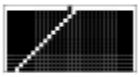
rule 170 

rule 171 

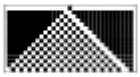
rule 172 

rule 173 

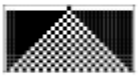
rule 174 

rule 175 

rule 176 

rule 177 

rule 178 

rule 179 

rule 180

rule 181

rule 182

rule 183

rule 184

rule 185

rule 186

rule 187

rule 188

rule 189

rule 190

rule 191

rule 192

rule 193

rule 194

rule 195

rule 196

rule 197

rule 198

rule 199

rule 200

rule 201

rule 202

rule 203

rule 204

rule 205

rule 206

rule 207

rule 208

rule 209

rule 210

rule 211

rule 212

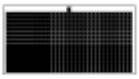
rule 213

rule 214

rule 245

rule 246

rule 247



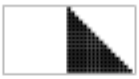
rule 248

rule 249

rule 250

rule 251

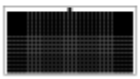
rule 252



rule 253

rule 254

rule 255



Acknowledgement

I would like to give special thanks to Dr. Rich without whom this project would not have been conducted. It was a great experience working on this project with him as an advisor. I would also like to thank Dr. Ahmad, who introduced me to the programming environment that made writing the software for this project easy. Finally I would like to thank Dr. Brumbaugh-Smith for being a second reader to this paper.

Bibliography

Wolfram, Stephen. A New Kind of Science. 2002.

Cook, Matthew. "Universality in Elementary Cellular Automata." Complex Systems (2004): 1-40.

Pesavento, Umberto. "An implementation of von Neumann's self-reproducing machine." Artificial Life (MIT Press) 1995: 337-354.

Tim Tyler, Howard Gutowitz. Cellular Automata FAQ. 23 6 2011 <http://cafaq.com/classify/index.php>.

Chapman, Paul. "Life Universal Computer." <http://www.igblan.free-online.co.uk/igblan/ca/>.